PRACTICAL-07: Implementing coding practices in Python using PEP8

PEP 8 exists to improve the readability of Python code.

1) Naming Conventions:

When you write Python code, you have to name a lot of things: variables, functions, classes, packages, and so on. Choosing sensible names will save you time and energy later. You'll be able to figure out, from the name, what a certain variable, function, or class represents. You'll also avoid using inappropriate names that might result in errors that are difficult to debug.

PEP8.py

```
O = 2  # This may look like you're trying to reassign 2 to zero
```

2) How to Choose Names:

When naming variables, you may be tempted choose simple, single-letter lowercase names, like x. But, unless you're using x as the argument of a mathematical function, it's not clear what x represents.

When naming variables, you may be tempted to choose simple, single-letter lowercase names, like x. But, unless you're using x as the argument of a mathematical function, it's not clear what x represents. Imagine you are storing a person's name as a string, and you want to use string slicing to format their name differently. You could end up with something like this:

```

x= 'Aman Upadhyay'
y, z = x.split()
print(z, y, sep=',')
'Aman, Upadhyay'
```

The following example is much clearer. If you come back to this code a couple of days after writing it, you'll still be able to read and understand the purpose of this function:

```python

name = 'Aman Upadhyay'
first_name, last_name = name.split()
print(last_name, first_name, sep=', ')
'Aman, Upadhyay'
```

3) Code Layout:

PEP 8 guidelines suggest that each line of code (as well as comment lines) should be 79 characters wide or less. This is a common standard that is also used in other languages including R.

PEP8.py

```
#CORRECT
# Perform some math
a = 1+2
b = 3+4
c = a+b

# Read in and Plot some
preceip_timeseries = pd.readcsv("precip-2019.csv")
preceip_timeseries.plot()
```

PEP8.py

```
#WRONG
a=1+2
b=3+4
c=a+b
date=pd.readcsv("precip=2019csv")
date.plot()
```

4) Whitespace in Expressions and Statements:

Adding space when there is more than one operator in a statement.

Surround the following binary operators with a single space on either side:

●Assignment operators (=, +=, -=, and so forth)

●Comparisons (==, !=, >, <. >=, <=) and (is, is not, in, not in)

●Booleans (and, not, or)

PEP8.py

```
# Recommanded
y = x**2 + 5
z = (x+y) * (x-y)
```

PEP8.py

```
# Not Recommanded
y = x ** 2 + 5
z = (x + y) * (x - y)
```

5) Comments:

Comments are lines that exist in computer programs that are ignored by compilers and interpreters.

Comment begins with a hash mark (#)

Generally, comment looks like this:

this a comment.

Because comment does not execute, when you will run program you will not see any indication of the comment there.

● Inline Comments: Inline comment should be separated by at least two spaces from the comment. They should start with a # and a single space. Inline comments are unnecessary and in fact distracting if they state the obvious.

-Anti pattern

PEP8.py

```
def print_name(self):
    print(self.name) # comment is correct nowneeds a space
```

-Best practice

PEP8.py

```
def print_name(self):
    print(self.name) # comment is correct now
```