**Python Language Coding Standards**

***Last update: 4 October 2024***

**Coding standards**, also called coding guidelines or style guides, are a set of rules developers follow when writing code. These rules help keep the code consistent and make it easier for developers to work together. Every development team should clearly define its **coding standards** based on the team's needs, the project, and the programming languages they use.

When **coding standards** are well-defined and followed, even new developers can easily understand the code. Ideally, the code should appear as if it was written, fixed, and maintained by one person.

**Comment Rules**

1. Single line comments should be written using #. Multiple lines comments should be written with “”” “””.

2. Every source file must include a comment at the beginning of the file that briefly describes the purpose of the source file.

For example:

“””

Demonstration program for Software Engineering

Program for personal financial tracking

Which contain many component.

“””

3. Every function, except the main function, must proceed with a comment that at least explains the purpose of the function. It should be a block comment on top of the function itself. **If** necessary (If the code is complicated), inline comment should be written at least two tab from the code (Otherwise it might be distracted) .

For example:

#Function printing name of people in transaction

#Define all name as x and print them

def my\_Transac(Name):

for x in Name:

print(x)

**Naming Rules**

1. For variable names that consist of multiple words, you may only use snake cases consistently. **Remember**, the name will only consist of lower case letters.

For example:

sum\_of\_transac = 0

Total\_transac = 0

2. Names for variables (as well as functions and other identifiers) should be meaningful, in order to help the reader understand the purpose of the variable.

For example:

my\_transaction = 42

def calculate\_total():

3. Avoid using single letters for variable names, except for loop variables or other temporary variables, or if you’re implementing a formula where the equation variables are single letters. In the same time also avoid using too long variable names which might be difficult to understand. Simply just name a variable as easy to understand as you can.

For example:

initial\_balance = 100

deposit\_amount = 200

final\_balance = initial\_balance + deposit\_amount

NOT like this

a = 100

b = 200

totalAmountOfMoneyInBankAccountAtEndOfMonth = a + b

4. Single-letter variables like i for loop counters or mass and acceleration for formulas are acceptable because they are temporary or represent well-known scientific equations.

For example:

for i in range(5):

print(f"Iteration {i}")

5. Defined constants must be all upper case. Otherwise, it might cause some confusion with some other variable.

**Code Format Rules**

1. All code must use consistent indentation to show the logical structure. Controlled statements must be indented one level from the statement that controls them. Some components will have some effect on making the code easier or harder to read etc. brackets or tab.

For example:

found = -1

i = 0

while found == -1 and i < student\_count:

if student\_name[i] == target\_name:

found = i

else:

i += 1

2. Long code lines are hard to read and understand. If a line of code is longer than 60 characters, break it into different lines.

3. Long functions are hard to read and understand. If a function is longer than about 60 lines of code, extract parts of the code into sub-functions.

4. Use parentheses to clarify the meaning of your expressions.

For example:

if (age >= 65) or (memberNumber > 0):

result = value1 + (value2 / count) - (value3 % 7)

5. A function should ideally do one thing and do it well. Large functions should be split into smaller, more manageable pieces.

6. Aim for functions/classes with a single responsibility to enhance maintainability and readability.

For example:

def process\_data(data):

def send\_email(data):

And NOT

def process\_data\_and\_send\_email(data):

process\_data(data)

send\_email(data)