HED Python

Release 0.0.1

HED Working Group

Jun 07, 2022

CONTENTS:

Links

- PDF docs
- Source code

Note: this is a work in progress. More information is coming.

CHAPTER

ONE

INTRODUCTION TO HED

Contents

1.1 Why HED?

Why use HED?

HED (Hierarchical Event Descriptors) is an infrastructure and a controlled vocabulary that allows researchers to annotate their experimental data, especially events, so that tools can automatically use this information in analysis.

For more information on using Hierarchical Event Descriptors (HED) visit HED examples:

1.2 Installing hedtools

Hedtools will be available soon on pypi, but in the meantime, you can install directly from the GitHub repository using the following command:

```
>>> pip install git+https://github.com/hed-standard/hed-python.git
```

1.3 Finding help

Mailing lists and forums

- Don't hesitate to ask questions about the python hedtools on NeuroStars.

Issues and problems

- If you notice a bug in the python hedtools code or encounter other problems using the tools, please open an issue in the hed-python repository on github.

CHAPTER

TWO

HED TOOLS USER GUIDE

CHAPTER

THREE

HED API REFERENCE

3.1 HED models

hed.models	Data structures for HED tag handling.
hed.models.BaseInput(file[, file_type, ...])	Superclass representing a basic columnar file.
hed.models.ColumnMapper([sidecars, ...])	Mapping of a base input file columns into HED tags.
hed.models.ColumnMetadata([column_type, ...])	Column in a ColumnMapper or top-level Sidecar dict.
hed.models.DefinitionDict()	Gathers definitions from a single source.
hed.models.DefinitionEntry(name, contents, ...)	A single definition.
hed.models.DefMapper([def_dicts])	Handles converting Def/ and Def-expand/.
hed.models.HedGroup([hed_string, startpos, ...])	A single parenthesized hed string.
hed.models.HedGroupBase([hed_string, ...])	Base class for the HedGroup API.
hed.models.HedGroupFrozen(contents[, hed_string])	A frozen version of the HedGroup.
hed.models.HedOps(*args, **kwargs)	Base class to support HedOps.
hed.models.HedString(hed_string[, ...])	A HED string.
hed.models.HedStringGroup(hed_string_obj_list)	A container with hed string objects.
hed.models.HedTag(hed_string[, span, hed_schema])	A single HED tag.
hed.models.OnsetMapper(def_mapper)	HedOps responsible for matching onset/offset pairs.
hed.models.Sidecar(file[, name])	Contents of a JSON file or merged file.
hed.models.SpreadsheetInput([file, ...])	A spreadsheet of HED tags.
hed.models.TabularInput([file, sidecar, ...])	A BIDS tabular tsv file with sidecar.
hed.models.hed_ops	Infrastructure for processing HED operations.

3.1.1 hed.models package

Submodules

hed.models.base_input module

class BaseInput(*file*, *file_type=None*, *worksheet_name=None*, *has_column_names=True*, *mapper=None*, *def_mapper=None*, *name=None*)

Bases: `object`

Superclass representing a basic columnar file.

COMMA_DELIMITER = ','

EXCEL_EXTENSION = ['.xlsx']

FILE_EXTENSION = ['.tsv', '.txt', '.xlsx']

FILE_INPUT = 'file'

STRING_INPUT = 'string'

TAB_DELIMITER = '\t'

TEXT_EXTENSION = ['.tsv', '.txt']

convert_to_long(*hed_schema*, *error_handler=None*)

Convert all tags to long form.

Parameters

- **hed_schema** (HedSchema) – The schema to use to convert tags.
- **error_handler** (*ErrorHandler*) – The error handler to use for context, uses a default if none.

Returns A list of issue dictionaries corresponding to issues found during conversion.

Return type dict

convert_to_short(*hed_schema*, *error_handler=None*)

Convert all tags to short form.

Parameters

- **hed_schema** (HedSchema) – The schema to use to convert tags.
- **error_handler** (*ErrorHandler*) – The error handler to use for context, uses a default if none.

Returns A list of issue dictionaries corresponding to issues found during conversion.

Return type dict

property dataframe

The underlying dataframe.

extract_definitions(*error_handler=None*)

Gather and validate all definitions.

Parameters **error_handler** (*ErrorHandler*) – The error handler to use for context or a default if None.

Returns Contains all the definitions located in the file.

Return type *DefinitionDict*

get_def_and_mapper_issues(*error_handler*, *check_for_warnings=False*)

Return definition and column issues.

Parameters

- **error_handler** (*ErrorHandler*) – The error handler to use.
- **check_for_warnings** (*bool*) – If True check for and return warnings as well as errors.

Returns A list of definition and mapping issues. Each issue is a dictionary.

Return type dict

get_worksheet(*worksheet_name=None*)

Get the requested worksheet.

Parameters **worksheet_name** (*str or None*) – The name of the requested worksheet by name or the first one if None.

Returns The workbook request.

Return type openpyxl.workbook.Workbook

Notes

If None, returns the first worksheet.

property has_column_names

True if dataframe has column names.

iter_dataframe(*hed_ops=None*, *mapper=None*, *return_string_only=True*, *run_string_ops_on_columns=False*, *error_handler=None*, *expand_defs=False*, *remove_definitions=True*, ***kwargs*)

Iterate rows based on the given column mapper.

Parameters

- **hed_ops** (*list, func, HedOps, or None*) – A func, a HedOps or a list of these to apply to the hed strings before returning.
- **mapper** (*ColumnMapper or None*) – The column name to column number mapper (or internal mapper if None).
- **return_string_only** (*bool*) – If True, do not return issues list, individual columns, attribute columns, etc.
- **run_string_ops_on_columns** (*bool*) – If true, run all tag and string ops on columns, rather than columns then rows.
- **error_handler** (*ErrorHandler or None*) – The error handler to use for context or a default if None.
- **expand_defs** (*bool*) – If True, expand def tags into def-expand groups.
- **remove_definitions** (*bool*) – If true, remove all definition tags found.
- **()** (*kwargs*) – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Yields *dict* – A dict with parsed row, including keys: "HED", "column_to_hed_tags", and possibly "column_issues".

iter_raw(*hed_ops=None*, *error_handler=None*, ***kwargs*)

Iterate all columns without substitutions

Parameters

- **hed_ops** (*list, func, HedOps, or None*) – A func, a HedOps or a list of these to apply to the hed strings before returning.
- **error_handler** (*ErrorHandler or None*) – Handler to use for context or a default one if None.
- **kwargs** –

Yields - *dict* – A dict with column_number keys and values corresponding to the cell at that position.

Notes

- See models.hed_ops.translate_ops or the specific hed_ops for additional options.
- Primarily for altering or re-saving the original file (e.g., convert short tags to long).
- Used for initial processing when trying to find definitions.

property loaded_workbook

The underlying loaded workbooks.

property name

Name of the data.

reset_mapper(*new_mapper*)

Set mapper to a different view of the file.

Parameters **new_mapper** (ColumnMapper) – A column mapper to be associated with this base input.

set_cell(*row_number*, *column_number*, *new_string_obj*, *include_column_prefix_if_exist=False*, *tag_form='short_tag'*)

Replace the specified cell with transformed text.

Parameters

- **row_number** (*int*) – The row number of the spreadsheet to set.
- **column_number** (*int*) – The column number of the spreadsheet to set.
- **new_string_obj** (HedString) – Object with text to put in the given cell.
- **include_column_prefix_if_exist** (*bool*) – If True and the column matches one from mapper _column_prefix_dictionary, remove the prefix.
- **tag_form** (*str*) – Version of the tags (short_tag, long_tag, base_tag, etc)

Notes

Any attribute of a HedTag that returns a string is a valid value of tag_form.

to_csv(*file=None*, *output_processed_file=False*)

Write to file or return as a string.

Parameters

- **file** (*str, file-like, or None*) – Location to save this file. If None, return as string.
- **output_processed_file** (*bool*) – Replace all definitions and labels in HED columns as appropriate. Also fills in things like categories.

Returns None if file is given or the contents as a str if file is None.

Return type None or str

to_excel(*file*, *output_processed_file=False*)

Output to an Excel file.

Parameters

- **file** (*str or file-like*) – Location to save this base input.
- **output_processed_file** (*bool*) – If True, replace definitions and labels in HED columns. Also fills in things like categories.

Raises HedFileError if empty file object or file cannot be opened. –

update_definition_mapper(*def_dict*)

Add definitions from dict(s) if mapper exists.

Parameters def_dict (*list or DefinitionDict*) – Add the DefDict or list of DefDict to the internal definition mapper.

validate_file(*hed_ops*, *name=None*, *error_handler=None*, *check_for_warnings=True*, ***kwargs*)

Run the hed_ops on columns and rows.

Parameters

- **hed_ops** (*func, HedOps, or list of func and/or HedOps*) – The HedOps of funcs to apply.
- **name** (*str*) – If present, use this as the filename for context, rather than using the actual filename Useful for temp filenames.
- **error_handler** (*ErrorHandler or None*) – Used to report errors a default one if None.
- **check_for_warnings** (*bool*) – If True check for and return warnings as well as errors.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Returns The list of validation issues found. The list elements are dictionaries.

Return type list

property worksheet_name

The worksheet name.

hed.models.column_mapper module

class ColumnMapper(*sidecars=None*, *tag_columns=None*, *column_prefix_dictionary=None*, *attribute_columns=None*, *optional_tag_columns=None*)

Bases: `object`

Mapping of a base input file columns into HED tags.

Notes

- Functions and variables column and row indexing starts at 0.

add_columns(*column_names_or_numbers*, *column_type=ColumnType.Attribute*)

Add blank columns in the given column category.

Parameters

- **column_names_or_numbers** (*list*) – A list of column names or numbers to add as the specified type.

- **column_type** (*ColumnType property*) – The category of column these should be.

add_sidecars(*sidecars*)

Add sidecar column info.

Parameters **sidecars** (*list*) – A list of filenames or loaded sidecar files in any mix.

expand_row_tags(*row_text*)

Expand all mapped columns for row.

Parameters **row_text** (*list*) – The text for the given row, one list entry per column number.

Returns A dictionary containing the keys COLUMN_TO_HED_TAGS, COLUMN_ISSUES, and arbitrary attributes.

Return type dict

Notes

- The "column_to_hed_tags" is each expanded column given separately as a list of HedStrings.
- Attributes are any column identified as an attribute. They will appear in the return value as {attribute_name: value_of_column}

get_column_mapping_issues()

Get all the issues with finalizing column mapping. Primarily a missing required column.

Returns A list dictionaries of all issues found from mapping column names to numbers.

Return type list

get_def_dicts()

Return def dicts from every column description.

Returns A list of DefinitionDict objects corresponding to each column entry.

Return type list

get_prefix_remove_func(*column_number*)

Return a function to removes name prefixes for column

Parameters **column_number** (*int*) – Column number to look up in the prefix dictionary.

Returns A function taking a tag and string, returning a string.

Return type func

set_column_map(*new_column_map=None*)

Set the column number to name mapping.

Parameters **new_column_map** (*list or dict*) – Either an ordered list of the column names or column_number:column name dictionary. In both cases column numbers start at 0

Returns List of issues. Each issue is a dictionary.

Return type list

set_column_prefix_dict(*column_prefix_dictionary*, *finalize_mapping=True*)

Replace the column prefix dictionary

Parameters

- **column_prefix_dictionary** (*dict*) – Dictionary with keys that are column numbers and values are HED tag prefixes to prepend to the tags in that column before processing.
- **finalize_mapping** (*bool*) – Re-generate the internal mapping if True, otherwise no effect until finalize.

Returns List of issues that occurred during this process. Each issue is a dictionary.

Return type list

set_tag_columns(*tag_columns=None*, *optional_tag_columns=None*, *finalize_mapping=True*)

Set tag columns and optional tag columns

Parameters

- **tag_columns** (*list*) – A list of ints or strings containing the columns that contain the HED tags. If None, clears existing tag_columns
- **optional_tag_columns** (*list*) – A list of ints or strings containing the columns that contain the HED tags, but not an error if missing. If None, clears existing tag_columns
- **finalize_mapping** (*bool*) – Re-generate the internal mapping if True, otherwise no effect until finalize.

Returns List of issues that occurred during this process. Each issue is a dictionary.

Return type list

validate_column_data(*hed_ops*, *error_handler=None*, ***kwargs*)

Validate the column data.

Parameters

- **hed_ops** (*list, func, or* HedOps) – A func, a HedOps or a list of these to apply to the hed strings in the sidecars.
- **error_handler** (*ErrorHandler or None*) – Used to report errors. Uses a default one if none passed in.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Returns A list of syntax and semantic issues found in the definitions. Each issue is a dictionary.

Return type list

hed.models.column_metadata module

class ColumnMetadata(*column_type=None*, *name=None*, *hed_dict=None*, *column_prefix=None*, *error_handler=None*)

Bases: object

Column in a ColumnMapper or top-level Sidecar dict.

property def_dict

Return the definition dictionary for this column.

Returns Contains all the definitions located in the column.

Return type *DefinitionDict*

expand(*input_text*)

Expand text using the rules for this column.

Parameters **input_text** (*str*) – Text to expand (generally from a single cell in a spreadsheet).

Returns

The expanded column as a hed_string. str or dict: If this is a string, contains the name of this column

as an attribute. If the first return value is None, this is an error message dictionary.

Return type str or None

Notes

- Examples are adding name_prefix, inserting a column hed_string from a category key, etc.

extract_definitions(*error_handler=None*)

Gather and validate definitions in metadata.

Parameters **error_handler** (*ErrorHandler*) – The error handler to use for context, uses a default one if None.

Returns Contains all the definitions located in the column. issues: List of issues encountered in extracting the definitions. Each issue is a dictionary.

Return type *DefinitionDict*

get_definition_issues()

Return the issues found extracting definitions.

Returns A list of issues found when parsing definitions. Individual issues are dictionaries.

Return type list

property hed_dict

The loaded dict for any given entry.

Returns A dict which generally contains a "HED" entry and optional others like description.

Return type dict

hed_string_iter(*hed_ops=None*, *error_handler=None*, ***kwargs*)

Iterator yielding column hed strings.

Parameters

- **hed_ops** (*func, HedOps, or list of these*) – The HedOps or funcs to apply to the hed strings before returning.
- **error_handler** (*ErrorHandler*) – The error handler to use for context, uses a default one if none.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options

Yields *tuple* – - HedString: The hed string at a given column and key position. - str: Indication of the where hed string was loaded from so it can be later set by the user. - list: Issues found applying hed_ops. Each issue is a dictionary.

remove_prefix(*original_tag*, *current_tag_text*)

Remove column_prefix if present from tag.

Parameters

- **original_tag** (HedTag) – The original hed tag being written.
- **current_tag_text** (*str*) – A single tag as a string, in any form.

Returns current_tag_text with required prefixes removed

Return type str

set_hed_string(*new_hed_string*, *position=None*, *set_def_removed=False*)

Set a hed string for a category key/etc.

Parameters

- **new_hed_string** (*str or* HedString) – The new hed_string to replace the value at position.
- **position** (*str, optional*) – This should only be a value returned from hed_string_iter.
- **set_def_removed** (*bool*) – If True, set the version with definitions removed, rather than the normal version.

Raises **TypeError** – If the mapping cannot occur.

validate_column(*hed_ops*, *error_handler*, ***kwargs*)

Run the given hed_ops on this column.

Parameters

- **hed_ops** (*list or func or* HedOps) – to the hed strings in the columns.
- **error_handler** (*ErrorHandler or None*) – Used to report errors. Uses a default one if none passed in.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Returns Issues found by the given hed_ops. Each issue is a dictionary.

Return type list

class ColumnType(*value*)

Bases: enum.Enum

The overall column_type of a column in column mapper, eg treat it as HED tags.

Mostly internal to column mapper related code

Attribute = 'attribute'

Categorical = 'categorical'

HEDTags = 'hed_tags'

Ignore = 'ignore'

Unknown = None

Value = 'value'

hed.models.def_mapper module

class DefMapper(*def_dicts=None*)

Bases: *hed.models.hed_ops.HedOps*

Handles converting Def/ and Def-expand/.

Notes

- The class provides string funcs but no tag funcs when extending HedOps.
- The class can expand or shrink definitions in hed strings via Def/XXX and (Def-expand/XXX ...).

add_definitions(*def_dicts*, *add_as_temp=False*)

Add definitions from dict(s) to mapper

Parameters

- **def_dicts** (*list or DefinitionDict*) – DefDict or list of DefDicts whose definitions should be added.
- **add_as_temp** (*bool*) – If true, mark these new definitions as temporary (easily purged).

add_definitions_from_string_as_temp(*hed_string_obj*)

Add definitions from hed string as temporary.

Parameters **hed_string_obj** (HedString) – Hed string object to search for definitions

Returns List of issues due to invalid definitions found in this string. Each issue is a dictionary.

Return type list

clear_temporary_definitions()

Remove any previously added temporary definitions.

expand_and_remove_definitions(*hed_string_obj*, *check_for_definitions=False*, *expand_defs=True*, *shrink_defs=False*, *remove_definitions=True*)

Validate and expand Def/Def-Expand tags.

Also removes definitions

Parameters

- **hed_string_obj** (HedString) – The string to search for definitions.
- **check_for_definitions** (*bool*) – If True, this will first check the hed string for any definitions.
- **expand_defs** (*bool*) – If True, replace Def tags to Def-expand tag groups.
- **shrink_defs** (*bool*) – If True, replace Def-expand groups with Def tags.
- **remove_definitions** (*bool*) – If true, this will remove all Definition tag groups.

Returns A list of issues for definition-related tags in this string. Each issue is a dictionary.

Return type def_issues (list)

Notes

- The check_for_definitions is mainly used for individual HedStrings in isolation.
- The defs can be expanded or shrunk, while definitions can be removed.
- This does not validate definitions, it will blindly remove invalid definitions as well.

expand_def_tags(*hed_string_obj*, *expand_defs=True*, *shrink_defs=False*)

Validate and expand Def/Def-Expand tags.

Parameters

- **hed_string_obj** (HedString) – The hed string to process.
- **expand_defs** (*bool*) – If true, convert def tags to def-expand tag groups that include definition content.
- **shrink_defs** (*bool*) – If True, replace all def-expand groups with corresponding def tags.

Returns Issues found related to validating defs. Each issue is a dictionary.

Return type list

Notes

- This function can optionally expand or shrink Def/ and Def-expand, respectively.
- Usually issues are mismatched placeholders or a missing definition.
- The expand_defs and shrink_defs cannot both be True.

get_def_entry(*def_name*)

Get the definition entry for the definition name.

Parameters **def_name** (*str*) – Name of the definition to retrieve.

Returns Definition entry for the requested definition.

Return type *DefinitionEntry*

property issues

hed.models.definition_dict module

class DefinitionDict

Bases: *hed.models.hed_ops.HedOps*

Gathers definitions from a single source.

This class extends HedOps because it has string_funcs to check for definitions. It has no tag_funcs.

check_for_definitions(*hed_string_obj*, *error_handler=None*)

Check string for definition tags, adding them to self.

Parameters

- **hed_string_obj** (HedString) – A single hed string to gather definitions from.
- **error_handler** (*ErrorHandler or None*) – Error context used to identify where definitions are found.

Returns List of issues encountered in checking for definitions. Each issue is a dictionary.

Return type list

property defs

Provides direct access to internal dictionary. Alter at your own risk.

Returns {str: DefinitionEntry}

Return type dict

get_definition_issues()

Return definition errors found during extraction.

Returns List of DefinitionErrors issues found. Each issue is a dictionary.

Return type list

class DefinitionEntry(*name*, *contents*, *takes_value*, *source_context*)

Bases: object

A single definition.

get_definition(*replace_tag*, *placeholder_value=None*)

Return a copy of the definition with the tag expanded and the placeholder plugged in.

Parameters

- **replace_tag** (HedTag) – The def hed tag to replace with an expanded version
- **placeholder_value** (*str or None*) – If present and required, will replace any pound signs in the definition contents.

Returns The expanded def tag name HedGroup: The contents of this definition(including the def tag itself)

Return type str

Raises **ValueError** – If a placeholder_value is passed, but this definition doesn't have a placeholder.

add_group_to_dict(*group*, *tag_dict=None*)

Add the tags and their values from a HED group to a tag dictionary.

Parameters

- **group** (HedGroup) – Contents to add to the tag dict.
- **tag_dict** (*dict*) – The starting tag dictionary to which to add to.

Returns The updated tag_dict containing the tags with a list of values.

Return type dict

Notes

- Expects tags to have forms calculated already.

hed.models.expression_parser module

class Expression(*token*, *left=None*, *right=None*)

Bases: object

handle_expr(*hed_group*, *exact=False*)

class ExpressionAnd(*token*, *left=None*, *right=None*)

Bases: *hed.models.expression_parser.Expression*

handle_expr(*hed_group*, *exact=False*)

class ExpressionContainingGroup(*token*, *left=None*, *right=None*)

Bases: *hed.models.expression_parser.Expression*

handle_expr(*hed_group*, *exact=False*)

class ExpressionExactGroup(*token*, *left=None*, *right=None*)

Bases: *hed.models.expression_parser.Expression*

handle_expr(*hed_group*, *exact=False*)

class ExpressionLogicalGroup(*token*, *left=None*, *right=None*)

Bases: *hed.models.expression_parser.Expression*

handle_expr(*hed_group*, *exact=False*)

class ExpressionNegation(*token*, *left=None*, *right=None*)

Bases: *hed.models.expression_parser.Expression*

handle_expr(*hed_group*, *exact=False*)

class ExpressionOr(*token*, *left=None*, *right=None*)

Bases: *hed.models.expression_parser.Expression*

handle_expr(*hed_group*, *exact=False*)

class TagExpressionParser(*expression_string*)

Bases: object

Parse a search expression into a form than can be used to search a hed string.

search_hed_string(*hed_string_obj*)

class Token(*text*)

Bases: object

And = 0

ContainingGroup = 8

ContainingGroupEnd = 9

ExactGroup = 2

ExactGroupEnd = 3

LogicalGroup = 5

LogicalGroupEnd = 6

LogicalNegation = 7

Or = 4

Tag = 1

hed.models.hed_group module

class HedGroup(*hed_string=''*, *startpos=None*, *endpos=None*, *contents=None*)

Bases: *hed.models.hed_group_base.HedGroupBase*

A single parenthesized hed string.

append(*tag_or_group*)

Add a tag or group to this group.

Parameters **tag_or_group** (*HedTag or HedGroup*) – The new object to add to this group.

Raises **ValueError** – If a HedGroupFrozen.

check_if_in_original(*tag_or_group*)

Check if the tag or group in original string.

Parameters **tag_or_group** (*HedTag or HedGroup*) – The HedTag or HedGroup to be looked for in this group.

Returns True if in this group.

Return type bool

get_frozen()

Return a frozen (non-mutable) copy of this HedGroup.

This is a deep copy if the group was not already frozen.

Returns A frozen copy of this HedGroup.

Return type *HedGroupFrozen*

remove(*items_to_remove*)

Remove any tags/groups in items_to_remove.

Parameters **items_to_remove** (*list*) – List of HedGroups and/or HedTags to remove.

Notes

- Any groups that become empty will also be pruned.
- Identity, not equivalence is used in determining whether to remove.

replace(*item_to_replace*, *new_contents*)

Replace an existing tag or group.

Parameters

- **item_to_replace** (HedTag *or* HedGroup) – The item to replace must exist or this will raise an error.
- **new_contents** (HedTag *or* HedGroup) – Replacement contents.

class HedGroupFrozen(*contents*, *hed_string=None*)

Bases: *hed.models.hed_group_base.HedGroupBase*

A frozen version of the HedGroup.

Notes

- Searching and getting all tags/groups will be faster.
- Additionally, HedGroupFrozen is order-agnostic: (a, b) = (b, a).

get_all_groups(*also_return_depth=False*)

Return HedGroups, including descendants and self.

Parameters **also_return_depth** (*bool*) – If True, this yields tuples (group, depth) rather than just groups.

Returns The list of all HedGroups in this group, including descendants and self.

Return type list

get_all_tags()

Return HedTags, including descendants.

Returns A list of all the HedTags in this group including descendants.

Return type list

Notes

- This list is cached when initially called since its contents never change.

get_frozen()

Return this frozen group.

Returns This same group.

Return type *HedGroupFrozen*

hed.models.hed_group_base module

class HedGroupBase(*hed_string=''*, *startpos=None*, *endpos=None*)

Bases: object

Base class for the HedGroup API.

Notes

- This interface is shared by HedGroup and HedGroupFrozen.

property children

A list of the direct children.

copy()

Return a deep copy of this group.

Returns The copied group.

Return type *HedGroupBase*

Notes

- The parent tag is removed.

find_def_tags(*recursive=False*, *include_groups=3*)

Find def and def-expand tags

Parameters

- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*int, 0, 1, 2, 3*) – options for how to expand or include groups

Returns A list of tuples. The contents depends on the values of the include group.

Return type list

Notes

- **The include_groups option controls the tag expansion as follows:**
  - If 0: Return only def and def expand tags/.
  - If 1: Return only def tags and def-expand groups.
  - If 2: Return only groups containing defs, or def-expand groups.
  - If 3 or any other value: Return all 3 as a tuple.

find_exact_tags(*tags_or_groups*, *recursive=False*)

Find the given tags or groups.

Parameters

- **tags_or_groups** (HedTag, HedGroupBase) – A container of tags to locate.
- **recursive** (*bool*) – If true, also check subgroups.

Returns A list of HedGroupBases the given tags/groups were found in.

Return type list

Notes

- If you pass in groups it will only find EXACT matches.
- This can only find identified tags.
- By default, definition, def, def-expand, onset, and offset are identified, even without a schema.
- If this is a HedGroup, order matters. (b, a) != (a, b)
- **If this is a HedGroupFrozen:**

  if "(a, b)" in tags_or_groups, then it will match 1 and 2, but not 3.

  1. (a, b)
  2. (b, a)
  3. (a, b, c)

`find_placeholder_tag()`

Return a placeholder tag, if present in this group.

Returns The placeholder tag if found.

Return type *HedTag* or None

Notes

- Assumes a valid HedString with no erroneous "#" characters.

`find_tags(search_tags, recursive=False, include_groups=2)`

Find the tags and their containing groups.

Parameters

- **search_tags** (*container*) – A container of short_base_tags to locate
- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*0, 1 or 2*) – Specify return values.

Returns The contents of the list depends on the value of include_groups.

Return type list

Notes

- If include_groups is 0, return a list of the HedTags.
- If include_groups is 1, return a list of the HedGroups containing the HedTags.
- If include_groups is 2, return a list of tuples (HedTag, HedGroup) for the found tags.
- This can only find identified tags.
- By default, definition, def, def-expand, onset, and offset are identified, even without a schema.

find_tags_with_term(*term*, *recursive=False*, *include_groups=2*)

Find any tags that contain the given term.

Note: This can only find identified tags.

Parameters

- **term** (*str*) – A single term to search for.
- **recursive** (*bool*) – If true, recursively check subgroups.
- **include_groups** (*int, 0, 1, 2, 3*) – 0, 1 or 2 If 0: Return only tags If 1: return only groups If 2 or any other value: return both
- **recursive** – If true, also check subgroups.
- **include_groups** – options for how to expand or include groups

Return type list

def_tag: HedTag The located def tag

def_expand_group: HedGroup or None If this is a def-expand rather than def tag, this will be the entire def-expand group.

group: HedGroup The group the def tag or def expand group is in.

Notes

- **The include_groups option controls the tag expansion as follows:**
  - If 0: Return only def and def expand tags/.
  - If 1: Return only def tags and def-expand groups.
  - If 2: Return only groups containing defs, or def-expand groups.
  - If 3 or any other value: Return all 3 as a tuple.

get_all_groups(*also_return_depth=False*)

Return HedGroups, including descendants and self.

Parameters **also_return_depth** (*bool*) – If True, yield tuples (group, depth) rather than just groups.

Returns The list of all HedGroups in this group, including descendants and self.

Return type list

get_all_tags()

Return HedTags, including descendants.

Returns A list of all the tags in this group including descendants.

Return type list

get_as_form(*tag_attribute*, *tag_transformer=None*)

Get the string corresponding to the specified form.

Parameters

- **tag_attribute** (*str*) – The hed_tag property to use to construct the string (usually short_tag or long_tag).

- **tag_transformer** (*func or None*) – A function that is applied to each tag string before returning.

Returns The constructed string after transformation

Return type str

Notes

- The signature of a tag_transformer is str def(HedTag, str).

get_as_long()

Return this HedGroup as a long tag string.

Returns The group as a string with all tags as long tags.

Return type str

get_as_short()

Return this HedGroup as a short tag string.

Returns The group as a string with all tags as short tags.

Return type str

get_original_hed_string()

Get the original hed string.

Returns The original string with no modification.

Return type str

groups()

Return the direct child groups of this group.

Returns All groups directly in this group, filtering out HedTag children.

Return type list

property is_group

True if this is a parenthesized group.

lower()

Convenience function, equivalent to str(self).lower()

property span

Return the source span.

Returns start index of the group (including parentheses) from the source string. int: end index of the group (including parentheses) from the source string.

Return type int

tags()

Return the direct child tags of this group.

Returns All tags directly in this group, filtering out HedGroup children.

Return type list

hed.models.hed_ops module

Infrastructure for processing HED operations.

class HedOps(**args*, ***kwargs*)

Bases: object

Base class to support HedOps.

Notes

- HED ops are operations that apply to HedStrings in a sequence.

translate_ops(*hed_ops*, *split_ops=False*, ***kwargs*)

Return functions to apply to a hed string object.

Parameters

- **hed_ops** (*list*) – A list of func or HedOps or HedSchema to apply to hed strings.
- **split_ops** (*bool*) – If true, will split the operations into separate lists of tag and string operations.

kwargs (dict): An optional dictionary of name-value pairs representing parameters passed to each HedOps

Returns A list of functions to apply or a tuple containing separate lists of tag and string ops.

Return type list or tuple

Notes

- The distinction between tag and string ops primarily applies to spreadsheets.
- **Splitting the ops into two lists is mainly used for parsing spreadsheets where any given** column isn't an entire hed string, but additional detail is needed on which column an issue original came from.
- **The currently accepted values of kwargs are:**
  - allow_placeholders
  - check_for_definitions
  - expand_defs
  - shrink_defs
  - error_handler
  - check_for_warnings
  - remove_definitions

hed.models.hed_string module

This module is used to split tags in a HED string.

class HedString(*hed_string*, *hed_schema=None*, *_contents=None*)

Bases: *hed.models.hed_group.HedGroup*

A HED string.

CLOSING_GROUP_CHARACTER = ')'

OPENING_GROUP_CHARACTER = '('

apply_funcs(*string_funcs*)

Run functions on this string.

Parameters **string_funcs** (*list*) – A list of functions that take a hed string object and return a list of issues.

Returns A list of issues found by these operations. Each issue is a dictionary.

Return type list

Notes

- This method potentially modifies the hed string object.

convert_to_canonical_forms(*hed_schema*)

Identify all tags using the given schema.

If schema is None, still identify "key" tags such as definitions.

Parameters **hed_schema** (*HedSchema or None*) – The schema to use to validate/convert tags.

Returns A list of issues found while converting the string. Each issue is a dictionary.

Return type list

convert_to_long(*hed_schema*)

Compute canonical forms and return the long form.

Parameters **hed_schema** (*HedSchema or None*) – The schema to use to calculate forms.

Returns

- str: The string with all tags converted to long form.
- list: A list of issues found during conversion. Each issue is a dictionary.

Return type tuple

Notes

- No issues will be found if no schema is passed.

convert_to_original()

Return the original form of this string.

Returns The string with all the tags in their original form.

Return type str

Notes

Potentially with some extraneous spaces removed on returned string.

convert_to_short(*hed_schema*)

Compute canonical forms and return the short form.

Parameters **hed_schema** (HedSchema *or None*) – The schema to use to calculate forms.

Returns

- str: The string with all tags converted to short form.
- list: A list of issues found during conversion. Each issue is a dictionary.

Return type tuple

Notes

- No issues will be found if no schema is passed.

find_top_level_tags(*anchor_tags*, *include_groups=2*)

Find top level groups with an anchor tag.

Parameters

- **anchor_tags** (*container*) – A list/set/etc of short_base_tags to find groups by.
- **include_groups** (*0, 1 or 2*) – Parameter indicating what return values to include.

Returns

The returned result depends on include_groups:

- If 0: return only tags.
- If 1: return only groups.
- If 2 or any other value: return both.

Return type list or tuple

Notes

- A max of 1 tag located per top level group.

classmethod from_hed_strings(*contents*)

Factory for creating HedStrings via combination.

Parameters **contents** (*list(HedString) or None*) – A list of HedString objects to combine. This takes ownership of their children.

get_frozen()

Return a frozen copy of this HedString.

This is a deep copy if the group was not already frozen.

Returns A frozen copy of this HedString.

Return type *HedStringFrozen*

property is_group

Always False since the underlying string is not a group with parentheses.

remove_definitions()

Remove definition tags and groups from this string.

This does not validate definitions and will blindly removing invalid ones as well.

Returns An empty list as there are no possible issues, this list is always blank.

Return type list

static split_hed_string(*hed_string*)

Split a HED string into delimiters and tags.

Parameters **hed_string** (*str*) – The HED string to split.

Returns A list of tuples where each tuple is (is_hed_tag, (start_pos, end_pos)).

Return type list

Notes

- **The tuple format is as follows**
  - is_hed_tag (bool): A (possible) hed tag if true, delimiter if not.
  - start_pos (int): Index of start of string in hed_string.
  - end_pos (int): Index of end of string in hed_string
- This function does not validate tags or delimiters in any form.

static split_into_groups(*hed_string*, *hed_schema=None*)

Split the HED string into a parse tree.

Parameters

- **hed_string** (*str*) – A hed string consisting of tags and tag groups to be processed.
- **hed_schema** (*HedSchema or None*) – Hed schema to use to identify tags.

Returns A list of HedTag and/or HedGroupBase.

Return type list

Raises **ValueError** – If the string is significantly malformed, such as mismatched parentheses.

Notes

- The parse tree consists of tag groups, tags, and delimiters.

validate(*hed_ops=None*, *error_handler=None*, ***kwargs*)

Run the given hed_ops on this string.

Parameters

- **hed_ops** – (func, HedOps, or list): Operations to apply to this object.
- **error_handler** (*ErrorHandler or None*) – Used to report errors in context. Uses a default if None.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options

Returns A list of issues encountered in applying these operations. Each issue is a dictionary.

Return type list

Notes

- Although this function is called validation, the HedOps can represent other transformations.

class HedStringFrozen(*hed_string*, *hed_schema=None*)

Bases: *hed.models.hed_group.HedGroupFrozen*

Class representing an immutable hed string.

apply_funcs(*string_funcs*)

Run functions on this string.

Parameters **string_funcs** (*list*) – A list of functions that take a hed string object and return a list of issues.

Returns A list of issues found by these operations. Each issue is a dictionary.

Return type list

Notes

- This method potentially modifies the hed string object.

find_top_level_tags(*anchor_tags*, *include_groups=2*)

Find top level groups with an anchor tag.

Parameters

- **anchor_tags** (*container*) – A list/set/etc of short_base_tags to find groups by.
- **include_groups** (*0, 1 or 2*) – Parameter indicating what return values to include.

Returns

The returned result depends on include_groups:

- If 0: return only tags.
- If 1: return only groups.
- If 2 or any other value: return both.

Return type list or tuple

Notes

- A max of 1 tag located per top level group.

`property is_group`

Return False since this does not have parentheses.

`validate`(*hed_ops=None*, *error_handler=None*, ***kwargs*)

Run the given hed_ops on this string.

Parameters

- **hed_ops** – (func, HedOps, or list): Operations to apply to this object.
- **error_handler** (*ErrorHandler or None*) – Used to report errors in context. Uses a default if None.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options

Returns A list of issues encountered in applying these operations. Each issue is a dictionary.

Return type list

Notes

- Although this function is called validation, the HedOps can represent other transformations.

hed.models.hed_string_group module

This module is used to easily concatenate multiple hed strings in place

`class HedStringGroup`(*hed_string_obj_list*)

Bases: *hed.models.hed_string.HedString*

A container with hed string objects.

Notes

- Often this is used for assembling the hed strings from multiple columns.
- The HedStringGroup passes through many of the HedString operations.

`property children`

Return the direct children of this string.

Returns a list of direct children of this group.

Return type list

get_original_hed_string()

Get the original hed string.

Returns The original string with no modification.

Return type str

remove(*items_to_remove*)

Remove any tags/groups in items_to_remove.

Parameters **items_to_remove** (*list*) – A list of HedGroup and HedTag objects to remove.

Notes

- Any groups that become empty will also be pruned.
- This goes by identity, not equivalence.

replace(*item_to_replace*, *new_contents*)

Replace an existing tag or group.

Parameters

- **item_to_replace** (HedTag *or* HedGroup) – The tag to replace.
- **new_contents** (HedTag *or* HedGroup *or* *list*) – The replacements for the tag.

Notes

- It tag must exist in this an error is raised.

property span

Return the source span of this group from the source hed string.

Returns

- int: start index of the group (including parentheses) from the source string.
- int: end index of the group (including parentheses) from the source string.

Return type tuple

hed.models.hed_tag module

class HedTag(*hed_string*, *span=None*, *hed_schema=None*)

Bases: object

A single HED tag.

Notes

- HedTag is a smart class in that it keeps track of its original value and positioning

as well as pointers to the relevant HED schema information, if relevant.

add_prefix_if_needed(*required_prefix*)

Add a prefix to this tag *unless* already formatted.

Parameters **required_prefix** (*str*) – The full name_prefix to add if not present.

Notes

- This means we verify the tag does not have the required name_prefix, or any partial name_prefix.

Examples

Required: KnownTag1/KnownTag2

Case 1: KnownTag1/KnownTag2/ColumnValue Will not be changed, has name_prefix already

Case 2: KnownTag2/ColumnValue Will not be changed, has partial name_prefix already

Case 3: ColumnValue Prefix will be added.

any_parent_has_attribute(*attribute*)

Check if the tag or any of its parents has the attribute.

Parameters **attribute** (*str*) – The name of the attribute to check for.

Returns True if the tag has the given attribute. False, if otherwise.

Return type bool

property attributes

Return a dict of all the attributes this tag has.

Returns empty dict if this is not a value tag.

