Assignment.

Priya Ashok Nichit
F41T.140

* PEP 8 :

- Introduction:-

The document gives coding conventions for the python code comprising the standard library in main python distribution. This documend and PEP 257 was adpted from Guido's orginal python Style Guide essay, with some additions from Barry's style guide.

The style guide evolves over time as additional conventions are identified and past conventions are rendered obsolete by changes in the language itself.

Many projects have their own coding style guidelines. In the event of any conflicts, such projects spectific guides take precedence for the project.

- Code Lay-out

→ Indentation

Use 4 spaces per indentation level.

Continuation lines should align wrapped elements either vertically using python's implicit line joining inside parentheses, brackets and braces, or using a

Page No.
Date: / /

hang

PRIYA ASHOK NIKAM
FYIT. 140

hanging indent. When using a hanging indent the following should be consider there should be no argument on the first line and further indentation should be used to clearly distinguish itself as continuation lines.

The 4-space line rule is optional for continuation lines :

When the conditional part of an if statement is long enough to require that it be written across multiple line its worth noting that the combination of a two character keywork, plus a single space, plus an opening parenthesis creates a natural 4 space indent for subsequent line of the multiline conditional. This PEP takes no explicit position on how to further visually distinguish such conditional lines from the nested suite inside the if-statement. Acceptable options in this situation include, but are not limited to:

The closing brace / bracket on multiline constructs may either line up under the first non-whitespace character

of the last line of list. or it may be lined up under the first character of the line that starts multiline construct.

→ Tabs or Spaces?

Spaces are the preferred indentation method. Tabs should be used solely to remain consistent with code that is already indented with tabs. Python disallows mixing tabs and spaces for indentation.

→ Maximum Line Length:

Limit all lines to a maximum of 79 characters. For flowing long blocks of text with fewer structural restrictions, the line length should be limited to 72 characters. Limiting the required editor window width makes it possible to have several files open side by side, and works well when using code review tools that present the two versions in adjacent columns. The limits are chosen to avoide wrapping in editors with the window width set to 80, even if the tool places a marker glyph in final

column when wrapping lines. Some web based tools may not offer dynamic line wrapping at all.
Some teams strongly prefer a longer line length. And it is okey to increase the line length up to 99 characters, provided that comments and docstrings are still wrapped at 72 characters.
The python standard library is conservative and requires limiting lines to 79 characters. Python's implied line continuation inside parentheses, brakets and braces. Long lines can be broken over multiple lines by wrapping expressions in parentheses. These should be used in preference to using a backslash for line continuations.
Backslashes may still be appropriate at times. For example, long, multiple with statements could not use implicit continuation before python 3.10, so backslashes were acceptable for that cases.
Another such case is with assert statements.
Make sure to indent the continued line appropriately.

Priya Ashok Nichit
FYIT. 140

→ Should a Line Break before or After a Binary operator?

For decades the recommended style was to break after binary operators But this can hurt readability in two ways. The operators tend to get scattered across different columns on the screen, and each operator is moved away from its operand and onto the previous line. Here, the eye has to do extra work to tell which items are added and which are subtracted.
To solve this readability problem, mathematacians and their publishers follow the opposite conventions.
In python code, it is permissible to break before or after a binary operator, as long as the convention is consistent locally. For new code Knuth's style is suggested.

→ Blank lines.

Surround top-level function and class definitions with two blank lines.

Priya Ashok Mishra
F4IT, 140

Method definitions inside a class are surrounded by single blank line Extra blank lines may be used to seprate groups of related functions. Blank lines may be omitted between a bunch of related one-liners (e.g set of dummy implementations) Use blank lines in function, sparingly to indicate logical sections.

→ 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, an UTF-8 encoding should be used only for test purpose. Use non-ASCII character sparingly, preferably only to denote places and human names If using non-ASCII characters as data, avoide noisy unicode characters like zalgo and byte order marks.

Page No.
Date: / /

Priya Ashok Nichit
E41T, 140

All Indentifiers in the python standard library MUST use ASCII-only identifiers, and should use English word wherver feasible.
Open source projects with a global audience are encourage to adopt a similar policy.

→ Imports.

- Imports should usually be on seprate lines:.
  Eg: Import os
      Import sys.
- Imports are always put at the top of file, just after any module comments and docstring, and before module globals and constants.
  It should be grouped in following order:
  1. Standard libary imports.
  2. Related third party imports
  3. Local application / libary specific imports.
- Absolute import are recommended as they are usually more readable and tend to be better be haved if

Priya Ashok Nichi
FYIT, 140

if the import system is incorrectly configured. (such as when a directory inside a package ends up on sys. (path) Standard library code should avoid complex package layouts and always use absolute imports.

- When important class from a class-containing module, it's usually okay to spell this:
  If this spelling causes local name clashes then spell them explicitly:
  import myclass
  import foo.bar.yourclass
- Wild import should be avoided (from <module> import *), as they make it unclear which names are present in the namespace, confusing both readers and many automated tools.

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

9

Page No.
Date: / /

Priya Ashok Nichit
FYIT, 140

→ Module level Dunder Names.

