pyversion-info

Release 1.2.2

John T. Wodder II

2024 Feb 04

CONTENTS

GitHub | PyPI | Documentation | Issues | *Changelog*

CHAPTER

ONE

API

Versions are passed to & returned from methods as strings in the form `"X"` (a major version), `"X.Y"` (a minor version), or `"X.Y.Z"` (a micro version).

All dates are returned as `datetime.date` objects.

class pyversion_info.**VersionDatabase**

New in version 1.0.0.

A database of CPython and PyPy version information. Instances are constructed from JSON objects following this JSON Schema.

classmethod fetch(*url: str = DATA_URL*, *cache_dir: str | pathlib.Path | None = CACHE_DIR*) → *pyversion_info.VersionDatabase*

Fetches the latest version information from the JSON document at `url` and returns a new *VersionDatabase* instance

Parameters

- **url** (*str*) – The URL from which to fetch the data
- **cache_dir** (*str | Path | None*) – The directory to use for caching HTTP requests. May be `None` to disable caching.

Return type
VersionDatabase

cpython: *CPythonVersionInfo*

A database of CPython version information

last_modified: *datetime*

The date & time when the database was last updated

classmethod parse_file(*filepath: str | Path*) → *VersionDatabase*

Parses a version database from a JSON file and returns a new *VersionDatabase* instance

classmethod parse_obj(*data: dict*) → *VersionDatabase*

Parses a version database from a `dict` deserialized from a JSON document and returns a new *VersionDatabase* instance

pypy: *PyPyVersionInfo*

A database of PyPy version information

class pyversion_info.**VersionInfo**

New in version 1.0.0.

A base class for storing & querying versions and their release dates

`is_released`(*version: str*) → bool

Returns whether the given version has been released yet. For a major or minor version, this is the whether the first (in version order) micro version has been released.

Parameters
version (*str*) – the version to query the release status of

Return type
bool

Raises

- *UnknownVersionError* – if there is no micro version corresponding to `version` in the database
- `ValueError` – if `version` is not a valid version string

`major_versions`() → list[str]

Returns a list in version order of all known major versions (as strings).

Changed in version 1.0.0: Now returns all known versions, released & unreleased

`micro_versions`() → list[str]

Returns a list in version order of all known micro versions

Changed in version 1.0.0: Now returns all known versions, released & unreleased

`minor_versions`() → list[str]

Returns a list in version order of all known minor versions

Changed in version 1.0.0: Now returns all known versions, released & unreleased

`release_date`(*version: str*) → date | None

Returns the release date of the given version. For a major or minor version, this is the release date of its first (in version order) micro version. The return value may be `None`, indicating that, though the version is known to the database, its release date is not; use *is_released()* to determine whether such a version has been released yet.

Changed in version 1.0.0: Unknown release dates are now always returned as `None`

Parameters
version (*str*) – the version to fetch the release date of

Return type
Optional[datetime.date]

Raises

- *UnknownVersionError* – if there is no micro version corresponding to `version` in the database
- `ValueError` – if `version` is not a valid version string

`subversions`(*version: str*) → list[str]

Returns a list in version order of all known subversions of the given version. If `version` is a major version, this is all of its minor versions. If `version` is a minor version, this is all of its micro versions.

Changed in version 1.0.0: Now returns all known subversions, released & unreleased

Parameters
version (*str*) – a major or minor version

Raises

- *UnknownVersionError* – if there is no entry for `version` in the database
- `ValueError` – if `version` is not a valid major or minor version string

class pyversion_info.**CPythonVersionInfo**

Bases: *VersionInfo*

A class for storing & querying CPython versions, their release dates, and series EOL dates

Changed in version 1.0.0: This class was previously named `PyVersionInfo`

eol_date(*version: str*) → date | None

Returns the end-of-life date of the given CPython version. The return value may be `None`, indicating that, though the version is known to the database, its EOL date is not; use *is_eol()* to determine whether such a version has reached end-of-life yet.

For a major version, this method returns the EOL date of the last subversion **if** every subversion is already end-of-life; otherwise, it returns `None`. For a micro version, this returns the EOL date of the corresponding minor version.

Changed in version 1.0.0: Unknown end-of-life dates are now always returned as `None`

Changed in version 1.1.0: Major and micro versions are now accepted

Parameters
version (*str*) – the version to fetch the end-of-life date of

Return type
Optional[datetime.date]

Raises

- *UnknownVersionError* – if there is no entry for `version` in the end-of-life table
- `ValueError` – if `version` is not a valid version string

is_eol(*series: str*) → bool

Returns whether the given version has reached end-of-life yet. For a major version, this is whether every subversion has reached end-of-life. For a micro version, this is whether the corresponding minor version has reached end-of-life.

Changed in version 1.1.0: Major and micro versions are now accepted

Parameters
series (*str*) – a Python version number

Return type
bool

Raises

- *UnknownVersionError* – if there is no entry for `version` in the end-of-life table
- `ValueError` – if `version` is not a valid version string

is_supported(*version: str*) → bool

Returns whether the given version is currently supported. For a micro version, this is whether it has been released and the corresponding minor version is not yet end-of-life. For a major or minor version, this is whether at least one subversion is supported.