Returns A dict of attributes this tag has.

Return type dict

Notes

- Returns empty dict if this is not a unit class tag.
- The dictionary has unit name as the key and HedSchemaEntry as value.

property base_tag

Long form without value or extension.

Returns The long form of the tag, without value or extension.

Return type base_tag (str)

Notes

- Only valid after calling convert_to_canonical_forms.

base_tag_has_attribute(*tag_attribute*)

Check to see if the tag has a specific attribute.

Parameters **tag_attribute** (*str*) – A tag attribute.

Returns True if the tag has the specified attribute. False, if otherwise.

Return type bool

convert_to_canonical_forms(*hed_schema*)

Update internal state based on schema.

Parameters **hed_schema** (HedSchema) – The schema to use to validate this tag

Returns A list of issues found during conversion. Each element is a dictionary.

Return type list

property extension_or_value_portion

Get the extension or value of tag

Generally this is just the portion after the last slash. Returns an empty string if no extension or value.

Returns The tag name.

Return type str

Notes

- This tag must have been computed first.

get_stripped_unit_value()

Return the extension portion without units.

Returns The extension portion with the units removed. unit (str or None): None if no valid unit found.

Return type stripped_unit_value (str)

Examples

'Duration/3 ms' will return '3'

get_tag_unit_class_units()

Get the unit class units associated with a particular tag.

Args:

Returns A list containing the unit class units associated with a particular tag or an empty list.

Return type list

get_unit_class_default_unit()

Get the default unit class unit for this tag.

Returns The default unit class unit associated with the specific tag or an empty string.

Return type str

has_attribute(*attribute*)

Return true if this is an attribute this tag has.

Parameters **attribute** (*str*) – Name of the attribute.

Returns True if this tag has the attribute.

Return type bool

is_basic_tag()

Return True if a known tag with no extension or value.

Returns True if this is a known tag without extension or value.

Return type bool

is_extension_allowed_tag()

Check if tag has 'extensionAllowed' attribute.

Recursively checks parent tag entries for the attribute as well.

Returns True if the tag has the 'extensionAllowed' attribute. False, if otherwise.

Return type bool

is_takes_value_tag()

Return true if this is a takes value tag.

Returns True if this is a takes value tag.

Return type bool

is_unit_class_tag()

Return true if this is a unit class tag.

Returns True if this is a unit class tag.

Return type bool

is_value_class_tag()

Return true if this is a value class tag.

Returns True if this is a a tag with a value class.

Return type bool

property library_prefix

Library prefix for this tag if one exists.

Returns The library prefix, including the colon.

Return type prefix (str)

property long_tag

Long form including value or extension.

Returns The long form of this tag.

Return type str

lower()

Convenience function, equivalent to str(self).lower().

property org_base_tag

Original form without value or extension.

Returns The original form of the tag, without value or extension.

Return type base_tag (str)

Notes

- Warning: This could be empty if the original tag had a name_prefix prepended.

e.g. a column where "Label/" is prepended, thus the column value has zero base portion. - Only valid after calling convert_to_canonical_forms.

property org_tag

Return the original unmodified tag.

Returns The original unmodified tag.

Return type str

replace_placeholder(*placeholder_value*)

If tag has a placeholder character(#), replace with value.

Parameters **placeholder_value** (*str*) – Value to replace placeholder with.

property short_base_tag

Short form without value or extension

Returns The short non-extension port of a tag.

Return type base_tag (str)

Notes

- ParentNodes/Def/DefName would return just "Def".

property short_tag

Short form including value or extension.

Returns The short form of the tag, including value or extension.

Return type

short_tag (str)

Note: Only valid after calling convert_to_canonical_forms

property tag

Return the entire user editable attribute in the tag.

Returns the original tag if no user form set.

Returns The custom set user form of the tag.

Return type tag (str)

tag_exists_in_schema()

Get the schema entry for this tag.

Returns True if this tag exists.

Return type bool

Notes

- This does NOT assure this is a valid tag.

tag_modified()

Return true if tag has been modified from original.

Returns Return True if the tag is modified.

Return type bool

Notes

- Modifications can include adding a column name_prefix.

property tag_terms

Return a tuple of all the terms in this tag Lowercase.

Returns Tuple of terms or empty tuple for unidentified tag.

Return type tag_terms (str)

Notes

- Does not include any extension.

property unit_classes

Return a dict of all the unit classes this tag accepts.

Returns A dict of unit classes this tag accepts.

Return type unit_classes (dict)

Notes

- Returns empty dict if this is not a unit class tag.
- The dictionary has unit name as the key and HedSchemaEntry as value.

property value_classes

Return a dict of all the value classes this tag accepts.

Returns A dictionary of HedSchemaEntry value classes this tag accepts.

Return type dict

Notes

- Returns empty dict if this is not a value class.
- The dictionary has unit name as the key and HedSchemaEntry as value.

hed.models.model_constants module

class DefTagNames

Bases: `object`

Source names for definitions, def labels, and expanded labels

DEFINITION_KEY = 'definition'

DEFINITION_ORG_KEY = 'Definition'

DEF_EXPAND_KEY = 'def-expand'

DEF_EXPAND_ORG_KEY = 'Def-expand'

DEF_KEY = 'def'

DEF_ORG_KEY = 'Def'

OFFSET_KEY = 'offset'

OFFSET_ORG_KEY = 'Offset'

ONSET_KEY = 'onset'

ONSET_ORG_KEY = 'Onset'

hed.models.onset_mapper module

class OnsetMapper(*def_mapper*)

Bases: *hed.models.hed_ops.HedOps*

HedOps responsible for matching onset/offset pairs.

check_for_onset_offset(*hed_string_obj*)

Check for onset or offset and track context.

Parameters **hed_string_obj** (HedString) – The hed string to check. Finds a maximum of one onset tag.

Returns A list of issues found in validating onsets (i.e., out of order onsets, unknown def names).

Return type list

Notes

- Each issue in the return list is a dictionary.

hed.models.sidecar module

class Sidecar(*file*, *name=None*)

Bases: object

Contents of a JSON file or merged file.

Notes

- The Sidecar maintains its own definition dictionaries.

get_as_json_string()

Return this sidecar's column metadata as a string.

Returns The json string representing this sidecar.

Return type str

get_def_dicts(*extra_def_dicts=None*)

Return DefinitionDicts for the columns in this sidecar.

Parameters **extra_def_dicts** (*list, DefinitionDict, or None*) – Extra dicts to add to the list.

Returns A list of definition dicts for each column plus any found in extra_def_dicts.

Return type list

hed_string_iter(*hed_ops=None*, *error_handler=None*, *expand_defs=False*, *remove_definitions=False*, *allow_placeholders=True*, *extra_def_dicts=None*, ***kwargs*)

Iterator over hed strings in columns.

Parameters

- **hed_ops** (*func, HedOps, list*) – A HedOps, funcs or list of these to apply to the hed strings before returning
- **error_handler** (*ErrorHandler*) – The error handler to use for context, uses a default one if none.
- **expand_defs** (*bool*) – If True, expand all def tags located in the strings.
- **remove_definitions** (*bool*) – If True, remove all definitions found in the string.

- **allow_placeholders** (*bool*) – If False, placeholders will be marked as validation warnings.
- **extra_def_dicts** (DefinitionDict, *list*, *None*) – Extra dicts to add to the list.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Yields *tuple* – - HedString: A HedString at a given column and key position. - tuple: Indicates where hed_string was loaded from so it can be later set by the user - list: A list of issues found performing ops. Each issue is a dictionary.

static load_multiple_sidecars(*input_list*)

Utility for loading multiple json files.

Parameters **input_list** (*list*) – A list of filenames or Sidecar files in any mix.

Returns A list sidecars.

Return type list

Raises **HedFileError** – If any of the files are not found.

load_sidecar_file(*file*)

Load column metadata from a given json file.

Parameters **file** (*str or FileLike*) – If a string, this is a filename. Otherwise, it will be parsed as a file-like.

Raises **HedFileError** – If the file was not found or could not be parsed into JSON.

Notes

- Multiple files can be loaded into one Sidecar, but it is discouraged.

save_as_json(*save_filename*)

Save column metadata to a JSON file.

Parameters **save_filename** (*str*) – Path to save file

set_hed_string(*new_hed_string*, *position*)

Set a provided column/category key/etc.

Parameters

- **new_hed_string** (*str or* HedString) – The new hed_string to replace the value at position.
- **position** (*tuple*) – The (HedString, str, list) tuple returned from hed_string_iter.

validate_entries(*hed_ops=None*, *name=None*, *extra_def_dicts=None*, *error_handler=None*, ***kwargs*)

Run the given hed_ops on all columns in this sidecar.

Parameters

- **hed_ops** (*list, func, or* HedOps) – A HedOps, func or list of these to apply to hed strings in this sidecar.
- **name** (*str*) – If present, will use this as the filename for context, rather than using the actual filename Useful for temp filenames.
- **extra_def_dicts** – (DefinitionDict, list, or None): If present use these in addition to sidecar's def dicts.

- **error_handler** (*ErrorHandler or None*) – Used to report errors. Uses a default one if none passed in.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Returns The list of validation issues found. Individual issues are in the form of a dict.

Return type list

hed.models.spreadsheet_input module

class SpreadsheetInput(*file=None*, *file_type=None*, *worksheet_name=None*, *tag_columns=None*, *has_column_names=True*, *column_prefix_dictionary=None*, *def_dicts=None*, *name=None*)

Bases: *hed.models.base_input.BaseInput*

A spreadsheet of HED tags.

hed.models.tabular_input module

class TabularInput(*file=None*, *sidecar=None*, *attribute_columns=None*, *extra_def_dicts=None*, *also_gather_defs=True*, *name=None*)

Bases: *hed.models.base_input.BaseInput*

A BIDS tabular tsv file with sidecar.

HED_COLUMN_NAME = 'HED'

create_def_mapper(*column_mapper*, *extra_def_dicts=None*)

Create the definition mapper for this file.

Parameters

- **column_mapper** (ColumnMapper) – The column mapper to gather definitions from.
- **extra_def_dicts** (DefinitionDict *or [*DefinitionDict*]*) – Additional definitions to add to mapper.

Returns A class to validate or expand definitions with the given def dicts.

Return type def mapper (*DefMapper*)

Notes

- The extra_def_dicts are definitions not included in the column mapper.

reset_column_mapper(*sidecars=None*, *attribute_columns=None*)

Change the sidecars and settings.

Parameters

- **sidecars** (*str or [str] or* Sidecar *or [*Sidecar*]*) – A list of json filenames to pull sidecar info from.
- **attribute_columns** (*str or int or [str] or [int]*) – Column names or numbers to treat as attributes. Default: [“duration”, “onset”]

validate_file_sidecars(*hed_ops=None*, *error_handler=None*, ***kwargs*)

Validate column definitions and hed strings.

Parameters

- **hed_ops** (*list*) – A list of HedOps of funcs to apply to the hed strings in the sidecars.
- **error_handler** (*ErrorHandler or None*) – Used to report errors. Uses a default one if none passed in.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Returns A list of syntax and semantic issues found in the definitions. Each issue is a dictionary.

Return type list

Notes

- For full validation you should validate the sidecar separately.

hed.models.timeseries_input module

class TimeseriesInput(*file=None*, *sidecar=None*, *extra_def_dicts=None*, *name=None*)

Bases: *hed.models.base_input.BaseInput*

A BIDS time series tsv file.

HED_COLUMN_NAME = 'HED'

Module contents

Data structures for HED tag handling.

3.1.2 hed.models.BaseInput

class BaseInput(*file*, *file_type=None*, *worksheet_name=None*, *has_column_names=True*, *mapper=None*, *def_mapper=None*, *name=None*)

Superclass representing a basic columnar file.

__init__(*file*, *file_type=None*, *worksheet_name=None*, *has_column_names=True*, *mapper=None*, *def_mapper=None*, *name=None*)

Constructor for the BaseInput class.

Parameters

- **file** (*str or file-like*) – An xlsx/tsv file to open.
- **file_type** (*str or None*) – ".xlsx" (Excel), ".tsv" or ".txt" (tab-separated text). Derived from file if file is a filename.
- **worksheet_name** (*str or None*) – Name of Excel workbook worksheet name to use. (Not applicable to tsv files.)
- **has_column_names** (*bool*) – True if file has column names.
- **mapper** (*ColumnMapper or None*) – Indicates which columns have HED tags.
- **name** (*str or None*) – Optional field for how this file will report errors.

Notes

- See SpreadsheetInput or TabularInput for examples of how to use built-in a ColumnMapper.

Methods

__init__(file[, file_type, worksheet_name, ...])	Constructor for the BaseInput class.
convert_to_long(hed_schema[, error_handler])	Convert all tags to long form.
convert_to_short(hed_schema[, error_handler])	Convert all tags to short form.
extract_definitions([error_handler])	Gather and validate all definitions.
get_def_and_mapper_issues(error_handler[, ...])	Return definition and column issues.
get_worksheet([worksheet_name])	Get the requested worksheet.
iter_dataframe([hed_ops, mapper, ...])	Iterate rows based on the given column mapper.
iter_raw([hed_ops, error_handler])	Iterate all columns without substitutions
reset_mapper(new_mapper)	Set mapper to a different view of the file.
set_cell(row_number, column_number, ...[, ...])	Replace the specified cell with transformed text.
to_csv([file, output_processed_file])	Write to file or return as a string.
to_excel(file[, output_processed_file])	Output to an Excel file.
update_definition_mapper(def_dict)	Add definitions from dict(s) if mapper exists.
validate_file(hed_ops[, name, ...])	Run the hed_ops on columns and rows.

Attributes

COMMA_DELIMITER	
EXCEL_EXTENSION	
FILE_EXTENSION	
FILE_INPUT	
STRING_INPUT	
TAB_DELIMITER	
TEXT_EXTENSION	
dataframe	The underlying dataframe.
has_column_names	True if dataframe has column names.
loaded_workbook	The underlying loaded workbooks.
name	Name of the data.
worksheet_name	The worksheet name.

class BaseInput(*file*, *file_type=None*, *worksheet_name=None*, *has_column_names=True*, *mapper=None*, *def_mapper=None*, *name=None*)

Bases: `object`

Superclass representing a basic columnar file.

COMMA_DELIMITER = ','

EXCEL_EXTENSION = ['.xlsx']

FILE_EXTENSION = ['.tsv', '.txt', '.xlsx']

FILE_INPUT = 'file'

STRING_INPUT = 'string'

TAB_DELIMITER = '\t'

TEXT_EXTENSION = ['.tsv', '.txt']

convert_to_long(*hed_schema*, *error_handler=None*)

Convert all tags to long form.

Parameters

- **hed_schema** (HedSchema) – The schema to use to convert tags.
- **error_handler** (*ErrorHandler*) – The error handler to use for context, uses a default if none.

Returns A list of issue dictionaries corresponding to issues found during conversion.

Return type dict

convert_to_short(*hed_schema*, *error_handler=None*)

Convert all tags to short form.

Parameters

- **hed_schema** (HedSchema) – The schema to use to convert tags.
- **error_handler** (*ErrorHandler*) – The error handler to use for context, uses a default if none.

Returns A list of issue dictionaries corresponding to issues found during conversion.

Return type dict

property dataframe

The underlying dataframe.

extract_definitions(*error_handler=None*)

Gather and validate all definitions.

Parameters **error_handler** (*ErrorHandler*) – The error handler to use for context or a default if None.

Returns Contains all the definitions located in the file.

Return type *DefinitionDict*

get_def_and_mapper_issues(*error_handler*, *check_for_warnings=False*)

Return definition and column issues.

Parameters

- **error_handler** (*ErrorHandler*) – The error handler to use.
- **check_for_warnings** (*bool*) – If True check for and return warnings as well as errors.

Returns A list of definition and mapping issues. Each issue is a dictionary.

Return type dict

get_worksheet(*worksheet_name=None*)

Get the requested worksheet.

Parameters **worksheet_name** (*str or None*) – The name of the requested worksheet by name or the first one if None.

Returns The workbook request.

Return type openpyxl.workbook.Workbook

Notes

If None, returns the first worksheet.

property has_column_names

True if dataframe has column names.

iter_dataframe(*hed_ops=None*, *mapper=None*, *return_string_only=True*, *run_string_ops_on_columns=False*, *error_handler=None*, *expand_defs=False*, *remove_definitions=True*, ***kwargs*)

Iterate rows based on the given column mapper.

Parameters

- **hed_ops** (*list, func, HedOps, or None*) – A func, a HedOps or a list of these to apply to the hed strings before returning.
- **mapper** (*ColumnMapper or None*) – The column name to column number mapper (or internal mapper if None).
- **return_string_only** (*bool*) – If True, do not return issues list, individual columns, attribute columns, etc.
- **run_string_ops_on_columns** (*bool*) – If true, run all tag and string ops on columns, rather than columns then rows.
- **error_handler** (*ErrorHandler or None*) – The error handler to use for context or a default if None.
- **expand_defs** (*bool*) – If True, expand def tags into def-expand groups.
- **remove_definitions** (*bool*) – If true, remove all definition tags found.
- **()** (*kwargs*) – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Yields *dict* – A dict with parsed row, including keys: "HED", "column_to_hed_tags", and possibly "column_issues".

iter_raw(*hed_ops=None*, *error_handler=None*, ***kwargs*)

Iterate all columns without substitutions

Parameters

- **hed_ops** (*list, func, HedOps, or None*) – A func, a HedOps or a list of these to apply to the hed strings before returning.
- **error_handler** (*ErrorHandler or None*) – Handler to use for context or a default one if None.
- **kwargs** –

Yields - *dict* – A dict with column_number keys and values corresponding to the cell at that position.

Notes

- See models.hed_ops.translate_ops or the specific hed_ops for additional options.
- Primarily for altering or re-saving the original file (e.g., convert short tags to long).
- Used for initial processing when trying to find definitions.

property loaded_workbook

The underlying loaded workbooks.

property name

Name of the data.

reset_mapper(*new_mapper*)

Set mapper to a different view of the file.

Parameters **new_mapper** (ColumnMapper) – A column mapper to be associated with this base input.

set_cell(*row_number*, *column_number*, *new_string_obj*, *include_column_prefix_if_exist=False*, *tag_form='short_tag'*)

Replace the specified cell with transformed text.

Parameters

- **row_number** (*int*) – The row number of the spreadsheet to set.
- **column_number** (*int*) – The column number of the spreadsheet to set.
- **new_string_obj** (HedString) – Object with text to put in the given cell.
- **include_column_prefix_if_exist** (*bool*) – If True and the column matches one from mapper _column_prefix_dictionary, remove the prefix.
- **tag_form** (*str*) – Version of the tags (short_tag, long_tag, base_tag, etc)

Notes

Any attribute of a HedTag that returns a string is a valid value of tag_form.

to_csv(*file=None*, *output_processed_file=False*)

Write to file or return as a string.

Parameters

- **file** (*str, file-like, or None*) – Location to save this file. If None, return as string.
- **output_processed_file** (*bool*) – Replace all definitions and labels in HED columns as appropriate. Also fills in things like categories.

Returns None if file is given or the contents as a str if file is None.

Return type None or str

to_excel(*file*, *output_processed_file=False*)

Output to an Excel file.

Parameters

- **file** (*str or file-like*) – Location to save this base input.
- **output_processed_file** (*bool*) – If True, replace definitions and labels in HED columns. Also fills in things like categories.

Raises HedFileError if empty file object or file cannot be opened. –

update_definition_mapper(*def_dict*)

Add definitions from dict(s) if mapper exists.

Parameters def_dict (*list or* DefinitionDict) – Add the DefDict or list of DefDict to the internal definition mapper.

validate_file(*hed_ops*, *name=None*, *error_handler=None*, *check_for_warnings=True*, ***kwargs*)

Run the hed_ops on columns and rows.

Parameters

- **hed_ops** (*func,* HedOps*, or list of func and/or HedOps*) – The HedOps of funcs to apply.
- **name** (*str*) – If present, use this as the filename for context, rather than using the actual filename Useful for temp filenames.
- **error_handler** (*ErrorHandler or None*) – Used to report errors a default one if None.
- **check_for_warnings** (*bool*) – If True check for and return warnings as well as errors.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Returns The list of validation issues found. The list elements are dictionaries.

Return type list

property worksheet_name

The worksheet name.

3.1.3 hed.models.ColumnMapper

class ColumnMapper(*sidecars=None*, *tag_columns=None*, *column_prefix_dictionary=None*, *attribute_columns=None*, *optional_tag_columns=None*)

Mapping of a base input file columns into HED tags.

Notes

- Functions and variables column and row indexing starts at 0.

__init__(*sidecars=None*, *tag_columns=None*, *column_prefix_dictionary=None*, *attribute_columns=None*, *optional_tag_columns=None*)

Constructor for ColumnMapper.

Parameters

- **sidecars** (Sidecar*, string, or list of these*) –

A list of Sidecars or filenames to gather ColumnDefinitions from.

Sidecars later in the list override those earlier in the list.

- **tag_columns** – (list): A list of ints or strings containing the columns that contain the HED tags. Sidecar column definitions will take precedent if there is a conflict with tag_columns.
- **column_prefix_dictionary** (*dict*) – Dictionary with keys that are column numbers and values are HED tag prefixes to prepend to the tags in that column before processing.
- **attribute_columns** (*str, int or list*) – A column name, column number or a list of column names or numbers to treat as attributes.
- **optional_tag_columns** (*list*) – A list of ints or strings containing the columns that contain the HED tags. If the column is otherwise unspecified, convert this column type to HEDTags.

Notes

- All column numbers are 0 based.

Examples

column_prefix_dictionary = {3: 'Description/', 4: 'Label/'}

The third column contains tags that need Description/ tag prepended, while the fourth column contains tag that needs Label/ prepended.

Methods

__init__([sidecars, tag_columns, ...])	Constructor for ColumnMapper.
add_columns(column_names_or_numbers[, ...])	Add blank columns in the given column category.
add_sidecars(sidecars)	Add sidecar column info.
expand_row_tags(row_text)	Expand all mapped columns for row.
get_column_mapping_issues()	Get all the issues with finalizing column mapping.
get_def_dicts()	Return def dicts from every column description.
get_prefix_remove_func(column_number)	Return a function to removes name prefixes for column
set_column_map([new_column_map])	Set the column number to name mapping.
set_column_prefix_dict(column_prefix_dictionary)	Replace the column prefix dictionary
set_tag_columns([tag_columns, ...])	Set tag columns and optional tag columns
validate_column_data(hed_ops[, error_handler])	Validate the column data.

class ColumnMapper(*sidecars=None*, *tag_columns=None*, *column_prefix_dictionary=None*, *attribute_columns=None*, *optional_tag_columns=None*)

Bases: `object`

Mapping of a base input file columns into HED tags.

Notes

- Functions and variables column and row indexing starts at 0.

add_columns(*column_names_or_numbers*, *column_type=ColumnType.Attribute*)

Add blank columns in the given column category.

Parameters

- **column_names_or_numbers** (*list*) – A list of column names or numbers to add as the specified type.
- **column_type** (*ColumnType property*) – The category of column these should be.

add_sidecars(*sidecars*)

Add sidecar column info.

Parameters **sidecars** (*list*) – A list of filenames or loaded sidecar files in any mix.

expand_row_tags(*row_text*)

Expand all mapped columns for row.

Parameters **row_text** (*list*) – The text for the given row, one list entry per column number.

Returns A dictionary containing the keys COLUMN_TO_HED_TAGS, COLUMN_ISSUES, and arbitrary attributes.

Return type dict

Notes

- The "column_to_hed_tags" is each expanded column given separately as a list of HedStrings.
- Attributes are any column identified as an attribute. They will appear in the return value as {attribute_name: value_of_column}

get_column_mapping_issues()

Get all the issues with finalizing column mapping. Primarily a missing required column.

Returns A list dictionaries of all issues found from mapping column names to numbers.

Return type list

get_def_dicts()

Return def dicts from every column description.

Returns A list of DefinitionDict objects corresponding to each column entry.

Return type list

get_prefix_remove_func(*column_number*)

Return a function to removes name prefixes for column

Parameters **column_number** (*int*) – Column number to look up in the prefix dictionary.

Returns A function taking a tag and string, returning a string.

Return type func

set_column_map(*new_column_map=None*)

Set the column number to name mapping.

Parameters **new_column_map** (*list or dict*) – Either an ordered list of the column names or column_number:column name dictionary. In both cases column numbers start at 0

Returns List of issues. Each issue is a dictionary.

Return type list

set_column_prefix_dict(*column_prefix_dictionary*, *finalize_mapping=True*)

Replace the column prefix dictionary

Parameters

- **column_prefix_dictionary** (*dict*) – Dictionary with keys that are column numbers and values are HED tag prefixes to prepend to the tags in that column before processing.
- **finalize_mapping** (*bool*) – Re-generate the internal mapping if True, otherwise no effect until finalize.

Returns List of issues that occurred during this process. Each issue is a dictionary.

Return type list

set_tag_columns(*tag_columns=None*, *optional_tag_columns=None*, *finalize_mapping=True*)

Set tag columns and optional tag columns

Parameters

- **tag_columns** (*list*) – A list of ints or strings containing the columns that contain the HED tags. If None, clears existing tag_columns
- **optional_tag_columns** (*list*) – A list of ints or strings containing the columns that contain the HED tags, but not an error if missing. If None, clears existing tag_columns
- **finalize_mapping** (*bool*) – Re-generate the internal mapping if True, otherwise no effect until finalize.

Returns List of issues that occurred during this process. Each issue is a dictionary.

Return type list

validate_column_data(*hed_ops*, *error_handler=None*, ***kwargs*)

Validate the column data.

Parameters

- **hed_ops** (*list, func, or* HedOps) – A func, a HedOps or a list of these to apply to the hed strings in the sidecars.
- **error_handler** (*ErrorHandler or None*) – Used to report errors. Uses a default one if none passed in.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Returns A list of syntax and semantic issues found in the definitions. Each issue is a dictionary.

Return type list

3.1.4 hed.models.ColumnMetadata

class ColumnMetadata(*column_type=None*, *name=None*, *hed_dict=None*, *column_prefix=None*, *error_handler=None*)

Column in a ColumnMapper or top-level Sidecar dict.

__init__(*column_type=None*, *name=None*, *hed_dict=None*, *column_prefix=None*, *error_handler=None*)

A single column entry in the column mapper.

Parameters

- **column_type** (*ColumnType or None*) – How to treat this column when reading data.
- **name** (*str, int, or None*) – The column_name or column number identifying this column. If name is a string, you'll need to use a column map to set the number later.
- **hed_dict** (*dict or None*) – The loaded data (usually from json) for the given def At a minimum, this needs "HED" in the dict for several ColumnType
- **column_prefix** (*str or None*) – If present, prepend the given column_prefix to all hed tags in the columns. Only works on ColumnType HedTags.
- **error_handler** (*ErrorHandler or None*) – Used to report errors. Uses a default if None.

Notes

- Each column from which data is retrieved must have a ColumnMetadata representing its contents.
- The column_prefix dictionaries are used when the column is processed.

Methods

__init__([column_type, name, hed_dict, ...])	A single column entry in the column mapper.
expand(input_text)	Expand text using the rules for this column.
extract_definitions([error_handler])	Gather and validate definitions in metadata.
get_definition_issues()	Return the issues found extracting definitions.
hed_string_iter([hed_ops, error_handler])	Iterator yielding column hed strings.
remove_prefix(original_tag, current_tag_text)	Remove column_prefix if present from tag.
set_hed_string(new_hed_string[, position, ...])	Set a hed string for a category key/etc.
validate_column(hed_ops, error_handler, **kwargs)	Run the given hed_ops on this column.

Attributes

def_dict	Return the definition dictionary for this column.
hed_dict	The loaded dict for any given entry.

class ColumnMetadata(*column_type=None*, *name=None*, *hed_dict=None*, *column_prefix=None*, *error_handler=None*)

Bases: `object`

Column in a ColumnMapper or top-level Sidecar dict.

property def_dict

Return the definition dictionary for this column.

Returns Contains all the definitions located in the column.

Return type *DefinitionDict*

expand(*input_text*)

Expand text using the rules for this column.

Parameters **input_text** (*str*) – Text to expand (generally from a single cell in a spreadsheet).

Returns

The expanded column as a hed_string. str or dict: If this is a string, contains the name of this column

as an attribute. If the first return value is None, this is an error message dictionary.

Return type str or None

Notes

- Examples are adding name_prefix, inserting a column hed_string from a category key, etc.

extract_definitions(*error_handler=None*)

Gather and validate definitions in metadata.

Parameters **error_handler** (*ErrorHandler*) – The error handler to use for context, uses a default one if None.

Returns Contains all the definitions located in the column. issues: List of issues encountered in extracting the definitions. Each issue is a dictionary.

Return type *DefinitionDict*

get_definition_issues()

Return the issues found extracting definitions.

Returns A list of issues found when parsing definitions. Individual issues are dictionaries.

Return type list

property hed_dict

The loaded dict for any given entry.

Returns A dict which generally contains a "HED" entry and optional others like description.

Return type dict

hed_string_iter(*hed_ops=None*, *error_handler=None*, ***kwargs*)

Iterator yielding column hed strings.

Parameters

- **hed_ops** (*func, HedOps, or list of these*) – The HedOps or funcs to apply to the hed strings before returning.
- **error_handler** (*ErrorHandler*) – The error handler to use for context, uses a default one if none.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options

Yields *tuple* – - HedString: The hed string at a given column and key position. - str: Indication of the where hed string was loaded from so it can be later set by the user. - list: Issues found applying hed_ops. Each issue is a dictionary.

remove_prefix(*original_tag*, *current_tag_text*)

Remove column_prefix if present from tag.

Parameters

- **original_tag** (HedTag) – The original hed tag being written.
- **current_tag_text** (*str*) – A single tag as a string, in any form.

Returns current_tag_text with required prefixes removed

Return type str

set_hed_string(*new_hed_string*, *position=None*, *set_def_removed=False*)

Set a hed string for a category key/etc.

Parameters

- **new_hed_string** (*str or* HedString) – The new hed_string to replace the value at position.
- **position** (*str, optional*) – This should only be a value returned from hed_string_iter.
- **set_def_removed** (*bool*) – If True, set the version with definitions removed, rather than the normal version.

Raises **TypeError** – If the mapping cannot occur.

validate_column(*hed_ops*, *error_handler*, ***kwargs*)

Run the given hed_ops on this column.

Parameters

- **hed_ops** (*list or func or* HedOps) – to the hed strings in the columns.
- **error_handler** (*ErrorHandler or None*) – Used to report errors. Uses a default one if none passed in.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Returns Issues found by the given hed_ops. Each issue is a dictionary.

Return type list

3.1.5 hed.models.DefinitionDict

class DefinitionDict

Gathers definitions from a single source.

This class extends HedOps because it has string_funcs to check for definitions. It has no tag_funcs.

__init__()

Definitions to be considered a single source.

Methods

__init__()	Definitions to be considered a single source.
check_for_definitions(hed_string_obj[, ...])	Check string for definition tags, adding them to self.
get_definition_issues()	Return definition errors found during extraction.

Attributes

defs	Provides direct access to internal dictionary.

class DefinitionDict

Bases: *hed.models.hed_ops.HedOps*

Gathers definitions from a single source.

This class extends HedOps because it has string_funcs to check for definitions. It has no tag_funcs.

check_for_definitions(*hed_string_obj*, *error_handler=None*)

Check string for definition tags, adding them to self.

Parameters

- **hed_string_obj** (HedString) – A single hed string to gather definitions from.
- **error_handler** (*ErrorHandler or None*) – Error context used to identify where definitions are found.

Returns List of issues encountered in checking for definitions. Each issue is a dictionary.

Return type list

property defs

Provides direct access to internal dictionary. Alter at your own risk.

Returns {str: DefinitionEntry}

Return type dict

get_definition_issues()

Return definition errors found during extraction.

Returns List of DefinitionErrors issues found. Each issue is a dictionary.

Return type list

3.1.6 hed.models.DefinitionEntry

class DefinitionEntry(*name*, *contents*, *takes_value*, *source_context*)

A single definition.

__init__(*name*, *contents*, *takes_value*, *source_context*)

Initialize info for a single definition.

Parameters

- **name** (*str*) – The label portion of this name (not including Definition/).
- **contents** (HedGroup) – The contents of this definition.

- **takes_value** (*bool*) – If True, expects ONE tag to have a single # sign in it.
- **source_context** (*dict*) – Info about where this definition was declared.

Methods

__init__(name, contents, takes_value, ...)	Initialize info for a single definition.
get_definition(replace_tag[, placeholder_value])	Return a copy of the definition with the tag expanded and the placeholder plugged in.

class DefinitionEntry(*name*, *contents*, *takes_value*, *source_context*)

Bases: object

A single definition.

get_definition(*replace_tag*, *placeholder_value=None*)

Return a copy of the definition with the tag expanded and the placeholder plugged in.

Parameters

- **replace_tag** (HedTag) – The def hed tag to replace with an expanded version
- **placeholder_value** (*str or None*) – If present and required, will replace any pound signs in the definition contents.

Returns The expanded def tag name HedGroup: The contents of this definition(including the def tag itself)

Return type str

Raises **ValueError** – If a placeholder_value is passed, but this definition doesn't have a placeholder.

3.1.7 hed.models.DefMapper

class DefMapper(*def_dicts=None*)

Handles converting Def/ and Def-expand/.

Notes

- The class provides string funcs but no tag funcs when extending HedOps.
- The class can expand or shrink definitions in hed strings via Def/XXX and (Def-expand/XXX ...).

__init__(*def_dicts=None*)

Initialize mapper for definitions in hed strings.

Parameters **def_dicts** (*list or DefDict*) – DefDicts containing the definitions this mapper should initialize with.

Notes

- More definitions can be added later.

Methods

__init__([def_dicts])	Initialize mapper for definitions in hed strings.
add_definitions(def_dicts[, add_as_temp])	Add definitions from dict(s) to mapper
add_definitions_from_string_as_temp(...)	Add definitions from hed string as temporary.
clear_temporary_definitions()	Remove any previously added temporary definitions.
expand_and_remove_definitions(hed_string_obj)	Validate and expand Def/Def-Expand tags.
expand_def_tags(hed_string_obj[, ...])	Validate and expand Def/Def-Expand tags.
get_def_entry(def_name)	Get the definition entry for the definition name.

Attributes

issues

class DefMapper(*def_dicts=None*)

Bases: *hed.models.hed_ops.HedOps*

Handles converting Def/ and Def-expand/.

Notes

- The class provides string funcs but no tag funcs when extending HedOps.
- The class can expand or shrink definitions in hed strings via Def/XXX and (Def-expand/XXX ...).

add_definitions(*def_dicts*, *add_as_temp=False*)

Add definitions from dict(s) to mapper

Parameters

- **def_dicts** (*list or* DefinitionDict) – DefDict or list of DefDicts whose definitions should be added.
- **add_as_temp** (*bool*) – If true, mark these new definitions as temporary (easily purged).

add_definitions_from_string_as_temp(*hed_string_obj*)

Add definitions from hed string as temporary.

Parameters **hed_string_obj** (HedString) – Hed string object to search for definitions

Returns List of issues due to invalid definitions found in this string. Each issue is a dictionary.

Return type list

clear_temporary_definitions()

Remove any previously added temporary definitions.

expand_and_remove_definitions(*hed_string_obj*, *check_for_definitions=False*, *expand_defs=True*, *shrink_defs=False*, *remove_definitions=True*)

Validate and expand Def/Def-Expand tags.

Also removes definitions

Parameters

- **hed_string_obj** (HedString) – The string to search for definitions.
- **check_for_definitions** (*bool*) – If True, this will first check the hed string for any definitions.
- **expand_defs** (*bool*) – If True, replace Def tags to Def-expand tag groups.
- **shrink_defs** (*bool*) – If True, replace Def-expand groups with Def tags.
- **remove_definitions** (*bool*) – If true, this will remove all Definition tag groups.

Returns A list of issues for definition-related tags in this string. Each issue is a dictionary.

Return type def_issues (list)

Notes

- The check_for_definitions is mainly used for individual HedStrings in isolation.
- The defs can be expanded or shrunk, while definitions can be removed.
- This does not validate definitions, it will blindly remove invalid definitions as well.

expand_def_tags(*hed_string_obj*, *expand_defs=True*, *shrink_defs=False*)

Validate and expand Def/Def-Expand tags.

Parameters

- **hed_string_obj** (HedString) – The hed string to process.
- **expand_defs** (*bool*) – If true, convert def tags to def-expand tag groups that include definition content.
- **shrink_defs** (*bool*) – If True, replace all def-expand groups with corresponding def tags.

Returns Issues found related to validating defs. Each issue is a dictionary.

Return type list

Notes

- This function can optionally expand or shrink Def/ and Def-expand, respectively.
- Usually issues are mismatched placeholders or a missing definition.
- The expand_defs and shrink_defs cannot both be True.

get_def_entry(*def_name*)

Get the definition entry for the definition name.

Parameters **def_name** (*str*) – Name of the definition to retrieve.

Returns Definition entry for the requested definition.

Return type *DefinitionEntry*

property issues

3.1.8 hed.models.HedGroup

class HedGroup(*hed_string=''*, *startpos=None*, *endpos=None*, *contents=None*)

A single parenthesized hed string.

__init__(*hed_string=''*, *startpos=None*, *endpos=None*, *contents=None*)

Return an empty HedGroup object.

Parameters

- **hed_string** (*str or None*) – Source hed string for this group.
- **startpos** (*int or None*) – Starting index of group(including parentheses) in hed_string.
- **endpos** (*int or None*) – Position after the end (including parentheses) in hed_string.
- **contents** (*list or None*) – A list of HedTags and/or HedGroups that will be set as the contents of this group.

Notes

- contents parameter is mainly used for processing definitions.