Module level 'dunders' such as __all__, __author__, __version__, etc. should be placed after the module docstring but before any import statement except from __future__ imports.

```
""" 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-quoted strings and double-quoted strings are the same. This PEP does not make a recommendation for this. Pick a rule and stick to it. When a string contains single or double quote characters, however use the other one to avoide backslashes in the string. It improves readability

Page No.
Date: / /

Priya Ashok Nichi
FYIT, 140

For triple-quoted strings, always use double quote characters to be consiste with the docstring convention in PEP-25

* Whitespace in Expressions and Stateme

→ Pet Peeves.

Avoide extraneous whitespace in the follo

- Immediately inside parentheses, brackets or braces:

```
spam(ham[1], {eggs: 2})
```

- Between a trailing comma and following close parenthesis:

```
foo = (0,)
```

- Immediately before a comma, semicolon, or colon:

```
if x == 4: print(x, y);
x, y = y, x
```

- However, in a slice the colon acts like a binary operator, and should have equal amounts on either side. In an extended slice, both colons must have the same amount of spacing applied.

Priya Ashok Nichit
EYIT, 140

Correct:

```
ham[1:9], ham[1:9:3],
ham[:9:3], ham[1::3],
ham[1:9:]
ham[lower:upper],
ham[lower:upper:];
ham[lower::step]
ham[lower+offset :
upper+offset]
ham[:upper_fn(x):
step_fn(x)], ham[::
step_fn(x)]
ham[lower+offset:upper
+offset]
```

Wrong

```
ham[lower + offset:upper +
offset)
ham[1:9], ham[1:9],
ham[1:9:3]
ham[lower::upper]
ham[:upper]
```

- Immediately before the open parenthesis that starts the argument list of a function call:

Correct.

```
spam(1)
```

Wrong

```
spam (1)
```

Page No.
Date: / /

Priya Ashok Nikal
F41T.140

- Immediately before the open parenthesis that starts an indexing or slicing:

```
# Correct.
dct['key'] = lst[index]

#wrong
dct ['key'] = lst [index]
```

- M.
- More than one space around an assignment operator to align it with another.

```
#correct.
x = 1
y = 2
long_variables = 3

#wrong
x              = 1
yr             = 2
long_variable  = 3
```

→ Other Recommendations.

- Avoide trailing whitespace anywhere. Because it's usually invisble, it can be confusing: eg. a backslash followed by a space and a newline does not count as a line continuation marker. Some editor don't preserve it and many projects (like Cpython itself) have pre-commit hooks that reject it.

Page No.
Date: / /

Priya A. Nichit
FYIT, 140

- Always surround these binary operators with a single space on either side: assignment (=), augmented assignment (+=, -=, etc), comparisons (==, <, >, !=, <>, <=, >=, in, not in, is, is not), Booleans (and, or, not).

- If operators with different priorities are used, consider adding whitespace around the operators with the lowest priority. Use your own judgment; however, never use more than one space, & always have the same amount of whitespace on both side of binary operators.

Correct.

```
i = i + 1
submitted += 1
x = x * 2 - 1
hypot2 = x * x + y * y
c = (a + b) * (a - b)
```

Wrong:

```
i=i+1
submitted +=1
x = x * 2 - 1
hypot2 = x * x + y * y
c = (a + b) * (a - b)
```

Page No
Date : / /

Bijya Ashok Nic
FY17.140.

- Function annotations should be use the normal rules for colons and always have space around the → arrow if present.

```
# correct :
def munge(input: AnyStr):
def munge() -> PosInt:

# wrong:
def munge(input : AnyStr):
def munge() -> PosInt:...
```

- Don't use spaces around the = sign when used to indicate a keyword argument, or when used to indicate a default value for an unannotated function parameter :

```
# Correct :
def complex(real,
imag = 0.0):
    return magic(r = real,
i = imag)

# wrong.
def complex(real, image =
0.0):
    return magic(r =
real, i = imag)
```

Page No.
Date: / /

Priyo A. Nichit
FYIT. 140

- When compound statements (multiple statements on the same line) are generally discouraged

```
# correct
if foo == 'blah':
    do_blah_things()
do_one()
do_two()
do_three()
```

```
# wrong
if foo == 'blah': do_blah_thing()
do_one(); do_two(); do_three()
```

- While something its okay to put an if/for/while with a small body on the same line, never do this for multi-clause statement. Also avoid folding such long lines!

When to use Trailing Commas

Trailing commas are usually optional, except they are mandatory when making a tuple of one element. For clarity, it's recommended to surround the latter in parentheses:

```
# correct:
FILES = ('setup.cfg',)

