PRACTICAL 7

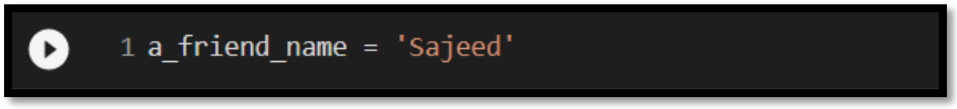
Implementing Coding Practices in Python Using PEP8

What is PEP 8?

Pep 8 is a document that provides guidelines and best practices on how to write Python code. It was written in 2001 by Guido van Rossum, Barry Warsaw and Nick Coghlan. The primary focus of PEP 8 is to improve the readability and consistency of Python code. PEP stands for Python Enchantment Proposal and there are several of them. A PEP is a document that describes new features proposed for python and document aspects of Python, like design and style for the community.

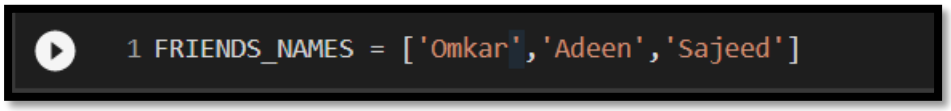
Code 1: regular_variables

- Variable names should be lowercase, where necessary separating words by underscores



Code 2: CONSTANTS

- In python, all variables can be modified
- Therefore, real constants don’t exist
- But to indicate that a variable should be treated as if it were a constant, names should be uppercase, where necessary separating words by underscores



Code 3: function_names()

- Names of functions and class methods should be lowercase, where necessary separating words by underscores

```
import random

def random_friend_name():
  return random.choice(FRIENDS_NAMES)

print(random_friend_name())
```

```
Omkar
```

Code 4: ClassNames

- Class names should capitalize the first letter of each word

```
class FriendName:

  def __init__(self, name):
    self.name = name

  def __str__(self):
    return self.name

adeen = FriendName(name='Adeen')
print(adeen)
```

```
Adeen
```

Code 5: FactoryFunctionNames()

- Factory functions return objects
- Therefore, to users of your code, factory functions act like class definitions
- To reflect this, factory-function names should also capitalize the first letter of each word

```
def Omkar():
  return FriendName(name='Omkar')

omkar = Omkar()
print(omkar)
```

```
Omkar
```

Code 6: _non_public_properties

- In Python, properties can be accessed from anywhere
- Therefore, private class properties don’t exist
- But to indicate that a property should be treated as if it were private, names should be prefixed with an underscore

```
class Sajeed(FriendName):

  def __init__(self):
    FriendName.__init__(self, name='Sajeed')
    self._age = '18'

  @property
  def age(self):

    return self._age

sajeed = Sajeed()
print(sajeed.age)
```

```
18
```

Code 7: conflicting_names_

- If a name is already taken, suffix an underscore