Parameters
version (*str*) – the version to query the support status of

Return type
bool

Raises

UnknownVersionError – if there is no entry for `version` in the database

supported_series() → list[str]

Returns a list in version order of all CPython version series (i.e., minor versions like 3.5) that are currently supported (i.e., that have at least one release made and are not yet end-of-life)

class pyversion_info.**PyPyVersionInfo**

Bases: *VersionInfo*

New in version 1.0.0.

A class for storing & querying PyPy versions, their release dates, and their corresponding CPython versions

supported_cpython(*version: str*) → list[str]

Given a PyPy micro version, returns a list of the corresponding CPython micro versions in version order.

Raises

- *UnknownVersionError* – if there is no entry for `version` in the database
- **ValueError** – if `version` is not a valid micro version string

supported_cpython_series(*version: str*, *released: bool = False*) → list[str]

Given a PyPy version, returns a list of all CPython series supported by that version or its subversions in version order. If `released` is true, only released versions are considered.

```
>>> db.supported_cpython_series("7.3.5")
['2.7', '3.7']
>>> db.supported_cpython_series("7.3")
['2.7', '3.6', '3.7', '3.8']
>>> db.supported_cpython_series("7")
['2.7', '3.5', '3.6', '3.7', '3.8']
```

Raises

- *UnknownVersionError* – if there is no entry for `version` in the database
- **ValueError** – if `version` is not a valid version string

exception pyversion_info.**UnknownVersionError**(*version: str*)

Bases: ValueError

Subclass of ValueError raised when a *VersionInfo* instance is asked for information about a version that does not appear in its database. Operations that result in an *UnknownVersionError* may succeed later as more Python versions are announced & released.

version

The unknown version the caller asked about

pyversion_info.**DATA_URL**

The default URL from which the version database is downloaded

pyversion_info.**CACHE_DIR**

The default directory in which the downloaded version database is cached

CHAPTER

TWO

COMMAND-LINE PROGRAM

```
pyversion-info [<global-options>] <command> [<args> ...]
```

`pyversion-info` provides a command of the same name for querying information about Python versions from the command line.

Currently, the **`pyversion-info`** command has two subcommands, **`list`** and **`show`**. By default, the commands provide information about CPython versions; to get information about PyPy versions instead, pass the `--pypy` option to the subcommand.

2.1 Global Options

`-d` <database>, **`--database`** <database>

Use the given JSON file as the version information database instead of fetching data from the default URL. <database> can be either an HTTP or HTTPS URL or a path to a local file.

2.2 `pyversion-info list`

```
pyversion-info [<global-options>] list [<options>] {major|minor|micro}
```

List all major, minor, or micro Python versions, one per line.

2.2.1 Options

`-a`, **`--all`**

List all known versions of the given level

`--cpython`

Show information about CPython versions. This is the default.

`-n`, **`--not-eol`**

Only list versions that have not yet reached end-of-life (i.e., all supported versions plus all unreleased versions). This may only be used when querying information about CPython versions.

`--pypy`

Show information about PyPy versions

-r, --released
: Only list released versions. This is the default.

-s, --supported
: Only list currently-supported versions

2.3 pyversion-info show

```
pyversion-info [<global-options>] show [<options>] <version>
```

Show various information about a given Python version.

Sample outputs:

```
$ pyversion-info show 3
Version: 3
Level: major
Release-date: 2008-12-03
Is-released: yes
Is-supported: yes
EOL-Date: UNKNOWN
Is-EOL: no
Subversions: 3.0, 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9
```

```
$ pyversion-info show 3.3
Version: 3.3
Level: minor
Release-date: 2012-09-29
Is-released: yes
Is-supported: no
EOL-date: 2017-09-29
Is-EOL: yes
Subversions: 3.3.0, 3.3.1, 3.3.2, 3.3.3, 3.3.4, 3.3.5, 3.3.6, 3.3.7
```

```
$ pyversion-info show 3.9.5
Version: 3.9.5
Level: micro
Release-date: 2021-05-03
Is-released: yes
Is-supported: yes
EOL-Date: 2025-10-01
Is-EOL: no
```

```
$ pyversion-info show --pypy 7.3
Version: 7.3
Level: minor
Release-Date: 2019-12-23
Is-Released: yes
Subversions: 7.3.0, 7.3.1, 7.3.2, 7.3.3, 7.3.4, 7.3.5, 7.3.6, 7.3.7
CPython-Series: 2.7, 3.6, 3.7, 3.8
```

```
$ pyversion-info show --pypy 7.3.7
Version: 7.3.7
Level: micro
Release-Date: 2021-10-25
Is-Released: yes
CPython: 3.7.12, 3.8.12
```

2.3.1 Options

--cpython

Show information about CPython versions. This is the default.

