Python Fix Imports Documentation

Release 1.0

Gaetan Semet

April 29, 2016

Contents

CHAPTER 1

Introduction

Python Fix Imports is a Python module that can automatically reorganize the `import` statements of your Python script. Please read the "Rationale" section for more information.

This module originally from a script that has been written for the Buildbot project, in order to help developers ensuring they properly organize their import statements in their Python files.

CHAPTER 2

Rationale

The beginning of each Python script is the part of the code that is likely to evolve the most over the lifetime of the file. Imports statements gets added, removed, reorganized all over the time.

Thanks to distributed versioning systems such as Git, several persons can easily work on the same time on the same file. And the management of the `import` statements is likely to cause conflict when each developer adds his modifications.

We really started having the need for an automatic reorganization script when we have set up an automatic merge of several branches alltogether. Most of the time, the conflicts were found to be on the `import` lines.

Here are the rules this `fiximports` script enforces:

2.1 Rule 1

Each import statement only imports one method, class or module.

Yes:

```
from abc import dce
from abc import fgh
```

No:

```
from abc import dce, fgh
from abc import (dce,
                 fgh)
from abc import dce, \
                fgh
```

`fiximports` automatically splits `import` statements that use a comma. `\` and parenthesis are not supported.

Bonus: let's say you want where and how an object "object_name" is imported. This rules ensures you will always find the import occurences of the following search pattern: `import object_name`. No need to do regex, only ‘‘import ‘‘ + what you are looking for.

2.2 Rule 2

Import statements are organized in blocks, separated by an empty line. Each block is alphabetically sorted.

This removes any ambiguity in the placement of an import line in a given block. When two developers on two different branches want to add the same import in the same file, the location of this line will be the same and so the merge if any will be obvious.

Yes:

```
from abc import aaaa
from abc import bbbb
from abc import cccc
```

No:

```
from abc import bbbb
from abc import aaaa
from abc import cccc
```

Sorting only occurs on a given block, if for any reason an import statement needs to be placed after another one, just add an empty line.

`fiximports` can sort all `import` statements at once (preserving the 'group' splitting).

In some project, I tend to enforce the ordering of the groups themself:

- first the standard library imports:

```
    import json
    import login
    import os
```

- Standart libraries in the form `from ... import`:

```
    from textwrap import dedent
    from twisted.internet import defer
```

- Project modules with their complete name (always uses `from __future__ import absolute_import`)

```
    from myproject.the.module.name import ClassName
    from myproject.the.other.module.name import TheOtherClassName
```

CHAPTER 3

Example

Let's look at the following code:

```
import datetime
import collections

from io import BytesIO, UnsupportedOperation
from .hooks import default_hooks
from .structures import CaseInsensitiveDict

from .auth import HTTPBasicAuth
from .cookies import cookiejar_from_dict, get_cookie_header
from .packages.urllib3.fields import RequestField
from .packages.urllib3.filepost import encode_multipart_formdata
from .packages.urllib3.util import parse_url
from .packages.urllib3.exceptions import DecodeError, ReadTimeoutError, ProtocolError, LocationParseE
from .exceptions import HTTPError, MissingSchema, InvalidURL, ChunkedEncodingError, ContentDecodingEr
from .utils import guess_filename, get_auth_from_url, requote_uri, stream_decode_response_unicode, to
from .compat import cookielib, urlunparse, urlsplit, urlencode, str, bytes, StringIO, is_py2, chardet
from .status_codes import codes
```

This automatically becomes with this plugin:

```
import collections
import datetime

from .hooks import default_hooks
from .structures import CaseInsensitiveDict
from io import BytesIO
from io import UnsupportedOperation

from .auth import HTTPBasicAuth
from .compat import StringIO
from .compat import basestring
from .compat import builtin_str
from .compat import bytes
from .compat import chardet
from .compat import cookielib
from .compat import is_py2
from .compat import json
from .compat import str
from .compat import urlencode
from .compat import urlsplit
from .compat import urlunparse
```

```
from .cookies import cookiejar_from_dict
from .cookies import get_cookie_header
from .exceptions import ChunkedEncodingError
from .exceptions import ConnectionError
from .exceptions import ContentDecodingError
from .exceptions import HTTPError
from .exceptions import InvalidURL
from .exceptions import MissingSchema
from .exceptions import StreamConsumedError
from .packages.urllib3.exceptions import DecodeError
from .packages.urllib3.exceptions import LocationParseError
from .packages.urllib3.exceptions import ProtocolError
from .packages.urllib3.exceptions import ReadTimeoutError
from .packages.urllib3.fields import RequestField
from .packages.urllib3.filepost import encode_multipart_formdata
from .packages.urllib3.util import parse_url
from .status_codes import codes
from .utils import get_auth_from_url
from .utils import guess_filename
from .utils import guess_json_utf
from .utils import iter_slices
from .utils import parse_header_links
from .utils import requote_uri
from .utils import stream_decode_response_unicode
from .utils import super_len
from .utils import to_key_val_list
from .utils import to_native_string
```

Indeed, the beginning of the file is much more verbose, but merges will be easier (since when we switched to this paradigm, we almost have not conflict on these lines).

CHAPTER 4

Usage

```
$ fiximports --help
usage: fiximports [-h] FILENAME

Fix Python Import Statements

positional arguments:
  FILENAME    Path or glob of Python files to fix

optional arguments:
  -h, --help  show this help message and exit
```

CHAPTER 5

License