Methods

__init__([hed_string, startpos, endpos, ...])	Return an empty HedGroup object.
append(tag_or_group)	Add a tag or group to this group.
check_if_in_original(tag_or_group)	Check if the tag or group in original string.
copy()	Return a deep copy of this group.
find_def_tags([recursive, include_groups])	Find def and def-expand tags
find_exact_tags(tags_or_groups[, recursive])	Find the given tags or groups.
find_placeholder_tag()	Return a placeholder tag, if present in this group.
find_tags(search_tags[, recursive, ...])	Find the tags and their containing groups.
find_tags_with_term(term[, recursive, ...])	Find any tags that contain the given term.
get_all_groups([also_return_depth])	Return HedGroups, including descendants and self.
get_all_tags()	Return HedTags, including descendants.
get_as_form(tag_attribute[, tag_transformer])	Get the string corresponding to the specified form.
get_as_long()	Return this HedGroup as a long tag string.
get_as_short()	Return this HedGroup as a short tag string.
get_frozen()	Return a frozen (non-mutable) copy of this HedGroup.
get_original_hed_string()	Get the original hed string.
groups()	Return the direct child groups of this group.
lower()	Convenience function, equivalent to str(self).lower()
remove(items_to_remove)	Remove any tags/groups in items_to_remove.
replace(item_to_replace, new_contents)	Replace an existing tag or group.
tags()	Return the direct child tags of this group.

Attributes

children	A list of the direct children.
is_group	True if this is a parenthesized group.
span	Return the source span.

class HedGroup(*hed_string=''*, *startpos=None*, *endpos=None*, *contents=None*)

Bases: *hed.models.hed_group_base.HedGroupBase*

A single parenthesized hed string.

append(*tag_or_group*)

Add a tag or group to this group.

Parameters **tag_or_group** (*HedTag or HedGroup*) – The new object to add to this group.

Raises **ValueError** – If a HedGroupFrozen.

check_if_in_original(*tag_or_group*)

Check if the tag or group in original string.

Parameters **tag_or_group** (*HedTag or HedGroup*) – The HedTag or HedGroup to be looked for in this group.

Returns True if in this group.

Return type bool

property children

A list of the direct children.

copy()

Return a deep copy of this group.

Returns The copied group.

Return type *HedGroupBase*

Notes

- The parent tag is removed.

find_def_tags(*recursive=False*, *include_groups=3*)

Find def and def-expand tags

Parameters

- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*int, 0, 1, 2, 3*) – options for how to expand or include groups

Returns A list of tuples. The contents depends on the values of the include group.

Return type list

Notes

- **The include_groups option controls the tag expansion as follows:**
  - If 0: Return only def and def expand tags/.
  - If 1: Return only def tags and def-expand groups.
  - If 2: Return only groups containing defs, or def-expand groups.
  - If 3 or any other value: Return all 3 as a tuple.

`find_exact_tags`(*tags_or_groups*, *recursive=False*)

Find the given tags or groups.

Parameters

- **tags_or_groups** (HedTag, HedGroupBase) – A container of tags to locate.
- **recursive** (*bool*) – If true, also check subgroups.

Returns A list of HedGroupBases the given tags/groups were found in.

Return type list

Notes

- If you pass in groups it will only find EXACT matches.
- This can only find identified tags.
- By default, definition, def, def-expand, onset, and offset are identified, even without a schema.
- If this is a HedGroup, order matters. (b, a) != (a, b)
- **If this is a HedGroupFrozen:**

  if "(a, b)" in tags_or_groups, then it will match 1 and 2, but not 3.

  1. (a, b)
  2. (b, a)
  3. (a, b, c)

`find_placeholder_tag`()

Return a placeholder tag, if present in this group.

Returns The placeholder tag if found.

Return type *HedTag* or None

Notes

- Assumes a valid HedString with no erroneous "#" characters.

find_tags(*search_tags*, *recursive=False*, *include_groups=2*)

Find the tags and their containing groups.

Parameters

- **search_tags** (*container*) – A container of short_base_tags to locate
- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*0, 1 or 2*) – Specify return values.

Returns The contents of the list depends on the value of include_groups.

Return type list

Notes

- If include_groups is 0, return a list of the HedTags.
- If include_groups is 1, return a list of the HedGroups containing the HedTags.
- If include_groups is 2, return a list of tuples (HedTag, HedGroup) for the found tags.
- This can only find identified tags.
- By default, definition, def, def-expand, onset, and offset are identified, even without a schema.

find_tags_with_term(*term*, *recursive=False*, *include_groups=2*)

Find any tags that contain the given term.

Note: This can only find identified tags.

Parameters

- **term** (*str*) – A single term to search for.
- **recursive** (*bool*) – If true, recursively check subgroups.
- **include_groups** (*int, 0, 1, 2, 3*) – 0, 1 or 2 If 0: Return only tags If 1: return only groups If 2 or any other value: return both
- **recursive** – If true, also check subgroups.
- **include_groups** – options for how to expand or include groups

Return type list

def_tag: HedTag The located def tag

def_expand_group: HedGroup or None If this is a def-expand rather than def tag, this will be the entire def-expand group.

group: HedGroup The group the def tag or def expand group is in.

Notes

- **The include_groups option controls the tag expansion as follows:**
  - If 0: Return only def and def expand tags/.
  - If 1: Return only def tags and def-expand groups.
  - If 2: Return only groups containing defs, or def-expand groups.
  - If 3 or any other value: Return all 3 as a tuple.

`get_all_groups`(*also_return_depth=False*)

Return HedGroups, including descendants and self.

Parameters `also_return_depth` (*bool*) – If True, yield tuples (group, depth) rather than just groups.

Returns The list of all HedGroups in this group, including descendants and self.

Return type list

`get_all_tags`()

Return HedTags, including descendants.

Returns A list of all the tags in this group including descendants.

Return type list

`get_as_form`(*tag_attribute*, *tag_transformer=None*)

Get the string corresponding to the specified form.

Parameters

- `tag_attribute` (*str*) – The hed_tag property to use to construct the string (usually short_tag or long_tag).
- `tag_transformer` (*func or None*) – A function that is applied to each tag string before returning.

Returns The constructed string after transformation

Return type str

Notes

- The signature of a tag_transformer is str def(HedTag, str).

`get_as_long`()

Return this HedGroup as a long tag string.

Returns The group as a string with all tags as long tags.

Return type str

`get_as_short`()

Return this HedGroup as a short tag string.

Returns The group as a string with all tags as short tags.

Return type str

get_frozen()

Return a frozen (non-mutable) copy of this HedGroup.

This is a deep copy if the group was not already frozen.

Returns A frozen copy of this HedGroup.

Return type *HedGroupFrozen*

get_original_hed_string()

Get the original hed string.

Returns The original string with no modification.

Return type str

groups()

Return the direct child groups of this group.

Returns All groups directly in this group, filtering out HedTag children.

Return type list

property is_group

True if this is a parenthesized group.

lower()

Convenience function, equivalent to str(self).lower()

remove(*items_to_remove*)

Remove any tags/groups in items_to_remove.

Parameters **items_to_remove** (*list*) – List of HedGroups and/or HedTags to remove.

Notes

- Any groups that become empty will also be pruned.
- Identity, not equivalence is used in determining whether to remove.

replace(*item_to_replace*, *new_contents*)

Replace an existing tag or group.

Parameters

- **item_to_replace** (HedTag *or* HedGroup) – The item to replace must exist or this will raise an error.
- **new_contents** (HedTag *or* HedGroup) – Replacement contents.

property span

Return the source span.

Returns start index of the group (including parentheses) from the source string. int: end index of the group (including parentheses) from the source string.

Return type int

tags()

Return the direct child tags of this group.

Returns All tags directly in this group, filtering out HedGroup children.

Return type list

3.1.9 hed.models.HedGroupBase

class HedGroupBase(*hed_string=''*, *startpos=None*, *endpos=None*)

Base class for the HedGroup API.

Notes

- This interface is shared by HedGroup and HedGroupFrozen.

__init__(*hed_string=''*, *startpos=None*, *endpos=None*)

Return an empty HedGroupBase object.

Parameters

- **hed_string** (*str*) – Source hed string for this group.
- **startpos** (*int*) – Starting index of group (including parentheses) in hed_string.
- **endpos** (*int*) – Ending index of group (including parentheses) in hed_string.

Methods

__init__([hed_string, startpos, endpos])	Return an empty HedGroupBase object.
copy()	Return a deep copy of this group.
find_def_tags([recursive, include_groups])	Find def and def-expand tags
find_exact_tags(tags_or_groups[, recursive])	Find the given tags or groups.
find_placeholder_tag()	Return a placeholder tag, if present in this group.
find_tags(search_tags[, recursive, ...])	Find the tags and their containing groups.
find_tags_with_term(term[, recursive, ...])	Find any tags that contain the given term.
get_all_groups([also_return_depth])	Return HedGroups, including descendants and self.
get_all_tags()	Return HedTags, including descendants.
get_as_form(tag_attribute[, tag_transformer])	Get the string corresponding to the specified form.
get_as_long()	Return this HedGroup as a long tag string.
get_as_short()	Return this HedGroup as a short tag string.
get_original_hed_string()	Get the original hed string.
groups()	Return the direct child groups of this group.
lower()	Convenience function, equivalent to str(self).lower()
tags()	Return the direct child tags of this group.

Attributes

children	A list of the direct children.
is_group	True if this is a parenthesized group.
span	Return the source span.

class HedGroupBase(*hed_string=''*, *startpos=None*, *endpos=None*)

Bases: `object`

Base class for the HedGroup API.

Notes

- This interface is shared by HedGroup and HedGroupFrozen.

property children

A list of the direct children.

copy()

Return a deep copy of this group.

Returns The copied group.

Return type *HedGroupBase*

Notes

- The parent tag is removed.

find_def_tags(*recursive=False*, *include_groups=3*)

Find def and def-expand tags

Parameters

- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*int, 0, 1, 2, 3*) – options for how to expand or include groups

Returns A list of tuples. The contents depends on the values of the include group.

Return type list

Notes

- **The include_groups option controls the tag expansion as follows:**
  - If 0: Return only def and def expand tags/.
  - If 1: Return only def tags and def-expand groups.
  - If 2: Return only groups containing defs, or def-expand groups.
  - If 3 or any other value: Return all 3 as a tuple.

find_exact_tags(*tags_or_groups*, *recursive=False*)

Find the given tags or groups.

Parameters

- **tags_or_groups** (HedTag, HedGroupBase) – A container of tags to locate.
- **recursive** (*bool*) – If true, also check subgroups.

Returns A list of HedGroupBases the given tags/groups were found in.

Return type list

Notes

- If you pass in groups it will only find EXACT matches.
- This can only find identified tags.
- By default, definition, def, def-expand, onset, and offset are identified, even without a schema.
- If this is a HedGroup, order matters. (b, a) != (a, b)
- **If this is a HedGroupFrozen:**

  if "(a, b)" in tags_or_groups, then it will match 1 and 2, but not 3.

  1. (a, b)
  2. (b, a)
  3. (a, b, c)

find_placeholder_tag()

Return a placeholder tag, if present in this group.

Returns The placeholder tag if found.

Return type *HedTag* or None

Notes

- Assumes a valid HedString with no erroneous "#" characters.

find_tags(*search_tags*, *recursive=False*, *include_groups=2*)

Find the tags and their containing groups.

Parameters

- **search_tags** (*container*) – A container of short_base_tags to locate
- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*0, 1 or 2*) – Specify return values.

Returns The contents of the list depends on the value of include_groups.

Return type list

Notes

- If include_groups is 0, return a list of the HedTags.
- If include_groups is 1, return a list of the HedGroups containing the HedTags.
- If include_groups is 2, return a list of tuples (HedTag, HedGroup) for the found tags.
- This can only find identified tags.
- By default, definition, def, def-expand, onset, and offset are identified, even without a schema.

find_tags_with_term(*term*, *recursive=False*, *include_groups=2*)

Find any tags that contain the given term.

Note: This can only find identified tags.

Parameters

- **term** (*str*) – A single term to search for.
- **recursive** (*bool*) – If true, recursively check subgroups.
- **include_groups** (*int, 0, 1, 2, 3*) – 0, 1 or 2 If 0: Return only tags If 1: return only groups If 2 or any other value: return both
- **recursive** – If true, also check subgroups.
- **include_groups** – options for how to expand or include groups

Return type list

def_tag: HedTag The located def tag

def_expand_group: HedGroup or None If this is a def-expand rather than def tag, this will be the entire def-expand group.

group: HedGroup The group the def tag or def expand group is in.

Notes

- **The include_groups option controls the tag expansion as follows:**
  - If 0: Return only def and def expand tags/.
  - If 1: Return only def tags and def-expand groups.
  - If 2: Return only groups containing defs, or def-expand groups.
  - If 3 or any other value: Return all 3 as a tuple.

get_all_groups(*also_return_depth=False*)

Return HedGroups, including descendants and self.

Parameters **also_return_depth** (*bool*) – If True, yield tuples (group, depth) rather than just groups.

Returns The list of all HedGroups in this group, including descendants and self.

Return type list

get_all_tags()

Return HedTags, including descendants.

Returns A list of all the tags in this group including descendants.

Return type list

get_as_form(*tag_attribute*, *tag_transformer=None*)

Get the string corresponding to the specified form.

Parameters

- **tag_attribute** (*str*) – The hed_tag property to use to construct the string (usually short_tag or long_tag).
- **tag_transformer** (*func or None*) – A function that is applied to each tag string before returning.

Returns The constructed string after transformation

Return type str

Notes

- The signature of a tag_transformer is str def(HedTag, str).

get_as_long()

Return this HedGroup as a long tag string.

Returns The group as a string with all tags as long tags.

Return type str

get_as_short()

Return this HedGroup as a short tag string.

Returns The group as a string with all tags as short tags.

Return type str

get_original_hed_string()

Get the original hed string.

Returns The original string with no modification.

Return type str

groups()

Return the direct child groups of this group.

Returns All groups directly in this group, filtering out HedTag children.

Return type list

property is_group

True if this is a parenthesized group.

lower()

Convenience function, equivalent to str(self).lower()

property span

Return the source span.

Returns start index of the group (including parentheses) from the source string. int: end index of the group (including parentheses) from the source string.

Return type int

tags()

Return the direct child tags of this group.

Returns All tags directly in this group, filtering out HedGroup children.

Return type list

3.1.10 hed.models.HedGroupFrozen

class HedGroupFrozen(*contents*, *hed_string=None*)

A frozen version of the HedGroup.

Notes

- Searching and getting all tags/groups will be faster.
- Additionally, HedGroupFrozen is order-agnostic: (a, b) = (b, a).

__init__(*contents*, *hed_string=None*)

Initialize a frozen group.

Parameters

- **contents** (*HedGroupBase or list*) – Used to set the contents of this group.
- **hed_string** (*str or None*) – If contents is a list, this is the raw string the contents came from.

Notes

- This makes a complete copy of the HedString.
- If HedGroupBase, get the source hed_string from the group.
- If a list, may be a mixture of HedTags and HedGroups. Use the hed_string parameter as the source.

Methods

__init__(contents[, hed_string])	Initialize a frozen group.
copy()	Return a deep copy of this group.
find_def_tags([recursive, include_groups])	Find def and def-expand tags
find_exact_tags(tags_or_groups[, recursive])	Find the given tags or groups.
find_placeholder_tag()	Return a placeholder tag, if present in this group.
find_tags(search_tags[, recursive, ...])	Find the tags and their containing groups.
find_tags_with_term(term[, recursive, ...])	Find any tags that contain the given term.
get_all_groups([also_return_depth])	Return HedGroups, including descendants and self.
get_all_tags()	Return HedTags, including descendants.
get_as_form(tag_attribute[, tag_transformer])	Get the string corresponding to the specified form.
get_as_long()	Return this HedGroup as a long tag string.
get_as_short()	Return this HedGroup as a short tag string.
get_frozen()	Return this frozen group.
get_original_hed_string()	Get the original hed string.
groups()	Return the direct child groups of this group.
lower()	Convenience function, equivalent to str(self).lower()
tags()	Return the direct child tags of this group.

Attributes

children	A list of the direct children.
is_group	True if this is a parenthesized group.
span	Return the source span.

class HedGroupFrozen(*contents*, *hed_string=None*)

Bases: *hed.models.hed_group_base.HedGroupBase*

A frozen version of the HedGroup.

Notes

- Searching and getting all tags/groups will be faster.
- Additionally, HedGroupFrozen is order-agnostic: (a, b) = (b, a).

property children

A list of the direct children.

copy()

Return a deep copy of this group.

Returns The copied group.

Return type *HedGroupBase*

Notes

- The parent tag is removed.

find_def_tags(*recursive=False*, *include_groups=3*)

Find def and def-expand tags

Parameters

- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*int, 0, 1, 2, 3*) – options for how to expand or include groups

Returns A list of tuples. The contents depends on the values of the include group.

Return type list

Notes

- **The include_groups option controls the tag expansion as follows:**
  - If 0: Return only def and def expand tags/.
  - If 1: Return only def tags and def-expand groups.
  - If 2: Return only groups containing defs, or def-expand groups.
  - If 3 or any other value: Return all 3 as a tuple.

find_exact_tags(*tags_or_groups*, *recursive=False*)

Find the given tags or groups.

Parameters

- **tags_or_groups** (HedTag, HedGroupBase) – A container of tags to locate.
- **recursive** (*bool*) – If true, also check subgroups.

Returns A list of HedGroupBases the given tags/groups were found in.

Return type list

Notes

- If you pass in groups it will only find EXACT matches.
- This can only find identified tags.
- By default, definition, def, def-expand, onset, and offset are identified, even without a schema.
- If this is a HedGroup, order matters. (b, a) != (a, b)
- **If this is a HedGroupFrozen:**

  if "(a, b)" in tags_or_groups, then it will match 1 and 2, but not 3.

  1. (a, b)
  2. (b, a)
  3. (a, b, c)

find_placeholder_tag()

Return a placeholder tag, if present in this group.

Returns The placeholder tag if found.

Return type *HedTag* or None

Notes

- Assumes a valid HedString with no erroneous "#" characters.

find_tags(*search_tags*, *recursive=False*, *include_groups=2*)

Find the tags and their containing groups.

Parameters

- **search_tags** (*container*) – A container of short_base_tags to locate
- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*0, 1 or 2*) – Specify return values.

Returns The contents of the list depends on the value of include_groups.

Return type list

Notes

- If include_groups is 0, return a list of the HedTags.
- If include_groups is 1, return a list of the HedGroups containing the HedTags.
- If include_groups is 2, return a list of tuples (HedTag, HedGroup) for the found tags.
- This can only find identified tags.
- By default, definition, def, def-expand, onset, and offset are identified, even without a schema.

find_tags_with_term(*term*, *recursive=False*, *include_groups=2*)

Find any tags that contain the given term.

Note: This can only find identified tags.

Parameters

- **term** (*str*) – A single term to search for.
- **recursive** (*bool*) – If true, recursively check subgroups.
- **include_groups** (*int, 0, 1, 2, 3*) – 0, 1 or 2 If 0: Return only tags If 1: return only groups If 2 or any other value: return both
- **recursive** – If true, also check subgroups.
- **include_groups** – options for how to expand or include groups

Return type list

def_tag: HedTag The located def tag

def_expand_group: HedGroup or None If this is a def-expand rather than def tag, this will be the entire def-expand group.

group: HedGroup The group the def tag or def expand group is in.

Notes

- **The include_groups option controls the tag expansion as follows:**
  - If 0: Return only def and def expand tags/.
  - If 1: Return only def tags and def-expand groups.
  - If 2: Return only groups containing defs, or def-expand groups.
  - If 3 or any other value: Return all 3 as a tuple.

`get_all_groups`(*also_return_depth=False*)

Return HedGroups, including descendants and self.

Parameters `also_return_depth` (*bool*) – If True, this yields tuples (group, depth) rather than just groups.

Returns The list of all HedGroups in this group, including descendants and self.

Return type list

`get_all_tags`()

Return HedTags, including descendants.

Returns A list of all the HedTags in this group including descendants.

Return type list

Notes

- This list is cached when initially called since its contents never change.

`get_as_form`(*tag_attribute*, *tag_transformer=None*)

Get the string corresponding to the specified form.

Parameters

- `tag_attribute` (*str*) – The hed_tag property to use to construct the string (usually short_tag or long_tag).
- `tag_transformer` (*func or None*) – A function that is applied to each tag string before returning.

Returns The constructed string after transformation

Return type str

Notes

- The signature of a tag_transformer is str def(HedTag, str).

get_as_long()

Return this HedGroup as a long tag string.

Returns The group as a string with all tags as long tags.

Return type str

get_as_short()

Return this HedGroup as a short tag string.

Returns The group as a string with all tags as short tags.

Return type str

get_frozen()

Return this frozen group.

Returns This same group.

Return type *HedGroupFrozen*

get_original_hed_string()

Get the original hed string.

Returns The original string with no modification.

Return type str

groups()

Return the direct child groups of this group.

Returns All groups directly in this group, filtering out HedTag children.

Return type list

property is_group

True if this is a parenthesized group.

lower()

Convenience function, equivalent to str(self).lower()

property span

Return the source span.

Returns start index of the group (including parentheses) from the source string. int: end index of the group (including parentheses) from the source string.

Return type int

tags()

Return the direct child tags of this group.

Returns All tags directly in this group, filtering out HedGroup children.

Return type list

3.1.11 hed.models.HedOps

class HedOps(**args*, ***kwargs*)

Base class to support HedOps.

Notes

- HED ops are operations that apply to HedStrings in a sequence.

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)

class HedOps(**args*, ***kwargs*)

Bases: `object`

Base class to support HedOps.

Notes

- HED ops are operations that apply to HedStrings in a sequence.

3.1.12 hed.models.HedString

class HedString(*hed_string*, *hed_schema=None*, *_contents=None*)

A HED string.

__init__(*hed_string*, *hed_schema=None*, *_contents=None*)

Constructor for the HedString class.

Parameters

- **hed_string** (*str*) – A HED string consisting of tags and tag groups.
- **hed_schema** (*HedSchema or None*) – The schema to use to identify tags. Can be passed later.
- **_contents** (*[HedGroupBase and/or HedTag] or None*) – Create a HedString from this exact list of children. Does not make a copy.

Notes

- The HedString object parses its component tags and groups into a tree-like structure.

Methods

__init__(hed_string[, hed_schema, _contents])	Constructor for the HedString class.
append(tag_or_group)	Add a tag or group to this group.
apply_funcs(string_funcs)	Run functions on this string.
check_if_in_original(tag_or_group)	Check if the tag or group in original string.
convert_to_canonical_forms(hed_schema)	Identify all tags using the given schema.
convert_to_long(hed_schema)	Compute canonical forms and return the long form.
convert_to_original()	Return the original form of this string.
convert_to_short(hed_schema)	Compute canonical forms and return the short form.
copy()	Return a deep copy of this group.
find_def_tags([recursive, include_groups])	Find def and def-expand tags
find_exact_tags(tags_or_groups[, recursive])	Find the given tags or groups.
find_placeholder_tag()	Return a placeholder tag, if present in this group.
find_tags(search_tags[, recursive, ...])	Find the tags and their containing groups.
find_tags_with_term(term[, recursive, ...])	Find any tags that contain the given term.
find_top_level_tags(anchor_tags[, ...])	Find top level groups with an anchor tag.
from_hed_strings(contents)	Factory for creating HedStrings via combination.
get_all_groups([also_return_depth])	Return HedGroups, including descendants and self.
get_all_tags()	Return HedTags, including descendants.
get_as_form(tag_attribute[, tag_transformer])	Get the string corresponding to the specified form.
get_as_long()	Return this HedGroup as a long tag string.
get_as_short()	Return this HedGroup as a short tag string.
get_frozen()	Return a frozen copy of this HedString.
get_original_hed_string()	Get the original hed string.
groups()	Return the direct child groups of this group.
lower()	Convenience function, equivalent to str(self).lower()
remove(items_to_remove)	Remove any tags/groups in items_to_remove.
remove_definitions()	Remove definition tags and groups from this string.
replace(item_to_replace, new_contents)	Replace an existing tag or group.
split_hed_string(hed_string)	Split a HED string into delimiters and tags.
split_into_groups(hed_string[, hed_schema])	Split the HED string into a parse tree.
tags()	Return the direct child tags of this group.
validate([hed_ops, error_handler])	Run the given hed_ops on this string.

Attributes

CLOSING_GROUP_CHARACTER	
OPENING_GROUP_CHARACTER	
children	A list of the direct children.
is_group	Always False since the underlying string is not a group with parentheses.
span	Return the source span.

class HedString(*hed_string*, *hed_schema=None*, *_contents=None*)

Bases: *hed.models.hed_group.HedGroup*

A HED string.

CLOSING_GROUP_CHARACTER = ')'

OPENING_GROUP_CHARACTER = '('

append(*tag_or_group*)

Add a tag or group to this group.

Parameters **tag_or_group** (*HedTag or HedGroup*) – The new object to add to this group.

Raises **ValueError** – If a HedGroupFrozen.

apply_funcs(*string_funcs*)

Run functions on this string.

Parameters **string_funcs** (*list*) – A list of functions that take a hed string object and return a list of issues.

Returns A list of issues found by these operations. Each issue is a dictionary.

Return type list

Notes

- This method potentially modifies the hed string object.

check_if_in_original(*tag_or_group*)

Check if the tag or group in original string.

Parameters **tag_or_group** (*HedTag or HedGroup*) – The HedTag or HedGroup to be looked for in this group.

Returns True if in this group.

Return type bool

property children

A list of the direct children.

convert_to_canonical_forms(*hed_schema*)

Identify all tags using the given schema.

If schema is None, still identify "key" tags such as definitions.

Parameters **hed_schema** (*HedSchema or None*) – The schema to use to validate/convert tags.

Returns A list of issues found while converting the string. Each issue is a dictionary.

Return type list

convert_to_long(*hed_schema*)

Compute canonical forms and return the long form.

Parameters **hed_schema** (*HedSchema or None*) – The schema to use to calculate forms.

Returns

- str: The string with all tags converted to long form.

- list: A list of issues found during conversion. Each issue is a dictionary.

Return type tuple

Notes

- No issues will be found if no schema is passed.

convert_to_original()

Return the original form of this string.

Returns The string with all the tags in their original form.

Return type str

Notes

Potentially with some extraneous spaces removed on returned string.

convert_to_short(*hed_schema*)

Compute canonical forms and return the short form.

Parameters **hed_schema** (*HedSchema or None*) – The schema to use to calculate forms.

Returns

- str: The string with all tags converted to short form.
- list: A list of issues found during conversion. Each issue is a dictionary.

Return type tuple

Notes

- No issues will be found if no schema is passed.

copy()

Return a deep copy of this group.

Returns The copied group.

Return type *HedGroupBase*

Notes

- The parent tag is removed.

find_def_tags(*recursive=False*, *include_groups=3*)

Find def and def-expand tags

Parameters

- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*int, 0, 1, 2, 3*) – options for how to expand or include groups

Returns A list of tuples. The contents depends on the values of the include group.

Return type list

Notes

- **The include_groups option controls the tag expansion as follows:**
  - If 0: Return only def and def expand tags/.
  - If 1: Return only def tags and def-expand groups.
  - If 2: Return only groups containing defs, or def-expand groups.
  - If 3 or any other value: Return all 3 as a tuple.

find_exact_tags(*tags_or_groups*, *recursive=False*)

Find the given tags or groups.

Parameters

- **tags_or_groups** (HedTag, HedGroupBase) – A container of tags to locate.
- **recursive** (*bool*) – If true, also check subgroups.

Returns A list of HedGroupBases the given tags/groups were found in.

Return type list

Notes

- If you pass in groups it will only find EXACT matches.
- This can only find identified tags.
- By default, definition, def, def-expand, onset, and offset are identified, even without a schema.
- If this is a HedGroup, order matters. (b, a) != (a, b)
- **If this is a HedGroupFrozen:**

  if "(a, b)" in tags_or_groups, then it will match 1 and 2, but not 3.

  1. (a, b)
  2. (b, a)
  3. (a, b, c)

find_placeholder_tag()

Return a placeholder tag, if present in this group.

Returns The placeholder tag if found.

Return type *HedTag* or None

Notes

- Assumes a valid HedString with no erroneous "#" characters.

find_tags(*search_tags*, *recursive=False*, *include_groups=2*)

Find the tags and their containing groups.

Parameters

- **search_tags** (*container*) – A container of short_base_tags to locate
- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*0, 1 or 2*) – Specify return values.

Returns The contents of the list depends on the value of include_groups.

Return type list

Notes

- If include_groups is 0, return a list of the HedTags.
- If include_groups is 1, return a list of the HedGroups containing the HedTags.
- If include_groups is 2, return a list of tuples (HedTag, HedGroup) for the found tags.
- This can only find identified tags.
- By default, definition, def, def-expand, onset, and offset are identified, even without a schema.

find_tags_with_term(*term*, *recursive=False*, *include_groups=2*)

Find any tags that contain the given term.

Note: This can only find identified tags.

Parameters

- **term** (*str*) – A single term to search for.
- **recursive** (*bool*) – If true, recursively check subgroups.
- **include_groups** (*int, 0, 1, 2, 3*) – 0, 1 or 2 If 0: Return only tags If 1: return only groups If 2 or any other value: return both
- **recursive** – If true, also check subgroups.
- **include_groups** – options for how to expand or include groups

Return type list

def_tag: HedTag The located def tag

def_expand_group: HedGroup or None If this is a def-expand rather than def tag, this will be the entire def-expand group.

group: HedGroup The group the def tag or def expand group is in.

Notes

- **The include_groups option controls the tag expansion as follows:**
  - If 0: Return only def and def expand tags/.
  - If 1: Return only def tags and def-expand groups.
  - If 2: Return only groups containing defs, or def-expand groups.
  - If 3 or any other value: Return all 3 as a tuple.

find_top_level_tags(*anchor_tags*, *include_groups=2*)

Find top level groups with an anchor tag.

Parameters

- **anchor_tags** (*container*) – A list/set/etc of short_base_tags to find groups by.
- **include_groups** (*0, 1 or 2*) – Parameter indicating what return values to include.

Returns

The returned result depends on include_groups:

- If 0: return only tags.
- If 1: return only groups.
- If 2 or any other value: return both.

Return type list or tuple

Notes

- A max of 1 tag located per top level group.

classmethod from_hed_strings(*contents*)

Factory for creating HedStrings via combination.

Parameters **contents** (*list(HedString) or None*) – A list of HedString objects to combine. This takes ownership of their children.

get_all_groups(*also_return_depth=False*)

Return HedGroups, including descendants and self.

Parameters **also_return_depth** (*bool*) – If True, yield tuples (group, depth) rather than just groups.

Returns The list of all HedGroups in this group, including descendants and self.

Return type list

get_all_tags()

Return HedTags, including descendants.

Returns A list of all the tags in this group including descendants.

Return type list

get_as_form(*tag_attribute*, *tag_transformer=None*)

Get the string corresponding to the specified form.

Parameters

- **tag_attribute** (*str*) – The hed_tag property to use to construct the string (usually short_tag or long_tag).
- **tag_transformer** (*func or None*) – A function that is applied to each tag string before returning.

Returns The constructed string after transformation

Return type str

Notes

- The signature of a tag_transformer is str def(HedTag, str).

get_as_long()

Return this HedGroup as a long tag string.

Returns The group as a string with all tags as long tags.

Return type str

get_as_short()

Return this HedGroup as a short tag string.

Returns The group as a string with all tags as short tags.

Return type str

get_frozen()

Return a frozen copy of this HedString.

This is a deep copy if the group was not already frozen.

Returns A frozen copy of this HedString.

Return type *HedStringFrozen*

get_original_hed_string()

Get the original hed string.

Returns The original string with no modification.

Return type str

groups()

Return the direct child groups of this group.

Returns All groups directly in this group, filtering out HedTag children.

Return type list

property is_group

Always False since the underlying string is not a group with parentheses.

lower()

Convenience function, equivalent to str(self).lower()

remove(*items_to_remove*)

Remove any tags/groups in items_to_remove.

Parameters **items_to_remove** (*list*) – List of HedGroups and/or HedTags to remove.

Notes

- Any groups that become empty will also be pruned.
- Identity, not equivalence is used in determining whether to remove.

remove_definitions()

Remove definition tags and groups from this string.

This does not validate definitions and will blindly removing invalid ones as well.

Returns An empty list as there are no possible issues, this list is always blank.

Return type list

replace(*item_to_replace*, *new_contents*)

Replace an existing tag or group.

Parameters

- **item_to_replace** (HedTag *or* HedGroup) – The item to replace must exist or this will raise an error.
- **new_contents** (HedTag *or* HedGroup) – Replacement contents.

property span

Return the source span.

Returns start index of the group (including parentheses) from the source string. int: end index of the group (including parentheses) from the source string.

Return type int

static split_hed_string(*hed_string*)

Split a HED string into delimiters and tags.

Parameters **hed_string** (*str*) – The HED string to split.

Returns A list of tuples where each tuple is (is_hed_tag, (start_pos, end_pos)).

Return type list

Notes

- **The tuple format is as follows**
  - is_hed_tag (bool): A (possible) hed tag if true, delimiter if not.
  - start_pos (int): Index of start of string in hed_string.
  - end_pos (int): Index of end of string in hed_string
- This function does not validate tags or delimiters in any form.

static split_into_groups(*hed_string*, *hed_schema=None*)

Split the HED string into a parse tree.

Parameters

- **hed_string** (*str*) – A hed string consisting of tags and tag groups to be processed.
- **hed_schema** (*HedSchema or None*) – Hed schema to use to identify tags.

Returns A list of HedTag and/or HedGroupBase.

Return type list

Raises **ValueError** – If the string is significantly malformed, such as mismatched parentheses.

Notes

- The parse tree consists of tag groups, tags, and delimiters.

tags()

Return the direct child tags of this group.

Returns All tags directly in this group, filtering out HedGroup children.

Return type list

validate(*hed_ops=None*, *error_handler=None*, ***kwargs*)

Run the given hed_ops on this string.

Parameters

- **hed_ops** – (func, HedOps, or list): Operations to apply to this object.
- **error_handler** (*ErrorHandler or None*) – Used to report errors in context. Uses a default if None.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options

Returns A list of issues encountered in applying these operations. Each issue is a dictionary.

Return type list

Notes

- Although this function is called validation, the HedOps can represent other transformations.

3.1.13 hed.models.HedStringGroup

class HedStringGroup(*hed_string_obj_list*)

A container with hed string objects.

Notes

- Often this is used for assembling the hed strings from multiple columns.
- The HedStringGroup passes through many of the HedString operations.

__init__(*hed_string_obj_list*)

Constructor for the HedStringGroup class.

Parameters **hed_string_obj_list** ([HedString]) – A list of component HedStrings for this combined string.

Methods

__init__(hed_string_obj_list)	Constructor for the HedStringGroup class.
append(tag_or_group)	Add a tag or group to this group.
apply_funcs(string_funcs)	Run functions on this string.
check_if_in_original(tag_or_group)	Check if the tag or group in original string.
convert_to_canonical_forms(hed_schema)	Identify all tags using the given schema.
convert_to_long(hed_schema)	Compute canonical forms and return the long form.
convert_to_original()	Return the original form of this string.
convert_to_short(hed_schema)	Compute canonical forms and return the short form.
copy()	Return a deep copy of this group.
find_def_tags([recursive, include_groups])	Find def and def-expand tags
find_exact_tags(tags_or_groups[, recursive])	Find the given tags or groups.
find_placeholder_tag()	Return a placeholder tag, if present in this group.
find_tags(search_tags[, recursive, ...])	Find the tags and their containing groups.
find_tags_with_term(term[, recursive, ...])	Find any tags that contain the given term.
find_top_level_tags(anchor_tags[, ...])	Find top level groups with an anchor tag.
from_hed_strings(contents)	Factory for creating HedStrings via combination.
get_all_groups([also_return_depth])	Return HedGroups, including descendants and self.
get_all_tags()	Return HedTags, including descendants.
get_as_form(tag_attribute[, tag_transformer])	Get the string corresponding to the specified form.
get_as_long()	Return this HedGroup as a long tag string.
get_as_short()	Return this HedGroup as a short tag string.
get_frozen()	Return a frozen copy of this HedString.
get_original_hed_string()	Get the original hed string.
groups()	Return the direct child groups of this group.
lower()	Convenience function, equivalent to str(self).lower()
remove(items_to_remove)	Remove any tags/groups in items_to_remove.
remove_definitions()	Remove definition tags and groups from this string.
replace(item_to_replace, new_contents)	Replace an existing tag or group.
split_hed_string(hed_string)	Split a HED string into delimiters and tags.
split_into_groups(hed_string[, hed_schema])	Split the HED string into a parse tree.
tags()	Return the direct child tags of this group.
validate([hed_ops, error_handler])	Run the given hed_ops on this string.

Attributes

CLOSING_GROUP_CHARACTER	
OPENING_GROUP_CHARACTER	
children	Return the direct children of this string.
is_group	Always False since the underlying string is not a group with parentheses.
span	Return the source span of this group from the source hed string.

class HedStringGroup(*hed_string_obj_list*)

Bases: *hed.models.hed_string.HedString*

A container with hed string objects.

Notes

- Often this is used for assembling the hed strings from multiple columns.
- The HedStringGroup passes through many of the HedString operations.

CLOSING_GROUP_CHARACTER = ')'

OPENING_GROUP_CHARACTER = '('

append(*tag_or_group*)

Add a tag or group to this group.

Parameters **tag_or_group** (HedTag *or* HedGroup) – The new object to add to this group.

Raises **ValueError** – If a HedGroupFrozen.

apply_funcs(*string_funcs*)

Run functions on this string.

Parameters **string_funcs** (*list*) – A list of functions that take a hed string object and return a list of issues.

Returns A list of issues found by these operations. Each issue is a dictionary.

Return type list

Notes

- This method potentially modifies the hed string object.

check_if_in_original(*tag_or_group*)

Check if the tag or group in original string.

Parameters **tag_or_group** (HedTag *or* HedGroup) – The HedTag or HedGroup to be looked for in this group.

Returns True if in this group.

Return type bool

property children

Return the direct children of this string.

Returns a list of direct children of this group.

Return type list

convert_to_canonical_forms(*hed_schema*)

Identify all tags using the given schema.

If schema is None, still identify "key" tags such as definitions.

Parameters **hed_schema** (HedSchema *or None*) – The schema to use to validate/convert tags.

Returns A list of issues found while converting the string. Each issue is a dictionary.

Return type list

convert_to_long(*hed_schema*)

Compute canonical forms and return the long form.

Parameters **hed_schema** (HedSchema *or None*) – The schema to use to calculate forms.

Returns

- str: The string with all tags converted to long form.
- list: A list of issues found during conversion. Each issue is a dictionary.

Return type tuple

Notes

- No issues will be found if no schema is passed.

convert_to_original()

Return the original form of this string.

Returns The string with all the tags in their original form.

Return type str

Notes

Potentially with some extraneous spaces removed on returned string.

convert_to_short(*hed_schema*)

Compute canonical forms and return the short form.

Parameters **hed_schema** (HedSchema *or None*) – The schema to use to calculate forms.

Returns

- str: The string with all tags converted to short form.
- list: A list of issues found during conversion. Each issue is a dictionary.

Return type tuple

Notes

- No issues will be found if no schema is passed.

copy()

Return a deep copy of this group.

Returns The copied group.

Return type *HedGroupBase*

Notes

- The parent tag is removed.

find_def_tags(*recursive=False*, *include_groups=3*)

Find def and def-expand tags

Parameters

- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*int, 0, 1, 2, 3*) – options for how to expand or include groups

Returns A list of tuples. The contents depends on the values of the include group.

Return type list

Notes

- **The include_groups option controls the tag expansion as follows:**
  - If 0: Return only def and def expand tags/.
  - If 1: Return only def tags and def-expand groups.
  - If 2: Return only groups containing defs, or def-expand groups.
  - If 3 or any other value: Return all 3 as a tuple.

find_exact_tags(*tags_or_groups*, *recursive=False*)

Find the given tags or groups.

Parameters

- **tags_or_groups** (HedTag, HedGroupBase) – A container of tags to locate.
- **recursive** (*bool*) – If true, also check subgroups.

Returns A list of HedGroupBases the given tags/groups were found in.

Return type list

Notes

- If you pass in groups it will only find EXACT matches.
- This can only find identified tags.
- By default, definition, def, def-expand, onset, and offset are identified, even without a schema.
- If this is a HedGroup, order matters. (b, a) != (a, b)
- **If this is a HedGroupFrozen:**

  if "(a, b)" in tags_or_groups, then it will match 1 and 2, but not 3.

  1. (a, b)
  2. (b, a)
  3. (a, b, c)

find_placeholder_tag()

Return a placeholder tag, if present in this group.

Returns The placeholder tag if found.

Return type *HedTag* or None

Notes

- Assumes a valid HedString with no erroneous "#" characters.

find_tags(*search_tags*, *recursive=False*, *include_groups=2*)

Find the tags and their containing groups.

Parameters

- **search_tags** (*container*) – A container of short_base_tags to locate
- **recursive** (*bool*) – If true, also check subgroups.
- **include_groups** (*0, 1 or 2*) – Specify return values.

Returns The contents of the list depends on the value of include_groups.

Return type list

Notes

- If include_groups is 0, return a list of the HedTags.
- If include_groups is 1, return a list of the HedGroups containing the HedTags.
- If include_groups is 2, return a list of tuples (HedTag, HedGroup) for the found tags.
- This can only find identified tags.
- By default, definition, def, def-expand, onset, and offset are identified, even without a schema.

find_tags_with_term(*term*, *recursive=False*, *include_groups=2*)

Find any tags that contain the given term.

Note: This can only find identified tags.

Parameters

- **term** (*str*) – A single term to search for.
- **recursive** (*bool*) – If true, recursively check subgroups.
- **include_groups** (*int, 0, 1, 2, 3*) – 0, 1 or 2 If 0: Return only tags If 1: return only groups If 2 or any other value: return both
- **recursive** – If true, also check subgroups.
- **include_groups** – options for how to expand or include groups

Return type list

def_tag: HedTag The located def tag

def_expand_group: HedGroup or None If this is a def-expand rather than def tag, this will be the entire def-expand group.

group: HedGroup The group the def tag or def expand group is in.

Notes

- **The include_groups option controls the tag expansion as follows:**
  - If 0: Return only def and def expand tags/.
  - If 1: Return only def tags and def-expand groups.
  - If 2: Return only groups containing defs, or def-expand groups.
  - If 3 or any other value: Return all 3 as a tuple.

find_top_level_tags(*anchor_tags*, *include_groups=2*)

Find top level groups with an anchor tag.

Parameters

- **anchor_tags** (*container*) – A list/set/etc of short_base_tags to find groups by.
- **include_groups** (*0, 1 or 2*) – Parameter indicating what return values to include.

Returns

The returned result depends on include_groups:

- If 0: return only tags.
- If 1: return only groups.
- If 2 or any other value: return both.

Return type list or tuple

Notes

- A max of 1 tag located per top level group.

classmethod from_hed_strings(*contents*)

Factory for creating HedStrings via combination.

Parameters **contents** (*list(HedString) or None*) – A list of HedString objects to combine. This takes ownership of their children.

get_all_groups(*also_return_depth=False*)

Return HedGroups, including descendants and self.

Parameters **also_return_depth** (*bool*) – If True, yield tuples (group, depth) rather than just groups.

Returns The list of all HedGroups in this group, including descendants and self.

Return type list

get_all_tags()

Return HedTags, including descendants.

Returns A list of all the tags in this group including descendants.

Return type list

get_as_form(*tag_attribute*, *tag_transformer=None*)

Get the string corresponding to the specified form.

Parameters

- **tag_attribute** (*str*) – The hed_tag property to use to construct the string (usually short_tag or long_tag).
- **tag_transformer** (*func or None*) – A function that is applied to each tag string before returning.

Returns The constructed string after transformation

Return type str

Notes

- The signature of a tag_transformer is str def(HedTag, str).

get_as_long()

Return this HedGroup as a long tag string.

Returns The group as a string with all tags as long tags.

Return type str

get_as_short()

Return this HedGroup as a short tag string.

Returns The group as a string with all tags as short tags.

Return type str

get_frozen()

Return a frozen copy of this HedString.

This is a deep copy if the group was not already frozen.

Returns A frozen copy of this HedString.

Return type *HedStringFrozen*

get_original_hed_string()

Get the original hed string.

Returns The original string with no modification.

Return type str

groups()

Return the direct child groups of this group.

Returns All groups directly in this group, filtering out HedTag children.

Return type list

property is_group

Always False since the underlying string is not a group with parentheses.

lower()

Convenience function, equivalent to str(self).lower()

remove(*items_to_remove*)

Remove any tags/groups in items_to_remove.

Parameters **items_to_remove** (*list*) – A list of HedGroup and HedTag objects to remove.

Notes

- Any groups that become empty will also be pruned.
- This goes by identity, not equivalence.

remove_definitions()

Remove definition tags and groups from this string.

This does not validate definitions and will blindly removing invalid ones as well.

Returns An empty list as there are no possible issues, this list is always blank.

Return type list

replace(*item_to_replace*, *new_contents*)

Replace an existing tag or group.

Parameters

- **item_to_replace** (HedTag *or* HedGroup) – The tag to replace.
- **new_contents** (HedTag *or* HedGroup *or* *list*) – The replacements for the tag.

Notes

- It tag must exist in this an error is raised.

property span

Return the source span of this group from the source hed string.

Returns

- int: start index of the group (including parentheses) from the source string.
- int: end index of the group (including parentheses) from the source string.

Return type tuple

static split_hed_string(*hed_string*)

Split a HED string into delimiters and tags.

Parameters **hed_string** (*str*) – The HED string to split.

Returns A list of tuples where each tuple is (is_hed_tag, (start_pos, end_pos)).

Return type list

Notes

- **The tuple format is as follows**
  - is_hed_tag (bool): A (possible) hed tag if true, delimiter if not.
  - start_pos (int): Index of start of string in hed_string.
  - end_pos (int): Index of end of string in hed_string
- This function does not validate tags or delimiters in any form.

static split_into_groups(*hed_string*, *hed_schema=None*)

Split the HED string into a parse tree.

Parameters

- **hed_string** (*str*) – A hed string consisting of tags and tag groups to be processed.
- **hed_schema** (HedSchema *or None*) – Hed schema to use to identify tags.

Returns A list of HedTag and/or HedGroupBase.

Return type list

Raises **ValueError** – If the string is significantly malformed, such as mismatched parentheses.

Notes

- The parse tree consists of tag groups, tags, and delimiters.

tags()

Return the direct child tags of this group.

Returns All tags directly in this group, filtering out HedGroup children.

Return type list

validate(*hed_ops=None*, *error_handler=None*, ***kwargs*)

Run the given hed_ops on this string.

Parameters

- **hed_ops** – (func, HedOps, or list): Operations to apply to this object.
- **error_handler** (*ErrorHandler or None*) – Used to report errors in context. Uses a default if None.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options

Returns A list of issues encountered in applying these operations. Each issue is a dictionary.

Return type list

Notes

- Although this function is called validation, the HedOps can represent other transformations.

3.1.14 hed.models.HedTag

class HedTag(*hed_string*, *span=None*, *hed_schema=None*)

A single HED tag.

Notes

- HedTag is a smart class in that it keeps track of its original value and positioning

as well as pointers to the relevant HED schema information, if relevant.

__init__(*hed_string*, *span=None*, *hed_schema=None*)

Creates a HedTag.

Parameters

- **hed_string** (*str*) – Source hed string for this tag.
- **span** (*int, int*) – The start and end indexes of the tag in the hed_string.
- **hed_schema** (HedSchema *or None*) – A convenience parameter for calculating canonical forms on creation.

Notes

- This does not produce issues and is used primarily for testing.

Methods

__init__(hed_string[, span, hed_schema])	Creates a HedTag.
add_prefix_if_needed(required_prefix)	Add a prefix to this tag *unless* already formatted.
any_parent_has_attribute(attribute)	Check if the tag or any of its parents has the attribute.
base_tag_has_attribute(tag_attribute)	Check to see if the tag has a specific attribute.
convert_to_canonical_forms(hed_schema)	Update internal state based on schema.
get_stripped_unit_value()	Return the extension portion without units.
get_tag_unit_class_units()	Get the unit class units associated with a particular tag.
get_unit_class_default_unit()	Get the default unit class unit for this tag.
has_attribute(attribute)	Return true if this is an attribute this tag has.
is_basic_tag()	Return True if a known tag with no extension or value.
is_extension_allowed_tag()	Check if tag has 'extensionAllowed' attribute.
is_takes_value_tag()	Return true if this is a takes value tag.
is_unit_class_tag()	Return true if this is a unit class tag.
is_value_class_tag()	Return true if this is a value class tag.
lower()	Convenience function, equivalent to str(self).lower().
replace_placeholder(placeholder_value)	If tag has a placeholder character(#), replace with value.
tag_exists_in_schema()	Get the schema entry for this tag.
tag_modified()	Return true if tag has been modified from original.

Attributes

attributes	Return a dict of all the attributes this tag has.
base_tag	Long form without value or extension.
extension_or_value_portion	Get the extension or value of tag
library_prefix	Library prefix for this tag if one exists.
long_tag	Long form including value or extension.
org_base_tag	Original form without value or extension.
org_tag	Return the original unmodified tag.
short_base_tag	Short form without value or extension
short_tag	Short form including value or extension.
tag	Return the entire user editable attribute in the tag.
tag_terms	Return a tuple of all the terms in this tag Lowercase.
unit_classes	Return a dict of all the unit classes this tag accepts.
value_classes	Return a dict of all the value classes this tag accepts.

class HedTag(*hed_string*, *span=None*, *hed_schema=None*)

Bases: `object`

A single HED tag.

Notes

- HedTag is a smart class in that it keeps track of its original value and positioning

as well as pointers to the relevant HED schema information, if relevant.

add_prefix_if_needed(*required_prefix*)

Add a prefix to this tag *unless* already formatted.

Parameters **required_prefix** (*str*) – The full name_prefix to add if not present.

Notes

- This means we verify the tag does not have the required name_prefix, or any partial name_prefix.

Examples

Required: KnownTag1/KnownTag2

Case 1: KnownTag1/KnownTag2/ColumnValue Will not be changed, has name_prefix already

Case 2: KnownTag2/ColumnValue Will not be changed, has partial name_prefix already

Case 3: ColumnValue Prefix will be added.

any_parent_has_attribute(*attribute*)

Check if the tag or any of its parents has the attribute.

Parameters **attribute** (*str*) – The name of the attribute to check for.

Returns True if the tag has the given attribute. False, if otherwise.

Return type bool

property attributes

Return a dict of all the attributes this tag has.

Returns empty dict if this is not a value tag.

Returns A dict of attributes this tag has.

Return type dict

Notes

- Returns empty dict if this is not a unit class tag.
- The dictionary has unit name as the key and HedSchemaEntry as value.

property base_tag

Long form without value or extension.

Returns The long form of the tag, without value or extension.

Return type base_tag (str)

Notes

- Only valid after calling convert_to_canonical_forms.

base_tag_has_attribute(*tag_attribute*)

Check to see if the tag has a specific attribute.

Parameters **tag_attribute** (*str*) – A tag attribute.

Returns True if the tag has the specified attribute. False, if otherwise.

Return type bool

convert_to_canonical_forms(*hed_schema*)

Update internal state based on schema.

Parameters **hed_schema** (HedSchema) – The schema to use to validate this tag

Returns A list of issues found during conversion. Each element is a dictionary.

Return type list

property extension_or_value_portion

Get the extension or value of tag

Generally this is just the portion after the last slash. Returns an empty string if no extension or value.

Returns The tag name.

Return type str

Notes

- This tag must have been computed first.

get_stripped_unit_value()

Return the extension portion without units.

Returns The extension portion with the units removed. unit (str or None): None if no valid unit found.

Return type stripped_unit_value (str)

Examples

'Duration/3 ms' will return '3'

get_tag_unit_class_units()

Get the unit class units associated with a particular tag.

Args:

Returns A list containing the unit class units associated with a particular tag or an empty list.

Return type list

get_unit_class_default_unit()

Get the default unit class unit for this tag.

Returns The default unit class unit associated with the specific tag or an empty string.

Return type str

has_attribute(*attribute*)

Return true if this is an attribute this tag has.

Parameters **attribute** (*str*) – Name of the attribute.

Returns True if this tag has the attribute.

Return type bool

is_basic_tag()

Return True if a known tag with no extension or value.

Returns True if this is a known tag without extension or value.

Return type bool

is_extension_allowed_tag()

Check if tag has 'extensionAllowed' attribute.

Recursively checks parent tag entries for the attribute as well.

Returns True if the tag has the 'extensionAllowed' attribute. False, if otherwise.

Return type bool

is_takes_value_tag()

Return true if this is a takes value tag.

Returns True if this is a takes value tag.

Return type bool

is_unit_class_tag()

Return true if this is a unit class tag.

Returns True if this is a unit class tag.

Return type bool

is_value_class_tag()

Return true if this is a value class tag.

Returns True if this is a a tag with a value class.

Return type bool

property library_prefix

Library prefix for this tag if one exists.

Returns The library prefix, including the colon.

Return type prefix (str)

property long_tag

Long form including value or extension.

Returns The long form of this tag.

Return type str

lower()

Convenience function, equivalent to str(self).lower().

property org_base_tag

Original form without value or extension.

Returns The original form of the tag, without value or extension.

Return type base_tag (str)

Notes

- Warning: This could be empty if the original tag had a name_prefix prepended.

e.g. a column where "Label/" is prepended, thus the column value has zero base portion. - Only valid after calling convert_to_canonical_forms.

property org_tag

Return the original unmodified tag.

Returns The original unmodified tag.

Return type str

replace_placeholder(*placeholder_value*)

If tag has a placeholder character(#), replace with value.

Parameters **placeholder_value** (*str*) – Value to replace placeholder with.

property short_base_tag

Short form without value or extension

Returns The short non-extension port of a tag.

Return type base_tag (str)

Notes

- ParentNodes/Def/DefName would return just "Def".

property short_tag

Short form including value or extension.

Returns The short form of the tag, including value or extension.

Return type

short_tag (str)

Note: Only valid after calling convert_to_canonical_forms

property tag

Return the entire user editable attribute in the tag.

Returns the original tag if no user form set.

Returns The custom set user form of the tag.

Return type tag (str)

tag_exists_in_schema()

Get the schema entry for this tag.

Returns True if this tag exists.

Return type bool

Notes

- This does NOT assure this is a valid tag.

tag_modified()

Return true if tag has been modified from original.

Returns Return True if the tag is modified.

Return type bool

Notes

- Modifications can include adding a column name_prefix.

property tag_terms

Return a tuple of all the terms in this tag Lowercase.

Returns Tuple of terms or empty tuple for unidentified tag.

Return type tag_terms (str)

Notes

- Does not include any extension.

property unit_classes

Return a dict of all the unit classes this tag accepts.

Returns A dict of unit classes this tag accepts.

Return type unit_classes (dict)

Notes

- Returns empty dict if this is not a unit class tag.
- The dictionary has unit name as the key and HedSchemaEntry as value.

property value_classes

Return a dict of all the value classes this tag accepts.

Returns A dictionary of HedSchemaEntry value classes this tag accepts.

Return type dict

Notes

- Returns empty dict if this is not a value class.
- The dictionary has unit name as the key and HedSchemaEntry as value.

3.1.15 hed.models.OnsetMapper

class OnsetMapper(*def_mapper*)

HedOps responsible for matching onset/offset pairs.

__init__(*def_mapper*)

Methods

__init__(def_mapper)	
check_for_onset_offset(hed_string_obj)	Check for onset or offset and track context.

class OnsetMapper(*def_mapper*)

Bases: *hed.models.hed_ops.HedOps*

HedOps responsible for matching onset/offset pairs.

check_for_onset_offset(*hed_string_obj*)

Check for onset or offset and track context.

Parameters **hed_string_obj** (HedString) – The hed string to check. Finds a maximum of one onset tag.

Returns A list of issues found in validating onsets (i.e., out of order onsets, unknown def names).

Return type list

Notes

- Each issue in the return list is a dictionary.

3.1.16 hed.models.Sidecar

class Sidecar(*file*, *name=None*)

Contents of a JSON file or merged file.

Notes

- The Sidecar maintains its own definition dictionaries.

__init__(*file*, *name=None*)

Construct a Sidecar object representing a JSON file.

Parameters

- **file** (*str or FileLike*) – A string or file-like object representing a JSON file.
- **name** (*str or None*) – Optional name identifying this sidecar, generally a filename.

Methods

__init__(file[, name])	Construct a Sidecar object representing a JSON file.
get_as_json_string()	Return this sidecar's column metadata as a string.
get_def_dicts([extra_def_dicts])	Return DefinitionDicts for the columns in this sidecar.
hed_string_iter([hed_ops, error_handler, ...])	Iterator over hed strings in columns.
load_multiple_sidecars(input_list)	Utility for loading multiple json files.
load_sidecar_file(file)	Load column metadata from a given json file.
save_as_json(save_filename)	Save column metadata to a JSON file.
set_hed_string(new_hed_string, position)	Set a provided column/category key/etc.
validate_entries([hed_ops, name, ...])	Run the given hed_ops on all columns in this sidecar.

class Sidecar(*file*, *name=None*)

Bases: object

Contents of a JSON file or merged file.

Notes

- The Sidecar maintains its own definition dictionaries.

get_as_json_string()

Return this sidecar's column metadata as a string.

Returns The json string representing this sidecar.

Return type str

get_def_dicts(*extra_def_dicts=None*)

Return DefinitionDicts for the columns in this sidecar.

Parameters **extra_def_dicts** (*list, DefinitionDict, or None*) – Extra dicts to add to the list.

Returns A list of definition dicts for each column plus any found in extra_def_dicts.

Return type list

hed_string_iter(*hed_ops=None*, *error_handler=None*, *expand_defs=False*, *remove_definitions=False*, *allow_placeholders=True*, *extra_def_dicts=None*, ***kwargs*)

Iterator over hed strings in columns.

Parameters

- **hed_ops** (*func, HedOps, list*) – A HedOps, funcs or list of these to apply to the hed strings before returning
- **error_handler** (*ErrorHandler*) – The error handler to use for context, uses a default one if none.
- **expand_defs** (*bool*) – If True, expand all def tags located in the strings.
- **remove_definitions** (*bool*) – If True, remove all definitions found in the string.
- **allow_placeholders** (*bool*) – If False, placeholders will be marked as validation warnings.
- **extra_def_dicts** (*DefinitionDict, list, None*) – Extra dicts to add to the list.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Yields *tuple* – - HedString: A HedString at a given column and key position. - tuple: Indicates where hed_string was loaded from so it can be later set by the user - list: A list of issues found performing ops. Each issue is a dictionary.

static load_multiple_sidecars(*input_list*)

Utility for loading multiple json files.

Parameters **input_list** (*list*) – A list of filenames or Sidecar files in any mix.

Returns A list sidecars.

Return type list

Raises **HedFileError** – If any of the files are not found.

load_sidecar_file(*file*)

Load column metadata from a given json file.

Parameters **file** (*str or FileLike*) – If a string, this is a filename. Otherwise, it will be parsed as a file-like.

Raises **HedFileError** – If the file was not found or could not be parsed into JSON.

Notes

- Multiple files can be loaded into one Sidecar, but it is discouraged.

save_as_json(*save_filename*)

Save column metadata to a JSON file.

Parameters **save_filename** (*str*) – Path to save file

set_hed_string(*new_hed_string*, *position*)

Set a provided column/category key/etc.

Parameters

- **new_hed_string** (*str or* HedString) – The new hed_string to replace the value at position.
- **position** (*tuple*) – The (HedString, str, list) tuple returned from hed_string_iter.

validate_entries(*hed_ops=None*, *name=None*, *extra_def_dicts=None*, *error_handler=None*, ***kwargs*)

Run the given hed_ops on all columns in this sidecar.

Parameters

- **hed_ops** (*list, func, or* HedOps) – A HedOps, func or list of these to apply to hed strings in this sidecar.
- **name** (*str*) – If present, will use this as the filename for context, rather than using the actual filename Useful for temp filenames.
- **extra_def_dicts** – (DefinitionDict, list, or None): If present use these in addition to sidecar's def dicts.
- **error_handler** (*ErrorHandler or None*) – Used to report errors. Uses a default one if none passed in.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Returns The list of validation issues found. Individual issues are in the form of a dict.

Return type list

3.1.17 hed.models.SpreadsheetInput

class SpreadsheetInput(*file=None*, *file_type=None*, *worksheet_name=None*, *tag_columns=None*, *has_column_names=True*, *column_prefix_dictionary=None*, *def_dicts=None*, *name=None*)

A spreadsheet of HED tags.

__init__(*file=None*, *file_type=None*, *worksheet_name=None*, *tag_columns=None*, *has_column_names=True*, *column_prefix_dictionary=None*, *def_dicts=None*, *name=None*)

Constructor for the SpreadsheetInput class.

Parameters

- **file** (*str or file like*) – An xlsx/tsv file to open or a File object.
- **file_type** (*str or None*) – ".xlsx" for excel, ".tsv" or ".txt" for tsv. data. If file is a string, the

- **worksheet_name** (*str or None*) – The name of the Excel workbook worksheet that contains the HED tags. Not applicable to tsv files. If omitted for Excel, the first worksheet is assumed.
- **tag_columns** (*list*) – A list of ints containing the columns that contain the HED tags. The default value is [2] indicating only the second column has tags.
- **has_column_names** (*bool*) – True if file has column names. Validation will skip over the first line of the file if the spreadsheet as column names.
- **column_prefix_dictionary** (*dict*) – A dictionary with column number keys and prefix values.
- **def_dicts** (DefinitionDict *or list*) – A DefinitionDict or list of DefDicts containing definitions for this object other than the ones extracted from the SpreadsheetInput object itself.

Examples

A prefix dictionary {3: 'Label/', 5: 'Description/'} indicates that column 3 and 5 have HED tags that need to be prefixed by Label/ and Description/ respectively. Column numbers 3 and 5 should also be included in the tag_columns list.

Methods

__init__([file, file_type, worksheet_name, ...])	Constructor for the SpreadsheetInput class.
convert_to_long(hed_schema[, error_handler])	Convert all tags to long form.
convert_to_short(hed_schema[, error_handler])	Convert all tags to short form.
extract_definitions([error_handler])	Gather and validate all definitions.
get_def_and_mapper_issues(error_handler[, ...])	Return definition and column issues.
get_worksheet([worksheet_name])	Get the requested worksheet.
iter_dataframe([hed_ops, mapper, ...])	Iterate rows based on the given column mapper.
iter_raw([hed_ops, error_handler])	Iterate all columns without substitutions
reset_mapper(new_mapper)	Set mapper to a different view of the file.
set_cell(row_number, column_number, ...[, ...])	Replace the specified cell with transformed text.
to_csv([file, output_processed_file])	Write to file or return as a string.
to_excel(file[, output_processed_file])	Output to an Excel file.
update_definition_mapper(def_dict)	Add definitions from dict(s) if mapper exists.
validate_file(hed_ops[, name, ...])	Run the hed_ops on columns and rows.

Attributes

COMMA_DELIMITER
EXCEL_EXTENSION
FILE_EXTENSION
FILE_INPUT
STRING_INPUT
TAB_DELIMITER
TEXT_EXTENSION

dataframe	The underlying dataframe.
has_column_names	True if dataframe has column names.
loaded_workbook	The underlying loaded workbooks.
name	Name of the data.
worksheet_name	The worksheet name.

class SpreadsheetInput(*file=None*, *file_type=None*, *worksheet_name=None*, *tag_columns=None*, *has_column_names=True*, *column_prefix_dictionary=None*, *def_dicts=None*, *name=None*)

Bases: *hed.models.base_input.BaseInput*

A spreadsheet of HED tags.

COMMA_DELIMITER = ','

EXCEL_EXTENSION = ['.xlsx']

FILE_EXTENSION = ['.tsv', '.txt', '.xlsx']

FILE_INPUT = 'file'

STRING_INPUT = 'string'

TAB_DELIMITER = '\t'

TEXT_EXTENSION = ['.tsv', '.txt']

convert_to_long(*hed_schema*, *error_handler=None*)

Convert all tags to long form.

Parameters

- **hed_schema** (HedSchema) – The schema to use to convert tags.
- **error_handler** (*ErrorHandler*) – The error handler to use for context, uses a default if none.

Returns A list of issue dictionaries corresponding to issues found during conversion.

Return type dict

convert_to_short(*hed_schema*, *error_handler=None*)

Convert all tags to short form.

Parameters

- **hed_schema** (HedSchema) – The schema to use to convert tags.
- **error_handler** (*ErrorHandler*) – The error handler to use for context, uses a default if none.

Returns A list of issue dictionaries corresponding to issues found during conversion.

Return type dict

property dataframe

The underlying dataframe.

extract_definitions(*error_handler=None*)

Gather and validate all definitions.

Parameters **error_handler** (*ErrorHandler*) – The error handler to use for context or a default if None.

Returns Contains all the definitions located in the file.

Return type *DefinitionDict*

get_def_and_mapper_issues(*error_handler*, *check_for_warnings=False*)

Return definition and column issues.

Parameters

- **error_handler** (*ErrorHandler*) – The error handler to use.
- **check_for_warnings** (*bool*) – If True check for and return warnings as well as errors.

Returns A list of definition and mapping issues. Each issue is a dictionary.

Return type dict

get_worksheet(*worksheet_name=None*)

Get the requested worksheet.

Parameters **worksheet_name** (*str or None*) – The name of the requested worksheet by name or the first one if None.

Returns The workbook request.

Return type openpyxl.workbook.Workbook

Notes

If None, returns the first worksheet.

property has_column_names

True if dataframe has column names.

iter_dataframe(*hed_ops=None*, *mapper=None*, *return_string_only=True*, *run_string_ops_on_columns=False*, *error_handler=None*, *expand_defs=False*, *remove_definitions=True*, ***kwargs*)

Iterate rows based on the given column mapper.

Parameters

- **hed_ops** (*list, func, HedOps, or None*) – A func, a HedOps or a list of these to apply to the hed strings before returning.
- **mapper** (*ColumnMapper or None*) – The column name to column number mapper (or internal mapper if None).
- **return_string_only** (*bool*) – If True, do not return issues list, individual columns, attribute columns, etc.
- **run_string_ops_on_columns** (*bool*) – If true, run all tag and string ops on columns, rather than columns then rows.
- **error_handler** (*ErrorHandler or None*) – The error handler to use for context or a default if None.
- **expand_defs** (*bool*) – If True, expand def tags into def-expand groups.
- **remove_definitions** (*bool*) – If true, remove all definition tags found.
- **()** (*kwargs*) – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Yields *dict* – A dict with parsed row, including keys: "HED", "column_to_hed_tags", and possibly "column_issues".

iter_raw(*hed_ops=None*, *error_handler=None*, ***kwargs*)

Iterate all columns without substitutions

Parameters

- **hed_ops** (*list, func, HedOps, or None*) – A func, a HedOps or a list of these to apply to the hed strings before returning.
- **error_handler** (*ErrorHandler or None*) – Handler to use for context or a default one if None.
- **kwargs** –

Yields - *dict* – A dict with column_number keys and values corresponding to the cell at that position.

Notes

- See models.hed_ops.translate_ops or the specific hed_ops for additional options.
- Primarily for altering or re-saving the original file (e.g., convert short tags to long).
- Used for initial processing when trying to find definitions.

property loaded_workbook

The underlying loaded workbooks.

property name

Name of the data.

reset_mapper(*new_mapper*)

Set mapper to a different view of the file.

Parameters **new_mapper** (*ColumnMapper*) – A column mapper to be associated with this base input.

set_cell(*row_number*, *column_number*, *new_string_obj*, *include_column_prefix_if_exist=False*, *tag_form='short_tag'*)

Replace the specified cell with transformed text.

Parameters

- **row_number** (*int*) – The row number of the spreadsheet to set.
- **column_number** (*int*) – The column number of the spreadsheet to set.
- **new_string_obj** (HedString) – Object with text to put in the given cell.
- **include_column_prefix_if_exist** (*bool*) – If True and the column matches one from mapper _column_prefix_dictionary, remove the prefix.
- **tag_form** (*str*) – Version of the tags (short_tag, long_tag, base_tag, etc)

Notes

Any attribute of a HedTag that returns a string is a valid value of tag_form.

to_csv(*file=None*, *output_processed_file=False*)

Write to file or return as a string.

Parameters

- **file** (*str, file-like, or None*) – Location to save this file. If None, return as string.
- **output_processed_file** (*bool*) – Replace all definitions and labels in HED columns as appropriate. Also fills in things like categories.

Returns None if file is given or the contents as a str if file is None.

Return type None or str

to_excel(*file*, *output_processed_file=False*)

Output to an Excel file.

Parameters

- **file** (*str or file-like*) – Location to save this base input.
- **output_processed_file** (*bool*) – If True, replace definitions and labels in HED columns. Also fills in things like categories.

Raises **HedFileError if empty file object or file cannot be opened.** –

update_definition_mapper(*def_dict*)

Add definitions from dict(s) if mapper exists.

Parameters **def_dict** (*list or* DefinitionDict) – Add the DefDict or list of DefDict to the internal definition mapper.

validate_file(*hed_ops*, *name=None*, *error_handler=None*, *check_for_warnings=True*, ***kwargs*)

Run the hed_ops on columns and rows.

Parameters

- **hed_ops** (*func,* HedOps*, or list of func and/or HedOps*) – The HedOps of funcs to apply.
- **name** (*str*) – If present, use this as the filename for context, rather than using the actual filename Useful for temp filenames.

- **error_handler** (*ErrorHandler or None*) – Used to report errors a default one if None.
- **check_for_warnings** (*bool*) – If True check for and return warnings as well as errors.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Returns The list of validation issues found. The list elements are dictionaries.

Return type list

property worksheet_name

The worksheet name.

3.1.18 hed.models.TabularInput

class TabularInput(*file=None*, *sidecar=None*, *attribute_columns=None*, *extra_def_dicts=None*, *also_gather_defs=True*, *name=None*)

A BIDS tabular tsv file with sidecar.

__init__(*file=None*, *sidecar=None*, *attribute_columns=None*, *extra_def_dicts=None*, *also_gather_defs=True*, *name=None*)

Constructor for the TabularInput class.

Parameters

- **file** (*str or file like*) – A tsv file to open.
- **sidecar** (*str or* Sidecar) – A Sidecar filename or Sidecar
- **attribute_columns** (*str or int or [str] or [int]*) – A list of column names or numbers to treat as attributes. Default: ["duration", "onset"]
- **extra_def_dicts** (*[*DefinitionDict*], *DefinitionDict*, or None*) – DefinitionDict objects containing all the definitions this file should use other than the ones coming from the file itself and from the sidecar. These are added as the last entries, so names will override earlier ones.
- **also_gather_defs** (*bool*) – If False, do NOT extract any definitions from column groups, assume they are already in the def_dict list.
- **name** (*str*) – The name to display for this file for error purposes.

Methods

__init__([file, sidecar, attribute_columns, ...])	Constructor for the TabularInput class.
convert_to_long(hed_schema[, error_handler])	Convert all tags to long form.
convert_to_short(hed_schema[, error_handler])	Convert all tags to short form.
create_def_mapper(column_mapper[, ...])	Create the definition mapper for this file.
extract_definitions([error_handler])	Gather and validate all definitions.
get_def_and_mapper_issues(error_handler[, ...])	Return definition and column issues.
get_worksheet([worksheet_name])	Get the requested worksheet.
iter_dataframe([hed_ops, mapper, ...])	Iterate rows based on the given column mapper.
iter_raw([hed_ops, error_handler])	Iterate all columns without substitutions
reset_column_mapper([sidecars, ...])	Change the sidecars and settings.
reset_mapper(new_mapper)	Set mapper to a different view of the file.
set_cell(row_number, column_number, ...[, ...])	Replace the specified cell with transformed text.
to_csv([file, output_processed_file])	Write to file or return as a string.
to_excel(file[, output_processed_file])	Output to an Excel file.
update_definition_mapper(def_dict)	Add definitions from dict(s) if mapper exists.
validate_file(hed_ops[, name, ...])	Run the hed_ops on columns and rows.
validate_file_sidecars([hed_ops, error_handler])	Validate column definitions and hed strings.

Attributes

COMMA_DELIMITER	
EXCEL_EXTENSION	
FILE_EXTENSION	
FILE_INPUT	
HED_COLUMN_NAME	
STRING_INPUT	
TAB_DELIMITER	
TEXT_EXTENSION	
dataframe	The underlying dataframe.
has_column_names	True if dataframe has column names.
loaded_workbook	The underlying loaded workbooks.
name	Name of the data.
worksheet_name	The worksheet name.

class TabularInput(*file=None*, *sidecar=None*, *attribute_columns=None*, *extra_def_dicts=None*, *also_gather_defs=True*, *name=None*)

Bases: *hed.models.base_input.BaseInput*

A BIDS tabular tsv file with sidecar.

COMMA_DELIMITER = ','

EXCEL_EXTENSION = ['.xlsx']

FILE_EXTENSION = ['.tsv', '.txt', '.xlsx']

FILE_INPUT = 'file'

HED_COLUMN_NAME = 'HED'

STRING_INPUT = 'string'

TAB_DELIMITER = '\t'

TEXT_EXTENSION = ['.tsv', '.txt']

convert_to_long(*hed_schema*, *error_handler=None*)

Convert all tags to long form.

Parameters

- **hed_schema** (HedSchema) – The schema to use to convert tags.
- **error_handler** (*ErrorHandler*) – The error handler to use for context, uses a default if none.

Returns A list of issue dictionaries corresponding to issues found during conversion.

Return type dict

convert_to_short(*hed_schema*, *error_handler=None*)

Convert all tags to short form.

Parameters

- **hed_schema** (HedSchema) – The schema to use to convert tags.
- **error_handler** (*ErrorHandler*) – The error handler to use for context, uses a default if none.

Returns A list of issue dictionaries corresponding to issues found during conversion.

Return type dict

create_def_mapper(*column_mapper*, *extra_def_dicts=None*)

Create the definition mapper for this file.

Parameters

- **column_mapper** (ColumnMapper) – The column mapper to gather definitions from.
- **extra_def_dicts** (DefinitionDict *or* *[*DefinitionDict*]*) – Additional definitions to add to mapper.

Returns A class to validate or expand definitions with the given def dicts.

Return type def mapper (*DefMapper*)

Notes

- The extra_def_dicts are definitions not included in the column mapper.

property dataframe

The underlying dataframe.

extract_definitions(*error_handler=None*)

Gather and validate all definitions.

Parameters **error_handler** (*ErrorHandler*) – The error handler to use for context or a default if None.

Returns Contains all the definitions located in the file.

Return type *DefinitionDict*

get_def_and_mapper_issues(*error_handler*, *check_for_warnings=False*)

Return definition and column issues.

Parameters

- **error_handler** (*ErrorHandler*) – The error handler to use.
- **check_for_warnings** (*bool*) – If True check for and return warnings as well as errors.

Returns A list of definition and mapping issues. Each issue is a dictionary.

Return type dict

get_worksheet(*worksheet_name=None*)

Get the requested worksheet.

Parameters **worksheet_name** (*str or None*) – The name of the requested worksheet by name or the first one if None.

Returns The workbook request.

Return type openpyxl.workbook.Workbook

Notes

If None, returns the first worksheet.

property has_column_names

True if dataframe has column names.

iter_dataframe(*hed_ops=None*, *mapper=None*, *return_string_only=True*, *run_string_ops_on_columns=False*, *error_handler=None*, *expand_defs=False*, *remove_definitions=True*, ***kwargs*)

Iterate rows based on the given column mapper.

Parameters

- **hed_ops** (*list, func, HedOps, or None*) – A func, a HedOps or a list of these to apply to the hed strings before returning.
- **mapper** (*ColumnMapper or None*) – The column name to column number mapper (or internal mapper if None).
- **return_string_only** (*bool*) – If True, do not return issues list, individual columns, attribute columns, etc.

- **run_string_ops_on_columns** (*bool*) – If true, run all tag and string ops on columns, rather than columns then rows.
- **error_handler** (*ErrorHandler or None*) – The error handler to use for context or a default if None.
- **expand_defs** (*bool*) – If True, expand def tags into def-expand groups.
- **remove_definitions** (*bool*) – If true, remove all definition tags found.
- **()** (*kwargs*) – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Yields *dict* – A dict with parsed row, including keys: "HED", "column_to_hed_tags", and possibly "column_issues".

iter_raw(*hed_ops=None*, *error_handler=None*, ***kwargs*)

Iterate all columns without substitutions

Parameters

- **hed_ops** (*list, func, HedOps, or None*) – A func, a HedOps or a list of these to apply to the hed strings before returning.
- **error_handler** (*ErrorHandler or None*) – Handler to use for context or a default one if None.
- **kwargs** –

Yields - *dict* – A dict with column_number keys and values corresponding to the cell at that position.

