Vaishnavi. Nayak
FYIT-54

- Blank lines

→ Surround top-level function and class definitions with two blank lines.
→ Method definitions inside a class are surrounded by a single bank life.
→ Extra blank lines may be used to seperate groups of related functions. Black lines may be omitted between a bunch of related one-liners.
→ Use blank lines in functions, sparingly, to indicate logical sections.
→ Python accepts the control-L from feed character as whitespace; Manly tools treat these characters as page seperators, so you may use them to seperate pages of related sections of your file. Note, some editors & web-based code viewers may not recognize control-L as a form feed & will show another glyph in its place.

- Source File Encoding

→ Code in the core Python distribution should always use UTF-8 and should not have an encoding declaration.
→ In the standard library, non UTF-8 encodings should be used only for test purposes. Use non-ASCII characters sparingly preferably only to denote places and human

Vaishnavi. Nayak
FYIT-59

names. If using non-ASCII characters as data, avoid noisy Unicode characters like zalgo and byte order marks.

→ All identifiers in the Python standard library MUST use ASCII-only identifiers, and SHOULD use English use whenever feasible.

→ Open source projects with a global audience are encouraged to adapt a similar policy.

- Imports

→ Imports should usually be on seperate lines:

```
# Correct:
import os
import sys

# Wrong:
import sys, os
```

Its okay to say this though

```
# Correct:
from subprocess import Popen, PIPE
```

→ Imports are always put at top of the file, just after any module comments and docstrings & before module globals & constants.

Vaishnavi. Nayak
FYIT-54

→ Imports should be grouped in following orde
1. Standard library imports
2. Related third party imports
3. Local application / library specific imports
You should put a blank line between each group of imports.

⇒ Absolute imports are recommended, as they are usually more readable & tend to be better behaved, if the import system is incorrectly configured.

```
import mypkg.sibling
from mypkg import sibling
from mypkg.sibling import example
```

→ However, explicit relative imports are an acceptable alternative to absolute imports, especially when dealing with complex package layouts where using absolute imports would be unnecessarily verbose.

```
from .import sibling
from sibling import example.
```

→ Standard library code should avoid complex package layoutts and always use absolute imports.

Vaishnavi. Nayak
FYIT-59

→ When importing a class from a class-containing module, it's usually okey to spell

```
from myclass import MyClass
from foo.bar.yourclass import YourClass
```

→ If this spelling causes local name clashes, then spell them explicitly

```
import myclass
import foo.bar.yourclass
```

and use "myClass" and "foo.bar.yourclass.YourClass"

→ Wildcard imports should be avoided, as they make it unclear which names are present in the namespace, confusing both readers and many automated tools. There is one defensible use case for wildcard import, which is to republish an internal interface as part of a public API.

When republishing names this way, the guidelines below regarding public and internal interfaces still apply.

Vaishnavi. Nayak
FYIT-54

- Module Level Dunder Names
Module level "dunder" such as __all__, __author__, __version__ etc should be placed after the module docstring but before any import statements except from __future__ imports. Python mandates that imports must appear in the module before any other code except docstrings:

```
""" This is the example module.
This module does stuff.
"""

from __future__ import barry_as_FLUFL
__all__ = ['a', 'b', 'c']
__version__ = '0.1'
__author__ = 'Cardinal Biggles'
import os
import sys
```

* String Quotes
In Python, single quouted strings and double-quoted strings are same. This PEP does not make a recommedation for this. Pick a rule and stick to it. When a string contains single or double quote characters, however, use the other one to avoid backslashes in the string. It improves readability.