PRACTICAL-NO:7

**AIM**: Implementing coding practices in Python using PEP8.

**EXPLANATION**:

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 reliable, and every developer should keep in mind that Python strictly follows the way of order and format of the string.Adaptive a nice coding style makes the code more readable. The code becomes 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 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.

Why PEP 8 is Important?

PEP 8 enhances the readability of the Python code, but why is readability so important? Let's understand this concept.

Creator of Python, Guido van Rossum said, **"Code is much more often than it is 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 rewrite it again. But sometimes, we need to read 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 line. That's why readability is so much important.We will describe few important guidelines for writing effective code that can be read by others as well.

**Example -**

Single lowercase letter

a = 10

Single upper case letter

A = 10

Lowercase

var = 10

Lower\_case\_with\_underscores

number\_of\_apple = 5

UPPERCASE

VAR = 6

UPPER\_CASE\_WITH\_UNDERSCORES

NUM\_OF\_CARS =20

CapitalizedWords (or CamelCase)

NumberOfBooks = 100

* **Name Style**

Below is the table that specifies some of the common naming styles in Python. Consider the following table.

|  |  |  |
| --- | --- | --- |
| **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, |

* **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 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.

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.

* Tabs vs. Space

We can also use the tabs to provide the consecutive spaces to indicate the indentation, but whitespaces are the most preferable. Python 2 allows the mixing of tabs and spaces but we will get an error in Python 3.

ndentation 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. Let's understand the following example.

Example –

# Correct Way:

# Aligned with opening delimiter.

obj = func\_name(argument\_one, argument\_two,

argument\_three, argument\_four

We can use the following structure.

# 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 the two types of document strings or docstring - single line and multiple lines. We use the triple quotes to define a single line or multiline quotes. Basically, these are used to describe the function or particular program. Let's understand the following example.

def add(a, b):

"""This is simple add method"""

"""This is a

simple add program to add

the two numbers. """

* **Should a Line Break Before or After a Binary Operator?**

The lines 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. Let's understand the following example.

# easy to match operators with operands

Total\_marks = (English\_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. Put the Closing Braces

We can break lines inside parentheses, brackets using the Line continuations. PEP 8 allows us to use closing braces in implies line continuations. Let's understand the following example.

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

list\_numbers = [

5, 4, 1,

4, 6, 3,

7, 8, 9

]

Line up the closing braces with the first character of line.

list\_numbers = [

5, 4, 1,

4, 6, 3,

7, 8, 9

]

* **Block Comment**

Block comments 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 as iterating a loop. They help us to understand the purpose of the code.

* PEP 8 provides the following rules to write comment block.

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

Let's see the following code.

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')

* **Conclusion:**

We have discussed the PEP 8 guidelines to make the code remove ambiguity and enhance readability. These guidelines improve the code, especially when sharing the code with potential employees or collaborators. We have discussed what PEP is and why it uses, how to write code that is PEP 8 compliant. Moreover, we have a brief introduction to the naming conventions. If you want more information regarding the PEP 8, you can read the full documentation or visit PEP8.org