Notes

- See models.hed_ops.translate_ops or the specific hed_ops for additional options.
- Primarily for altering or re-saving the original file (e.g., convert short tags to long).
- Used for initial processing when trying to find definitions.

property loaded_workbook

The underlying loaded workbooks.

property name

Name of the data.

reset_column_mapper(*sidecars=None*, *attribute_columns=None*)

Change the sidecars and settings.

Parameters

- **sidecars** (*str or [str] or Sidecar or [Sidecar]*) – A list of json filenames to pull sidecar info from.
- **attribute_columns** (*str or int or [str] or [int]*) – Column names or numbers to treat as attributes. Default: ["duration", "onset"]

reset_mapper(*new_mapper*)

Set mapper to a different view of the file.

Parameters **new_mapper** (ColumnMapper) – A column mapper to be associated with this base input.

set_cell(*row_number*, *column_number*, *new_string_obj*, *include_column_prefix_if_exist=False*, *tag_form='short_tag'*)

Replace the specified cell with transformed text.

Parameters

- **row_number** (*int*) – The row number of the spreadsheet to set.
- **column_number** (*int*) – The column number of the spreadsheet to set.
- **new_string_obj** (HedString) – Object with text to put in the given cell.
- **include_column_prefix_if_exist** (*bool*) – If True and the column matches one from mapper _column_prefix_dictionary, remove the prefix.
- **tag_form** (*str*) – Version of the tags (short_tag, long_tag, base_tag, etc)

Notes

Any attribute of a HedTag that returns a string is a valid value of tag_form.

to_csv(*file=None*, *output_processed_file=False*)

Write to file or return as a string.

Parameters

- **file** (*str, file-like, or None*) – Location to save this file. If None, return as string.
- **output_processed_file** (*bool*) – Replace all definitions and labels in HED columns as appropriate. Also fills in things like categories.

Returns None if file is given or the contents as a str if file is None.

Return type None or str

to_excel(*file*, *output_processed_file=False*)

Output to an Excel file.

Parameters

- **file** (*str or file-like*) – Location to save this base input.
- **output_processed_file** (*bool*) – If True, replace definitions and labels in HED columns. Also fills in things like categories.

Raises **HedFileError if empty file object or file cannot be opened.** –

update_definition_mapper(*def_dict*)

Add definitions from dict(s) if mapper exists.

Parameters **def_dict** (*list or* DefinitionDict) – Add the DefDict or list of DefDict to the internal definition mapper.

validate_file(*hed_ops*, *name=None*, *error_handler=None*, *check_for_warnings=True*, ***kwargs*)

Run the hed_ops on columns and rows.

Parameters

- **hed_ops** (*func,* HedOps*, or list of func and/or HedOps*) – The HedOps of funcs to apply.
- **name** (*str*) – If present, use this as the filename for context, rather than using the actual filename Useful for temp filenames.

- **error_handler** (*ErrorHandler or None*) – Used to report errors a default one if None.
- **check_for_warnings** (*bool*) – If True check for and return warnings as well as errors.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Returns The list of validation issues found. The list elements are dictionaries.

Return type list

validate_file_sidecars(*hed_ops=None*, *error_handler=None*, ***kwargs*)

Validate column definitions and hed strings.

Parameters

- **hed_ops** (*list*) – A list of HedOps of funcs to apply to the hed strings in the sidecars.
- **error_handler** (*ErrorHandler or None*) – Used to report errors. Uses a default one if none passed in.
- **kwargs** – See models.hed_ops.translate_ops or the specific hed_ops for additional options.

Returns A list of syntax and semantic issues found in the definitions. Each issue is a dictionary.

Return type list

Notes

- For full validation you should validate the sidecar separately.

property worksheet_name

The worksheet name.

3.1.19 hed.models.hed_ops

Infrastructure for processing HED operations.

Functions

translate_ops(hed_ops[, split_ops])	Return functions to apply to a hed string object.

Classes

HedOps(*args, **kwargs)	Base class to support HedOps.

3.2 HED schema handling

hed.schema	Data structures for handling the HED schema.
hed.schema.HedSchema()	A HED schema suitable for processing.
hed.schema.HedSchemaEntry(name, section)	A single node in a HedSchema.
hed.schema.UnitClassEntry(*args, **kwargs)	A single unit class entry in the HedSchema.
hed.schema.UnitEntry(*args, **kwargs)	A single unit entry with modifiers in the HedSchema.
hed.schema.HedTagEntry(*args, **kwargs)	A single tag entry in the HedSchema.
hed.schema.HedSchemaGroup(schema_list)	Container for multiple HedSchema objects.
hed.schema.HedSchemaSection(section_key[, ...])	Container with entries in one section of the schema.
hed.schema.hed_cache	Infrastructure for caching HED schema.
hed.schema.hed_schema_io	Utilities for loading and outputting HED schema.
hed.schema.schema_compliance	Utilities for HED schema checking.
hed.schema.schema_validation_util	Utilities used in HED validation using a HED schema.

3.2.1 hed.schema package

Subpackages

hed.schema.schema_io package

Submodules

hed.schema.schema_io.schema2wiki module

Allows output of HedSchema objects as .mediawiki format

class HedSchema2Wiki

Bases: object

process_schema(*hed_schema*)

Takes a HedSchema object and returns a list of strings representing its .mediawiki version.

Parameters **hed_schema** (HedSchema) –

Returns **mediawiki_strings** – A list of strings representing the .mediawiki version of this schema.

Return type [str]

hed.schema.schema_io.schema2xml module

Allows output of HedSchema objects as .xml format

class HedSchema2XML

Bases: object

process_schema(*hed_schema*)

Takes a HedSchema object and returns an XML tree version.

Parameters **hed_schema** (HedSchema) –

Returns mediawiki_strings – A list of strings representing the .mediawiki version of this schema.

Return type [str]

hed.schema.schema_io.wiki2schema module

This module is used to create a HedSchema object from a .mediawiki file.

class HedSchemaWikiParser(*wiki_file_path*, *schema_as_string*)

Bases: object

static load_wiki(*wiki_file_path=None*, *schema_as_string=None*)

class HedWikiSection

Bases: object

Attributes = 9

EndHed = 12

EndSchema = 5

Epilogue = 11

HeaderLine = 2

Prologue = 3

Properties = 10

Schema = 4

UnitModifiers = 7

UnitsClasses = 6

ValueClasses = 8

hed.schema.schema_io.wiki_constants module

hed.schema.schema_io.xml2schema module

This module is used to create a HedSchema object from an XML file or tree.

class HedSchemaXMLParser(*hed_xml_file_path*, *schema_as_string=None*)

Bases: object

static load_xml(*xml_file_path=None*, *schema_as_string=None*)

hed.schema.schema_io.xml_constants module

Module contents

Submodules

hed.schema.hed_cache module

Infrastructure for caching HED schema.

cache_specific_url(*hed_xml_url*, *xml_version=None*, *library_name=None*, *cache_folder=None*)

Cache a file from a URL.

Parameters

- **hed_xml_url** (*str*) – Path to an exact file at a URL, or a GitHub API url to a directory.
- **xml_version** (*str*) – If not None and hed_xml_url is a directory, return this version or None.
- **library_name** (*str or None*) – Optional schema library name.
- **cache_folder** (*str*) – The path of the hed cache. Defaults to HED_CACHE_DIRECTORY.

Returns Path to local hed XML file to use.

Return type str

cache_xml_versions(*hed_base_urls=('https://api.github.com/repos/hed-standard/hed-specification/contents/hedxml', 'https://api.github.com/repos/hed-standard/hed-schema-library/contents/library_schemas')*, *skip_folders=('deprecated',)*, *cache_folder=None*)

Cache a file from a URL.

Parameters

- **hed_base_urls** (*str or list*) – Path or list of paths.
- **skip_folders** (*list*) – A list of subfolders to skip over when downloading.
- **cache_folder** (*str*) – The folder holding the cache.

Returns

Returns -1 if cache failed, a positive number meaning time in seconds since last update if it didn't cache, 0 if it cached successfully this time.

Return type float

Notes

- The Default skip_folders is 'deprecated'.
- The HED cache folder defaults to HED_CACHE_DIRECTORY.
- **The directories on Github are of the form:** https://api.github.com/repos/hed-standard/hed-specification/contents/hedxml

get_cache_directory()

Return the current value of HED_CACHE_DIRECTORY.

get_hed_version_path(*xml_version=None*, *library_name=None*, *local_hed_directory=None*)

Get latest HED XML file path in a directory.

Parameters

- **library_name** (*str or None*) – Optional the schema library name.
- **xml_version** (*str or None*) – If not None, return this version or None.
- **local_hed_directory** (*str*) – Path to local hed directory. Defaults to HED_CACHE_DIRECTORY

Returns The path to the latest HED version the hed directory.

Return type str

get_hed_versions(*local_hed_directory=None*, *library_name=None*, *get_libraries=False*)

Get the HED versions in the hed directory.

Parameters

- **local_hed_directory** (*str*) – Directory to check for versions which defaults to hed_cache.
- **library_name** (*str or None*) – An optional schema library name.
- **get_libraries** (*bool*) – If true, return a dictionary of version numbers, with an entry for each library name.

Returns List of version numbers or dictionary {library_name: [versions]}.

Return type list or dict

get_path_from_hed_version(*hed_version*, *library_name=None*, *local_hed_directory=None*)

Return the HED XML file path for a version.

Parameters

- **hed_version** (*str*) – The HED version that is in the hed directory.
- **library_name** (*str or None*) – An optional schema library name.
- **local_hed_directory** (*str*) – The local hed path to use.

Returns The HED XML file path in the hed directory that corresponds to the hed version specified.

Return type str

Notes

- Note if no local directory is given, it defaults to HED_CACHE_DIRECTORY.

set_cache_directory(*new_cache_dir*)

Set default global hed cache directory.

Parameters **new_cache_dir** (*str*) – Directory to check for versions.

hed.schema.hed_schema module

class HedSchema

Bases: object

A HED schema suitable for processing.

property all_tags

Return the tag schema section.

Returns The tag section.

Return type *HedSchemaSection*

check_compliance(*check_for_warnings=True*, *name=None*, *error_handler=None*)

Check for HED3 compliance of this schema.

Parameters

- **check_for_warnings** (*bool*) – If True, also checks for formatting issues
- **name** (*str*) – If present, use this as the filename for context
- **error_handler** (*ErrorHandler or None*) – Used to report errors.

Returns A list of all warnings and errors found in the file. Each issue is a dictionary.

Return type list

Notes

- Formatting issues include invalid characters and capitalization.
- The name parameter is useful when handling temporary files.
- A default error handler is created if none passed in.

property filename

The filename if one is known.

Returns The filename of this schema.

Return type str

finalize_dictionaries()

Call to finish loading.

find_duplicate_tags()

Find all tags that are not unique.

Returns A dictionary of all duplicate short tags

Return type dict

Notes

- The returned dictionary has the short-form tags as keys and lists

of long tags sharing the short form as the values.

find_tag_entry(*tag*, *library_prefix=''*)

Find the schema entry for a given source tag.

Parameters

- **tag** (*str, HedTag*) – Any form of tag to look up. Can have an extension, value, etc.
- **library_prefix** (*str*) – The prefix the library, if any.

Returns The located tag entry for this tag. str: The remainder of the tag that isn't part of the base tag. list: A list of errors while converting.

Return type *HedTagEntry*

Notes

Works right to left (which is mostly relevant for errors).

get_all_schema_tags(*return_last_term=False*)

Get a list of all hed terms from the schema.

Returns A list of all terms(short tags) from the schema.

Return type list

Notes

Compatible with Hed2 or Hed3.

get_all_tag_attributes(*tag_name*, *key_class='tags'*)

Gather all attributes for a given tag name.

Parameters

- **tag_name** (*str*) – The name of the tag to check.
- **key_class** (*str*) – The type of attributes requested. e.g. Tag, Units, Unit modifiers, or attributes.

Returns A dictionary of attribute name and attribute value.

Return type dict

Notes

If keys is None, gets all normal hed tag attributes.

get_as_mediawiki_string()

Return the schema to a mediawiki string.

Returns The schema as a string in mediawiki format.

Return type str

get_as_xml_string()

Return the schema to an XML string.

Returns Return the schema as an XML string.

Return type str

get_desc_iter()

Return an iterator over all the descriptions.

Yields *tuple* – - str: The tag node name. - str: The description associated with the node.

get_modifiers_for_unit(*unit*)

Return the valid modifiers for the given unit

Parameters **unit** (*str*) – A known unit.

Returns List of HedSchemaEntry.

Return type list

Notes

This is a lower level one that doesn't rely on the Unit entries being fully setup.

get_tag_attribute_names()

Return a dict of all allowed tag attributes.

Returns A dictionary whose keys are attribute names and values are HedSchemaEntry object.

Return type dict

get_tag_description(*tag_name*, *key_class='tags'*)

Return the description associated with the tag.

Parameters

- **tag_name** (*str*) – A hed tag name(or unit/unit modifier etc) with proper capitalization.
- **key_class** (*str*) – A string indicating type of description (e.g. All tags, Units, Unit modifier). The default is HedSectionKey.AllTags.

Returns A description of the specified tag.

Return type str

get_tag_entry(*name*, *key_class='tags'*, *library_prefix=''*)

Return the schema entry for this tag, if one exists.

Parameters

- **name** (*str*) – Any form of basic tag(or other section entry) to look up. This will not handle extensions or similar.

- **key_class** (HedSectionKey *or* *str*) – The type of entry to return.
- **library_prefix** (*str*) – Only used for HedSchemaGroup otherwise ignored.

Returns The schema entry for the given tag.

Return type *HedSchemaEntry*

get_tags_with_attribute(*key*, *section_key='tags'*)

Return tag entries with the given attribute.

Parameters

- **key** (*str*) – A tag attribute. Eg HedKey.ExtensionAllowed
- **section_key** (*str*) – The HedSectionKey for the section to retrieve from.

Returns A list of all tags with this attribute.

Return type list

Notes

- The result is cached so will be fast after first call.

get_unit_class_units(*unit_class_type*)

Get the list of unit class units this type will accept.

Parameters **unit_class_type** (*str*) – The unit class type to check for. e.g. "time".

Returns A list of each UnitEntry this type allows.

Return type list

Examples

Eg 'time' returns ['second', 's', 'day', 'minute', 'hour']

get_unknown_attributes()

Retrieve the current list of unknown attributes.

Returns The keys are attribute names and the values are lists of tags with this attribute.

Return type dict

Notes

This includes attributes found in the wrong section for example unitClass attribute found on a Tag.

The return tag list is in long form.

property has_duplicate_tags

Return True if this is a valid hed3.

Returns True if this is a valid hed3 schema with no duplicate short tags.

Return type bool

property is_hed3_schema

Return true if this is at least version HED3.

Returns True if this is a hed3 schema.

Return type bool

Notes

- This is considered true if the version number is >= 8.0 or it has a library name.

property library

The name of this library schema if one exists.

Returns Library name if any.

Return type str or None

save_as_mediawiki()

Save as mediawiki to a temporary file.

Returns The newly created schema filename.

Return type str

save_as_xml()

Save as XML to a temporary file.

Returns The name of the newly created schema file.

Return type str

schema_for_prefix(*prefix*)

Return HedSchema object for this prefix.

Parameters **prefix** (*str*) – The schema library name prefix.

Returns The HED schema object for this schema.

Return type *HedSchema*

Notes

-This is mostly a placeholder for HedSchemaGroup and may be refactored out later.

set_library_prefix(*library_prefix*)

Set library prefix associated for this schema.

Parameters **library_prefix** (*str*) – Should be empty, or end with a colon.(Colon will be automated added if missing).

property unit_classes

Return the unit classes schema section.

Returns The unit classes section.

Return type *HedSchemaSection*

property unit_modifiers

Return the modifiers classes schema section

Returns The unit modifiers section.

Return type *HedSchemaSection*

property valid_prefixes

Return a list of all prefixes this schema will accept

Returns A list of valid tag prefixes for this schema.

Return type list

Notes

- The return value is always length 1 if using a HedSchema.

property value_classes

Return the value classes schema section.

Returns The value classes section.

Return type *HedSchemaSection*

property version

The HED version of this schema.

Returns The version of this schema.

Return type str

hed.schema.hed_schema_constants module

class HedKey

Bases: `object`

Known property and attribute names.

Notes

- These names should match the attribute values in the XML/wiki.

AllowedCharacter = 'allowedCharacter'

BoolProperty = 'boolProperty'

DefaultUnits = 'defaultUnits'

ExtensionAllowed = 'extensionAllowed'

Recommended = 'recommended'

RelatedTag = 'relatedTag'

RequireChild = 'requireChild'

Required = 'required'

SIUnit = 'SIUnit'

SIUnitModifier = 'SIUnitModifier'

SIUnitSymbolModifier = 'SIUnitSymbolModifier'

SuggestedTag = 'suggestedTag'

TagGroup = 'tagGroup'

TakesValue = 'takesValue'

TopLevelTagGroup = 'topLevelTagGroup'

Unique = 'unique'

UnitClass = 'unitClass'

UnitClassProperty = 'unitClassProperty'

UnitModifierProperty = 'unitModifierProperty'

UnitPrefix = 'unitPrefix'

UnitProperty = 'unitProperty'

UnitSymbol = 'unitSymbol'

ValueClass = 'valueClass'

ValueClassProperty = 'valueClassProperty'

class HedSectionKey

Bases: object

Kegs designating specific sections in a HedSchema object.

AllTags = 'tags'

Attributes = 'attributes'

Properties = 'properties'

UnitClasses = 'unitClasses'

UnitModifiers = 'unitModifiers'

Units = 'units'

ValueClasses = 'valueClasses'

hed.schema.hed_schema_entry module

class HedSchemaEntry(*name*, *section*)

Bases: object

A single node in a HedSchema.

The structure contains all the node information including attributes and properties.

attribute_has_property(*attribute_name*, *property_name*)

Return True if attribute has property.

Parameters

- **attribute_name** (*str*) – Attribute name to check for property_name.
- **property_name** (*str*) – The property value to return.

Returns Returns True if this entry has the property.

Return type bool

finalize_entry(*schema*)

Called once after loading to set internal state.

Parameters **schema** (HedSchema) – The schema that holds the rules.

has_attribute(*attribute_name*, *return_value=False*)

Return True if this entry has the attribute.

Parameters

- **attribute_name** (*str*) – The attribute to check for.
- **return_value** (*bool*) – If True return the actual attribute value rather than just indicate presence.

Returns If return_value is false, a boolean is returned rather than the actual value.

Return type bool or str

Notes

- A return value of True does not indicate whether or not this attribute is valid

set_attribute_value(*attribute_name*, *attribute_value*)

Add attribute and set its value.

Parameters

- **attribute_name** (*str*) – The name of the schema entry attribute.
- **attribute_value** (*bool or str*) – The value of the attribute.

Notes

- If this an invalid attribute name, it will be also added as an unknown attribute.

class HedTagEntry(**args*, ***kwargs*)

Bases: *hed.schema.hed_schema_entry.HedSchemaEntry*

A single tag entry in the HedSchema.

any_parent_has_attribute(*attribute*)

Check if tag (or parents) has the attribute.

Parameters **attribute** (*str*) – The name of the attribute to check for.

Returns True if the tag has the given attribute. False, if otherwise.

Return type bool

Notes

- This is mostly used to check extension allowed. Could be cached.

base_tag_has_attribute(*tag_attribute*)

Check if the base tag has a specific attribute.

Parameters **tag_attribute** (*str*) – A tag attribute.

Returns True if the tag has the specified attribute. False, if otherwise.

Return type bool

Notes

This mostly is relevant for takes value tags.

finalize_entry(*schema*)

Called once after schema loading to set state.

Parameters **schema** (HedSchema) – The schema that the rules come from.

static get_fake_tag_entry(*tag*, *tags_to_identify*)

Create a tag entry if a given a tag has a match in a list of possible short tags.

Parameters

- **tag** (*str*) – The short/mid/long form tag to identify.
- **tags_to_identify** (*list*) – A list of lowercase short tags to identify.

Returns

- HedTagEntry or None: The fake entry showing the short tag name as the found tag.
- str: The remaining text after the located short tag, which may be empty.

Return type tuple

Notes

- The match is done left to right.

class UnitClassEntry(*args, **kwargs)

Bases: hed.schema.hed_schema_entry.HedSchemaEntry

A single unit class entry in the HedSchema.

add_unit(unit_entry)

Add the given unit entry to this unit class.

Parameters **unit_entry** (HedSchemaEntry) – Unit entry to add.

finalize_entry(schema)

Called once after schema load to set state.

Parameters **schema** (HedSchema) – The object with the schema rules.

class UnitEntry(*args, **kwargs)

Bases: hed.schema.hed_schema_entry.HedSchemaEntry

A single unit entry with modifiers in the HedSchema.

finalize_entry(schema)

Called once after loading to set internal state.

Parameters **schema** (HedSchema) – The schema rules come from.

hed.schema.hed_schema_group module

class HedSchemaGroup(schema_list)

Bases: object

Container for multiple HedSchema objects.

Notes

- The container class is useful when library schema are included.
- You cannot save/load/etc the combined schema object directly.

check_compliance(check_for_warnings=True, name=None, error_handler=None)

Check for hed3 compliance of this schema.

Parameters

- **check_for_warnings** (*bool*) – If True, checks for formatting issues like invalid characters, capitalization.
- **name** (*str*) – If present, use as the filename for context, rather than using the actual filename.
- **error_handler** (*ErrorHandler or None*) – Used to report errors. Uses a default one if none passed in.

Returns A list of all warnings and errors found in the file. Each issue is a dictionary.

Return type list

Notes

- Useful for temp filenames when supporting web services.

find_tag_entry(*tag*, *library_prefix=''*)

Find a schema entry for a source tag.

Parameters

- **tag** (*str or HedTag*) – Any form of tag to look up. Can have an extension, value, etc.
- **library_prefix** (*str*) – The prefix the library, if any.

Returns

- HedTagEntry: The located tag entry for this tag.
- str: The remainder of the tag that isn't part of the base tag.
- list: A list of errors while converting.

Return type tuple

Notes

- Works right to left.(mostly relevant for errors).

get_tag_entry(*name*, *key_class='tags'*, *library_prefix=''*)

Return the schema entry for this tag, if one exists.

Parameters

- **name** (*str*) – Any form of basic tag(or other section entry) to look up.
- **key_class** (HedSectionKey) – The tag section to search.
- **library_prefix** (*str*) – An optional prefix associated with this tag.

Returns The schema entry for the given tag.

Return type *HedSchemaEntry*

Notes

- This will not handle extensions or similar.

get_tags_with_attribute(*key*)

Return the tags with this attribute.

Parameters **key** (*str*) – The attributes.

property has_duplicate_tags

Return True if valid hed3 schema with no duplicate short tags.

Returns True if this is a valid hed3 schema with no duplicate short tags.

Return type bool

property is_hed3_compatible

A list of HED3-compliant schemas in this group.

property is_hed3_schema

HedSchemaGroup objects are always HED3.

schema_for_prefix(*prefix*)

Return the HedSchema for the library prefix.

Parameters **prefix** (*str*) – A schema library name prefix.

Returns The specific schema for this library name prefix if exists.

Return type *HedSchema* or None

property unit_classes

A list of all unit classes represented in this group.

property unit_modifiers

Return a list of all unit modifiers for all schema.

property valid_prefixes

Return a list of all prefixes this group will accept.

Returns A list of strings representing valid prefixes for this group.

Return type list

property value_classes

hed.schema.hed_schema_io module

Utilities for loading and outputting HED schema.

from_string(*schema_string*, *file_type='.xml'*, *library_prefix=None*)

Create a schema from the given string.

Parameters

- **schema_string** (*str*) – An XML or mediawiki file as a single long string.
- **file_type** (*str*) – The extension(including the .) corresponding to a file source.
- **library_prefix** (*str, None*) – The name_prefix all tags in this schema will accept.

Returns The loaded schema.

Return type (*HedSchema*)

Raises **HedFileError** – If empty string or invalid extension is passed.

Notes

- The loading is determined by file type.

get_hed_xml_version(*xml_file_path*)

Get the version number from a HED XML file.

Parameters **xml_file_path** (*str*) – The path to a HED XML file.

Returns The version number of the HED XML file.

Return type str

load_schema(*hed_path=None*, *library_prefix=None*)

Load a schema from the given file or URL path.

Parameters

- **hed_path** (*str or None*) – A filepath or url to open a schema from.
- **library_prefix** (*str or None*) – The name_prefix all tags in this schema will accept.

Returns The loaded schema.

Return type *HedSchema*

Raises **HedFileError** – If there are any fatal issues when loading the schema.

load_schema_version(*xml_folder=None*, *xml_version=None*, *library_name=None*, *library_prefix=None*)

Return specified version or latest if not specified.

Parameters

- **xml_folder** (*str*) – Path to a folder containing schema.
- **xml_version** (*str*) – HED version format string. Expected format: 'X.Y.Z'.
- **library_name** (*str or None*) – Optional library name
- **library_prefix** (*str or None*) – The name_prefix all tags in this schema will accept.

Returns The requested HedSchema object.

Return type *HedSchema*

Notes

- The library schema files have names of the form HED_(LIBRARY_NAME)_(version).xml.

hed.schema.hed_schema_section module

class HedSchemaSection(*section_key*, *case_sensitive=True*)

Bases: object

Container with entries in one section of the schema.

get(*key*)

Return the name associated with key.

Parameters **key** (*str*) – The name of the key.

get_entries_with_attribute(*attribute_name*, *return_name_only=False*, *library_prefix=''*)

Return entries or names with given attribute.

Parameters

- **attribute_name** (*str*) – The name of the attribute(generally a HedKey entry).
- **return_name_only** (*bool*) – If true, return the name as a string rather than the tag entry.
- **library_prefix** (*str*) – Prepends given prefix to each name if returning names.

Returns List of HedSchemaEntry or strings representing the names.

Return type list

items()

Return the items.

keys()

The names of the keys.

values()

All names of the sections.

class HedSchemaTagSection(**args*, *case_sensitive=False*, ***kwargs*)

Bases: *hed.schema.hed_schema_section.HedSchemaSection*

A section of the schema.

get(*key*)

Return the name associated with key.

Parameters **key** (*str*) – The name of the key.

hed.schema.schema_compliance module

Utilities for HED schema checking.

check_compliance(*hed_schema*, *check_for_warnings=True*, *name=None*, *error_handler=None*)

Check for hed3 compliance of a schema object.

Parameters

- **hed_schema** (HedSchema) – HedSchema object to check for hed3 compliance.
- **check_for_warnings** (*bool*) – If True, check for formatting issues like invalid characters, capitalization, etc.
- **name** (*str*) – If present, will use as filename for context.
- **error_handler** (*ErrorHandler or None*) – Used to report errors. Uses a default one if none passed in.

Returns A list of all warnings and errors found in the file. Each issue is a dictionary.

Return type list

Notes

- Useful for temp filenames in support of web services.

tag_exists_check(*hed_schema*, *tag_entry*, *possible_tags*, *force_issues_as_warnings=True*)

Check if comma separated list are valid HedTags.

Parameters

- **hed_schema** (HedSchema) – The schema to check if the tag exists.
- **tag_entry** (HedSchemaEntry) – The schema entry for this tag.
- **possible_tags** (*str*) – Comma separated list of tags. Short long or mixed form valid.
- **force_issues_as_warnings** (*bool*) – If True, set all the severity levels to warning.

Returns A list of issues. Each issue is a dictionary.

Return type list

tag_is_placeholder_check(*hed_schema*, *tag_entry*, *possible_tags*, *force_issues_as_warnings=True*)

Check if comma separated list has valid HedTags.

Parameters

- **hed_schema** (HedSchema) – The schema to check if the tag exists.
- **tag_entry** (HedSchemaEntry) – The schema entry for this tag.
- **possible_tags** (*str*) – Comma separated list of tags. Short long or mixed form valid.
- **force_issues_as_warnings** (*bool*) – If True sets all the severity levels to warning.

Returns A list of issues. Each issue is a dictionary.

Return type list

validate_schema_description(*tag_name*, *hed_description*)

Check the description of a single schema term.

Parameters

- **tag_name** (*str*) – A single hed tag - not validated here, just used for error messages.
- **hed_description** (*str*) – The description string to validate.

Returns A list of all formatting issues found in the description.

Return type list

validate_schema_term(*hed_term*)

Check short tag for capitalization and illegal characters.

Parameters **hed_term** (*str*) – A single hed term.

Returns A list of all formatting issues found in the term. Each issue is a dictionary.

Return type list

hed.schema.schema_validation_util module

Utilities used in HED validation using a HED schema.

is_hed3_version_number(*version_string*)

Check validity of the version.

Parameters **version_string** (*str*) – A version string.

Returns If True the version corresponds to a HED3 schema.

Return type bool

validate_attributes(*attrib_dict*, *filename*)

Validate attributes in the dictionary.

Parameters

- **attrib_dict** (*dict*) – Dictionary of attributes to be evaluated.
- **filename** (*str*) – File name to use in reporting errors.

Returns List of issues. Each issue is a dictionary.

Return type list

Raises **HedFileError** – if invalid or version not found in the dictionary.

Module contents

Data structures for handling the HED schema.

3.2.2 hed.schema.HedSchema

class HedSchema

A HED schema suitable for processing.

__init__()

Constructor for the HedSchema class.

Returns The constructed HED schema.

Return type *HedSchema*

Methods

__init__()	Constructor for the HedSchema class.
check_compliance([check_for_warnings, name, ...])	Check for HED3 compliance of this schema.
finalize_dictionaries()	Call to finish loading.
find_duplicate_tags()	Find all tags that are not unique.
find_tag_entry(tag[, library_prefix])	Find the schema entry for a given source tag.
get_all_schema_tags([return_last_term])	Get a list of all hed terms from the schema.
get_all_tag_attributes(tag_name[, key_class])	Gather all attributes for a given tag name.
get_as_mediawiki_string()	Return the schema to a mediawiki string.
get_as_xml_string()	Return the schema to an XML string.
get_desc_iter()	Return an iterator over all the descriptions.
get_modifiers_for_unit(unit)	Return the valid modifiers for the given unit
get_tag_attribute_names()	Return a dict of all allowed tag attributes.
get_tag_description(tag_name[, key_class])	Return the description associated with the tag.
get_tag_entry(name[, key_class, library_prefix])	Return the schema entry for this tag, if one exists.
get_tags_with_attribute(key[, section_key])	Return tag entries with the given attribute.
get_unit_class_units(unit_class_type)	Get the list of unit class units this type will accept.
get_unknown_attributes()	Retrieve the current list of unknown attributes.
save_as_mediawiki()	Save as mediawiki to a temporary file.
save_as_xml()	Save as XML to a temporary file.
schema_for_prefix(prefix)	Return HedSchema object for this prefix.
set_library_prefix(library_prefix)	Set library prefix associated for this schema.

Attributes

all_tags	Return the tag schema section.
filename	The filename if one is known.
has_duplicate_tags	Return True if this is a valid hed3.
is_hed3_schema	Return true if this is at least version HED3.
library	The name of this library schema if one exists.
unit_classes	Return the unit classes schema section.
unit_modifiers	Return the modifiers classes schema section
valid_prefixes	Return a list of all prefixes this schema will accept
value_classes	Return the value classes schema section.
version	The HED version of this schema.

class HedSchema

Bases: object

A HED schema suitable for processing.

property all_tags

Return the tag schema section.

Returns The tag section.

Return type *HedSchemaSection*

check_compliance(*check_for_warnings=True*, *name=None*, *error_handler=None*)

Check for HED3 compliance of this schema.

Parameters

- **check_for_warnings** (*bool*) – If True, also checks for formatting issues
- **name** (*str*) – If present, use this as the filename for context
- **error_handler** (*ErrorHandler or None*) – Used to report errors.

Returns A list of all warnings and errors found in the file. Each issue is a dictionary.

Return type list

Notes

- Formatting issues include invalid characters and capitalization.
- The name parameter is useful when handling temporary files.
- A default error handler is created if none passed in.

property filename

The filename if one is known.

Returns The filename of this schema.

Return type str

finalize_dictionaries()

Call to finish loading.

find_duplicate_tags()

Find all tags that are not unique.

Returns A dictionary of all duplicate short tags

Return type dict

Notes

- The returned dictionary has the short-form tags as keys and lists

of long tags sharing the short form as the values.

find_tag_entry(*tag*, *library_prefix=''*)

Find the schema entry for a given source tag.

Parameters

- **tag** (*str, HedTag*) – Any form of tag to look up. Can have an extension, value, etc.
- **library_prefix** (*str*) – The prefix the library, if any.

Returns The located tag entry for this tag. str: The remainder of the tag that isn't part of the base tag. list: A list of errors while converting.

Return type *HedTagEntry*

Notes

Works right to left (which is mostly relevant for errors).

get_all_schema_tags(*return_last_term=False*)

Get a list of all hed terms from the schema.

Returns A list of all terms(short tags) from the schema.

Return type list

Notes

Compatible with Hed2 or Hed3.

get_all_tag_attributes(*tag_name*, *key_class='tags'*)

Gather all attributes for a given tag name.

Parameters

- **tag_name** (*str*) – The name of the tag to check.
- **key_class** (*str*) – The type of attributes requested. e.g. Tag, Units, Unit modifiers, or attributes.

Returns A dictionary of attribute name and attribute value.

Return type dict

Notes

If keys is None, gets all normal hed tag attributes.

get_as_mediawiki_string()

Return the schema to a mediawiki string.

Returns The schema as a string in mediawiki format.

Return type str

get_as_xml_string()

Return the schema to an XML string.

Returns Return the schema as an XML string.

Return type str

get_desc_iter()

Return an iterator over all the descriptions.

Yields *tuple* – - str: The tag node name. - str: The description associated with the node.

get_modifiers_for_unit(*unit*)

Return the valid modifiers for the given unit

Parameters **unit** (*str*) – A known unit.

Returns List of HedSchemaEntry.

Return type list

Notes

This is a lower level one that doesn't rely on the Unit entries being fully setup.

get_tag_attribute_names()

Return a dict of all allowed tag attributes.

Returns A dictionary whose keys are attribute names and values are HedSchemaEntry object.

Return type dict

get_tag_description(*tag_name*, *key_class='tags'*)

Return the description associated with the tag.

Parameters

- **tag_name** (*str*) – A hed tag name(or unit/unit modifier etc) with proper capitalization.
- **key_class** (*str*) – A string indicating type of description (e.g. All tags, Units, Unit modifier). The default is HedSectionKey.AllTags.

Returns A description of the specified tag.

Return type str

get_tag_entry(*name*, *key_class='tags'*, *library_prefix=''*)

Return the schema entry for this tag, if one exists.

Parameters

- **name** (*str*) – Any form of basic tag(or other section entry) to look up. This will not handle extensions or similar.

- **key_class** (HedSectionKey *or* *str*) – The type of entry to return.
- **library_prefix** (*str*) – Only used for HedSchemaGroup otherwise ignored.

Returns The schema entry for the given tag.

Return type *HedSchemaEntry*

get_tags_with_attribute(*key*, *section_key='tags'*)

Return tag entries with the given attribute.

Parameters

- **key** (*str*) – A tag attribute. Eg HedKey.ExtensionAllowed
- **section_key** (*str*) – The HedSectionKey for the section to retrieve from.

Returns A list of all tags with this attribute.

Return type list

Notes

- The result is cached so will be fast after first call.

get_unit_class_units(*unit_class_type*)

Get the list of unit class units this type will accept.

Parameters **unit_class_type** (*str*) – The unit class type to check for. e.g. "time".

Returns A list of each UnitEntry this type allows.

Return type list

Examples

Eg 'time' returns ['second', 's', 'day', 'minute', 'hour']

get_unknown_attributes()

Retrieve the current list of unknown attributes.

Returns The keys are attribute names and the values are lists of tags with this attribute.

Return type dict

Notes

This includes attributes found in the wrong section for example unitClass attribute found on a Tag.

The return tag list is in long form.

property has_duplicate_tags

Return True if this is a valid hed3.

Returns True if this is a valid hed3 schema with no duplicate short tags.

Return type bool

property is_hed3_schema

Return true if this is at least version HED3.

Returns True if this is a hed3 schema.

Return type bool

Notes

- This is considered true if the version number is >= 8.0 or it has a library name.

property library

The name of this library schema if one exists.

Returns Library name if any.

Return type str or None

save_as_mediawiki()

Save as mediawiki to a temporary file.

Returns The newly created schema filename.

Return type str

save_as_xml()

Save as XML to a temporary file.

Returns The name of the newly created schema file.

Return type str

schema_for_prefix(*prefix*)

Return HedSchema object for this prefix.

Parameters **prefix** (*str*) – The schema library name prefix.

Returns The HED schema object for this schema.

Return type *HedSchema*

Notes

-This is mostly a placeholder for HedSchemaGroup and may be refactored out later.

set_library_prefix(*library_prefix*)

Set library prefix associated for this schema.

Parameters **library_prefix** (*str*) – Should be empty, or end with a colon.(Colon will be automated added if missing).

property unit_classes

Return the unit classes schema section.

Returns The unit classes section.

Return type *HedSchemaSection*

property unit_modifiers

Return the modifiers classes schema section

Returns The unit modifiers section.

Return type *HedSchemaSection*

property valid_prefixes

Return a list of all prefixes this schema will accept

Returns A list of valid tag prefixes for this schema.

Return type list

Notes

- The return value is always length 1 if using a HedSchema.

property value_classes

Return the value classes schema section.

Returns The value classes section.

Return type *HedSchemaSection*

property version

The HED version of this schema.

Returns The version of this schema.

Return type str

3.2.3 hed.schema.HedSchemaEntry

class HedSchemaEntry(*name*, *section*)

A single node in a HedSchema.

The structure contains all the node information including attributes and properties.

__init__(*name*, *section*)

Constructor for HedSchemaEntry.

Parameters

- **name** (*str*) – The name of the entry.
- **section** (HedSchemaSection) – The section to which it belongs.

Methods

__init__(name, section)	Constructor for HedSchemaEntry.
attribute_has_property(attribute_name, ...)	Return True if attribute has property.
finalize_entry(schema)	Called once after loading to set internal state.
has_attribute(attribute_name[, return_value])	Return True if this entry has the attribute.
set_attribute_value(attribute_name, ...)	Add attribute and set its value.

class HedSchemaEntry(*name*, *section*)

Bases: object

A single node in a HedSchema.