-J, --json

Output JSON

--pypy

Show information about PyPy versions

-S, --subversions <all|not-eol|released|supported>

Which subversions to list (and, for PyPy versions, which subversions to take into account when determining supported CPython versions); the choices have the same meanings as the **list** options of the same name [default: released]

CHAPTER

THREE

CHANGELOG

3.1 v1.2.2 (2024-02-04)

- Support platformdirs v4.0
- Migrated from setuptools to hatch
- Support cachecontrol v0.14

3.2 v1.2.1 (2023-11-09)

- Support Python 3.12
- Correct the order of results returned by *PyPyVersionInfo.supported_cpython_series()*

3.3 v1.2.0 (2023-09-23)

- Support platformdirs v3.0
- Update pydantic to v2.0

3.4 v1.1.1 (2023-06-01)

- Support Python 3.11
- Support cachecontrol 0.13

3.5 v1.1.0 (2021-11-08)

- Use pydantic internally for parsing & validating version databases
- *eol_date()* and *is_eol()* now accept major and micro versions

3.6 v1.0.0 (2021-11-04)

- Support Python 3.10
- Drop support for Python 3.6
- Support for fetching information on PyPy versions has been added. With it come the following changes:
  - The schema used by the database (and thus the URL for the default database) has been modified
  - `PyVersionInfo` has been renamed to *CPythonVersionInfo*
  - A new *PyPyVersionInfo* class has been added
  - A new *VersionDatabase* class has been added, containing a *CPythonVersionInfo* instance and a *PyPyVersionInfo* instance
  - `get_pyversion_info()` is now `VersionDatabase.fetch()`
  - The command-line interface now takes a `--pypy` option for showing details about PyPy versions
- The `unreleased` argument to *major_versions()*, *minor_versions()*, *micro_versions()*, and *subversions()* has been removed; the methods now return all known versions, released & unreleased
- *release_date()* now returns `None` for any known version whose release date is unknown, whether it's been released yet or not. Use *is_released()* to determine whether such a version has been released.
- *eol_date()* now returns `None` for any known version whose EOL date is unknown, whether it's EOL yet or not. Use *is_eol()* to determine whether such a version has reached end-of-life.
- Moved documentation from README file to a Read the Docs site

3.7 v0.4.0 (2021-10-03)

- *major_versions()*, *minor_versions()*, *micro_versions()*, and *subversions()* now take optional `unreleased` arguments for including unreleased versions
- *is_supported()* now accepts major and micro versions
- *UnknownVersionError* now inherits `ValueError`
- Added a command-line interface

3.8 v0.3.0 (2021-10-01)

- Add type annotations
- Switch from appdirs to platformdirs

3.9 v0.2.0 (2020-12-13)

- Support Python 3.8 and 3.9
- Add a note to the README about the possibility of release deadlines being missed
- Drop support for Python 2.7, 3.4, and 3.5
- Properly close the `requests` session after downloading the database

3.10 v0.1.0 (2019-04-23)

Initial release

Ever needed to know what Python versions were currently supported, or how many subversions a given Python version had? Wondering how long until a given version came out or reached end-of-life? Need to know what CPython versions a given PyPy version corresponds to? The answers to these and some other questions can be found with this library.

`pyversion-info` pulls its data every run from jwodder/pyversion-info-data on GitHub. Prerelease versions are not (currently) included. I promise 24-hour turnaround times for keeping the database up-to-date until I am hit by a bus.

CHAPTER

FOUR

INSTALLATION

`pyversion-info` requires Python 3.7 or higher. Just use pip for Python 3 (You have pip, right?) to install `pyversion-info` and its dependencies:

```
python3 -m pip install pyversion-info
```

CHAPTER

FIVE

EXAMPLES

(The given outputs are current as of 2021-11-04.)

Start out by fetching the database:

```
>>> from pyversion_info import VersionDatabase
>>> vd = VersionDatabase.fetch()
```

Get a list of all currently-supported CPython series:

```
>>> vd.cpython.supported_series()
['3.6', '3.7', '3.8', '3.9', '3.10']
```

When does 3.11 come out?

```
>>> vd.cpython.release_date("3.11")
datetime.date(2022, 10, 3)
```

When does 3.6 reach end-of-life?

```
>>> vd.cpython.eol_date("3.6")
datetime.date(2021, 12, 23)
```

Just how many micro versions does 3.9 have, anyway?

```
>>> vd.cpython.subversions("3.9")
['3.9.0', '3.9.1', '3.9.2', '3.9.3', '3.9.4', '3.9.5', '3.9.6', '3.9.7', '3.9.8', '3.9.9
↪', '3.9.10', '3.9.11']
```

What major versions of PyPy are there?

```
>>> vd.pypy.major_versions()
['1', '2', '4', '5', '6', '7']
```

What CPython series do PyPy 7.3.* support?

```
>>> vd.pypy.supported_cpython_series("7.3")
['2.7', '3.6', '3.7', '3.8']
```

CHAPTER

SIX

CAVEATS

The CPython database is generally only updated when an edit is made to a release schedule PEP. Occasionally, a deadline listed in a PEP is missed, but the PEP is not updated for a couple days, and so for a brief period this library will falsely report the given version as released.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

INDEX

Symbols

C

D

E

I

L

M

P

R

S

U

V