Package **yaslha** Manual

Sho Iwamoto / Misho

Dipartimento di Fisica e Astronomia, Università degli Studi di Padova
Via Marzolo 8, I-35131 Padova, Italia

sho.iwamoto@pd.infn.it

package **yaslha** for Python 3.5+
version: `0.2.1`
repository: https://github.com/misho104/yaslha
document: https://yaslha.readthedocs.io/
License: MIT License

Abstract

`yaslha` is an yet-another Python 3 package to handle SUSY Les Houches Accord (SLHA) files.

Important: This PDF is an auto-generated draft. Since finalization processes have not yet completed for this LaTeX source, there are several issues in typesetting and referencing, especially in bibliography. Official PDF documents may be released in GitHub repository.

---

1. Introduction

1.1. Background

SUSY Les Houches accord (SLHA) [1] is a data format widely used in particle physics community. It has been extended as SLHA2 [2] or flavor Les Houches accord (FLHA) [3]....

1.2. This Package

`yaslha` is an yet-another Python 3 package to handle SLHA data.

You can include this package in your Python code for more customization.... The appendices contain additional information of this package. Appendix A is the full API reference of this package.

2. Quick Start

2.1. Install and uninstall

This package requires Python 3.5 or above with `pip` package manager. You may check the versions by

```
$ python -V
Python 3.6.7
$ pip -V
pip 19.0.3
```

With the set-up, you can simply install this package via PyPI:

```
$ pip install yaslha            # to install
$ pip install --upgrade yaslha  # or to upgrade

Collecting yaslha
...
Successfully installed yaslha-(version)
```

You can also instantly uninstall this package by

```
$ pip uninstall yaslha
```

3. Conclusion

This paper introduces a Python 3 package "`yaslha`" to handle SLHA format, which can be installed via `PyPI`. ... (TBW)

This package is managed on GitHub: `misho104/yaslha`. Bug reports, comments, questions, and contribution are welcome on the website.

Acknowledgments

(To be written.)

This work is under much influence from the packages `PySLHA` by Andy Buckley and `pylha` by David M. Straub.

The author credits PyPI as the distributor, GitHub as the repository host, TravisCI and CoverAlls as the host of continuous integration with coverage, and ReadTheDocs as the host of documentation, of this package.

The author (SI) is supported by the MIUR-PRIN project 2015P5SBHT 003 "Search for the Fundamental Laws and Constituents" by INFN (to be clarified),...

A. API Reference: yaslha

A.1. yaslha package

Package to handle SLHA-format files and data.

yaslha.**parse**(*text*, *input_type='AUTO'*, *parser=None*, ***kwargs*)
Parse a text to return an SLHA object.

yaslha.**dump**(*slha*, *output_type='SLHA'*, *dumper=None*, ***kwargs*)
Output a dumped string of an SLHA object.

yaslha.**parse_file**(*path*, ***kwargs*)
Parse a file to return an SLHA object.

yaslha.**dump_file**(*data*, *path*, ***kwargs*)
Write into a file a dumped string of an SLHA object.

A.2. yaslha.slha module

Module of SLHA object class.

class yaslha.slha.**SLHA**
Bases: `object`

SLHA object, representing a SLHA-format text.

add_block(*obj: Union[Block, InfoBlock, Decay]*) → *None*
Add a block to SLHA file.

The name is automatically detected from the object.

__getitem__(*key: Any*) → *Any*
Get values of SLHA object or deeper.

`SLHA[str]` and `SLHA[int]` give the specified block and decay block, respectively. For ordinary blocks, further referencing is possible as `SLHA[str, *key]`, while decay blocks refuse such referencing for safety.

get(**key, default: Any = None*) → *Any*
Return the value if exists, or default.

__setitem__(*key: Any*, *value: Any*) → *None*
Set a value of SLHA object or deeper.

__delitem__(*key: Any*) → *None*
Delete values of SLHA object or deeper.

normalize(*blocks: bool = True*, *decays: bool = True*) → *None*
Normalize the head-lines so that names/pids match the dict keys.

merge(*another: yaslha.slha.SLHA*) → *None*
Merge another SLHA data into this object.

A.3. yaslha.block module

Block-like object of SLHA data.

class yaslha.block.**GenericBlock**
Bases: typing.Generic

Block-like object containing comments.

comment
Give the interface to comments.

classmethod new(*obj: Union[int, str, yaslha.line.DecayHeadLine, yaslha.line.BlockHeadLine]*) → *Union[yaslha.block.Block, yaslha.block.InfoBlock, yaslha.block.Decay]*
Create a GenericBlock object according to the argument.

class yaslha.block.**AbsBlock**(*obj: Union[yaslha.line.BlockHeadLine, str]*)
Bases: yaslha.block.GenericBlock, typing.Generic

Abstract class for SLHA blocks.

name
Return the name of block (always in upper case).

q
Return the Q value.