The structure contains all the node information including attributes and properties.

attribute_has_property(*attribute_name*, *property_name*)

Return True if attribute has property.

Parameters

- **attribute_name** (*str*) – Attribute name to check for property_name.
- **property_name** (*str*) – The property value to return.

Returns Returns True if this entry has the property.

Return type bool

finalize_entry(*schema*)

Called once after loading to set internal state.

Parameters **schema** (HedSchema) – The schema that holds the rules.

has_attribute(*attribute_name*, *return_value=False*)

Return True if this entry has the attribute.

Parameters

- **attribute_name** (*str*) – The attribute to check for.
- **return_value** (*bool*) – If True return the actual attribute value rather than just indicate presence.

Returns If return_value is false, a boolean is returned rather than the actual value.

Return type bool or str

Notes

- A return value of True does not indicate whether or not this attribute is valid

set_attribute_value(*attribute_name*, *attribute_value*)

Add attribute and set its value.

Parameters

- **attribute_name** (*str*) – The name of the schema entry attribute.
- **attribute_value** (*bool or str*) – The value of the attribute.

Notes

- If this an invalid attribute name, it will be also added as an unknown attribute.

3.2.4 hed.schema.UnitClassEntry

class UnitClassEntry(**args*, ***kwargs*)

A single unit class entry in the HedSchema.

__init__(**args*, ***kwargs*)

Constructor for HedSchemaEntry.

Parameters

- **name** (*str*) – The name of the entry.
- **section** (HedSchemaSection) – The section to which it belongs.

Methods

__init__(*args, **kwargs)	Constructor for HedSchemaEntry.
add_unit(unit_entry)	Add the given unit entry to this unit class.
attribute_has_property(attribute_name, ...)	Return True if attribute has property.
finalize_entry(schema)	Called once after schema load to set state.
has_attribute(attribute_name[, return_value])	Return True if this entry has the attribute.
set_attribute_value(attribute_name, ...)	Add attribute and set its value.

class UnitClassEntry(**args*, ***kwargs*)

Bases: *hed.schema.hed_schema_entry.HedSchemaEntry*

A single unit class entry in the HedSchema.

add_unit(*unit_entry*)

Add the given unit entry to this unit class.

Parameters **unit_entry** (HedSchemaEntry) – Unit entry to add.

attribute_has_property(*attribute_name*, *property_name*)

Return True if attribute has property.

Parameters

- **attribute_name** (*str*) – Attribute name to check for property_name.
- **property_name** (*str*) – The property value to return.

Returns Returns True if this entry has the property.

Return type bool

finalize_entry(*schema*)

Called once after schema load to set state.

Parameters **schema** (HedSchema) – The object with the schema rules.

has_attribute(*attribute_name*, *return_value=False*)

Return True if this entry has the attribute.

Parameters

- **attribute_name** (*str*) – The attribute to check for.
- **return_value** (*bool*) – If True return the actual attribute value rather than just indicate presence.

Returns If return_value is false, a boolean is returned rather than the actual value.

Return type bool or str

Notes

- A return value of True does not indicate whether or not this attribute is valid

set_attribute_value(*attribute_name*, *attribute_value*)

Add attribute and set its value.

Parameters

- **attribute_name** (*str*) – The name of the schema entry attribute.
- **attribute_value** (*bool or str*) – The value of the attribute.

Notes

- If this an invalid attribute name, it will be also added as an unknown attribute.

3.2.5 hed.schema.UnitEntry

class UnitEntry(**args*, ***kwargs*)

A single unit entry with modifiers in the HedSchema.

__init__(**args*, ***kwargs*)

Constructor for HedSchemaEntry.

Parameters

- **name** (*str*) – The name of the entry.
- **section** (HedSchemaSection) – The section to which it belongs.

Methods

__init__(*args, **kwargs)	Constructor for HedSchemaEntry.
attribute_has_property(attribute_name, ...)	Return True if attribute has property.
finalize_entry(schema)	Called once after loading to set internal state.
has_attribute(attribute_name[, return_value])	Return True if this entry has the attribute.
set_attribute_value(attribute_name, ...)	Add attribute and set its value.

class UnitEntry(*args, **kwargs)

Bases: hed.schema.hed_schema_entry.HedSchemaEntry

A single unit entry with modifiers in the HedSchema.

attribute_has_property(*attribute_name*, *property_name*)

Return True if attribute has property.

Parameters

- **attribute_name** (*str*) – Attribute name to check for property_name.
- **property_name** (*str*) – The property value to return.

Returns Returns True if this entry has the property.

Return type bool

finalize_entry(*schema*)

Called once after loading to set internal state.

Parameters **schema** (HedSchema) – The schema rules come from.

has_attribute(*attribute_name*, *return_value=False*)

Return True if this entry has the attribute.

Parameters

- **attribute_name** (*str*) – The attribute to check for.
- **return_value** (*bool*) – If True return the actual attribute value rather than just indicate presence.

Returns If return_value is false, a boolean is returned rather than the actual value.

Return type bool or str

Notes

- A return value of True does not indicate whether or not this attribute is valid

set_attribute_value(*attribute_name*, *attribute_value*)

Add attribute and set its value.

Parameters

- **attribute_name** (*str*) – The name of the schema entry attribute.
- **attribute_value** (*bool or str*) – The value of the attribute.

Notes

- If this an invalid attribute name, it will be also added as an unknown attribute.

3.2.6 hed.schema.HedTagEntry

class HedTagEntry(**args*, ***kwargs*)

A single tag entry in the HedSchema.

__init__(**args*, ***kwargs*)

Constructor for HedSchemaEntry.

Parameters

- **name** (*str*) – The name of the entry.
- **section** (HedSchemaSection) – The section to which it belongs.

Methods

__init__(*args, **kwargs)	Constructor for HedSchemaEntry.
any_parent_has_attribute(attribute)	Check if tag (or parents) has the attribute.
attribute_has_property(attribute_name, ...)	Return True if attribute has property.
base_tag_has_attribute(tag_attribute)	Check if the base tag has a specific attribute.
finalize_entry(schema)	Called once after schema loading to set state.
get_fake_tag_entry(tag, tags_to_identify)	Create a tag entry if a given a tag has a match in a list of possible short tags.
has_attribute(attribute_name[, return_value])	Return True if this entry has the attribute.
set_attribute_value(attribute_name, ...)	Add attribute and set its value.

class HedTagEntry(**args*, ***kwargs*)

Bases: *hed.schema.hed_schema_entry.HedSchemaEntry*

A single tag entry in the HedSchema.

any_parent_has_attribute(*attribute*)

Check if tag (or parents) has the attribute.

Parameters **attribute** (*str*) – The name of the attribute to check for.

Returns True if the tag has the given attribute. False, if otherwise.

Return type bool

Notes

- This is mostly used to check extension allowed. Could be cached.

attribute_has_property(*attribute_name*, *property_name*)

Return True if attribute has property.

Parameters

- **attribute_name** (*str*) – Attribute name to check for property_name.
- **property_name** (*str*) – The property value to return.

Returns Returns True if this entry has the property.

Return type bool

base_tag_has_attribute(*tag_attribute*)

Check if the base tag has a specific attribute.

Parameters **tag_attribute** (*str*) – A tag attribute.

Returns True if the tag has the specified attribute. False, if otherwise.

Return type bool

Notes

This mostly is relevant for takes value tags.

finalize_entry(*schema*)

Called once after schema loading to set state.

Parameters **schema** (HedSchema) – The schema that the rules come from.

static get_fake_tag_entry(*tag*, *tags_to_identify*)

Create a tag entry if a given a tag has a match in a list of possible short tags.

Parameters

- **tag** (*str*) – The short/mid/long form tag to identify.
- **tags_to_identify** (*list*) – A list of lowercase short tags to identify.

Returns

- HedTagEntry or None: The fake entry showing the short tag name as the found tag.
- str: The remaining text after the located short tag, which may be empty.

Return type tuple

Notes

- The match is done left to right.

has_attribute(*attribute_name*, *return_value=False*)

Return True if this entry has the attribute.

Parameters

- **attribute_name** (*str*) – The attribute to check for.
- **return_value** (*bool*) – If True return the actual attribute value rather than just indicate presence.

Returns If return_value is false, a boolean is returned rather than the actual value.

Return type bool or str

Notes

- A return value of True does not indicate whether or not this attribute is valid

set_attribute_value(*attribute_name*, *attribute_value*)

Add attribute and set its value.

Parameters

- **attribute_name** (*str*) – The name of the schema entry attribute.
- **attribute_value** (*bool or str*) – The value of the attribute.

Notes

- If this an invalid attribute name, it will be also added as an unknown attribute.

3.2.7 hed.schema.HedSchemaGroup

class HedSchemaGroup(*schema_list*)

Container for multiple HedSchema objects.

Notes

- The container class is useful when library schema are included.
- You cannot save/load/etc the combined schema object directly.

__init__(*schema_list*)

Combine multiple HedSchema objects from a list.

Parameters **schema_list** (*list*) – A list of HedSchema for the container.

Returns the container created.

Return type *HedSchemaGroup*

Raises **HedFileError** – If multiple schemas have the same library prefixes.

Methods

__init__(schema_list)	Combine multiple HedSchema objects from a list.
check_compliance([check_for_warnings, name, ...])	Check for hed3 compliance of this schema.
find_tag_entry(tag[, library_prefix])	Find a schema entry for a source tag.
get_tag_entry(name[, key_class, library_prefix])	Return the schema entry for this tag, if one exists.
get_tags_with_attribute(key)	Return the tags with this attribute.
schema_for_prefix(prefix)	Return the HedSchema for the library prefix.

Attributes

has_duplicate_tags	Return True if valid hed3 schema with no duplicate short tags.
is_hed3_compatible	A list of HED3-compliant schemas in this group.
is_hed3_schema	HedSchemaGroup objects are always HED3.
unit_classes	A list of all unit classes represented in this group.
unit_modifiers	Return a list of all unit modifiers for all schema.
valid_prefixes	Return a list of all prefixes this group will accept.
value_classes	

class HedSchemaGroup(*schema_list*)

Bases: `object`

Container for multiple HedSchema objects.

Notes

- The container class is useful when library schema are included.
- You cannot save/load/etc the combined schema object directly.

check_compliance(*check_for_warnings=True*, *name=None*, *error_handler=None*)

Check for hed3 compliance of this schema.

Parameters

- **check_for_warnings** (*bool*) – If True, checks for formatting issues like invalid characters, capitalization.
- **name** (*str*) – If present, use as the filename for context, rather than using the actual filename.
- **error_handler** (*ErrorHandler or None*) – Used to report errors. Uses a default one if none passed in.

Returns A list of all warnings and errors found in the file. Each issue is a dictionary.

Return type list

Notes

- Useful for temp filenames when supporting web services.

find_tag_entry(*tag*, *library_prefix=''*)

Find a schema entry for a source tag.

Parameters

- **tag** (*str or HedTag*) – Any form of tag to look up. Can have an extension, value, etc.
- **library_prefix** (*str*) – The prefix the library, if any.

Returns

- HedTagEntry: The located tag entry for this tag.

- str: The remainder of the tag that isn't part of the base tag.
- list: A list of errors while converting.

Return type tuple

Notes

- Works right to left.(mostly relevant for errors).

get_tag_entry(*name*, *key_class='tags'*, *library_prefix=''*)

Return the schema entry for this tag, if one exists.

Parameters

- **name** (*str*) – Any form of basic tag(or other section entry) to look up.
- **key_class** (HedSectionKey) – The tag section to search.
- **library_prefix** (*str*) – An optional prefix associated with this tag.

Returns The schema entry for the given tag.

Return type *HedSchemaEntry*

Notes

- This will not handle extensions or similar.

get_tags_with_attribute(*key*)

Return the tags with this attribute.

Parameters **key** (*str*) – The attributes.

property has_duplicate_tags

Return True if valid hed3 schema with no duplicate short tags.

Returns True if this is a valid hed3 schema with no duplicate short tags.

Return type bool

property is_hed3_compatible

A list of HED3-compliant schemas in this group.

property is_hed3_schema

HedSchemaGroup objects are always HED3.

schema_for_prefix(*prefix*)

Return the HedSchema for the library prefix.

Parameters **prefix** (*str*) – A schema library name prefix.

Returns The specific schema for this library name prefix if exists.

Return type *HedSchema* or None

property unit_classes

A list of all unit classes represented in this group.

property unit_modifiers

Return a list of all unit modifiers for all schema.

property valid_prefixes

Return a list of all prefixes this group will accept.

Returns A list of strings representing valid prefixes for this group.

Return type list

property value_classes

3.2.8 hed.schema.HedSchemaSection

class HedSchemaSection(*section_key*, *case_sensitive=True*)

Container with entries in one section of the schema.

__init__(*section_key*, *case_sensitive=True*)

Construct schema section.

Parameters

- **section_key** (*str*) – Name of the schema section.
- **case_sensitive** (*bool*) – If True, names are case sensitive.

Methods

__init__(section_key[, case_sensitive])	Construct schema section.
get(key)	Return the name associated with key.
get_entries_with_attribute(attribute_name[, ...])	Return entries or names with given attribute.
items()	Return the items.
keys()	The names of the keys.
values()	All names of the sections.

class HedSchemaSection(*section_key*, *case_sensitive=True*)

Bases: object

Container with entries in one section of the schema.

get(*key*)

Return the name associated with key.

Parameters **key** (*str*) – The name of the key.

get_entries_with_attribute(*attribute_name*, *return_name_only=False*, *library_prefix=''*)

Return entries or names with given attribute.

Parameters

- **attribute_name** (*str*) – The name of the attribute(generally a HedKey entry).
- **return_name_only** (*bool*) – If true, return the name as a string rather than the tag entry.
- **library_prefix** (*str*) – Prepends given prefix to each name if returning names.

Returns List of HedSchemaEntry or strings representing the names.

Return type list

items()

Return the items.

keys()

The names of the keys.

values()

All names of the sections.

3.2.9 hed.schema.hed_cache

Infrastructure for caching HED schema.

Functions

cache_specific_url(hed_xml_url[, ...])	Cache a file from a URL.
cache_xml_versions([hed_base_urls, ...])	Cache a file from a URL.
get_cache_directory()	Return the current value of HED_CACHE_DIRECTORY.
get_hed_version_path([xml_version, ...])	Get latest HED XML file path in a directory.
get_hed_versions([local_hed_directory, ...])	Get the HED versions in the hed directory.
get_path_from_hed_version(hed_version[, ...])	Return the HED XML file path for a version.
set_cache_directory(new_cache_dir)	Set default global hed cache directory.

3.2.10 hed.schema.hed_schema_io

Utilities for loading and outputting HED schema.

Functions

from_string(schema_string[, file_type, ...])	Create a schema from the given string.
get_hed_xml_version(xml_file_path)	Get the version number from a HED XML file.
load_schema([hed_path, library_prefix])	Load a schema from the given file or URL path.
load_schema_version([xml_folder, ...])	Return specified version or latest if not specified.

3.2.11 hed.schema.schema_compliance

Utilities for HED schema checking.

Functions

check_compliance(hed_schema[, ...])	Check for hed3 compliance of a schema object.
tag_exists_check(hed_schema, tag_entry, ...)	Check if comma separated list are valid HedTags.
tag_is_placeholder_check(hed_schema, ...[, ...])	Check if comma separated list has valid HedTags.
validate_schema_description(tag_name, ...)	Check the description of a single schema term.
validate_schema_term(hed_term)	Check short tag for capitalization and illegal characters.

3.2.12 hed.schema.schema_validation_util

Utilities used in HED validation using a HED schema.

Functions

is_hed3_version_number(version_string)	Check validity of the version.
validate_attributes(attrib_dict, filename)	Validate attributes in the dictionary.

3.3 HED tools

hed.tools	HED tools for analysis and summarization.
hed.tools.BidsDataset(root_path[, schema, ...])	A BIDS dataset primarily focused on HED evaluation.
hed.tools.BidsDatasetSummary(dataset)	Summary of a BIDS dataset events and other info.
hed.tools.BidsFile(file_path)	A BIDS file with entity dictionary.
hed.tools.BidsFileDictionary(...[, entities])	A dictionary of BidsFile keyed by entity pairs.
hed.tools.BidsFileGroup(root_path[, suffix, ...])	Container for BIDS files with a specified suffix.
hed.tools.BidsSidecarFile(file_path)	A BIDS sidecar file.
hed.tools.BidsTabularDictionary(...[, entities])	A key tabular-file dictionary for tabular files.
hed.tools.BidsTabularFile(file_path)	A BIDS tabular file including its associated sidecar.
hed.tools.BidsTabularSummary([value_cols, ...])	Summarize the contents of BIDS tabular files.
hed.tools.FileDictionary(collection_name, ...)	A file dictionary keyed by entity pair indices.
hed.tools.KeyMap(key_cols[, target_cols, name])	A map of unique column values for remapping columns.
hed.tools.TagSummary(file_group, schema[, ...])	A HED tag summary for a BID file group.
hed.tools.analysis.analysis_util	Utilities for downstream analysis such as searching.
hed.tools.analysis.annotation_util	Utilities to facilitate annotation of events in BIDS.
hed.tools.analysis.summary_util	Utilities used in computing dataset annotation.
hed.tools.HedLogger([name])	Log status messages organized by key.

3.3.1 hed.tools package

Subpackages

hed.tools.analysis package

Submodules

hed.tools.analysis.analysis_util module

Utilities for downstream analysis such as searching.

assemble_hed(*data_input*, *columns_included=None*, *expand_defs=False*)

Return assembled HED annotations in a dataframe.

Parameters

- **data_input** (TabularInput) – The tabular input file whose HED annotations are to be assembled.
- **columns_included** (*list or None*) – A list of additional column names to include. If None, only the list of assembled tags is included.
- **expand_defs** (*bool*) – If True, definitions are expanded when the events are assembled.

Returns A DataFrame with the assembled events.

Return type DataFrame or None

get_assembled_strings(*table*, *hed_schema=None*, *expand_defs=False*)

Return HED string objects for a tabular file.

Parameters

- **table** (TabularInput) – The input file to be searched.
- **hed_schema** (HedSchema *or HedschemaGroup*) – If provided the HedStrings are converted to canonical form.
- **expand_defs** (*bool*) – If True, definitions are expanded when the events are assembled.

Returns A list of HedString or HedStringGroup objects.

Return type List

search_tabular(*data_input*, *hed_schema*, *query*, *columns_included=None*)

Return a dataframe with results of query.

Parameters

- **data_input** (TabularInput) – The tabular input file (e.g., events) to be searched.
- **hed_schema** (HedSchema *or* HedSchemaGroup) – The schema(s) under which to make the query.
- **query** (*str*) – The str query to make.
- **columns_included** (*list or None*) – List of names of columns to include

Returns A DataFrame with the results of the query or None if no events satisfied the query.

Return type DataFrame or None

hed.tools.analysis.annotation_util module

Utilities to facilitate annotation of events in BIDS.

check_df_columns(*df*, *required_cols=('column_name', 'column_value', 'description', 'HED')*)

Return a list of the specified columns that are missing from a dataframe.

Parameters

- **df** (*DataFrame*) – Spreadsheet to check the columns of.
- **required_cols** (*tuple*) – List of column names that must be present.

Returns List of column names that are missing.

Return type list

df_to_hed(*dataframe*, *description_tag=True*)

Create sidecar-like dictionary from a 4-column dataframe.

Parameters

- **dataframe** (*DataFrame*) – A four-column Pandas DataFrame with specific columns.
- **description_tag** (*bool*) – If True description tag is included.

Returns A dictionary compatible compatible with BIDS JSON tabular file that includes HED.

Return type dict

Notes

- The DataFrame must have the columns with names: column_name, column_value, description, and HED.

extract_tags(*hed_string*, *search_tag*)

Extract all instances of specified tag from a tag_string.

Parameters

- **hed_string** (*str*) – Tag string from which to extract tag.
- **search_tag** (*str*) – HED tag to extract.

Returns

- str: Tag string without the tags.
- list: A list of the tags that were extracted, for example descriptions.

Return type tuple

generate_sidecar_entry(*column_name*, *column_values=None*)

Create a sidecar column dictionary for column.

Parameters

- **column_name** (*str*) – Name of the column.
- **column_values** – List of column values.

hed_to_df(*sidecar_dict*, *col_names=None*)

Return a 4-column dataframe of HED portions of sidecar.

Parameters

- **sidecar_dict** (*dict*) – A dictionary conforming to BIDS JSON events sidecar format.
- **col_names** (*list, None*) – A list of the cols to include in the flattened side car.

Returns Four-column spreadsheet representing HED portion of sidecar.

Return type DataFrame

Notes

- The returned DataFrame has columns: column_name, column_value, description, and HED.

merge_hed_dict(*sidecar_dict*, *hed_dict*)

Update a JSON sidecar based on the hed_dict values.

Parameters

- **sidecar_dict** (*dict*) – Dictionary representation of a BIDS JSON sidecar.
- **hed_dict** (*dict*) – Dictionary derived from a dataframe representation of HED in sidecar.

trim_back(*tag_string*)

Return a trimmed copy of tag_string.

Parameters **tag_string** (*str*) – A tag string to be trimmed.

Returns A copy of tag_string that has been trimmed.

Return type str

Notes

- The trailing blanks and commas are removed from the copy.

trim_front(*tag_string*)

Return a copy of tag_string with leading blanks and commas removed.

Parameters **tag_string** (*str*) – A tag string to be trimmed.

Returns A copy of tag_string that has been trimmed.

Return type str

hed.tools.analysis.file_dictionary module

class FileDictionary(*collection_name*, *file_list*, *key_indices=(0, 2)*, *separator='_'*)

Bases: object

A file dictionary keyed by entity pair indices.

Notes

- The entities are identified as 0, 1, ... depending on order in the base filename.
- The entity key-value pairs are assumed separated by '_' unless a separator is provided.

create_file_dict(*file_list*, *key_indices*, *separator*)

Create new dict based on key indices.

Parameters

- **file_list** (*list*) – Paths of the files to include.
- **key_indices** (*tuple*) – A tuple of integers representing order of entities for key.
- **separator** (*str*) – The separator used between entities to form the key.

property file_list

List of path values in this dictionary.

get_file_path(*key*)

Return file path corresponding to key.

Parameters **key** (*str*) – Key used to retrieve the file path.

Returns File path.

Return type str

iter_files()

Iterator over the files in this dictionary.

Yields - *str* – Key into the dictionary. - file: File path.

key_diffs(*other_dict*)

Return symmetric key difference with other.

Parameters **other_dict** (FileDictionary) –

Returns The symmetric difference of the keys in this dictionary and the other one.

Return type list

property key_list

Keys in this dictionary.

static make_file_dict(*file_list*, *key_indices=(0, 2)*, *separator='_'*)

Return a dictionary of files using entity keys.

Parameters

- **file_list** (*list*) – Paths to files to use.
- **key_indices** (*tuple*) – Positions of entities to use for key.
- **separator** (*str*) – Separator character used to construct key.

static make_key(*key_string*, *indices=(0, 2)*, *separator='_'*)

Create a key from specified entities.

Parameters

- **key_string** (*str*) – The string from which to extract the key (usually a filename or path).
- **indices** (*tuple*) – Positions of entity pairs to use as key.

- **separator** (*str*) – Separator between entity pairs in the created key.

Returns The created key.

Return type str

property name

Name of this dictionary

output_files(*title=None*, *logger=None*)

Return a string with the output of the list.

Parameters

- **title** (*None, str*) –
- **logger** (HedLogger) –

Returns The dictionary in string form.

Return type str

Notes

- The logger is updated if available.

hed.tools.analysis.key_map module

class KeyMap(*key_cols*, *target_cols=None*, *name=''*)

Bases: object

A map of unique column values for remapping columns.

name

Name of this remap for identification purposes.

Type str

key_cols

A list of column names that will be hashed into the keys for the map.

Type list

target_cols

An optional list of column names that will be inserted into data and later remapped.

Type list

property columns

make_template(*additional_cols=None*)

Return a dataframe template.

Parameters **additional_cols** (*list or None*) – Optional list of additional columns to append to the returned dataframe.

Returns A dataframe containing the template.

Return type DataFrame

Raises **HedFileError** – If additional columns are not disjoint from the key columns.

Notes

- The template consists of the unique key columns in this map plus additional columns.

remap(*data*)

Remap the columns of a dataframe or columnar file.

Parameters **data** (*DataFrame, str*) – Columnar data (either DataFrame or filename) whose columns are to be remapped.

Returns

- DataFrame: New dataframe with columns remapped.
- list: List of row numbers that had no correspondence in the mapping.

Return type tuple

Raises **HedFileError** – If data is missing some of the key columns.

resort()

Sort the col_map in place by the key columns.

update(*data*, *allow_missing=True*, *keep_counts=True*)

Update the existing map with information from data.

Parameters

- **data** (*DataFrame or str*) – DataFrame or filename of an events file or event map.
- **allow_missing** (*bool*) – If true allow missing keys and add as n/a columns.
- **keep_counts** (*bool*) – If true keep a count of the times each key is present.

Returns The indices of duplicates.

Return type list

Raises **HedFileError** – If there are missing keys and allow_missing is False.

hed.tools.analysis.summary_util module

Utilities used in computing dataset annotation.

add_tag_list_to_dict(*tag_list*, *tag_dict*, *hed_schema=None*)

Convert tags and groups to a dictionary.

Parameters

- **tag_list** (*list*) – List of HedTag and HedGroup objects to transform.
- **tag_dict** (*dict*) –
- **hed_schema** (*HedSchema or HedSchemaGroup*) – Hed schema to use to convert tags to canonical form. If None, assumes already converted

Returns Dictionary of tags as keys with a dict of values as the value.

Return type dict

Notes

- The returned dictionary has short tag as key and dict of values as the value.

breakout_tags(*schema*, *tag_list*, *breakout_list*)

Create a dictionary with tags split into groups.

Parameters

- **schema** (HedSchema, *HedSchemas*) – Schemas to use to break out the tags.
- **tag_list** (*list, dict*) – Iterable of tags to be broken out.
- **breakout_list** (*list*) – List of hed tag node strings that should be summarized separately.

Returns

Dictionary where the keys are the tags from tag_list and all of their parents. The values are each a list of all of the tags from the breakout_list and their parents.

Return type dict

Notes

- The tags that aren't in the breakout list appear in the returned dictionary under "leftovers".

extract_dict_values(*tag_dict*, *tag_name*, *tags*)

Get the tags associated with tag name from the tag dictionary.

Parameters

- **tag_dict** (*dict*) – Dictionary with keys that a tag node names and values are dictionaries.
- **tag_name** (*str*) – Name of a node.
- **tags** (*list*) – List of tags left to process.

Returns

- list: Tags associated with tag_name.
- bool: True if tag_name was found in tag_dict.

Return type tuple

Notes

- Side-effect the tags list is modified.

get_schema_entries(*hed_schema*, *tag*, *library_prefix=''*)

Get schema entries for tag and to its parents.

Parameters

- **hed_schema** (HedSchema, HedSchemaGroup) – The schema in which to search for tag.
- **tag** (HedTag, *str*) – The HED tag to look for.
- **library_prefix** (*str*) – The library prefix to use in the search.

Returns A list of HedTagEntry objects for the tag and its parent nodes in the schema.

Return type list

`unfold_tag_list`(*tag_list*)

Unfold a list to a single-level list of HedTags.

Parameters `tag_list` (*list*) – List of HedTags and HedGroup objects.

Returns A list of HedTags.

Return type list

hed.tools.analysis.tabular_reports module

`report_diffs`(*tsv_dict1*, *tsv_dict2*, *logger*)

Reports and logs the contents and differences of two equivalent BidsTabularDictionary objects

Parameters

- **tsv_dict1** (BidsTabularDictionary) – A dictionary representing BIDS-keyed tsv files.
- **tsv_dict2** (BidsTabularDictionary) – A dictionary representing BIDS-keyed tsv files.
- **logger** (HedLogger) – A HedLogger object for reporting the values by key.

Returns A string with the differences.

Return type str

hed.tools.analysis.tag_summary module

`class TagSummary`(*file_group*, *schema*, *breakout_list=None*)

Bases: `object`

A HED tag summary for a BID file group.

`file_group`

The BIDS file group to be summarized.

Type *BidsFileGroup*

`schema (HedSchema or Hed`

`all_tags_dict`

The keys are all of the unique tags in the file group. The values are dictionaries of the unique values that these tags take on.

Type dict

`breakout_list`

The tag nodes that are to be specially summarized.

Type list

`breakout_dict`

The keys are the breakout nodes. The values are dictionaries of the child nodes and the nodes themselves that appear in the dataset.

Type dict

task_dict

The keys are definition names and the values are dictionaries with info.

Type dict

cond_dict

The keys are definition names and the values are dictionaries with info.

Type dict

static extract_summary_info(*entry_dict*, *tag_name*)

Extract the summary of tag in this entry.

Parameters

- **entry_dict** (*dict*) – Keys are individual tag node names.
- **tag_name** (*str*) – Name of an individual node.

Returns A dictionary of the extracted tag information.

Return type dict

get_design_matrices()

Return information about the condition variables.

Returns

- dict: Dictionary with condition variable levels corresponding to a design matrix.
- list: List with the other condition variables that aren't associated with levels.
- list: List of errors.

Return type tuple

Module contents

hed.tools.bids package

Submodules

hed.tools.bids.bids_dataset module

class BidsDataset(*root_path*, *schema=None*, *tabular_types=None*)

Bases: object

A BIDS dataset primarily focused on HED evaluation.

root_path

Real root path of the BIDS dataset.

Type str

schema

The schema used for evaluation.

Type *HedSchema* or *HedSchemaGroup*

tabular_files

A dictionary of BidsTabularDictionary objects containing a given type.

Type dict

get_schema_versions()

Return the schema versions used in this dataset.

Returns List of schema versions used in this dataset.

Return type list

get_summary()

Return an abbreviated summary of the dataset.

get_tabular_group(*obj_type='events'*)

Return the specified tabular file group.

Parameters **obj_type** (*str*) – Suffix of the BidsFileGroup to be returned.

Returns The requested tabular group.

Return type *BidsFileGroup* or None

validate(*types=None*, *check_for_warnings=True*)

Validate the specified file group types.

Parameters

- **types** (*list*) – A list of strings indicating the file group types to be validated.
- **check_for_warnings** (*bool*) – If True, check for warnings.

Returns List of issues encountered during validation. Each issue is a dictionary.

Return type list

hed.tools.bids.bids_dataset_summary module

class BidsDatasetSummary(*dataset*)

Bases: object

Summary of a BIDS dataset events and other info.

hed.tools.bids.bids_file module

class BidsFile(*file_path*)

Bases: object

A BIDS file with entity dictionary.

file_path

Real path of the file.

Type str

suffix

Suffix part of the filename.

Type str

ext

Extension (including the .).

Type str

entity_dict

Dictionary of entity-names (keys) and entity-values (values).

Type dict

sidecar

Merged sidecar for this file.

Type *BidsSidecarFile*

contents

Contents of this file

Notes

- This class may hold the merged sidecar giving metadata for this file as well as contents.

clear_contents()

Set the contents attribute of this object to None.

property get_contents

Return the current contents of this object.

get_key(*entities*)

Return a key for this BIDS file given a list of entities.

Parameters **entities** (*tuple*) – A tuple of strings representing entities.

Returns A key based on this object.

Return type str

set_contents(*content_info=None*, *overwrite=False*)

Set the contents of this object.

Parameters

- **content_info** – The contents appropriate for this object.
- **overwrite** (*bool*) – If False and the contents are not empty, do nothing.

Notes

- Do not set if the contents are already set and no_overwrite is True.

hed.tools.bids.bids_file_dictionary module

class BidsFileDictionary(*collection_name*, *files*, *entities=('sub', 'ses', 'task', 'run')*)

Bases: *hed.tools.analysis.file_dictionary.FileDictionary*

A dictionary of BidsFile keyed by entity pairs.

The keys are simplified entity key-value pairs and the values are BidsFile objects.

correct_file(*the_file*)

Transform to BidsFile if needed.

Parameters **the_file** (*str or* BidsFile) – If a str, create a new BidsFile object, otherwise pass the original on.

Returns Either the original file or a newly created BidsTabularFile.

Return type *BidsFile*

Raises **HedFileError** – If the_file isn't str or BidsFile.

property file_list

Files contained in this dictionary.

get_file_path(*key*)

Return the file path corresponding to key.

Parameters **key** (*str*) – The key to use to look up the file in this dictionary.

Returns The real path of the file being looked up.

Return type str

Notes

- None is returned if the key is not present.

get_new_dict(*name*, *files*)

Create a dictionary with these files.

Parameters

- **name** (*str*) – Name of this dictionary
- **files** (*list or dict*) – List or dictionary of files. These could be paths or objects.

Returns The newly created dictionary.

Return type *BidsFileDictionary*

Notes

- The new dictionary uses the same type of entities for keys as this dictionary.

iter_files()

Iterator over the files in this dictionary.

Yields *tuple* – - str: The next entity-based key. - BidsFile: The next BidsFile.

key_diffs(*other_dict*)

Return the symmetric key difference with other.

Parameters **other_dict** (FileDictionary) –

Returns The symmetric difference of the keys in this dictionary and the other one.

Return type list

property key_list

The dictionary keys.

make_dict(*files*, *entities*)

Make a dictionary from files or a dict.

Parameters

- **files** (*list or dict*) – List or dictionary of file-like objs to use.
- **entities** (*tuple*) – Tuple of entity names to use as keys, e.g. ('sub', 'run').

Returns A dictionary whose keys are entity keys and values are BidsFile objects.

Return type dict

Raises **HedFileError** – If incorrect format is passed or something not recognizable as a Bids file.

make_query(*query_dict={'sub': '*'}*)

Return a dictionary of files matching query.

Parameters **query_dict** (*dict*) – A dictionary whose keys are entities and whose values are entity values to match.

Returns A dictionary entries in this dictionary that match the query.

Return type dict

Notes

- A query dictionary key a valid BIDS entity name such as sub or task.
- A query dictionary value may be a string or a list.
- A query value string should contain a specific value of the entity or a '*' indicating any value matches.
- A query value list should be a list of valid values for the corresponding entity.

static match_query(*query_dict*, *entity_dict*)

Return True if query has a match in dictionary.

Parameters

- **query_dict** (*dict*) – A dictionary representing a query about entities.

- **entity_dict** (*dict*) – A dictionary containing the entity representation for a BIDS file.

Returns True if the query matches the entities representing the file.

Return type bool

Notes

- A query is a dictionary whose keys are entity names and whose values are specific entity values or '*'.

Examples

{'sub', '001', 'run', '*'} requests all runs from subject 001.

split_by_entity(*entity*)

Split this dictionary based on an entity.

Parameters **entity** (*str*) – Entity name (for example task).

Returns

- dict: A dictionary unique values of entity as keys and BidsFileDictionary objs as values.
- dict: A BidsFileDictionary containing the files that don't have entity in their names.

Return type tuple

Notes

- This function is used for analysis where a single subject or single type of task is being analyzed.

hed.tools.bids.bids_file_group module

class BidsFileGroup(*root_path*, *suffix='_events'*, *obj_type='tabular'*)

Bases: object

Container for BIDS files with a specified suffix.

root_path

Real root path of the Bids dataset.

Type str

suffix

The file suffix specifying the class of file represented in this group (e.g., events).

Type str

obj_type

Type of file in this group (e.g., Tabular or Timeseries).

Type str

sidecar_dict

A dictionary of sidecars associated with this suffix .

Type dict

datafile_dict

A dictionary with values either BidsTabularFile or BidsTimeseriesFile.

Type dict

sidecar_dir_dict

Dictionary whose keys are directory paths and values are list of sidecars in the corresponding directory.

Type dict

get_sidecars_from_path(*obj*)

Return applicable sidecars for the object.

Parameters **obj** (*BidsTabularFile or BidsSidecarFile*) – The BIDS file object to get the sidecars for.

Returns A list of the applicable sidecars for obj starting at the root.

Return type list

summarize(*value_cols=None*, *skip_cols=None*)

Return a BidsTabularSummary of group files.

Parameters

- **value_cols** (*list*) – Column names designated as value columns.
- **skip_cols** (*list*) – Column names designated as columns to skip.

Returns A summary of the number of values in different columns if tabular group.

Return type *BidsTabularSummary* or None

Notes

- The columns that are not value_cols or skip_col are summarized by counting

the number of times each unique value appears in that column.

validate_datafiles(*hed_ops*, *check_for_warnings=True*, *keep_contents=False*)

Validate the datafiles and return an error list.

Parameters

- **hed_ops** (*[func or HedOps], func, HedOps*) – Validation functions to apply.
- **check_for_warnings** (*bool*) – If True, include warnings in the check.
- **keep_contents** (*bool*) – If True, the underlying data files are read and their contents retained.

Returns A list of validation issues found. Each issue is a dictionary.

Return type list

validate_sidecars(*hed_ops*, *check_for_warnings=True*)

Validate merged sidecars.

Parameters

- **hed_ops** (*[func or HedOps], func, HedOps*) – Validation functions to apply.
- **check_for_warnings** (*bool*) – If True, include warnings in the check.

Returns A list of validation issues found. Each issue is a dictionary.

Return type list

hed.tools.bids.bids_sidecar_file module

class BidsSidecarFile(*file_path*)

Bases: *hed.tools.bids.bids_file.BidsFile*

A BIDS sidecar file.

static get_merged(*file_list*)

Return merged contents of JSON files as dict.

Parameters **file_list** (*list or None*) – A list of JSON files representing sidecars in the order they are to be merged.

Returns A merged JSON dictionary.

Return type dict

Notes

- Merging takes place from front to back with overwriting of top-level keys.

static is_hed(*json_dict*)

Return True if the json has HED.

Parameters **json_dict** (*dict*) – A dictionary representing a JSON file or merged file.

Returns True if the dictionary has HED or HED_assembled as a first or second-level key.

Return type bool

is_sidecar_for(*obj*)

Return true if this is a sidecar for obj.

Parameters **obj** (BidsFile) – A BidsFile object to check.

Returns True if this is a BIDS parent of obj and False otherwise.

Return type bool

Notes

- A sidecar is a sidecar for itself.

set_contents(*content_info=None*, *overwrite=False*)

Set the contents of the sidecar.

Parameters

- **content_info** (*list, str, dict or None*) – If None, create a Sidecar from the objects file-path.
- **overwrite** (*bool*) – If True, overwrite contents if already set.

