Training Report Day-13

20 June 2024

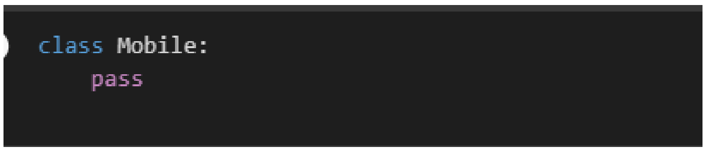
OOP Basics in Python:

Object oriented programming approach allows us to club together the data and behaviour so that it becomes easier to code real world problems.

Classes and Objects:

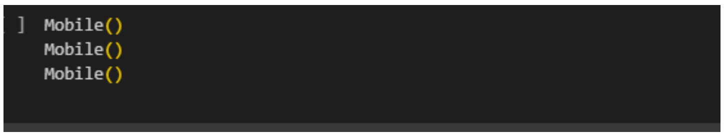
- Objects are real world entities. Anything you can describe in this world is an object.
- Classes on the other hand are not real. They are just a concept.

Classes are defined by using ‘class’ keyword.



```
class Mobile:
    pass
```

To create an object, we need a class. The syntax for creating an object is "classname()", where classname is the name of the class.

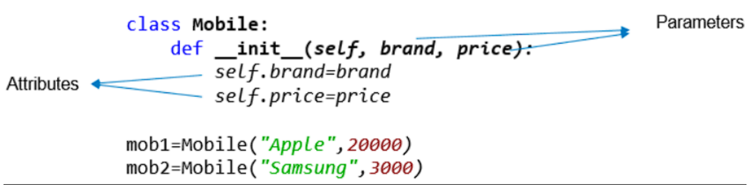


```
Mobile()
Mobile()
Mobile()
```

Attributes of an Object:

Attributes are declared by using dot(.) operator with object.

Example:



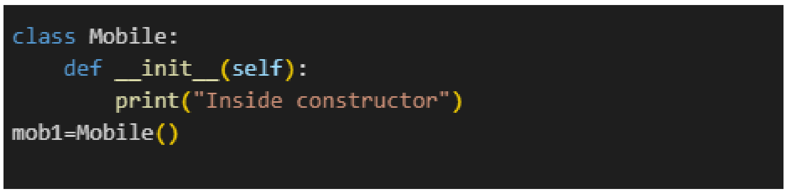
```
class Mobile:
    def __init__(self, brand, price):
        self.brand=brand
        self.price=price

mob1=Mobile("Apple",20000)
mob2=Mobile("Samsung",3000)
```

Name-Jasleen kaur **Branch-D2 CSE (C2)** **URN-2302723** **CRN-2215220**

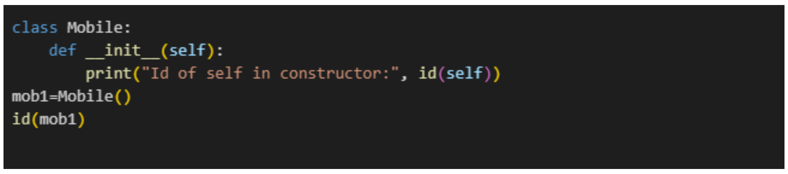
Constructor & Self Introduction:

When we create an object, the special __ init __() method inside the class of that object is invoked automatically. This special function is called as a constructor.



```
class Mobile:
    def __init__(self):
        print("Inside constructor")
mob1=Mobile()
```

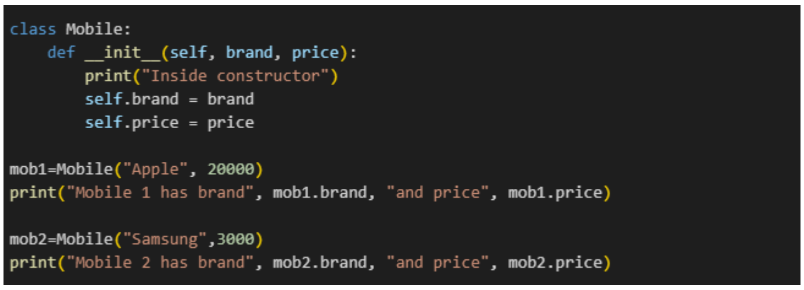
self is not a keyword. self refers to the current object being executed.



```
class Mobile:
    def __init__(self):
        print("Id of self in constructor:", id(self))
mob1=Mobile()
id(mob1)
```

Parameterized constructor :

If a constructor takes parameters then it would be called as parameterized constructor.



```
class Mobile:
    def __init__(self, brand, price):
        print("Inside constructor")
        self.brand = brand
        self.price = price

mob1=Mobile("Apple", 20000)
print("Mobile 1 has brand", mob1.brand, "and price", mob1.price)

mob2=Mobile("Samsung",3000)
print("Mobile 2 has brand", mob2.brand, "and price", mob2.price)
```