# wrong
FILES = 'setup.cfg',
```

Priya A. Nichi
FMIT, 140

Comments.

Comments.

Comments that contradict the code are worse than no comments, Always make a priority of keeping the comments up-to-date when the code changes!
Comments should be complete sentences. The first word should be capatalized unless it's an indentifier that begins with a lower case letter.
Block comments generally consist of one or more paragraphs built out complete sentence, with each ~~statement~~ sentence ending in a period.
You should use 2 spaces after a sentence ending period in multi-sentence comments, except after the final sentence.
Ensure that your comments are clear and easily understandable to other speakers of the language you are writing in.
Python coder from non English speaking cantries: please write your comment in Engl. unless you are 120% sure that code is not read by people who don't speak your lang.

Priya A. Nichit
FYIT, 140

* Block Comments:
Block comments generally apply to some code that follows them, and are indented to the same level as that code. Each line of a block comment starts with a # and a single space. Paragraphs inside a block comment are seprated by a line containg a single #.

* Inline Comments.
Use inline comments sparingly.
An inline comment is a comment to the same line as a statement. Inline comments should be seprated by atleast two spaces from the statements. They should start with a # and a single space.
Inline comments are unnecessary and in fact distracting if they state the obvious. Don't do this:

```
x = x + 1                 # Increment x
```

But sometimes

```
x = x + 1                 # compensate for border
```

Page No.
Date: / /

Priya Ashok Nic
EyIT, 140

* Documentation Strings

Conventions for writing good documentation strings are immortalized in PEP 257.

- Write docstrings for all public modules, functions, classes, and methods. Docstrings are not necessary for non-public method, but you should have a comment that describes what the method does. The comment should appear after the def line.
- For one liner docstrings, please keep the closing """ on the same line
  """Return an ex-parrot."""

* Naming Conventions

The naming conventions of Python's library are a bit of a mess, so we'll never get this completely consistent -- nevertheless, here are the currently recommended naming standards. New modules and packages should be written to these standards, but where an existing library has a different style, internal consistency is preferred

Page No.
Date

Pruja Ashok Mohite
E111, 140

→ Overriding Principle

Names that are visible to the user as public parts of the API should follow conventions that reflect usage rather than implementation.

→ Descriptive : Naming Styles.

There are a lot of different naming styles. It helps to be able to recognize what naming style is being used, independently from what they are used for.
The following naming styles are commonly distinguished:

- b (single lowercase letter)
- B (single uppercase letter)
- lowercase
- lower_case_with_underscores.
- UPPERCASE
- UPPER_CASE_WITH_UNDERSCORES
- Capitalizeword.
- mixed case
- Capitalized_Words_With_Underscores.

Priya Ashok Nichi
EYIT, 140

→ Prescriptive: Naming Conventions.

- Names to Avoide
Never use the character 'l', 'O', 'I' as single character variable names. In some fonts these characters are indistinguishable from the numerals one and zero. When tempted to use 'l' use 'L' instead

- ASCII Compatibility.
Identifiers used in the standard library must be ASCII compatible as described in the policy section of PEP 3131

- Package and Module Names.
Modules should have short, all-lowercase names. Underscores can be used in the module names if it improved readability. Python package should also have short, all-lower-case names, although the use of underscores is discouraged.

- Class Names.
Class names should normally use the CapWords convention.
The naming convention for functions may be

~~used~~

Priya A. Nichit
F4IT, 140

The naming convention for functions maybe used instead in cases where the interface is documented and used primarily as a callable.

- Type Variable Names.
Names of type variables introduced in PEP 484 should normally use CapWords preferring short names: T, AnyStr, Num.

- Exception Names:
Because exception should be classes, the class naming convention applies here. However you should use the suffix "Error" on your exception names.

- Global Variable Names.
Modules that are designed for use via from M import * should use the __all__ mech to prevent exporting globals or use the older convention of prefixing such globals with an underscore.

- Function and Variable Names.
Function names should be lowercase, with words separated by underscores

Priya A. Nichit
FYIT, 140

as necesary to improve readability.
Varsable names follow the same convent as function names.
mixed case is allowed only in contents where that's already the prevailing sty to retain backwards compatility.

- Function and Method Arguments:
always use self for the first Argument to Instance methods. always use cls for the first argument to class methods. If a function argument's name clashes with a reserved keyword, it's generally better to append a single trailing underscor rather than use an abbreviation or spelling corruption.

- Constants.
Constants are usually defined on a module level and written in all ceptial letters with underscores separating words. Example include MAX_OVERFLOW and TOTAL.

- Public and Internal Interfaces.
Any backwards compatibility guarantee aply only to public Interfaces. Acc.

Priya A. Nichit
FYIT, 140

- Public and Internal Interfaces.
  Any backwards compatibity guarantee aply only to public interfaces. Acc, it's important that users be able to clearly distinguish between public and and internal interfaces.

- Programming Recommendations.
  - Code should be written in a way that does not disadvange other implementations of python.
  - Sequences use the fact that empty sequences are false.

```
# correct:
  if not seq:
  if seq.

# wrong.
  if len(seq):
  if not len(seq):
```

- Don't compare boolean values to True or False using ==

```
# Correct
  if greeting:

# wrong
  if greeting == True
```

Briya A. Nichol
FYIT, 140

* Variable Annotation.

- Annotation for module level varibles, clays, and instance varibles. & local varibles should have a sigle space after colon.
- There should be no space before the colon.

* References:

- Barry's GNU Mailname style guide
- Donald Knuth's The TeXBook, pages 195 and 196
- Typeshed repo