IT TOOLS

Tanvi
FYIT:112

Pep in python

Q.1] 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 quality. The developers coding style make the code much reliable, and every developer should keep in mind that python strictly is follows the way of order and format of the string. Adaptive a nice coding style makes the code more readable. The code becomes the easy for 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 in officially written in 2001 by guide can Rossum. Barry warsaw. and Nick Coghlan. The main aim of PEP is to

enhance the readability and in this constisency of code.

Q.2] What PEP 8 is important? why?
→ PEP 8 enhances the readability of the python code, but why is this in readability so important? Creator of python, Guide van Rossum said, "Code is much more often than it is in written"! The code can be written in a few minutes, a few hours. or a whole day but once we have written the code, we will never rewritten it again. But sometime, we need to retred the code again and again.

At this point, we must have an idea of why we wrote the particular line in the code. The code should reflect the meaning of each other. and in line. That's why readability is so much important.

Tanvi
FYIT:112
③

Naming Convention:
When we write the code, we need to assign name to many things such as variables, functions, classes, and packages, alot more things. Selecting a peroper name will save time and energy. When we look back of the file after sometime, we can easily recall what a certain variable, functions or class represent developers should avoid choosing inappointment names. The naming convention in python is slighty messy but there are certain. convention that we can follow easily.

Examples: Single lowercase letter.
a = 10

Single uppercase letter
A = 10

lowercase
var = 10

lowercase_width_underscores
number_of_apple = 5

UPPERCASE
VAR = 6

UPPER_CASE_WITH_UNDERSCORES
NUM_OF_CARS = 20

Capitalized Words (or Camelcase)
Number of Book = 100

* Name style:

Types	Naming convenvetion	Examples.
Function	We should use the lowercase words or separates words by the underscore	my function my_function
variable	We should use the lowercase letters, words, or separate in words to enhance the readability	a, var, variable_name

method	We should use a lowercase letter words or seperate words to enhance readability	class-method
Constant	We should use a short, uppercase letters, words or seperate the words to enhance the read-ability.	MY CONS-CONSTANT
Module	We should use a short lowercase letters, words or seperate the words to enhance the read-ability.	Module name.py, py.
Package	We should use a lowercase letter, words, or seperate words to enhance the readability. Do not seperate words with the underscore.	package, my pack.

* Code layout :
The code layout defines how much the code is readable, In this section, we will learn, how to use whitespace to improve code readability.

* Indentation :
Unlike other programming language, the indentation is used to define the code block in Python. The indentations are the important part of python to the programming language and it determines the level of lines of code. So generally we use the 4 space for indentation.

example : x = 5
if x = =5 :
print ("x is larger than 5").

In 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 to execute when a function is called or condition trigger.

- Tabs VS space.
  We can also use the tabs to provide the consecutive space to indicate the indentation but whitespaces are the most "preferable". Python 2 allows the mixing of tabs and space but we will get an error in python 3.

* Indentation following line Break:
It is essential to use indential when this is using continuations to keep the lines to fever than 19. characters. It provides the festixibility to the determining between two lines of code and a single line of code that extends two lines.

example : 1] # Correct way:
# Aligned with opening delimiter.
obj = func-name (argument_one, argue-
two, argument_three - argue-fr).

2] # First line doesn't has any argument.
We add 4 spaces from the second link
to discriminate arguements from the rest,
def function_name (
arguement-one, arguement-two,
arguement_three, arguement-four):
print (arguement-two).

# 4 space indentation to add a level.
foo = long-function-name (
var-one, var-two,
var-three, var-four).

* Use docstring :
Python describes the two types of document strings or docstring - single line and multiple line. We use the triple quotes to define a single line or multiline quotes. So, these are used to describe the function or particular program.

Examples: def (a, b):
"" This is Simple Method ""
""" This is
a
simple add program to add
the two number """

* Should a line Break before or after a Binary operator?
→ This line break or after a binary operation is a traditional approach. But is affects the readability extensively because the operator are scattered

accross the different screens and each operator is kept away from its operand and onto the previous line.

Example : 1] # Wrong.
Operator lit far away from their operands marks = (english_marks + math_marks + science_marks - biology_marks + physics_marks.

example : 2] # Correct :
# easy to match operators with operands.
total_marks = (english_marks - marks + math_marks + science_marks - biology_marks + physics_marks).

Python allows us to break line before or after a binary operator, as long as the convention is consistent locally.

Tanvi
FYIT:112
(11)

* Importing module

- We should import the modules in the separate line as follows:

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

- Wrong

```
import sys, os....
```

- We can also use the following approach

```
from subpoints import Popen, PIPE :
```

- The import statement should be written at the top of the File or just after any module comments. Absolute import are the recommended because they

are more readable and tend to be letter behaved.

