Python Standard Library List Documentation

Release 0.9.0

Jack Maney

Jun 29, 2023

CONTENTS

This package includes lists of all of the standard libraries for Python 2.6 through 3.11.

Note: If you're on Python 3.10 or newer, you **probably don't need this library**. See sys.stdlib_module_names and sys.builtin_module_names for similar functionality.

CHAPTER

ONE

CONTENTS

1.1 Installation

Most end users should use `pip` to install this package:

```
python -m pip install stdlib-list
```

If for whatever reason you need to install `stdlib-list` from the source repository instead:

```
git clone https://github.com/pypi/stdlib-list
cd stdlib-list
python -m pip install .
```

1.2 Usage

1.2.1 Getting The List of Libraries

`stdlib_list.stdlib_list` returns the list of libraries in stdlib for any given version (by default, current python version).

In particular:

```
In [1]: from stdlib_list import stdlib_list

In [2]: libs = stdlib_list("3.4")

In [3]: libs[:6]
Out[3]: ['__future__', '__main__', '_dummy_thread', '_thread', 'abc', 'aifc']
```

1.2.2 Checking if a Module is part of stdlib

`stdlib_list.in_stdlib` provides an efficient way to check if a module name is part of stdlib. It relies on `@lru_cache` to cache the stdlib list and query results for similar calls. Therefore it is much more efficient than `module_name in stdlib_list()` especially if you wish to perform multiple checks.

In particular:

```
>>> from stdlib_list import in_stdlib
>>> in_stdlib('zipimport')  # built in
True
>>> in_stdlib('math')       # C-API stdlib module, but linked as extension (on my␣
↪machine)
True
>>> in_stdlib('numpy')      # C-API extension, not stdlib
False
>>> in_stdlib('sys')        # built-in (and special)
True
>>> in_stdlib('os')         # Python code in stdlib
True
>>> in_stdlib('requests')   # Python code, not stdlib
False
```

`stdlib_list.`**`in_stdlib`**(*module_name: str*, *version: str | None = None*) → bool

Return a `bool` indicating if module `module_name` is in the list of stdlib symbols for python version `version`. If `version` is `None` (default), the version of current python interpreter is used.

Note that `True` will be returned for built-in modules too, since this project considers they are part of stdlib. See :issue:21.

It relies on `@lru_cache` to cache the stdlib list and query results for similar calls. Therefore it is much more efficient than `module_name in stdlib_list()` especially if you wish to perform multiple checks.

Parameters

- **`module_name`** (*`str`* | *`None`*) – The module name (as a string) to query for.
- **`version`** (*`str`* | *`None`*) – The version (as a string) whose list of libraries you want (formatted as `X.Y`, e.g. `"2.7"` or `"3.10"`).

  If not specified, the current version of Python will be used.

Returns

A bool indicating if the given module name is part of standard libraries for the specified version of Python.

Return type

list

`stdlib_list.`**`stdlib_list`**(*version: str | None = None*) → list[str]

Given a `version`, return a `list` of names of the Python Standard Libraries for that version.

Parameters

`version` (*`str`* | *`None`*) – The version (as a string) whose list of libraries you want (formatted as `X.Y`, e.g. `"2.7"` or `"3.10"`).

If not specified, the current version of Python will be used.

Returns

A list of standard libraries from the specified version of Python

Return type
list

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX