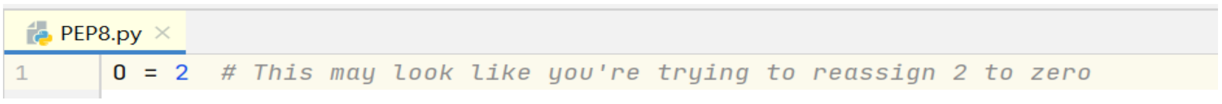
PRACTICAL-07: Implementing coding practices in Python using PEP8

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.

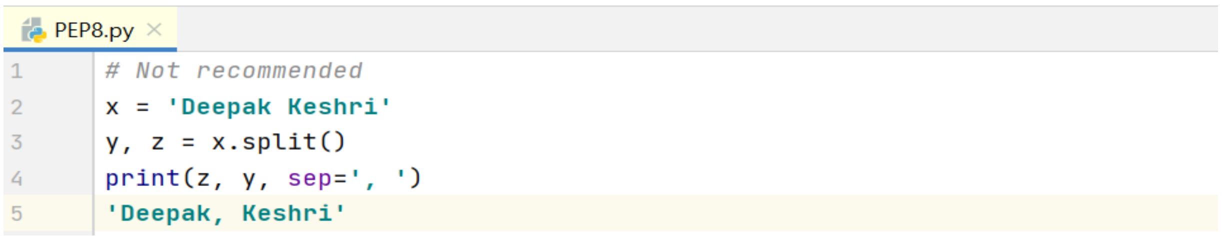


```python
O = 2   # This may look like you're trying to reassign 2 to zero
```

2) How to Choose Names:

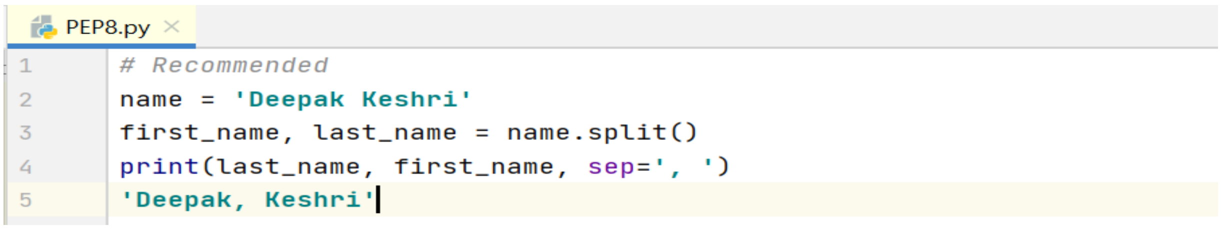
When naming variables, you may be tempted choose simple, single-letter lowercase names, like x. But, unless you're using x as the argument of a mathematical function, it's not clear what x represents.

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:



```python
# Not recommended
x = 'Deepak Keshri'
y, z = x.split()
print(z, y, sep=', ')
'Deepak, Keshri'
```

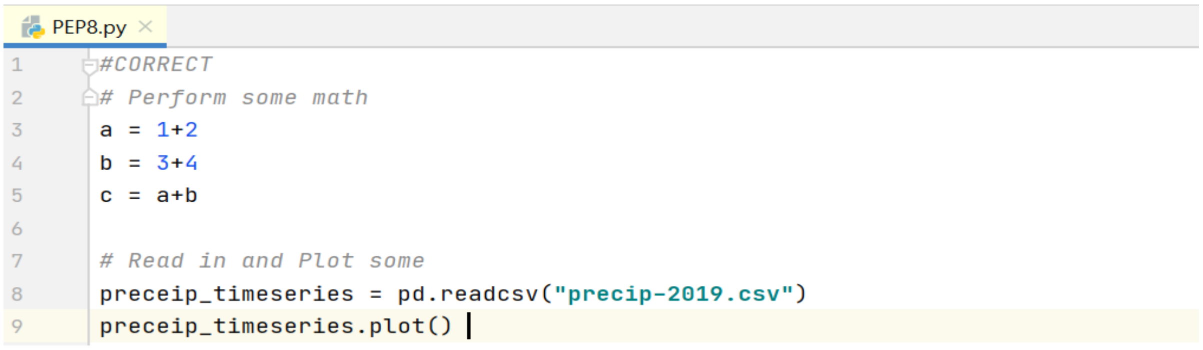
The following example is much clearer. If you come back to this code a couple of days after writing it, you'll still be able to read and understand the purpose of this function:



```python
# Recommended
name = 'Deepak Keshri'
first_name, last_name = name.split()
print(last_name, first_name, sep=', ')
'Deepak, Keshri'
```