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

However, we can use the explicit the relative imports instead of absolute imports, especially dealing with the complex package.

* Blank lines:

Blank lines can be improved the readability of python code. If many lines of code bunched together the code will become harder to read. We can remove this by using the many blank virtual lines and the reader might need to scroll more than necessary. Follow the below instruction to add vertical whitespace.

- 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.
```

- Second blank line inside classes - The function that we define in the class is related to one another.

example :
```
class Firstclass:
    def method-one (self)
        return None.
```

```
def second - two (self):
    return None.
```

- Use blank line inside the function:
Sometimes, we need to write a thare complicated function has consits of the several steps before the return to the statement. So we can add the blank line between each step.

```
example: def cal_variance (n_list)
    list - sum = 0
    for n in n_list:
        list - sum = list - sum t n.
    mean = list = sum / in (n-list)

square_sum = 0
    for n in n-list:
        square_sum = square - sum t n**2.
mean - square = square_sum / len (n-list)
return mean_square_mean **2
```

- The above way can remove the whitespace to improve the readalibility the code.

* Put the closing Braces:
We can break lines inside those parentnesses, brackets using the line continuations. PEP 8 allows us to use closing braces in implies line to the continuations.

- line up the closing brace with the first non_whitespace.

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

- line up the closing braces with the first characters of lines.

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

Both methods are suitable to use, but consistency is key, so choose

any one and continue with it.

* Comments.
Comments are the integral part of the any programming language. Those are the best way to explain the code. When we documented our code with the proper comments anymore can able to understand the code. But we should remember the following points:

- start with the capital letter, 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:

→ Block comment are the good choice for the small section of code. Such comments are useful when we write several line codes to perform a single action such asiterating a loop. They helps in to understand the purpose of the codes.

PEP 8 provides the following rules to write comment block.

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

example : for i in range (0,5):
loop will iterate over i five times.
new line character.
print (i, '\n').

We can use more than paragraph for the technical code.

* Inline comments.
Inline comments are used to explain the single statement in a piece of code. we can quickly get the idea of why we wrote that particular line of code.
PEP 8 specifies the following rules for the inline comments.

- Start comments with the # and single space.
- Use inline comments carefully.

- We should separate the inline comment on the same line as the statement they refer.

Example: a = 10 # There is a variable that holds integer value.

Sometimes, we can use the naming convention to replace the inline the comments.

Example: x = 'Prefer Decosta'
This is a student name.

we can say or use the following naming convention.

Example: student_name:
'Peter Shah'

Inline comments are essential but block comments make the code more readable.

* Avoid Unnecary Adding whitespace:
→ In some cases, use of whitespace can make the code much harder to read. Too much whitespaces can make code overly sparse and too difficult to understand. We should avoid adding whitespaces at the end of a line. this is known as training whitespaces.

example :1] # Recommended.
list 1 = [1, 2, 3)

Not Recommended.
list 1 = [1, 2, 3].

example 2] : x = 5
y = 6

```
# Recommended.
print (x, y).

# Not recommended.
print (x,y)
```

* Programming Recommendation :

→ As we know that, there are the several method to perform similar tasks in Python. In this section, we will see some of the suggestions of PEP 8 to improve the consistency.

Avoid comparing Boolean values for using the equivalence operator:

Example 1]
```
# Not Recommended.
bool_value = 10 > 5.
if bool_value == True :
return '10 is bigger than 5'.
```

We shouldn't use the equilance of operator == to compare the Boolean value It can only take the True or False.

example 2] # Recommended.
```
if my_bool:
    return '10 is bigger than 5'.
```

* Empty sequences are false in if those statement.
If we want to check whether a given list is empty, we might need to check the length of list, so we need to avoid.

Example: # Not recommended.
```
list 1 = []
if not len (list 1):
    print ('list is empty!")
```

However, if there is any empty list, set or tuple.

example : Recommended

```
list 1 = []
if not list:
    print ('list is empty!')
```

The second method is more appropriate thats why PEP 8 is encourages of it to us:

* Don't use not is in if statement.

→ There are two options to check whether a variable has a defined value. the first option is with x is not None, as in the example.

Example : # Recommended.
if x is not none:
return 'x exists!'

A second portion is to evaluate x is None and if statement based on not the outcome.

Example : # Not Reomended:
if x is not None:
return 'x exists!'.

Conclusion : We have discussed the PEP 8 guidelines to make the code to remove ambiguity and enhance readability. These guidelinesss to improve the code, especially when sharing the code with potential employees or collaborators.