Notes

- **The handling of content_info is as follows:**
  - None: This object's file_path is used to read a JSON dictionary from.
  - str: The string is interpreted as a path and used to read a JSON dictionary from.
  - list: The list is of paths and/or BidsSideCarFiles and a merged dictionary is created.

hed.tools.bids.bids_tabular_dictionary module

class BidsTabularDictionary(*collection_name*, *files*, *entities=('sub', 'ses', 'task', 'run')*)

Bases: *hed.tools.bids.bids_file_dictionary.BidsFileDictionary*

A key tabular-file dictionary for tabular files.

column_dict

Dictionary with an entity key and a list of column names for the file as the value.

Type dict

rowcount_dict

Dictionary with an entity key and a count of number of rows for the file as the value.

Type dict

correct_file(*the_file*)

Transform to BidsTabularFile if needed.

Parameters **the_file** (*str or* BidsFile) – If a str, create a new BidsTabularFile object, otherwise pass the original on.

Returns Either the original file or a newly created BidsTabularFile.

Return type *BidsTabularFile*

Raises **HedFileError** – If the_file isn't str or BidsTabularFile.

count_diffs(*other_dict*)

Return keys in which the number of rows differ.

Parameters **other_dict** (FileDictionary) – A file dictionary object.

Returns A list containing 3-element tuples.

Return type list

Notes

- **The returned tuples consist of**
  - str: The key representing the file.
  - int: Number of rows in the file in this dictionary.
  - int: Number of rows in the file in the other dictionary.

get_info(*key*)

Return a dict with key, row count, and column count.

Parameters **key** (*str*) – The key for file whose information is to be returned.

Returns A dictionary with key, row_count, and columns entries.

Return type dict

get_new_dict(*name*, *files*)

Create a new BidsTabularDictionary.

Parameters

- **name** (*str*) – Name of the new object.
- **files** (*list, dict*) – List or dictionary specifying the files to include.

Returns The object contains just the specified files.

Return type *BidsTabularDictionary*

Notes

- The created object uses the entities from this object

iter_files()

Iterator over the files in this dictionary.

Yields *tuple* – - str: The next key. - BidsTabularFile: The next object. - int: Number of rows - list: List of column names

make_new(*name*, *files*)

Create a dictionary with these files.

Parameters

- **name** (*str*) – Name of this dictionary
- **files** (*list or dict*) – List or dictionary of files. These could be paths or objects.

Returns The newly created dictionary.

Return type *BidsTabularDictionary*

hed.tools.bids.bids_tabular_file module

class BidsTabularFile(*file_path*)

Bases: *hed.tools.bids.bids_file.BidsFile*

A BIDS tabular file including its associated sidecar.

set_contents(*content_info=None*, *overwrite=False*)

Set the contents of this tabular file.

Parameters

- **content_info** (*str or None*) – If string should be a file path if None use self.file_path.
- **overwrite** – If False, do not overwrite existing contents if any.

hed.tools.bids.bids_tabular_summary module

class BidsTabularSummary(*value_cols=None*, *skip_cols=None*, *name=''*)

Bases: `object`

Summarize the contents of BIDS tabular files.

extract_sidecar_template()

Extract a BIDS sidecar-compatible dictionary.

static get_columns_info(*dataframe*, *skip_cols=None*)

Extract unique value counts for columns.

Parameters

- **dataframe** (*DataFrame*) – The DataFrame to be analyzed.
- **skip_cols** (*list*) – List of names of columns to be skipped in the extraction.

Returns

A dictionary with keys that are column names and values that are dictionaries of unique value counts.

Return type dict

get_number_unique(*column_names=None*)

Return the number of unique values in columns.

Parameters **column_names** (*list, None*) – A list of column names to analyze or all columns if None.

Returns Column names are the keys and the number of unique values in the column are the values.

Return type dict

static make_combined_dicts(*file_dictionary*, *skip_cols=None*)

Return combined and individual summaries.

Parameters

- **file_dictionary** (FileDictionary) – Dictionary of file name keys and full path.
- **skip_cols** (*list*) – Name of the column.

Returns

- BidsTabularSummary: Summary of the file dictionary.
- dict: of individual BidsTabularSummary objects.

Return type tuple

update(*data*)

Update the counts based on data.

Parameters **data** (*DataFrame, str, or list*) – DataFrame containing data to update.

update_summary(*col_sum*)

Add ColumnSummary values to this object.

Parameters **col_sum** (BidsTabularSummary) – A ColumnSummary to be combined.

Notes

- The value_cols and skip_cols are updated as long as they are not contradictory.
- A new skip column cannot used.

hed.tools.bids.bids_timeseries_file module

class BidsTimeseriesFile(*file_path*)

Bases: *hed.tools.bids.bids_file.BidsFile*

A BIDS continuous file including its associated sidecar.

set_contents(*content_info=None*, *no_overwrite=True*)

Set the contents of this object.

Parameters

- **content_info** – The contents appropriate for this object.
- **overwrite** (*bool*) – If False and the contents are not empty, do nothing.

Notes

- Do not set if the contents are already set and no_overwrite is True.

Module contents

Submodules

hed.tools.hed_logger module

class HedLogger(*name=''*)

Bases: object

Log status messages organized by key.

add(*key*, *msg*, *level=''*, *also_print=False*)

get_log(*key*)

get_log_keys()

get_log_string(*level=None*)

Return the log as a string, with entries separated by newlines.

load_log(*root_path*, *sub_path='code'*, *log_name='hed_log.json'*)

print_log(*level=None*)

save_log(*save_path*, *log_name='hed_log.json'*)

Module contents

HED tools for analysis and summarization.

3.3.2 hed.tools.BidsDataset

class BidsDataset(*root_path*, *schema=None*, *tabular_types=None*)

A BIDS dataset primarily focused on HED evaluation.

root_path

Real root path of the BIDS dataset.

Type str

schema

The schema used for evaluation.

Type *HedSchema* or *HedSchemaGroup*

tabular_files

A dictionary of BidsTabularDictionary objects containing a given type.

Type dict

__init__(*root_path*, *schema=None*, *tabular_types=None*)

Constructor for a BIDS dataset.

Parameters

- **root_path** (*str*) – Root path of the BIDS dataset.
- **schema** (HedSchema *or* HedSchemaGroup) – A schema that overrides the one specified in dataset.
- **tabular_types** (*list or None*) – List of strings specifying types of tabular types to include. If None or empty, then ['events'] is assumed.

Methods

__init__(root_path[, schema, tabular_types])	Constructor for a BIDS dataset.
get_schema_versions()	Return the schema versions used in this dataset.
get_summary()	Return an abbreviated summary of the dataset.
get_tabular_group([obj_type])	Return the specified tabular file group.
validate([types, check_for_warnings])	Validate the specified file group types.

class BidsDataset(*root_path*, *schema=None*, *tabular_types=None*)

Bases: object

A BIDS dataset primarily focused on HED evaluation.

root_path

Real root path of the BIDS dataset.

Type str

schema

The schema used for evaluation.

Type *HedSchema* or *HedSchemaGroup*

tabular_files

A dictionary of BidsTabularDictionary objects containing a given type.

Type dict

get_schema_versions()

Return the schema versions used in this dataset.

Returns List of schema versions used in this dataset.

Return type list

get_summary()

Return an abbreviated summary of the dataset.

get_tabular_group(*obj_type='events'*)

Return the specified tabular file group.

Parameters **obj_type** (*str*) – Suffix of the BidsFileGroup to be returned.

Returns The requested tabular group.

Return type *BidsFileGroup* or None

validate(*types=None*, *check_for_warnings=True*)

Validate the specified file group types.

Parameters

- **types** (*list*) – A list of strings indicating the file group types to be validated.
- **check_for_warnings** (*bool*) – If True, check for warnings.

Returns List of issues encountered during validation. Each issue is a dictionary.

Return type list

3.3.3 hed.tools.BidsDatasetSummary

class BidsDatasetSummary(*dataset*)

Summary of a BIDS dataset events and other info.

__init__(*dataset*)

Constructor for producing a BIDS dataset JSON summary.

Parameters **dataset** (BidsDataset *or* *str*) – The dataset to be summarized.

Methods

__init__(dataset)	Constructor for producing a BIDS dataset JSON summary.

class BidsDatasetSummary(*dataset*)

Bases: object

Summary of a BIDS dataset events and other info.

3.3.4 hed.tools.BidsFile

class BidsFile(*file_path*)

A BIDS file with entity dictionary.

file_path

Real path of the file.

Type str

suffix

Suffix part of the filename.

Type str

ext

Extension (including the .).

Type str

entity_dict

Dictionary of entity-names (keys) and entity-values (values).

Type dict

sidecar

Merged sidecar for this file.

Type *BidsSidecarFile*

contents

Contents of this file

Notes

- This class may hold the merged sidecar giving metadata for this file as well as contents.

__init__(*file_path*)

Constructor for a file path.

Parameters **file_path** (*str*) – Full path of the file.

Methods

__init__(file_path)	Constructor for a file path.
clear_contents()	Set the contents attribute of this object to None.
get_key(entities)	Return a key for this BIDS file given a list of entities.
set_contents([content_info, overwrite])	Set the contents of this object.

Attributes

get_contents	Return the current contents of this object.

class BidsFile(*file_path*)

Bases: object

A BIDS file with entity dictionary.

file_path

Real path of the file.

Type str

suffix

Suffix part of the filename.

Type str

ext

Extension (including the .).

Type str

entity_dict

Dictionary of entity-names (keys) and entity-values (values).

Type dict

sidecar

Merged sidecar for this file.

Type *BidsSidecarFile*

contents

Contents of this file

Notes

- This class may hold the merged sidecar giving metadata for this file as well as contents.

clear_contents()

Set the contents attribute of this object to None.

property get_contents

Return the current contents of this object.

get_key(*entities*)

Return a key for this BIDS file given a list of entities.

Parameters **entities** (*tuple*) – A tuple of strings representing entities.

Returns A key based on this object.

Return type str

set_contents(*content_info=None*, *overwrite=False*)

Set the contents of this object.

Parameters

- **content_info** – The contents appropriate for this object.
- **overwrite** (*bool*) – If False and the contents are not empty, do nothing.

Notes

- Do not set if the contents are already set and no_overwrite is True.

3.3.5 hed.tools.BidsFileDictionary

class BidsFileDictionary(*collection_name*, *files*, *entities=('sub', 'ses', 'task', 'run')*)

A dictionary of BidsFile keyed by entity pairs.

The keys are simplified entity key-value pairs and the values are BidsFile objects.

__init__(*collection_name*, *files*, *entities=('sub', 'ses', 'task', 'run')*)

Create the dictionary keyed to entities.

Parameters

- **collection_name** (*str*) – Name of this collection.
- **files** (*list or dict*) – Full paths of files to include.
- **entities** (*tuple*) – Entity names to use in creating the keys.

Raises **HedFileError** – If files has inappropriate values.

Notes

- This function is used for cross listing BIDS style files for different studies.

Examples

If entities is ('sub', 'ses', 'task', 'run'), a typical key might be sub-001_ses-01_task-memory_run-01.

Methods

__init__(collection_name, files[, entities])	Create the dictionary keyed to entities.
correct_file(the_file)	Transform to BidsFile if needed.
create_file_dict(file_list, key_indices, ...)	Create new dict based on key indices.
get_file_path(key)	Return the file path corresponding to key.
get_new_dict(name, files)	Create a dictionary with these files.
iter_files()	Iterator over the files in this dictionary.
key_diffs(other_dict)	Return the symmetric key difference with other.
make_dict(files, entities)	Make a dictionary from files or a dict.
make_file_dict(file_list[, key_indices, ...])	Return a dictionary of files using entity keys.
make_key(key_string[, indices, separator])	Create a key from specified entities.
make_query([query_dict])	Return a dictionary of files matching query.
match_query(query_dict, entity_dict)	Return True if query has a match in dictionary.
output_files([title, logger])	Return a string with the output of the list.
split_by_entity(entity)	Split this dictionary based on an entity.

Attributes

file_list	Files contained in this dictionary.
key_list	The dictionary keys.
name	Name of this dictionary

class BidsFileDictionary(*collection_name*, *files*, *entities=('sub', 'ses', 'task', 'run')*)

Bases: *hed.tools.analysis.file_dictionary.FileDictionary*

A dictionary of BidsFile keyed by entity pairs.

The keys are simplified entity key-value pairs and the values are BidsFile objects.

correct_file(*the_file*)

Transform to BidsFile if needed.

Parameters **the_file** (*str or* BidsFile) – If a str, create a new BidsFile object, otherwise pass the original on.

Returns Either the original file or a newly created BidsTabularFile.

Return type *BidsFile*

Raises **HedFileError** – If the_file isn't str or BidsFile.

create_file_dict(*file_list*, *key_indices*, *separator*)

Create new dict based on key indices.

Parameters

- **file_list** (*list*) – Paths of the files to include.
- **key_indices** (*tuple*) – A tuple of integers representing order of entities for key.
- **separator** (*str*) – The separator used between entities to form the key.

property file_list

Files contained in this dictionary.

get_file_path(*key*)

Return the file path corresponding to key.

Parameters **key** (*str*) – The key to use to look up the file in this dictionary.

Returns The real path of the file being looked up.

Return type str

Notes

- None is returned if the key is not present.

get_new_dict(*name*, *files*)

Create a dictionary with these files.

Parameters

- **name** (*str*) – Name of this dictionary
- **files** (*list or dict*) – List or dictionary of files. These could be paths or objects.

Returns The newly created dictionary.

Return type *BidsFileDictionary*

Notes

- The new dictionary uses the same type of entities for keys as this dictionary.

iter_files()

Iterator over the files in this dictionary.

Yields *tuple* – - str: The next entity-based key. - BidsFile: The next BidsFile.

key_diffs(*other_dict*)

Return the symmetric key difference with other.

Parameters **other_dict** (FileDictionary) –

Returns The symmetric difference of the keys in this dictionary and the other one.

Return type list

property key_list

The dictionary keys.

make_dict(*files*, *entities*)

Make a dictionary from files or a dict.

Parameters

- **files** (*list or dict*) – List or dictionary of file-like objs to use.
- **entities** (*tuple*) – Tuple of entity names to use as keys, e.g. ('sub', 'run').

Returns A dictionary whose keys are entity keys and values are BidsFile objects.

Return type dict

Raises **HedFileError** – If incorrect format is passed or something not recognizable as a Bids file.

static make_file_dict(*file_list*, *key_indices=(0, 2)*, *separator='_'*)

Return a dictionary of files using entity keys.

Parameters

- **file_list** (*list*) – Paths to files to use.
- **key_indices** (*tuple*) – Positions of entities to use for key.
- **separator** (*str*) – Separator character used to construct key.

static make_key(*key_string*, *indices=(0, 2)*, *separator='_'*)

Create a key from specified entities.

Parameters

- **key_string** (*str*) – The string from which to extract the key (usually a filename or path).
- **indices** (*tuple*) – Positions of entity pairs to use as key.
- **separator** (*str*) – Separator between entity pairs in the created key.

Returns The created key.

Return type str

make_query(*query_dict={'sub': '*'}*)

Return a dictionary of files matching query.

Parameters **query_dict** (*dict*) – A dictionary whose keys are entities and whose values are entity values to match.

Returns A dictionary entries in this dictionary that match the query.

Return type dict

Notes

- A query dictionary key a valid BIDS entity name such as sub or task.
- A query dictionary value may be a string or a list.
- A query value string should contain a specific value of the entity or a '*' indicating any value matches.
- A query value list should be a list of valid values for the corresponding entity.

static match_query(*query_dict*, *entity_dict*)

Return True if query has a match in dictionary.

Parameters

- **query_dict** (*dict*) – A dictionary representing a query about entities.
- **entity_dict** (*dict*) – A dictionary containing the entity representation for a BIDS file.

Returns True if the query matches the entities representing the file.

Return type bool

Notes

- A query is a dictionary whose keys are entity names and whose values are specific entity values or '*'.

Examples

{'sub', '001', 'run', '*'} requests all runs from subject 001.

property name

Name of this dictionary

output_files(*title=None*, *logger=None*)

Return a string with the output of the list.

Parameters

- **title** (*None, str*) –
- **logger** (HedLogger) –

Returns The dictionary in string form.

Return type str

Notes

- The logger is updated if available.

split_by_entity(*entity*)

Split this dictionary based on an entity.

Parameters **entity** (*str*) – Entity name (for example task).

Returns

- dict: A dictionary unique values of entity as keys and BidsFileDictionary objs as values.
- dict: A BidsFileDictionary containing the files that don't have entity in their names.

Return type tuple

Notes

- This function is used for analysis where a single subject or single type of task is being analyzed.

3.3.6 hed.tools.BidsFileGroup

class BidsFileGroup(*root_path*, *suffix='_events'*, *obj_type='tabular'*)

Container for BIDS files with a specified suffix.

root_path

Real root path of the Bids dataset.

Type str

suffix

The file suffix specifying the class of file represented in this group (e.g., events).

Type str

obj_type

Type of file in this group (e.g., Tabular or Timeseries).

Type str

sidecar_dict

A dictionary of sidecars associated with this suffix .

Type dict

datafile_dict

A dictionary with values either BidsTabularFile or BidsTimeseriesFile.

Type dict

sidecar_dir_dict

Dictionary whose keys are directory paths and values are list of sidecars in the corresponding directory.

Type dict

__init__(*root_path*, *suffix='_events'*, *obj_type='tabular'*)

Constructor for a BidsFileGroup.

Parameters

- **root_path** (*str*) – Path of the root of the BIDS dataset.
- **suffix** (*str*) – Suffix indicating the type this group represents (e.g. events, or channels, etc.).
- **obj_type** (*str*) – Indicates the type of underlying file represents the contents.

Methods

__init__(root_path[, suffix, obj_type])	Constructor for a BidsFileGroup.
get_sidecars_from_path(obj)	Return applicable sidecars for the object.
summarize([value_cols, skip_cols])	Return a BidsTabularSummary of group files.
validate_datafiles(hed_ops[, ...])	Validate the datafiles and return an error list.
validate_sidecars(hed_ops[, check_for_warnings])	Validate merged sidecars.

class BidsFileGroup(*root_path*, *suffix='_events'*, *obj_type='tabular'*)

Bases: `object`

Container for BIDS files with a specified suffix.

root_path

Real root path of the Bids dataset.

Type str

suffix

The file suffix specifying the class of file represented in this group (e.g., events).

Type str

obj_type

Type of file in this group (e.g., Tabular or Timeseries).

Type str

sidecar_dict

A dictionary of sidecars associated with this suffix .

Type dict

datafile_dict

A dictionary with values either BidsTabularFile or BidsTimeseriesFile.

Type dict

sidecar_dir_dict

Dictionary whose keys are directory paths and values are list of sidecars in the corresponding directory.

Type dict

get_sidecars_from_path(*obj*)

Return applicable sidecars for the object.

Parameters **obj** (*BidsTabularFile or BidsSidecarFile*) – The BIDS file object to get the sidecars for.

Returns A list of the applicable sidecars for obj starting at the root.

Return type list

summarize(*value_cols=None*, *skip_cols=None*)

Return a BidsTabularSummary of group files.

Parameters

- **value_cols** (*list*) – Column names designated as value columns.
- **skip_cols** (*list*) – Column names designated as columns to skip.

Returns A summary of the number of values in different columns if tabular group.

Return type *BidsTabularSummary* or None

Notes

- The columns that are not value_cols or skip_col are summarized by counting

the number of times each unique value appears in that column.

validate_datafiles(*hed_ops*, *check_for_warnings=True*, *keep_contents=False*)

Validate the datafiles and return an error list.

Parameters

- **hed_ops** (*[func or HedOps], func, HedOps*) – Validation functions to apply.
- **check_for_warnings** (*bool*) – If True, include warnings in the check.

- **keep_contents** (*bool*) – If True, the underlying data files are read and their contents retained.

Returns A list of validation issues found. Each issue is a dictionary.

Return type list

validate_sidecars(*hed_ops*, *check_for_warnings=True*)

Validate merged sidecars.

Parameters

- **hed_ops** (*[func or HedOps], func, HedOps*) – Validation functions to apply.
- **check_for_warnings** (*bool*) – If True, include warnings in the check.

Returns A list of validation issues found. Each issue is a dictionary.

Return type list

3.3.7 hed.tools.BidsSidecarFile

class BidsSidecarFile(*file_path*)

A BIDS sidecar file.

__init__(*file_path*)

Constructor for a file path.

Parameters **file_path** (*str*) – Full path of the file.

Methods

__init__(file_path)	Constructor for a file path.
clear_contents()	Set the contents attribute of this object to None.
get_key(entities)	Return a key for this BIDS file given a list of entities.
get_merged(file_list)	Return merged contents of JSON files as dict.
is_hed(json_dict)	Return True if the json has HED.
is_sidecar_for(obj)	Return true if this is a sidecar for obj.
set_contents([content_info, overwrite])	Set the contents of the sidecar.

Attributes

get_contents	Return the current contents of this object.

class BidsSidecarFile(*file_path*)

Bases: *hed.tools.bids.bids_file.BidsFile*

A BIDS sidecar file.

clear_contents()

Set the contents attribute of this object to None.

property get_contents

Return the current contents of this object.

get_key(*entities*)

Return a key for this BIDS file given a list of entities.

Parameters **entities** (*tuple*) – A tuple of strings representing entities.

Returns A key based on this object.

Return type str

static get_merged(*file_list*)

Return merged contents of JSON files as dict.

Parameters **file_list** (*list or None*) – A list of JSON files representing sidecars in the order they are to be merged.

Returns A merged JSON dictionary.

Return type dict

Notes

- Merging takes place from front to back with overwriting of top-level keys.

static is_hed(*json_dict*)

Return True if the json has HED.

Parameters **json_dict** (*dict*) – A dictionary representing a JSON file or merged file.

Returns True if the dictionary has HED or HED_assembled as a first or second-level key.

Return type bool

is_sidecar_for(*obj*)

Return true if this is a sidecar for obj.

Parameters **obj** (BidsFile) – A BidsFile object to check.

Returns True if this is a BIDS parent of obj and False otherwise.

Return type bool

Notes

- A sidecar is a sidecar for itself.

set_contents(*content_info=None*, *overwrite=False*)

Set the contents of the sidecar.

Parameters

- **content_info** (*list, str, dict or None*) – If None, create a Sidecar from the objects file-path.
- **overwrite** (*bool*) – If True, overwrite contents if already set.

Notes

- **The handling of content_info is as follows:**
  - None: This object's file_path is used to read a JSON dictionary from.
  - str: The string is interpreted as a path and used to read a JSON dictionary from.
  - list: The list is of paths and/or BidsSideCarFiles and a merged dictionary is created.

3.3.8 hed.tools.BidsTabularDictionary

class BidsTabularDictionary(*collection_name*, *files*, *entities=('sub', 'ses', 'task', 'run')*)

A key tabular-file dictionary for tabular files.

column_dict

Dictionary with an entity key and a list of column names for the file as the value.

Type dict

rowcount_dict

Dictionary with an entity key and a count of number of rows for the file as the value.

Type dict

__init__(*collection_name*, *files*, *entities=('sub', 'ses', 'task', 'run')*)

Create a dictionary of full paths.

Parameters

- **collection_name** (*str*) – Name of the collection.
- **files** (*list, dict*) – Contains the full paths or BidsFile representation of files of interest.
- **entities** (*tuple*) – List of indices into base file names of pieces to assemble for the key.

Notes

- Used for cross listing BIDS style files for different studies.

Methods

__init__(collection_name, files[, entities])	Create a dictionary of full paths.
correct_file(the_file)	Transform to BidsTabularFile if needed.
count_diffs(other_dict)	Return keys in which the number of rows differ.
create_file_dict(file_list, key_indices, ...)	Create new dict based on key indices.
get_file_path(key)	Return the file path corresponding to key.
get_info(key)	Return a dict with key, row count, and column count.
get_new_dict(name, files)	Create a new BidsTabularDictionary.
iter_files()	Iterator over the files in this dictionary.
key_diffs(other_dict)	Return the symmetric key difference with other.
make_dict(files, entities)	Make a dictionary from files or a dict.
make_file_dict(file_list[, key_indices, ...])	Return a dictionary of files using entity keys.
make_key(key_string[, indices, separator])	Create a key from specified entities.
make_new(name, files)	Create a dictionary with these files.
make_query([query_dict])	Return a dictionary of files matching query.
match_query(query_dict, entity_dict)	Return True if query has a match in dictionary.
output_files([title, logger])	Return a string with the output of the list.
split_by_entity(entity)	Split this dictionary based on an entity.

Attributes

file_list	Files contained in this dictionary.
key_list	The dictionary keys.
name	Name of this dictionary

class BidsTabularDictionary(*collection_name*, *files*, *entities=('sub', 'ses', 'task', 'run')*)

Bases: *hed.tools.bids.bids_file_dictionary.BidsFileDictionary*

A key tabular-file dictionary for tabular files.

column_dict

Dictionary with an entity key and a list of column names for the file as the value.

Type dict

rowcount_dict

Dictionary with an entity key and a count of number of rows for the file as the value.

Type dict

correct_file(*the_file*)

Transform to BidsTabularFile if needed.

Parameters **the_file** (*str or* BidsFile) – If a str, create a new BidsTabularFile object, otherwise pass the original on.

Returns Either the original file or a newly created BidsTabularFile.

Return type *BidsTabularFile*

Raises **HedFileError** – If the_file isn't str or BidsTabularFile.

count_diffs(*other_dict*)

Return keys in which the number of rows differ.

Parameters **other_dict** (FileDictionary) – A file dictionary object.

Returns A list containing 3-element tuples.

Return type list

Notes

- **The returned tuples consist of**
  - str: The key representing the file.
  - int: Number of rows in the file in this dictionary.
  - int: Number of rows in the file in the other dictionary.

create_file_dict(*file_list*, *key_indices*, *separator*)

Create new dict based on key indices.

Parameters

- **file_list** (*list*) – Paths of the files to include.
- **key_indices** (*tuple*) – A tuple of integers representing order of entities for key.
- **separator** (*str*) – The separator used between entities to form the key.

property file_list

Files contained in this dictionary.

get_file_path(*key*)

Return the file path corresponding to key.

Parameters **key** (*str*) – The key to use to look up the file in this dictionary.

Returns The real path of the file being looked up.

Return type str

Notes

- None is returned if the key is not present.

get_info(*key*)

Return a dict with key, row count, and column count.

Parameters **key** (*str*) – The key for file whose information is to be returned.

Returns A dictionary with key, row_count, and columns entries.

Return type dict

get_new_dict(*name*, *files*)

Create a new BidsTabularDictionary.

Parameters

- **name** (*str*) – Name of the new object.

- **files** (*list, dict*) – List or dictionary specifying the files to include.

Returns The object contains just the specified files.

Return type *BidsTabularDictionary*

Notes

- The created object uses the entities from this object

iter_files()

Iterator over the files in this dictionary.

Yields *tuple* – - str: The next key. - BidsTabularFile: The next object. - int: Number of rows - list: List of column names

key_diffs(*other_dict*)

Return the symmetric key difference with other.

Parameters **other_dict** (FileDictionary) –

Returns The symmetric difference of the keys in this dictionary and the other one.

Return type list

property key_list

The dictionary keys.

make_dict(*files*, *entities*)

Make a dictionary from files or a dict.

Parameters

- **files** (*list or dict*) – List or dictionary of file-like objs to use.
- **entities** (*tuple*) – Tuple of entity names to use as keys, e.g. ('sub', 'run').

Returns A dictionary whose keys are entity keys and values are BidsFile objects.

Return type dict

Raises **HedFileError** – If incorrect format is passed or something not recognizable as a Bids file.

static make_file_dict(*file_list*, *key_indices=(0, 2)*, *separator='_'*)

Return a dictionary of files using entity keys.

Parameters

- **file_list** (*list*) – Paths to files to use.
- **key_indices** (*tuple*) – Positions of entities to use for key.
- **separator** (*str*) – Separator character used to construct key.

static make_key(*key_string*, *indices=(0, 2)*, *separator='_'*)

Create a key from specified entities.

Parameters

- **key_string** (*str*) – The string from which to extract the key (usually a filename or path).
- **indices** (*tuple*) – Positions of entity pairs to use as key.

- **separator** (*str*) – Separator between entity pairs in the created key.

Returns The created key.

Return type str

make_new(*name*, *files*)

Create a dictionary with these files.

Parameters

- **name** (*str*) – Name of this dictionary
- **files** (*list or dict*) – List or dictionary of files. These could be paths or objects.

Returns The newly created dictionary.

Return type *BidsTabularDictionary*

make_query(*query_dict={'sub': '*'}*)

Return a dictionary of files matching query.

Parameters **query_dict** (*dict*) – A dictionary whose keys are entities and whose values are entity values to match.

Returns A dictionary entries in this dictionary that match the query.

Return type dict

Notes

- A query dictionary key a valid BIDS entity name such as sub or task.
- A query dictionary value may be a string or a list.
- A query value string should contain a specific value of the entity or a '*' indicating any value matches.
- A query value list should be a list of valid values for the corresponding entity.

static match_query(*query_dict*, *entity_dict*)

Return True if query has a match in dictionary.

Parameters

- **query_dict** (*dict*) – A dictionary representing a query about entities.
- **entity_dict** (*dict*) – A dictionary containing the entity representation for a BIDS file.

Returns True if the query matches the entities representing the file.

Return type bool

Notes

- A query is a dictionary whose keys are entity names and whose values are specific entity values or '*'.

Examples

{'sub', '001', 'run', '*'} requests all runs from subject 001.

property name

Name of this dictionary

output_files(*title=None*, *logger=None*)

Return a string with the output of the list.

Parameters

- **title** (*None, str*) –
- **logger** (HedLogger) –

Returns The dictionary in string form.

Return type str

Notes

- The logger is updated if available.

split_by_entity(*entity*)

Split this dictionary based on an entity.

Parameters **entity** (*str*) – Entity name (for example task).

Returns

- dict: A dictionary unique values of entity as keys and BidsFileDictionary objs as values.
- dict: A BidsFileDictionary containing the files that don't have entity in their names.

Return type tuple

Notes

- This function is used for analysis where a single subject or single type of task is being analyzed.

3.3.9 hed.tools.BidsTabularFile

class BidsTabularFile(*file_path*)

A BIDS tabular file including its associated sidecar.

__init__(*file_path*)

Constructor for a BIDS tabular file.

Parameters **file_path** (*str*) – Path of the tabular file.

Methods

__init__(file_path)	Constructor for a BIDS tabular file.
clear_contents()	Set the contents attribute of this object to None.
get_key(entities)	Return a key for this BIDS file given a list of entities.
set_contents([content_info, overwrite])	Set the contents of this tabular file.

Attributes

get_contents	Return the current contents of this object.

class BidsTabularFile(*file_path*)

Bases: *hed.tools.bids.bids_file.BidsFile*

A BIDS tabular file including its associated sidecar.

clear_contents()

Set the contents attribute of this object to None.

property get_contents

Return the current contents of this object.

get_key(*entities*)

Return a key for this BIDS file given a list of entities.

Parameters **entities** (*tuple*) – A tuple of strings representing entities.

Returns A key based on this object.

Return type str

set_contents(*content_info=None*, *overwrite=False*)

Set the contents of this tabular file.

Parameters

- **content_info** (*str or None*) – If string should be a file path if None use self.file_path.
- **overwrite** – If False, do not overwrite existing contents if any.

3.3.10 hed.tools.BidsTabularSummary

class BidsTabularSummary(*value_cols=None*, *skip_cols=None*, *name=''*)

Summarize the contents of BIDS tabular files.

__init__(*value_cols=None*, *skip_cols=None*, *name=''*)

Constructor for a BIDS tabular file summary.

Parameters

- **value_cols** (*list, None*) – List of columns to be treated as value columns.
- **skip_cols** (*list, None*) – List of columns to be skipped.
- **name** (*str*) – Name associated with the dictionary.

Methods

__init__([value_cols, skip_cols, name])	Constructor for a BIDS tabular file summary.
extract_sidecar_template()	Extract a BIDS sidecar-compatible dictionary.
get_columns_info(dataframe[, skip_cols])	Extract unique value counts for columns.
get_number_unique([column_names])	Return the number of unique values in columns.
make_combined_dicts(file_dictionary[, skip_cols])	Return combined and individual summaries.
update(data)	Update the counts based on data.
update_summary(col_sum)	Add ColumnSummary values to this object.

class BidsTabularSummary(*value_cols=None*, *skip_cols=None*, *name=''*)

Bases: `object`

Summarize the contents of BIDS tabular files.

extract_sidecar_template()

Extract a BIDS sidecar-compatible dictionary.

static get_columns_info(*dataframe*, *skip_cols=None*)

Extract unique value counts for columns.

Parameters

- **dataframe** (*DataFrame*) – The DataFrame to be analyzed.
- **skip_cols** (*list*) – List of names of columns to be skipped in the extraction.

Returns

A dictionary with keys that are column names and values that are dictionaries of unique value counts.

Return type dict

get_number_unique(*column_names=None*)

Return the number of unique values in columns.

Parameters **column_names** (*list, None*) – A list of column names to analyze or all columns if None.

Returns Column names are the keys and the number of unique values in the column are the values.

Return type dict

static make_combined_dicts(*file_dictionary*, *skip_cols=None*)

Return combined and individual summaries.

Parameters

- **file_dictionary** (FileDictionary) – Dictionary of file name keys and full path.
- **skip_cols** (*list*) – Name of the column.

Returns

- BidsTabularSummary: Summary of the file dictionary.
- dict: of individual BidsTabularSummary objects.

Return type tuple

update(*data*)

Update the counts based on data.

Parameters **data** (*DataFrame, str, or list*) – DataFrame containing data to update.

update_summary(*col_sum*)

Add ColumnSummary values to this object.

Parameters **col_sum** (BidsTabularSummary) – A ColumnSummary to be combined.

Notes

- The value_cols and skip_cols are updated as long as they are not contradictory.
- A new skip column cannot used.

3.3.11 hed.tools.FileDictionary

class FileDictionary(*collection_name*, *file_list*, *key_indices=(0, 2)*, *separator='_'*)

A file dictionary keyed by entity pair indices.

Notes

- The entities are identified as 0, 1, ... depending on order in the base filename.
- The entity key-value pairs are assumed separated by '_' unless a separator is provided.

__init__(*collection_name*, *file_list*, *key_indices=(0, 2)*, *separator='_'*)

Create a dictionary with full paths as values.

Parameters

- **collection_name** (*str*) – Name of the file collection for reference.
- **file_list** (*list, None*) – List containing full paths of files of interest.
- **key_indices** (*tuple, None*) – List of order of key-value pieces to assemble for the key.
- **separator** (*str*) – Character used to separate pieces of key name.

Notes

- This dictionary is used for cross listing BIDS style files for different studies.
-

Examples

If key_indices is (0, 2), the key generated for /tmp/sub-001_task-FaceCheck_run-01_events.tsv is sub_001_run-01.

Methods

__init__(collection_name, file_list[, ...])	Create a dictionary with full paths as values.
create_file_dict(file_list, key_indices, ...)	Create new dict based on key indices.
get_file_path(key)	Return file path corresponding to key.
iter_files()	Iterator over the files in this dictionary.
key_diffs(other_dict)	Return symmetric key difference with other.
make_file_dict(file_list[, key_indices, ...])	Return a dictionary of files using entity keys.
make_key(key_string[, indices, separator])	Create a key from specified entities.
output_files([title, logger])	Return a string with the output of the list.

Attributes

file_list	List of path values in this dictionary.
key_list	Keys in this dictionary.
name	Name of this dictionary

class FileDictionary(*collection_name*, *file_list*, *key_indices=(0, 2)*, *separator='_'*)

Bases: object

A file dictionary keyed by entity pair indices.

Notes

- The entities are identified as 0, 1, ... depending on order in the base filename.
- The entity key-value pairs are assumed separated by '_' unless a separator is provided.

create_file_dict(*file_list*, *key_indices*, *separator*)

Create new dict based on key indices.

Parameters

- **file_list** (*list*) – Paths of the files to include.
- **key_indices** (*tuple*) – A tuple of integers representing order of entities for key.
- **separator** (*str*) – The separator used between entities to form the key.

property file_list

List of path values in this dictionary.

get_file_path(*key*)

Return file path corresponding to key.

Parameters **key** (*str*) – Key used to retrieve the file path.

Returns File path.

Return type str

`iter_files()`

Iterator over the files in this dictionary.

Yields - *str* – Key into the dictionary. - file: File path.

`key_diffs(other_dict)`

Return symmetric key difference with other.

Parameters **other_dict** (FileDictionary) –

Returns The symmetric difference of the keys in this dictionary and the other one.

Return type list

`property key_list`

Keys in this dictionary.

`static make_file_dict(file_list, key_indices=(0, 2), separator='_')`

Return a dictionary of files using entity keys.

Parameters

- **file_list** (*list*) – Paths to files to use.
- **key_indices** (*tuple*) – Positions of entities to use for key.
- **separator** (*str*) – Separator character used to construct key.

`static make_key(key_string, indices=(0, 2), separator='_')`

Create a key from specified entities.

Parameters

- **key_string** (*str*) – The string from which to extract the key (usually a filename or path).
- **indices** (*tuple*) – Positions of entity pairs to use as key.
- **separator** (*str*) – Separator between entity pairs in the created key.

Returns The created key.

Return type str

`property name`

Name of this dictionary

`output_files(title=None, logger=None)`

Return a string with the output of the list.

Parameters

- **title** (*None, str*) –
- **logger** (HedLogger) –

Returns The dictionary in string form.

Return type str

Notes

- The logger is updated if available.

3.3.12 hed.tools.KeyMap

class KeyMap(*key_cols*, *target_cols=None*, *name=''*)

A map of unique column values for remapping columns.

name

Name of this remap for identification purposes.

Type str

key_cols

A list of column names that will be hashed into the keys for the map.

Type list

target_cols

An optional list of column names that will be inserted into data and later remapped.

Type list

__init__(*key_cols*, *target_cols=None*, *name=''*)

Information for remapping columns of tabular files.

Parameters

- **key_cols** (*list*) – List of columns to be replaced (assumed in the DataFrame)
- **target_cols** (*list*) – List of replacement columns (assumed to not be in the DataFrame)
- **name** (*str*) – Name associated with this remap (usually a pathname of the events file).