__getitem__(*key: KT*) → *Union[VT, Sequence[VT]]*
Return the value corresponding to the key.

__setitem__(*key: KT, value: VT*) → *None*
Set the value for the key.

__delitem__(*key: KT*) → *None*
Delete the value for the key.

update_line(*line: LT*) → *None*
Add the line to the block, overriding if exists.

keys(*sort: bool = False*) → *Iterator[KT]*
Return the keys.

__iter__(*sort: bool = False*) → *Iterator[KT]*
Return the keys.

items(*sort: bool = False*) → *Iterator[Tuple[KT, VT]]*
Return (key, value) tuples.

class yaslha.block.**Block**(*obj: Union[yaslha.line.BlockHeadLine, str]*)
Bases: yaslha.block.AbsBlock

SLHA block that has one value for one key.

__getitem__(*key: Union[None, int, Sequence[int]]*) → *float*
Return the value corresponding to the key.

__setitem__(*key: Union[None, int, Sequence[int]], value: float*) → *None*
Set the value for the key.

__delitem__(*key: Union[None, int, Sequence[int]]*) → *None*
Delete the value for the key.

update_line(*line: yaslha.line.ValueLine*) → *None*
Add the line to the block, overriding if exists.

merge(*another: Union[yaslha.block.Block, yaslha.block.InfoBlock]*) → *None*
Merge another block.

get(**key, default: T*) → *Union[float, T]*
Return the value for the key if exists, or default value.

keys(*sort: bool = False*) → *Iterator[Union[None, int, Sequence[int]]]*
Return the keys.

__iter__(*sort: bool = False*) → *Iterator[Union[None, int, Sequence[int]]]*
Return the keys.

items(*sort: bool = False*) → *Iterator[Tuple[Union[None, int, Sequence[int]], float]]*
Return (key, value) tuples.

class yaslha.block.**InfoBlock**(*obj: Union[yaslha.line.BlockHeadLine, str]*)
Bases: yaslha.block.AbsBlock

SLHA block that may have multiple values for one key.

__getitem__(*key: int*) → *Sequence[str]*
Return the value corresponding to the key.

__setitem__(*key: int, value: Sequence[str]*) → *None*
Set the value for the key.

__delitem__(*key: int*) → *None*
Delete the value for the key.

update_line(*line: yaslha.line.InfoLine*) → *None*
Add the line to the block, overriding if exists.

append_line(*line: yaslha.line.InfoLine*) → *None*
Add the line, appending to the existing one if exists.

append(*key: int, value: str*) → *None*
Append the value for the key.

merge(*another: Union[yaslha.block.Block, yaslha.block.InfoBlock]*) → *None*
Merge another block.

keys(*sort: bool = False*) → *Iterator[int]*
Return the keys.

__iter__(*sort: bool = False*) → *Iterator[int]*
Return the keys.

items(*sort: bool = False*) → *Iterator[Tuple[int, str]]*
Return (key, value) tuples.

class yaslha.block.**Decay**(*obj: Union[yaslha.line.DecayHeadLine, int]*)
Bases: yaslha.block.GenericBlock

Decay block.

br_normalize_threshold = 1e-06

pid
Return the pid of mother particle.

width
Return the total width.

update_line(*line: yaslha.line.DecayLine*) → *None*
Add the line to the block, overriding if exists.

br(**key*) → *float*
Return the BR of given channel.

partial_width(**key*) → *float*
Return the width of given channel.

keys(*sort: bool = False*) → *Iterator[Sequence[int]]*
Return the keys.

__iter__(*sort: bool = False*) → *Iterator[Sequence[int]]*
Return the keys.

items_br(*sort=False*)
Return (key, BR) tuples, sorted by the BR.

items_partial_width(*sort=False*)
Return (key, width) tuples, sorted by the BR.

normalize(*force: bool = False*) → *None*
Normalize the branching ratios.

This method normalize all the branching ratios so that the sum becomes unity or less. In particular, if `force` is set True, they are normalized so that the sum becomes unity, regardless of the current value.

If `force` is False and the sum is less than one, the branching ratio is not normalized, assuming that some decay channels are not listed. If `force` is False and the sum slightly exceeds the unity, the branching ratios are normalized, while if the excess is larger than `br_normalize_threshold`, `ValueError` is raised.

set_partial_width(**args*) → *None*
Update the partial width and recalculate BRs of all channels.

remove(**key*) → *None*
Remove the channel and recalculate BRs of all the other channels.

A.4. yaslha.comment module

Module of a class to handle comment interface.

class yaslha.comment.**CommentInterface**(*block: GenericBlock[KTG, CT]*)
Bases: `typing.Generic`

Accessor object to the comments in blocks.

pre
Return pre-comment interface.

