JYOTI SINGH PRACTICAL7
FYCS_54

Practical 07 Implementing coding practices in Python using PEP8.

01.What is Pep ?

The PEP is an abbreviation form of Python Enterprise Proposal. Writing code with proper logic is a key factor of programming, but many other important factors can affect the code's quality. The developer's coding style makes the code much more reliable, and every developer should keep in mind that Python strictly follows the order and format of the string.

Adapting a nice coding style makes the code more readable. The code becomes easy for the end-user.

PEP 8 is a document that provides various guidelines to write the readable in Python. PEP 8 describes how the developer can write beautiful code. It was officially written in 2001 by Guido van Rossum, Barry Warsaw, and Nick Coghlan. The main aim of PEP is to enhance the readability and consistency of code.

02.Naming Convention :-

When we write the code, we need to assign name to many things such as variables, functions, classes, packages, and a lot more things. Selecting a proper name will save time and energy. When we look back to the file after sometime, we can easily recall what a certain variable, function, or class represents. Developers should avoid choosing inappropriate names.

The naming convention in Python is slightly messy, but there are certain conventions that we can follow easily. Let's see the following naming convention.

Single lowercase letter

1. a = 10

Single upper case letter

1. A = 10

Lowercase

1. var = 10

Lower_case_with_underscores

1. number_of_apple = 5

UPPERCASE

1. VAR = 6

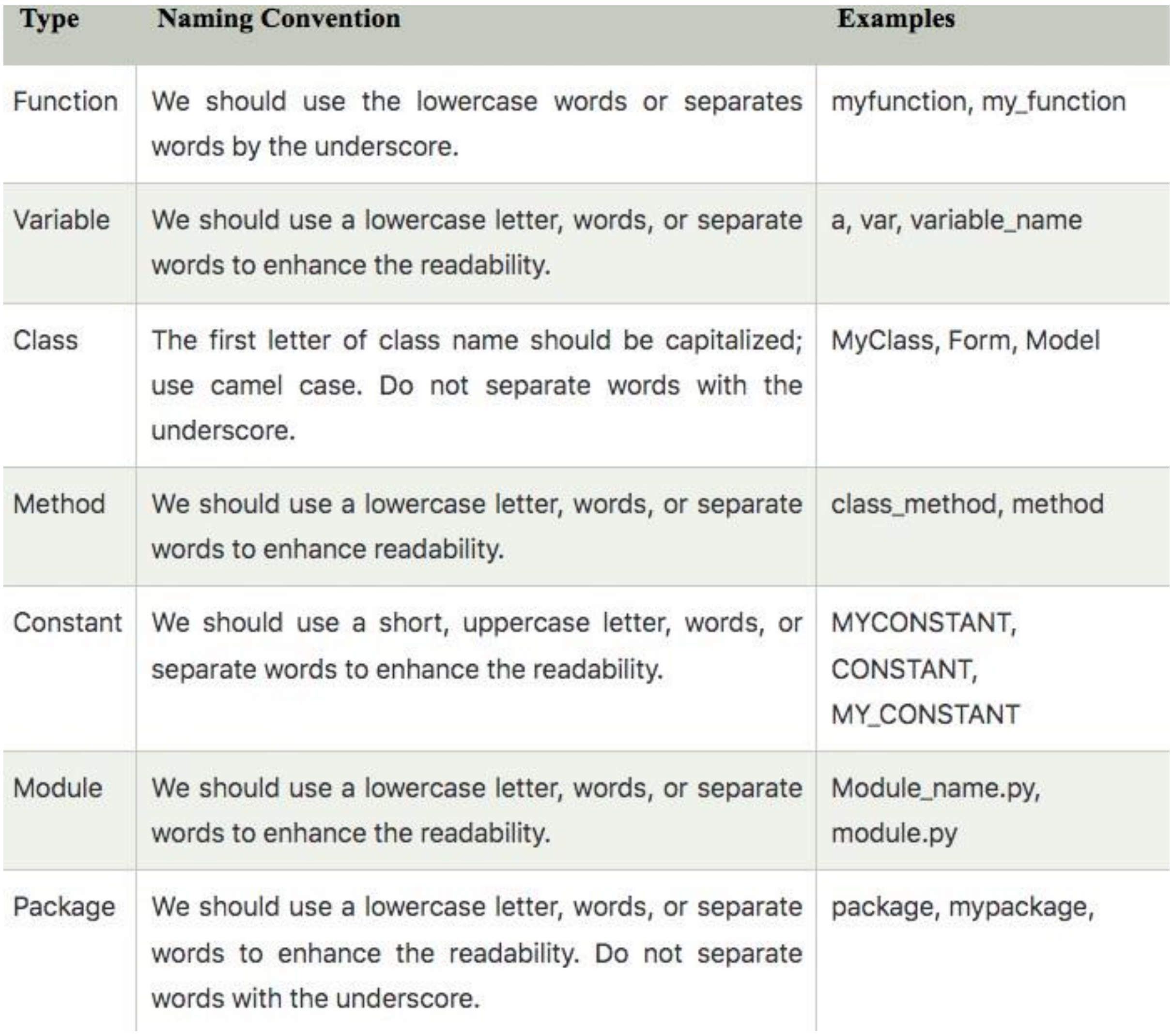
UPPER_CASE_WITH_UNDERSCORES

1. NUM_OF_CARS =20

CapitalizedWords (or CamelCase)