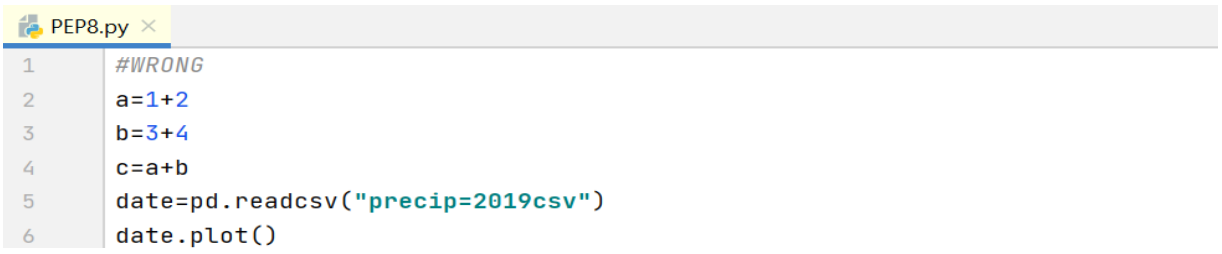
3) Code Layout:

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.



```
#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()
```

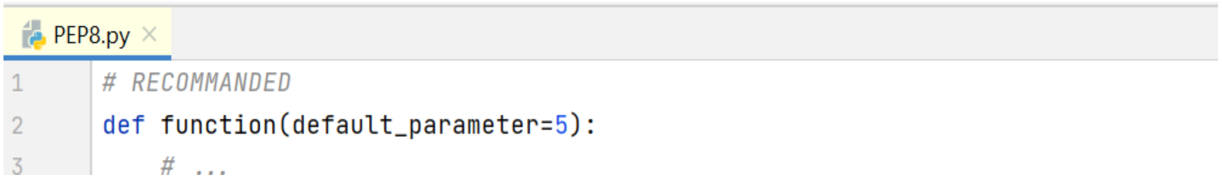


```
#WRONG
a=1+2
b=3+4
c=a+b
date=pd.readcsv("precip=2019csv")
date.plot()
```

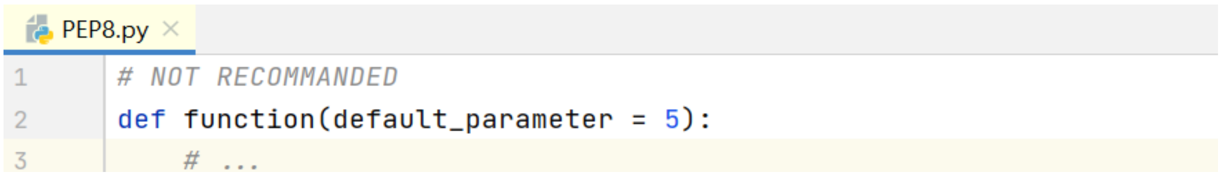
4) Whitespace in Expressions and Statements:

a) Whitespace Around Binary Operators

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)
note: When = is used to assign a default value to a function argument, do not surround it with spaces.



```
# RECOMMANDED
def function(default_parameter=5):
    # ...
```



```
# NOT RECOMMANDED
def function(default_parameter = 5):
    # ...
```

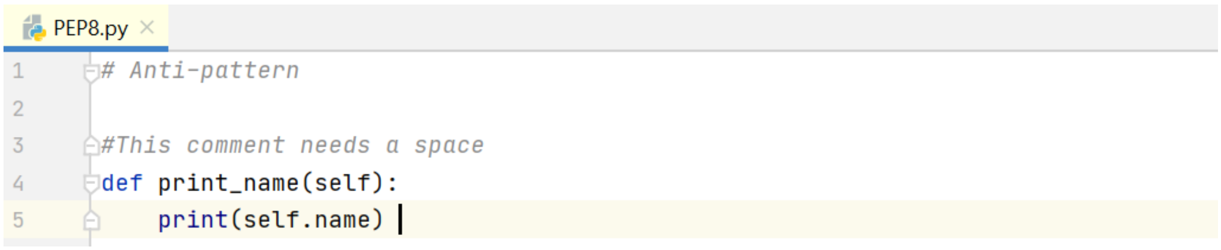
5) Comments:

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

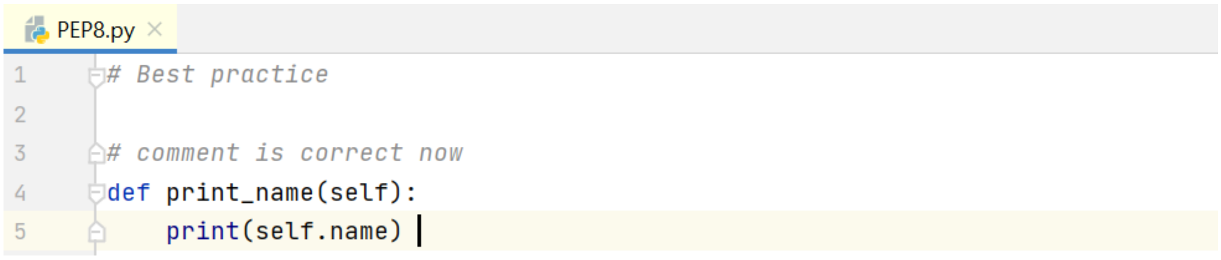
Because comment does not execute, when you will run program you will not see any indication of the comment there.

●Block Comments
Each line of block comments starts with a # and a single space.
Paragraphs inside a block comment are separated by a line containing a single #.



```python
# Anti-pattern

#This comment needs a space
def print_name(self):
    print(self.name)
```



```python
# Best practice

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