__getitem__(*key: Union[KTG, typing_extensions.Literal['head'][head]]*) → *Union[CT, str]*
Return comment.

__setitem__(*key: Union[KTG, typing_extensions.Literal['head'][head]], value: Union[CT, str, None]*) → *None*
Set comment.

__call__(*key: Union[KTG, typing_extensions.Literal['head'][head]], **kw*) → *Union[CT, str]*
Return comment in specified format.

class yaslha.comment.**PreCommentInterface**(*block: GenericBlock[KTG, CT]*)
Bases: typing.Generic

Accessor object to the pre-line comments in blocks.

__getitem__(*key: Union[KTG, typing_extensions.Literal['head'][head]]*) → *List[str]*
Return pre-line comment.

__setitem__(*key: Union[KTG, typing_extensions.Literal['head'][head]], value: Optional[List[str]]*) → *None*
Set pre-line comment.

__call__(*key: Union[KTG, typing_extensions.Literal['head'][head]], **kw*) → *List[str]*
Return pre-line comment in specified format.

A.5. yaslha.line module

Module to describe each type of lines.

The line hierarchy is given by:

- AbsLine - BlockHeadLine - DecayHeadLine - InfoLine - ValueLine
  - NoIndexLine
  - OneIndexLine
  - TwoIndexLine
  - ThreeIndexLine
  - DecayLine
  - CommentLine

class yaslha.line.**LineOutputOption**
Bases: object

Class to hold all the options on dumping lines.

class yaslha.line.**AbsLine**(***kwargs*)
Bases: object

Abstract class for SLHA-line like objects.

output_option = <yaslha.line.LineOutputOption object>

classmethod pattern() → *Pattern[str]*
Return a regexp pattern matching a SLHA line for the type.

classmethod construct(*line: str*) → *Optional[LT]*
Construct an object from a line if it maches the pattern.

to_slha(*option: Optional[yaslha.line.LineOutputOption] = None*) → *List[str]*
Return the lines for SLHA-format, including pre-comments.

dump() → *List[Union[str, float]]*
Return a list representing the line.

classmethod from_dump(*dump: Sequence[Any], **kw*) → *LT*
Return an object from dumped value.

class yaslha.line.**BlockHeadLine**(*name, q=None, comment=None*)
Bases: yaslha.line.AbsLine

Line for block header.

name
Return name in upper case.

class yaslha.line.**DecayHeadLine**(*pid: Union[str, int], width: Union[str, float], comment: Optional[str] = None*)
Bases: yaslha.line.AbsLine

A line with format `('DECAY',1x,I9,3x,1P,E16.8,0P,3x,'#',1x,A)`.

class yaslha.line.**InfoLine**(*key, value, comment=None*)
Bases: yaslha.line.AbsLine

Class for lines of INFO blocks.

An info-block line is given by `format(1x,I5,3x,A)`, which is not exclusive and matches other patterns. This class accept multi-lines, and thus values are List[str] and comments are multi-lined string, internally kept as List[str].

class yaslha.line.**ValueLine**(*key: Union[None, int, Sequence[int]], value: Union[str, float], comment: Optional[str] = None*)
Bases: yaslha.line.AbsLine

Abstract class for value lines in ordinary blocks.

classmethod new(*key: Union[None, int, Sequence[int]], value: Union[str, float], comment: Optional[str] = None*) → *LT2*
Construct line object according to the type of key.

The resulting object is an instance of a subclass of ValueLine but not DecayLine.

class yaslha.line.**NoIndexLine**(*value, comment=None*)
Bases: yaslha.line.ValueLine

A line with `format(9x, 1P, E16.8, 0P, 3x, '#', 1x, A)`.

class yaslha.line.**OneIndexLine**(*i, value, comment=None*)
Bases: yaslha.line.ValueLine

A line with `format(1x,I5,3x,1P,E16.8,0P,3x,'#',1x,A)`.

class yaslha.line.**TwoIndexLine**(*i1, i2, value, comment=None*)
Bases: yaslha.line.ValueLine

A line with `format(1x,I2,1x,I2,3x,1P,E16.8,0P,3x,'#',1x,A)`.

class yaslha.line.**ThreeIndexLine**(*i1, i2, i3, value, comment=None*)
Bases: yaslha.line.ValueLine

A line with `format(1x,I2,1x,I2,1x,I2,3x,1P,E16.8,0P,3x,'#',1x,A)`.

class yaslha.line.**DecayLine**(*br, channel, nda=None, comment=None*)
Bases: yaslha.line.ValueLine

A decay line `(3x,1P,E16.8,0P,3x,I2,3x,N (I9,1x),2x,'#',1x,A)`.

br
Return the branching ratio.

class yaslha.line.**CommentLine**(*comment: Optional[str] = None*)
Bases: yaslha.line.AbsLine

Comment line.