Methods

__init__(key_cols[, target_cols, name])	Information for remapping columns of tabular files.
make_template([additional_cols])	Return a dataframe template.
remap(data)	Remap the columns of a dataframe or columnar file.
resort()	Sort the col_map in place by the key columns.
update(data[, allow_missing, keep_counts])	Update the existing map with information from data.

Attributes

columns

class KeyMap(*key_cols*, *target_cols=None*, *name=''*)

Bases: `object`

A map of unique column values for remapping columns.

name

Name of this remap for identification purposes.

Type str

key_cols

A list of column names that will be hashed into the keys for the map.

Type list

target_cols

An optional list of column names that will be inserted into data and later remapped.

Type list

property columns

make_template(*additional_cols=None*)

Return a dataframe template.

Parameters **additional_cols** (*list or None*) – Optional list of additional columns to append to the returned dataframe.

Returns A dataframe containing the template.

Return type DataFrame

Raises **HedFileError** – If additional columns are not disjoint from the key columns.

Notes

- The template consists of the unique key columns in this map plus additional columns.

remap(*data*)

Remap the columns of a dataframe or columnar file.

Parameters **data** (*DataFrame, str*) – Columnar data (either DataFrame or filename) whose columns are to be remapped.

Returns

- DataFrame: New dataframe with columns remapped.
- list: List of row numbers that had no correspondence in the mapping.

Return type tuple

Raises **HedFileError** – If data is missing some of the key columns.

resort()

Sort the col_map in place by the key columns.

update(*data*, *allow_missing=True*, *keep_counts=True*)

Update the existing map with information from data.

Parameters

- **data** (*DataFrame or str*) – DataFrame or filename of an events file or event map.
- **allow_missing** (*bool*) – If true allow missing keys and add as n/a columns.
- **keep_counts** (*bool*) – If true keep a count of the times each key is present.

Returns The indices of duplicates.

Return type list

Raises **HedFileError** – If there are missing keys and allow_missing is False.

3.3.13 hed.tools.TagSummary

class TagSummary(*file_group*, *schema*, *breakout_list=None*)

A HED tag summary for a BID file group.

file_group

The BIDS file group to be summarized.

Type *BidsFileGroup*

schema (HedSchema or Hed

all_tags_dict

The keys are all of the unique tags in the file group. The values are dictionaries of the unique values that these tags take on.

Type dict

breakout_list

The tag nodes that are to be specially summarized.

Type list

breakout_dict

The keys are the breakout nodes. The values are dictionaries of the child nodes and the nodes themselves that appear in the dataset.

Type dict

task_dict

The keys are definition names and the values are dictionaries with info.

Type dict

cond_dict

The keys are definition names and the values are dictionaries with info.

Type dict

__init__(*file_group*, *schema*, *breakout_list=None*)

Constructor for TagSummary.

Parameters

- **file_group** (BidsFileGroup) – Container holding the files with a particular suffix.
- **schema** (HedSchema *or* HedSchemaGroup) – The HED schema(s) used in the summary.
- **breakout_list** (*list or None*) – Used to arrange the tags in specified groupings.

Methods

__init__(file_group, schema[, breakout_list])	Constructor for TagSummary.
extract_summary_info(entry_dict, tag_name)	Extract the summary of tag in this entry.
get_design_matrices()	Return information about the condition variables.

class TagSummary(*file_group*, *schema*, *breakout_list=None*)

Bases: object

A HED tag summary for a BID file group.

file_group

The BIDS file group to be summarized.

Type *BidsFileGroup*

schema (HedSchema or Hed

all_tags_dict

The keys are all of the unique tags in the file group. The values are dictionaries of the unique values that these tags take on.

Type dict

breakout_list

The tag nodes that are to be specially summarized.

Type list

breakout_dict

The keys are the breakout nodes. The values are dictionaries of the child nodes and the nodes themselves that appear in the dataset.

Type dict

task_dict

The keys are definition names and the values are dictionaries with info.

Type dict

cond_dict

The keys are definition names and the values are dictionaries with info.

Type dict

static extract_summary_info(*entry_dict*, *tag_name*)

Extract the summary of tag in this entry.

Parameters

- **entry_dict** (*dict*) – Keys are individual tag node names.
- **tag_name** (*str*) – Name of an individual node.

Returns A dictionary of the extracted tag information.

Return type dict

get_design_matrices()

Return information about the condition variables.

Returns

- dict: Dictionary with condition variable levels corresponding to a design matrix.
- list: List with the other condition variables that aren't associated with levels.
- list: List of errors.

Return type tuple

3.3.14 hed.tools.analysis.analysis_util

Utilities for downstream analysis such as searching.

Functions

assemble_hed(data_input[, columns_included, ...])	Return assembled HED annotations in a dataframe.
get_assembled_strings(table[, hed_schema, ...])	Return HED string objects for a tabular file.
search_tabular(data_input, hed_schema, query)	Return a dataframe with results of query.

3.3.15 hed.tools.analysis.annotation_util

Utilities to facilitate annotation of events in BIDS.

Functions

check_df_columns(df[, required_cols])	Return a list of the specified columns that are missing from a dataframe.
df_to_hed(dataframe[, description_tag])	Create sidecar-like dictionary from a 4-column dataframe.
extract_tags(hed_string, search_tag)	Extract all instances of specified tag from a tag_string.
generate_sidecar_entry(column_name[, ...])	Create a sidecar column dictionary for column.
hed_to_df(sidecar_dict[, col_names])	Return a 4-column dataframe of HED portions of sidecar.
merge_hed_dict(sidecar_dict, hed_dict)	Update a JSON sidecar based on the hed_dict values.
trim_back(tag_string)	Return a trimmed copy of tag_string.
trim_front(tag_string)	Return a copy of tag_string with leading blanks and commas removed.

3.3.16 hed.tools.analysis.summary_util

Utilities used in computing dataset annotation.

Functions

add_tag_list_to_dict(tag_list, tag_dict[, ...])	Convert tags and groups to a dictionary.
breakout_tags(schema, tag_list, breakout_list)	Create a dictionary with tags split into groups.
extract_dict_values(tag_dict, tag_name, tags)	Get the tags associated with tag name from the tag dictionary.
get_schema_entries(hed_schema, tag[, ...])	Get schema entries for tag and to its parents.
unfold_tag_list(tag_list)	Unfold a list to a single-level list of HedTags.

3.3.17 hed.tools.HedLogger

class HedLogger(*name=''*)

Log status messages organized by key.

__init__(*name=''*)

Methods

__init__([name])	
add(key, msg[, level, also_print])	
get_log(key)	
get_log_keys()	
get_log_string([level])	Return the log as a string, with entries separated by newlines.
load_log(root_path[, sub_path, log_name])	
print_log([level])	
save_log(save_path[, log_name])	

class HedLogger(*name=''*)

Bases: `object`

Log status messages organized by key.

add(*key*, *msg*, *level=''*, *also_print=False*)

get_log(*key*)

get_log_keys()

get_log_string(*level=None*)

Return the log as a string, with entries separated by newlines.

load_log(*root_path*, *sub_path='code'*, *log_name='hed_log.json'*)

print_log(*level=None*)

save_log(*save_path*, *log_name='hed_log.json'*)

3.4 HED utilities

hed.util	Data and file handling utilities.
hed.util.data_util	Data handling utilities involving dataframes.
hed.util.file_util	Utilities for writing content to files and for other file manipulation.
hed.util.io_util	Utilities for generating and handling file names.

3.4.1 hed.util package

Submodules

hed.util.data_util module

Data handling utilities involving dataframes.

add_columns(*df*, *column_list*, *value='n/a'*)

Add specified columns to df if not there.

Parameters

- **df** (*DataFrame*) – Pandas dataframe.
- **column_list** (*list*) – List of columns to append to the dataframe.
- **value** (*str*) – Default fill value for the column.

check_match(*ds1*, *ds2*, *numeric=False*)

Check two Pandas data series have the same values.

Parameters

- **ds1** (*DataSeries*) – Pandas data series to check.
- **ds2** (*DataSeries*) – Pandas data series to check.
- **numeric** (*bool*) – If true, treat as numeric and do close-to comparison.

Returns Error messages indicating the mismatch or empty if the series match.

Return type list

delete_columns(*df*, *column_list*)

Delete the specified columns from a dataframe.

Parameters

- **df** (*DataFrame*) – Pandas dataframe from which to delete columns.

- **column_list** (*list*) – List of candidate column names for deletion.

Notes

- The deletion of columns is done in place.
- This does not raise an error if df does not have a column in the list.

delete_rows_by_column(*df*, *value*, *column_list=None*)

Delete rows where columns have this value.

Parameters

- **df** (*DataFrame*) – Pandas dataframe from which to delete rows.
- **value** (*str*) – Specified value to indicate row should be deleted.
- **column_list** (*list*) – List of columns to search for value.

Notes

- All values are converted to string before testing.
- Deletion is done in place.

get_key_hash(*key_tuple*)

Calculate a hash key for tuple of values.

Parameters **key_tuple** (*tuple, list*) – The key values in the correct order for lookup.

Returns A hash key for the tuple.

Return type int

get_new_dataframe(*data*)

Get a new dataframe representing a tsv file.

Parameters **data** (*DataFrame or str*) – DataFrame or filename representing a tsv file.

Returns

A dataframe containing the contents of the tsv file or if data was a DataFrame to start with, a new copy of the DataFrame.

Return type DataFrame

Raises **HedFileError** – If a filename is given and it cannot be read into a Dataframe.

get_row_hash(*row*, *key_list*)

Get a hash key from key column values for row.

Parameters

- **row** (*DataSeries*) –
- **key_list** (*list*) –

Returns Hash key constructed from the entries of row in the columns specified by key_list.

Return type str

Raises **HedFileError** – If row doesn't have all of the columns in key_list HedFileError is raised.

get_value_dict(*tsv_path*, *key_col='file_basename'*, *value_col='sampling_rate'*)

Get a dictionary of two columns of a dataframe.

Parameters

- **tsv_path** (*str*) –
- **key_col** (*str*) –
- **value_col** (*str*) –

Returns

Dictionary with key_col values as the keys and the corresponding value_col values as the values.

Return type dict

Raises **HedFileError** – When tsv_path does not correspond to a file that can be read into a DataFrame.

make_info_dataframe(*col_info*, *selected_col*)

Get a dataframe from selected columns.

Parameters

- **col_info** (*dict*) – Dictionary of dictionaries of column values and counts.
- **selected_col** (*str*) – Name of the column used as top level key for col_info.

Returns

A two-column dataframe with first column containing values from the dictionary whose key is selected_col and whose second column are the corresponding counts. The returned value is None if selected_col is not a top-level key in col_info.

Return type dataframe

remove_quotes(*df*)

Remove quotes from all columns.

Parameters **df** (*Dataframe*) – Dataframe to process by removing quotes.

Notes

- Replacement is done in place.

reorder_columns(*data*, *col_order*, *skip_missing=True*)

Create a new dataframe with columns reordered. :param data: Dataframe or filename of dataframe whose columns are to be reordered. :type data: DataFrame, str :param col_order: List of column names in desired order. :type col_order: list :param skip_missing: If true, col_order columns missing from data are skipped, otherwise error. :type skip_missing: bool

Returns DataFrame A new reordered dataframe.

Raises **HedFileError** – If col_order contains columns not in data and skip_missing is False or if data corresponds to a filename from which a dataframe cannot be created.

replace_values(*df*, *values=None*, *replace_value='n/a'*, *column_list=None*)

Replace string values in specified columns.

Parameters

- **df** (*DataFrame*) – Dataframe whose values will replaced.
- **values** (*list, None*) – List of strings to replace. If None, only empty strings are replaced.
- **replace_value** (*str*) – String replacement value.
- **column_list** (*list, None*) – List of columns in which to do replacement. If None all columns are processed.

Returns number of values replaced.

Return type int

separate_columns(*base_cols*, *target_cols*)

Get target columns from the base list.

Parameters

- **base_cols** (*list*) – List of columns to be tested.
- **target_cols** – List of desired column names.

hed.util.file_util module

Utilities for writing content to files and for other file manipulation.

get_version_from_xml(*hed_xml_tree*)

Get version from root node of an XML tree.

Parameters **hed_xml_tree** (*Element*) – The root node of an XML tree.

Returns The version of the HED schema (e.g. "8.0.0").

Return type str

Raises **KeyError or AttributeError** – If invalid.

TODO: This should be moved to the schema module

url_to_file(*resource_url*)

Write data from a URL resource into a file. Data is decoded as unicode.

Parameters **resource_url** (*str*) – The URL to the resource.

Returns The local temporary filename for the downloaded file,

Return type str

url_to_string(*resource_url*)

Get the data from the specified url as a string.

Parameters **resource_url** (*str*) – The URL to the resource.

Returns The data at the target url.

Return type str

write_errors_to_file(*issues*, *extension='.txt'*)

Write an array of issue dictionaries to a temporary file.

Parameters

- **issues** (*list*) – List of 2-element dictionaries containing code and message keys.
- **extension** (*str*) – Desired file extension.

Returns The name of the temporary file.

Return type str

write_strings_to_file(*output_strings*, *extension=None*)

Write output strings to a temporary file.

Parameters

- **output_strings** (*[str], str*) – Strings to output one per line.
- **extension** (*str*) – File extension of the temporary file.

Returns Opened temporary file.

Return type file

write_xml_tree_2_xml_file(*xml_tree*, *extension='.xml'*)

Write an XML element tree object into a XML file.

Parameters

- **xml_tree** (*Element*) – An element representing an XML file.
- **extension** (*string*) – The file extension to use for the temporary file.

Returns Name of the temporary file.

Return type str

TODO: Should this be in the schema module?

hed.util.io_util module

Utilities for generating and handling file names.

check_filename(*test_file*, *name_prefix=None*, *name_suffix=None*, *extensions=None*)

Return True if correct extension, suffix, and prefix.

Parameters

- **test_file** (*str*) – Path of filename to test.
- **name_prefix** (*str, None*) – An optional name_prefix for the base filename
- **name_suffix** (*str, None*) – An optional name_suffix for the base file name
- **extensions** (*list, None*) – An optional list of file extensions

Returns True if file has the appropriate format.

Return type bool

Notes

- Everything is converted to lower case prior to testing so this test should be case insensitive.

extract_suffix_path(*path*, *prefix_path*)

Return suffix of path after prefix path has been removed.

Parameters

- **path** (*str*) –

- **prefix_path** (*str*) –

Returns Suffix path.

Return type str

Notes

- This function is useful for creating files within BIDS datasets

generate_filename(*base_name*, *name_prefix=None*, *name_suffix=None*, *extension=None*)

Generate a filename for the attachment.

Parameters

- **base_name** (*str*) – Name of the base, usually the name of the file that the issues were generated from.
- **name_prefix** (*str*) – Prefix prepended to the front of the base name.
- **name_suffix** (*str*) – Suffix appended to the end of the base name.
- **extension** (*str*) – Extension to use.

Returns Name of the attachment other containing the issues.

Return type str

Notes

- The form prefix_basename_suffix + extension.

get_dir_dictionary(*dir_path*, *name_prefix=None*, *name_suffix=None*, *extensions=None*, *skip_empty=True*)

Create dictionary directory paths keys.

Parameters

- **dir_path** (*str*) – Full path of the directory tree to be traversed (no ending slash).
- **name_prefix** (*str, None*) – An optional name_prefix for the base filename.
- **name_suffix** (*str, None*) – An optional name_suffix for the base file name.
- **extensions** (*list, None*) – An optional list of file extensions.
- **skip_empty** (*bool*) – Do not put entry for directories that have no files.

Returns Dictionary with directories as keys and file lists values.

Return type dict

get_file_list(*root_path*, *name_prefix=None*, *name_suffix=None*, *extensions=None*, *exclude_dirs=None*)

Return paths satisfying various conditions.

Parameters

- **root_path** (*str*) – Full path of the directory tree to be traversed (no ending slash).
- **name_prefix** (*str, None*) – An optional name_prefix for the base filename.
- **name_suffix** (*str, None*) – The name_suffix of the paths to be extracted.
- **extensions** (*list, None*) – A list of extensions to be selected.

- **exclude_dirs** (*list, None*) – A list of paths to be excluded.

Returns The full paths.

Return type list

get_filtered_list(*file_list*, *name_prefix=None*, *name_suffix=None*, *extensions=None*)

Get list of filenames satisfying the criteria.

Everything is converted to lower case prior to testing so this test should be case insensitive.

Args: file_list (list): List of files to test. name_prefix (str): Optional name_prefix for the base filename. name_suffix (str): Optional name_suffix for the base filename. extensions (list): Optional list of file extensions (allows two periods (.tsv.gz)

Returns: list: The filtered file names.

get_path_components(*this_path*, *root_path*)

Get a list with root_path and remaining components.

Parameters

- **this_path** (*str*) – The path of a file or directory descendant of root_path
- **root_path** (*str*) – A path (no trailing separator)

Returns

A list with the first element being root_path and the remaining elements directory components to the file.

Return type list or None

Notes: this_path must be a descendant of root_path.

make_path(*root_path*, *sub_path*, *filename*)

Get path for a file, verifying all components exist.

Parameters

- **root_path** (*str*) –
- **sub_path** (*str*) –
- **filename** (*str*) –

Returns: (str) A valid realpath for the specified file.

Notes: This function is useful for creating files within BIDS datasets

parse_bids_filename(*file_path*)

Split a filename into BIDS-relevant components.

Parameters **file_path** (*str*) –

Returns: dict suffix (str) BIDS suffix name. ext (str) File extension (including the .). entity_dict (dict) Dictionary with key-value pair being (entity type, entity value).

Raises HedFileError when filename does not conform to name-value_suffix format. –

Notes

into BIDS suffix, extension, and a dictionary of entity name-value pairs.

Module contents

Data and file handling utilities.

3.4.2 hed.util.data_util

Data handling utilities involving dataframes.

Functions

add_columns(df, column_list[, value])	Add specified columns to df if not there.
check_match(ds1, ds2[, numeric])	Check two Pandas data series have the same values.
delete_columns(df, column_list)	Delete the specified columns from a dataframe.
delete_rows_by_column(df, value[, column_list])	Delete rows where columns have this value.
get_key_hash(key_tuple)	Calculate a hash key for tuple of values.
get_new_dataframe(data)	Get a new dataframe representing a tsv file.
get_row_hash(row, key_list)	Get a hash key from key column values for row.
get_value_dict(tsv_path[, key_col, value_col])	Get a dictionary of two columns of a dataframe.
make_info_dataframe(col_info, selected_col)	Get a dataframe from selected columns.
remove_quotes(df)	Remove quotes from all columns.
reorder_columns(data, col_order[, skip_missing])	Create a new dataframe with columns reordered.
replace_values(df[, values, replace_value, ...])	Replace string values in specified columns.
separate_columns(base_cols, target_cols)	Get target columns from the base list.

3.4.3 hed.util.file_util

Utilities for writing content to files and for other file manipulation.

Functions

get_version_from_xml(hed_xml_tree)	Get version from root node of an XML tree.
url_to_file(resource_url)	Write data from a URL resource into a file.
url_to_string(resource_url)	Get the data from the specified url as a string.
write_errors_to_file(issues[, extension])	Write an array of issue dictionaries to a temporary file.
write_strings_to_file(output_strings[, ...])	Write output strings to a temporary file.
write_xml_tree_2_xml_file(xml_tree[, extension])	Write an XML element tree object into a XML file.

3.4.4 hed.util.io_util

Utilities for generating and handling file names.

Functions

check_filename(test_file[, name_prefix, ...])	Return True if correct extension, suffix, and prefix.
extract_suffix_path(path, prefix_path)	Return suffix of path after prefix path has been removed.
generate_filename(base_name[, name_prefix, ...])	Generate a filename for the attachment.
get_dir_dictionary(dir_path[, name_prefix, ...])	Create dictionary directory paths keys.
get_file_list(root_path[, name_prefix, ...])	Return paths satisfying various conditions.
get_filtered_list(file_list[, name_prefix, ...])	Get list of filenames satisfying the criteria.
get_path_components(this_path, root_path)	Get a list with root_path and remaining components.
make_path(root_path, sub_path, filename)	Get path for a file, verifying all components exist.
parse_bids_filename(file_path)	Split a filename into BIDS-relevant components.

3.5 HED validators

hed.validator	Validation of HED tags.
hed.validator.HedValidator([hed_schema, ...])	Top level validation of HED strings.
hed.validator.TagValidator([hed_schema, ...])	Validation for individual HED tags.
hed.validator.tag_validator_util	Utilities to support HED validation.

3.5.1 hed.validator package

Submodules

hed.validator.hed_validator module

This module contains the HedValidator class which is used to validate the tags in a HED string or a file. The file types include .tsv, .txt, and .xlsx. To get the validation issues after creating a HedValidator class call the get_validation_issues() function.

class HedValidator(*hed_schema=None*, *run_semantic_validation=True*)

Bases: *hed.models.hed_ops.HedOps*

Top level validation of HED strings.

hed.validator.tag_validator module

This module is used to validate the HED tags as strings.

class TagValidator(*hed_schema=None*, *run_semantic_validation=True*)

Bases: `object`

Validation for individual HED tags.

CAMEL_CASE_EXPRESSION = '([A-Z-]+\\s*[a-z-]*)+'

CLOSING_GROUP_CHARACTER = ')'

COMMA = ','

DEFAULT_ALLOWED_PLACEHOLDER_CHARS = '.+-^ _#'

INVALID_STRING_CHARS = '[]{}~'

OPENING_GROUP_CHARACTER = '('

TAG_ALLOWED_CHARS = '-_/'

check_capitalization(*original_tag*)

Report warning if incorrect tag capitalization.

Parameters **original_tag** (HedTag) – The original tag used to report the warning.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_count_tag_group_parentheses(*hed_string*)

Report unmatched parentheses.

Parameters **hed_string** (*str*) – A hed string.

Returns A list of validation list. Each issue is a dictionary.

Return type list

check_delimiter_issues_in_hed_string(*hed_string*)

Report missing commas or commas in value tags.

Parameters **hed_string** (*str*) – A hed string.

Returns A validation issues list. Each issue is a dictionary.

Return type list

check_for_invalid_extension_chars(*original_tag*)

Report invalid characters in extension/value.

Parameters **original_tag** (HedTag) – The original tag that is used to report the error.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_for_placeholder(*original_tag*, *is_definition=False*)

Report invalid placeholder characters.

Parameters

- **original_tag** (HedTag) – The HedTag to be checked
- **is_definition** (*bool*) – If True, placeholders are allowed.

Returns Validation issues. Each issue is a dictionary.

Return type list

Notes

- Invalid placeholder may appear in the extension/value portion of a tag.

check_for_required_tags(*tags*)

Report missing required tags.

Parameters **tags** (*list*) – HedTags containing the tags.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_invalid_character_issues(*hed_string*)

Report invalid characters.

Parameters **hed_string** (*str*) – A hed string.

Returns Validation issues. Each issue is a dictionary.

Return type list

Notes

- Invalid tag characters are defined by TagValidator.INVALID_STRING_CHARS.

check_multiple_unique_tags_exist(*tags*)

Report if multiple identical unique tags exist

A unique Term can only appear once in a given HedString. Unique terms are terms with the 'unique' property in the schema.

Parameters **tags** (*list*) – HedTags containing the tags.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_tag_exists_in_schema(*original_tag*, *check_for_warnings=False*)

Report invalid tag or doesn't take a value.

Parameters

- **original_tag** (HedTag) – The original tag that is used to report the error.
- **check_for_warnings** (*bool*) – If True, also check for warnings.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_tag_formatting(*original_tag*)

Report repeated or erroneous slashes.

Parameters **original_tag** (HedTag) – The original tag that is used to report the error.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_tag_invalid_chars(*original_tag*, *allow_placeholders*)

Report invalid characters in the given tag.

Parameters

- **original_tag** (HedTag) – The original tag that is used to report the error.
- **allow_placeholders** (*bool*) – Allow placeholder characters(#) if True.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_tag_level_issue(*original_tag_list*, *is_top_level*, *is_group*)

Report tags incorrectly positioned in hierarchy.

Parameters

- **original_tag_list** (*list*) – HedTags containing the original tags.
- **is_top_level** (*bool*) – If True, this group is a "top level tag group"
- **is_group** (*bool*) – If true group should be contained by parenthesis

Returns Validation issues. Each issue is a dictionary.

Return type list

Notes

- Top-level groups can contain definitions, Onset, etc tags.

check_tag_requires_child(*original_tag*)

Report if tag is a leaf with 'requiredTag' attribute.

Parameters **original_tag** (HedTag) – The original tag that is used to report the error.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_tag_unit_class_units_are_valid(*original_tag*, *check_for_warnings*)

Report incorrect unit class or units.

Parameters

- **original_tag** (HedTag) – The original tag that is used to report the error.
- **check_for_warnings** (*bool*) – Indicates whether to check for warnings.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_tag_unit_class_units_exist(*original_tag*)

Report warning if tag has a unit class tag with no units.

Parameters **original_tag** (HedTag) – The original tag that is used to report the error.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_tag_value_class_valid(*original_tag*)

Report an invalid value portion.

Parameters **original_tag** (HedTag) – The original tag that is used to report the error.

Returns Validation issues.

Return type list

pattern_doubleslash = re.compile('([\\t/]{2,}|^/|/$)')

run_all_tags_validators(*tags*, *check_for_warnings*)

Validate the multi-tag properties in a hed string.

Parameters

- **tags** (*list*) – A list containing the HedTags in a HED string.
- **check_for_warnings** (*bool*) – If True, also check for warnings.

Returns The validation issues associated with the tags in a HED string. Each issue is a dictionary.

Return type list

Notes

- Multi-tag properties include required tags.

run_hed_string_validators(*hed_string_obj*)

Basic high level checks of the hed string

Parameters **hed_string_obj** (HedString) – A HED string.

Returns The validation issues associated with a HED string. Each issue is a dictionary.

Return type list

Notes

- Used for basic invalid characters or bad delimiters.

run_individual_tag_validators(*original_tag*, *check_for_warnings*, *allow_placeholders=False*, *is_definition=False*)

Runs the hed_ops on the individual tags.

Parameters

- **original_tag** (HedTag) – A original tag.
- **check_for_warnings** (*bool*) – If True, also check for warnings.
- **allow_placeholders** (*bool*) – Allow value class or extensions to be placeholders rather than a specific value.
- **is_definition** (*bool*) – This tag is part of a Definition, not a normal line.

Returns The validation issues associated with the top-level tags. Each issue is dictionary.

Return type list

run_tag_level_validators(*original_tag_list*, *is_top_level*, *is_group*)

Run hed_ops at each level in a HED string.

Parameters

- **original_tag_list** (*list*) – A list containing the original HedTags.
- **is_top_level** (*bool*) – If True, this group is a "top level tag group".
- **is_group** (*bool*) – If true, group is contained by parenthesis.

Returns The validation issues associated with each level in a HED string.

Return type list

Notes

- This is for the top-level, all groups, and nested groups.
- This can contain definitions, Onset, etc tags.

validate_value_class_type(*unit_or_value_portion*, *valid_types*)

Report invalid unit or valid class values.

Parameters

- **unit_or_value_portion** (*str*) – The value portion to validate.
- **valid_types** (*list*) – The names of value class or unit class types (e.g. dateTime or dateTimeClass).

Returns True if this is one of the valid_types validators.

Return type type_valid (bool)

hed.validator.tag_validator_util module

Utilities to support HED validation.

is_clock_face_time(*time_string*)

Check if a valid HH:MM time string.

Parameters **time_string** (*str*) – A time string.

Returns True if the time string is valid. False, if otherwise.

Return type bool

Notes

- This is deprecated and has no expected use going forward.

is_date_time(*date_time_string*)

Check if the specified string is a valid datetime.

Parameters **date_time_string** (*str*) – A datetime string.

Returns True if the datetime string is valid. False, if otherwise.

Return type bool

Notes

- ISO 8601 datetime string.

validate_numeric_value_class(*numeric_string*)

Checks to see if valid numeric value.

Parameters **numeric_string** (*str*) – A string that should be only a number with no units.

Returns True if the numeric string is valid. False, if otherwise.

Return type bool

validate_text_value_class(*text_string*)

Placeholder for eventual text value class validation

Parameters **text_string** (*str*) – Text class.

Returns True

Return type bool

Module contents

Validation of HED tags.

3.5.2 hed.validator.HedValidator

class HedValidator(*hed_schema=None*, *run_semantic_validation=True*)

Top level validation of HED strings.

__init__(*hed_schema=None*, *run_semantic_validation=True*)

Constructor for the HedValidator class.

Parameters

- **hed_schema** (*HedSchema or HedSchemaGroup*) – HedSchema object to use for validation.
- **run_semantic_validation** (*bool*) – True if the validator should check the HED data against a schema.

Methods

__init__([hed_schema, run_semantic_validation])	Constructor for the HedValidator class.

class HedValidator(*hed_schema=None*, *run_semantic_validation=True*)

Bases: *hed.models.hed_ops.HedOps*

Top level validation of HED strings.

3.5.3 hed.validator.TagValidator

class TagValidator(*hed_schema=None*, *run_semantic_validation=True*)

Validation for individual HED tags.

__init__(*hed_schema=None*, *run_semantic_validation=True*)

Constructor for the Tag_Validator class.

Parameters

- **hed_schema** (HedSchema) – A HedSchema object.
- **run_semantic_validation** (*bool*) – True if the validator should check the HED data against a schema.

Returns A Tag_Validator object.

Return type *TagValidator*

Methods

__init__([hed_schema, run_semantic_validation])	Constructor for the Tag_Validator class.
check_capitalization(original_tag)	Report warning if incorrect tag capitalization.
check_count_tag_group_parentheses(hed_string)	Report unmatched parentheses.
check_delimiter_issues_in_hed_string(hed_string)	Report missing commas or commas in value tags.
check_for_invalid_extension_chars(original_tag)	Report invalid characters in extension/value.
check_for_placeholder(original_tag[, ...])	Report invalid placeholder characters.
check_for_required_tags(tags)	Report missing required tags.
check_invalid_character_issues(hed_string)	Report invalid characters.
check_multiple_unique_tags_exist(tags)	Report if multiple identical unique tags exist
check_tag_exists_in_schema(original_tag[, ...])	Report invalid tag or doesn't take a value.
check_tag_formatting(original_tag)	Report repeated or erroneous slashes.
check_tag_invalid_chars(original_tag, ...)	Report invalid characters in the given tag.
check_tag_level_issue(original_tag_list, ...)	Report tags incorrectly positioned in hierarchy.
check_tag_requires_child(original_tag)	Report if tag is a leaf with 'requiredTag' attribute.
check_tag_unit_class_units_are_valid(...)	Report incorrect unit class or units.
check_tag_unit_class_units_exist(original_tag)	Report warning if tag has a unit class tag with no units.
check_tag_value_class_valid(original_tag)	Report an invalid value portion.
run_all_tags_validators(tags, check_for_warnings)	Validate the multi-tag properties in a hed string.
run_hed_string_validators(hed_string_obj)	Basic high level checks of the hed string
run_individual_tag_validators(original_tag, ...)	Runs the hed_ops on the individual tags.
run_tag_level_validators(original_tag_list, ...)	Run hed_ops at each level in a HED string.
validate_value_class_type(...)	Report invalid unit or valid class values.

Attributes

CAMEL_CASE_EXPRESSION
CLOSING_GROUP_CHARACTER
COMMA
DEFAULT_ALLOWED_PLACEHOLDER_CHARS
INVALID_STRING_CHARS
OPENING_GROUP_CHARACTER
TAG_ALLOWED_CHARS
pattern_doubleslash

class TagValidator(*hed_schema=None*, *run_semantic_validation=True*)

Bases: `object`

Validation for individual HED tags.

CAMEL_CASE_EXPRESSION = '([A-Z-]+\\s*[a-z-]*)+'

CLOSING_GROUP_CHARACTER = ')'

COMMA = ','

DEFAULT_ALLOWED_PLACEHOLDER_CHARS = '.+-^ _#'

INVALID_STRING_CHARS = '[]{}~'

OPENING_GROUP_CHARACTER = '('

TAG_ALLOWED_CHARS = '-_/'

check_capitalization(*original_tag*)

Report warning if incorrect tag capitalization.

Parameters **original_tag** (HedTag) – The original tag used to report the warning.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_count_tag_group_parentheses(*hed_string*)

Report unmatched parentheses.

Parameters **hed_string** (*str*) – A hed string.

Returns A list of validation list. Each issue is a dictionary.

Return type list

check_delimiter_issues_in_hed_string(*hed_string*)

Report missing commas or commas in value tags.

Parameters **hed_string** (*str*) – A hed string.

Returns A validation issues list. Each issue is a dictionary.

Return type list

check_for_invalid_extension_chars(*original_tag*)

Report invalid characters in extension/value.

Parameters **original_tag** (HedTag) – The original tag that is used to report the error.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_for_placeholder(*original_tag*, *is_definition=False*)

Report invalid placeholder characters.

Parameters

- **original_tag** (HedTag) – The HedTag to be checked
- **is_definition** (*bool*) – If True, placeholders are allowed.

Returns Validation issues. Each issue is a dictionary.

Return type list

Notes

- Invalid placeholder may appear in the extension/value portion of a tag.

check_for_required_tags(*tags*)

Report missing required tags.

Parameters **tags** (*list*) – HedTags containing the tags.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_invalid_character_issues(*hed_string*)

Report invalid characters.

Parameters **hed_string** (*str*) – A hed string.

Returns Validation issues. Each issue is a dictionary.

Return type list

Notes

- Invalid tag characters are defined by TagValidator.INVALID_STRING_CHARS.

check_multiple_unique_tags_exist(*tags*)

Report if multiple identical unique tags exist

A unique Term can only appear once in a given HedString. Unique terms are terms with the 'unique' property in the schema.

Parameters **tags** (*list*) – HedTags containing the tags.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_tag_exists_in_schema(*original_tag*, *check_for_warnings=False*)

Report invalid tag or doesn't take a value.

Parameters

- **original_tag** (HedTag) – The original tag that is used to report the error.
- **check_for_warnings** (*bool*) – If True, also check for warnings.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_tag_formatting(*original_tag*)

Report repeated or erroneous slashes.

Parameters **original_tag** (HedTag) – The original tag that is used to report the error.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_tag_invalid_chars(*original_tag*, *allow_placeholders*)

Report invalid characters in the given tag.

Parameters

- **original_tag** (HedTag) – The original tag that is used to report the error.
- **allow_placeholders** (*bool*) – Allow placeholder characters(#) if True.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_tag_level_issue(*original_tag_list*, *is_top_level*, *is_group*)

Report tags incorrectly positioned in hierarchy.

Parameters

- **original_tag_list** (*list*) – HedTags containing the original tags.
- **is_top_level** (*bool*) – If True, this group is a "top level tag group"
- **is_group** (*bool*) – If true group should be contained by parenthesis

Returns Validation issues. Each issue is a dictionary.

Return type list

Notes

- Top-level groups can contain definitions, Onset, etc tags.

check_tag_requires_child(*original_tag*)

Report if tag is a leaf with 'requiredTag' attribute.

Parameters **original_tag** (HedTag) – The original tag that is used to report the error.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_tag_unit_class_units_are_valid(*original_tag*, *check_for_warnings*)

Report incorrect unit class or units.

Parameters

- **original_tag** (HedTag) – The original tag that is used to report the error.
- **check_for_warnings** (*bool*) – Indicates whether to check for warnings.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_tag_unit_class_units_exist(*original_tag*)

Report warning if tag has a unit class tag with no units.

Parameters **original_tag** (HedTag) – The original tag that is used to report the error.

Returns Validation issues. Each issue is a dictionary.

Return type list

check_tag_value_class_valid(*original_tag*)

Report an invalid value portion.

Parameters **original_tag** (HedTag) – The original tag that is used to report the error.

Returns Validation issues.

Return type list

pattern_doubleslash = re.compile('([\\t/]{2,}|^/|/$)')

run_all_tags_validators(*tags*, *check_for_warnings*)

Validate the multi-tag properties in a hed string.

Parameters

- **tags** (*list*) – A list containing the HedTags in a HED string.
- **check_for_warnings** (*bool*) – If True, also check for warnings.

Returns The validation issues associated with the tags in a HED string. Each issue is a dictionary.

Return type list

Notes

- Multi-tag properties include required tags.

run_hed_string_validators(*hed_string_obj*)

Basic high level checks of the hed string

Parameters **hed_string_obj** (HedString) – A HED string.

Returns The validation issues associated with a HED string. Each issue is a dictionary.

Return type list

Notes

- Used for basic invalid characters or bad delimiters.

run_individual_tag_validators(*original_tag*, *check_for_warnings*, *allow_placeholders=False*, *is_definition=False*)

Runs the hed_ops on the individual tags.

Parameters

- **original_tag** (HedTag) – A original tag.
- **check_for_warnings** (*bool*) – If True, also check for warnings.
- **allow_placeholders** (*bool*) – Allow value class or extensions to be placeholders rather than a specific value.
- **is_definition** (*bool*) – This tag is part of a Definition, not a normal line.

Returns The validation issues associated with the top-level tags. Each issue is dictionary.

Return type list

run_tag_level_validators(*original_tag_list*, *is_top_level*, *is_group*)

Run hed_ops at each level in a HED string.

Parameters

- **original_tag_list** (*list*) – A list containing the original HedTags.
- **is_top_level** (*bool*) – If True, this group is a "top level tag group".
- **is_group** (*bool*) – If true, group is contained by parenthesis.

Returns The validation issues associated with each level in a HED string.

Return type list

Notes

- This is for the top-level, all groups, and nested groups.
- This can contain definitions, Onset, etc tags.

validate_value_class_type(*unit_or_value_portion*, *valid_types*)

Report invalid unit or valid class values.

Parameters

- **unit_or_value_portion** (*str*) – The value portion to validate.
- **valid_types** (*list*) – The names of value class or unit class types (e.g. dateTime or dateTimeClass).

Returns True if this is one of the valid_types validators.

Return type type_valid (bool)

3.5.4 hed.validator.tag_validator_util

Utilities to support HED validation.

Functions

is_clock_face_time(time_string)	Check if a valid HH:MM time string.
is_date_time(date_time_string)	Check if the specified string is a valid datetime.
validate_numeric_value_class(numeric_string)	Checks to see if valid numeric value.
validate_text_value_class(text_string)	Placeholder for eventual text value class validation

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

h

INDEX

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W