1. Number_Of_Books = 100

03. Name Style:-



Type	Naming Convention	Examples
Function	We should use the lowercase words or separates words by the underscore.	myfunction, my_function
Variable	We should use a lowercase letter, words, or separate words to enhance the readability.	a, var, variable_name
Class	The first letter of class name should be capitalized; use camel case. Do not separate words with the underscore.	MyClass, Form, Model
Method	We should use a lowercase letter, words, or separate words to enhance readability.	class_method, method
Constant	We should use a short, uppercase letter, words, or separate words to enhance the readability.	MYCONSTANT, CONSTANT, MY_CONSTANT
Module	We should use a lowercase letter, words, or separate words to enhance the readability.	Module_name.py, module.py
Package	We should use a lowercase letter, words, or separate words to enhance the readability. Do not separate words with the underscore.	package, mypackage,

04. Code Layout:-

Indentation:

Unlike other programming languages, the indentation is used to define the code block in Python. The indentations are the important part of the Python programming language and it determines the level of lines of code. Generally, we use the 4 space for indentation. Let's understand the following example. x = 5 **if** x == 5:

```
  print('x is larger than 5')
```

In the above example, the indented print statement will get executed if the condition of if statement is true. This indentation defines the code block and tells us what statements execute when a function is called or condition trigger.

Indentation following Line break

It is essential to use indentation when using line continuations to keep the line to fewer than 79 characters. It provides the flexibility to determining between two lines of code and a single line of code that extends two lines.

For eg.

```
# first line doesn't has any argument

# We add 4 spaces from the second line to discriminate arguments from the

rest.  def  function_name(  argument_one,  argument_two,  argument_three,

argument_four):

  print(argument_two)

# 4 space indentation to add a level.

foo                              =

  long_function_name(

  var_one,          var_two,

  var_three, var_four)
```

Use docstring

Python provides two types of document strings or docstring-

- Single line
- Multiple line

We use triple quotes to define a single line or multiline quotes. Basically, these are used to describe the function or particular program.

For eg.

```
def add(a, b):
  """This is simple add method"""
  """This is a simple add
  program to add the two
  numbers. """
```

The line break before or after a Binary operation is a traditional approach. But it affects the readability extensively because the operators are scattered across the different screens, and each operator is kept away from its operand and onto the previous line.

For eg.

```
# easy to match operators with operands
 Total_marks = (English_marks
       + math_marks
       + (science_marks - biology_marks)
       + physics_marks
```

Importing Module

We should import modules in separate lines as follows:

```
import    pygame
import  os  import
sys
```

OR

from subprocess **import** Popen, PIPE

The import statement should be written at the top of the file or just after any module comment. Absolute imports are the recommended because they are more readable and tend to be better behaved. **import** mypkg.sibling from mypkg **import** sibling from mypkg.sibling **import** example

However, we can use the explicit relative imports instead of absolutes import, especially dealing with complex packages.

Blank Lines

Blank lines can improve the readability of Python code.

Top-level function and classes with two lines- Put the extra vertical space around them so that it can be understandable. **class** FirstClass: pass

```
class  SecondClass:
    pass


def main_function():


    return None
```

2. Single blank line inside classes- The functions that we define in the class is related to one another.

```
class FirstClass:
    def method_one(self):
        return
        None


    def second_two(self):
        return
        None
```

3. Use Blank lines inside the function- While writing a complicated functions which includesseveral steps before the return statement hence blank lines between each step is added to make it readable.

```
def cal_variance(n_list):
    list_sum =
    0 for n in
    n_list:
      list_sum = list_sum + n
    mean = list_sum /
    len(n_list)

    square_sum = 0
    for n in n_list:
      square_sum = square_sum + n**2
    mean_squares = square_sum / len(n_list)

    return mean_squares - mean**2
```

Put the closing braces

PEP8 allows us to use closing braces in implies line continuation.

- Line up the closing brace with the first non-whitespace.

```
list_numbers = [
5, 4, 1,
4, 6, 3,
7, 8, 9
 ]
```

Line up closing braces with the first character of line.

```
list_numbers = [
  5, 4, 1,
  4, 6, 3,
        7, 8, 9
    ]
```

Both methods are suitable to use.

Comments

Comments are the integral part of any programming language. While writing a comment there are certain points to keep in mind :-

- Start with the capital latter, and write complete sentence.
- Update the comment in case of a change in code.
- Limit the line length of comments and docstrings to 72 characters.

Block Comment

They are useful while we perform a single action such as iterating a loop. There are certain rules to write a block comment:-

- Indent block comment should be at the same level.
- Start each line with the # followed by a single space.
- Separate line using the single #.

For eg.

```
for i in range(0, 5):
    # Loop will iterate over i five times and print out the value of i
    # new line character
    print(i, '\n')
```

Inline Comments

Inline comments are used to explain the single statement in a code. Certain points to remember:-

- Start comments with the # and single space.
- Use inline comments carefully
- We should separate the inline comments on the same line as the statement they refer.

For eg.

a = 10 # The a is a variable that holds integer value.

Avoid Unnecessary Whitespaces

Use of unnecessary whitespace can make the code much harder to understand therefore we should avoid adding whitespaces. This is known as trailing of whitespaces.

05. Programming Recommendation

- Avoid comparing Boolean values using the equivalence operator.
- Empty sequences are false in if statements.
- Do not use 'not' in the if statement. For eg.

```
#

Recommended

if x is not None:

return 'x

exists!
```