This object is prepared for compatibility and not intended to be inserted in blocks or decay-blocks; therefore dumping methods are not implemented.

A.6. yaslha.parser module

Parsers for SLHA package.

Parsers for SLHA-format, JSON-format, and YAML-format are provided.

class yaslha.parser.**SLHAParser**(***kw*)
Bases: object

SLHA-format file parser.

parse(*text: str*) → *yaslha.slha.SLHA*
Parse SLHA format text and return SLHA object.

A.7. yaslha.dumper module

Dumpers to write SLHA data in various format.

class yaslha.dumper.**BlocksOrder**
Bases: enum.Enum

Options for block ordering.

DEFAULT = 0

KEEP = 1

ABC = 2

class yaslha.dumper.**ValuesOrder**
Bases: enum.Enum

Options for value ordering.

DEFAULT = 0

KEEP = 1

SORTED = 2

class yaslha.dumper.**CommentsPreserve**
Bases: enum.Enum

Options for comment handling.

NONE = 0

TAIL = 1

ALL = 2

keep_line
Return if to keep line-level comments.

keep_tail
Return if to keep tail comments of value lines.

class yaslha.dumper.**AbsDumper**(***kw*)
Bases: object

Abstract class for YASLHA dumpers.

config(*k: str*) → *Any*
Get a current value of configuration.

set_config(*k: str, v: Any*) → *None*
Set configuration.

dump(*slha: yaslha.slha.SLHA*) → *str*
Return dumped string of an SLHA object.

class yaslha.dumper.**SLHADumper**(***kw*)
Bases: yaslha.dumper.AbsDumper

A dumper class for SLHA output.

config(*k: str*) → *Any*
Get a current value of configuration.

set_config(*k: str, v: Any*) → *None*
Set configuration.

dump(*slha: yaslha.slha.SLHA*) → *str*
Return SLHA-format text of an SLHA object.

dump_block(*block, document_block=False*)
Return SLHA-format text of a block.

class yaslha.dumper.**AbsMarshalDumper**(***kw*)
Bases: yaslha.dumper.AbsDumper

An abstract class for dumpers handling marshaled data.

SCHEME_VERSION = 3

config(*k: str*) → *Any*
Get a current value of configuration.

set_config(*k: str, v: Any*) → *None*
Set configuration.

marshal(*slha*)
Return Mashaled object of an SLHA object.

marshal_block(*block: Union[yaslha.block.Block, yaslha.block.InfoBlock, yaslha.block.Decay]*)
→ *Mapping[str, Any]*
Return marshaled object of a block.

class yaslha.dumper.**YAMLDumper**(***kw*)
Bases: yaslha.dumper.AbsMarshalDumper

A dumper for YAML output.

dump(*slha: yaslha.slha.SLHA*) → *str*
Return YAML-format text of an SLHA object.

class yaslha.dumper.**JSONDumper**(***kw*)
Bases: yaslha.dumper.AbsMarshalDumper

A dumper for JSON output.

dump(*slha: yaslha.slha.SLHA*) → *str*
Return JSON-format text of an SLHA object.

A.8. yaslha.script module

Scripts of this package.

A.9. yaslha.config module

Configuration handlers.

class yaslha.config.**SectionWrapper**(*data: configparser.SectionProxy*)
Bases: object

A wrapper class of configparser.SectionProxy.

get_enum(*key: str, enum_class: Type[EnumType]*) → *EnumType*
Get an item as an Enum-class object.

get_list(*key: str*) → *List[str]*
Get a List[str] object.

class yaslha.config.**Config**
Bases: configparser.ConfigParser

Dictionary to store the configurations.

A.10. yaslha.utility module

Utility module.

yaslha.utility.**sort_blocks_default**(*block_names: Sequence[str]*) → *List[str]*
Sort block names according to specified order.

yaslha.utility.**sort_pids_default**(*pids: Sequence[T]*) → *List[Union[T, int]]*
Sort block names according to specified order.

References

[1] P. Z. Skands *et al.*, *SUSY Les Houches accord: Interfacing SUSY spectrum calculators, decay packages, and event generators*, JHEP **07** (2004) 036 [hep-ph/0311123].

[2] B. C. Allanach *et al.*, *SUSY Les Houches Accord 2*, Comput. Phys. Commun. **180** (2009) 8–25 [arXiv:0801.0045].

[3] F. Mahmoudi *et al.*, *Flavour Les Houches Accord: Interfacing Flavour related Codes*, Comput. Phys. Commun. **183** (2012) 285–298 [arXiv:1008.0762].

Python Module Index

y

Index