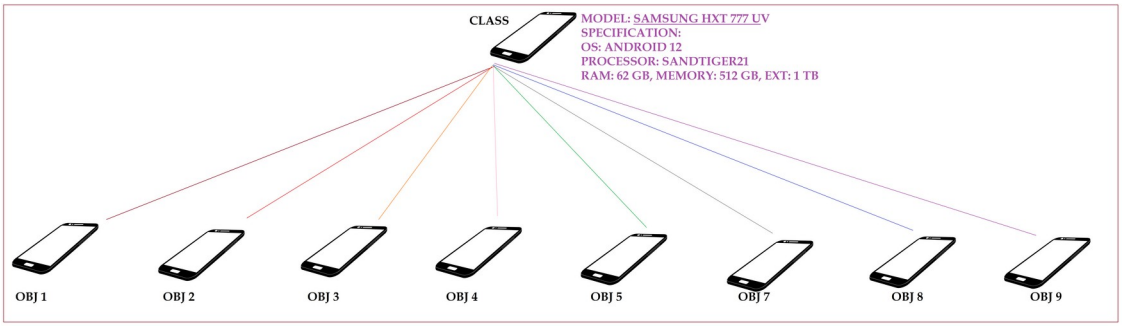
TheDataLytics

python™

ANUURAG EDLABADKAR



Python Classes and Objects

Object is instance of class. A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by their class) for modifying their state.

To understand the need for creating a class let's consider an example, Samsung company want to launch a new mobile in market Samsung HXT 777 UV. Specifications are OS Android 12, Processor Sandtiger21, RAM 62 GB, Int Memory 512, Ext Memory 1 TB. Company prepare blue print of same, finalize the architecture and prepare model. Once testing is done successful, Samsung HXT 777 UV is ready to be launched in market. 1 Million copy of Samsung HXT 777 UV is created in factory.

Now let us co-relate it with our senario. Blue print / prototype / architecture of Samsung HXT 777 UV is **class** and making it's 1 million copies means **objects**.

Class creates a user-defined data structure, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

Some points on Python class:

1. Classes are created by keyword class.
2. Attributes are the variables that belong to a class.
3. Attributes are always public and can be accessed using the dot (.) operator.

Example 1

```
# Creating "class"
class mobile:
  x = "Samsung HXT 777 UV"

# Creating "object" from mobile class
mob_obj1 = mobile()

# Accessing class value from object
mob_obj1.x
```

```
{"type":"string"}
```

self parameter/argument

1. Class methods must have an extra first parameter in the method definition. We do not give a value for this parameter when we call the method, Python provides it.

2. If we have a method that takes no arguments, then we still have to have one argument.

3. This is similar to this pointer in C++ and this reference in Java.

When we call a method of this object as myobject.method(arg1, arg2), this is automatically converted by Python into MyClass.method(myobject, arg1, arg2) – this is all the special self is about.

The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class.

init method The **init** method is similar to constructors in C++ and Java. Constructors are used to initializing the object's state. Like methods, a constructor also contains a collection of statements(i.e. instructions) that are executed at the time of Object creation. It runs as soon as an object of a class is instantiated. The method is useful to do any initialization you want to do with your object.

To understand the meaning of classes we have to understand the built-in **init**() function.

All classes have a function called **init**(), which is always executed when the class is being initiated.

Use the **init**() function to assign values to object properties, or other operations that are necessary to do when the object is being created.

Example 2

```
# creating a **class cl_mobile1**

class cl_mobile1:
  def __init__(self, model, processor, ram, intmem, extmem):
    self.mob_model = model
    self.mob_processor = processor
    self.mob_ram = ram
    self.mob_intmem = intmem
    self.mob_extmem = extmem

# creating **object** of **class cl_mobile1**
obj_cl_mob1 = cl_mobile1("Samsung HXT 777 UV", "Sandtiger21", "RAM 62
GB", "512 GB", "1 TB")

print(obj_cl_mob1.mob_model,
      obj_cl_mob1.mob_processor,
      obj_cl_mob1.mob_ram,
      obj_cl_mob1.mob_intmem,
      obj_cl_mob1.mob_extmem)
```

```
Samsung HXT 777 UV Sandtiger21 RAM 62 GB 512 GB 1 TB
```

Example 3

```
# creating a **class cl_mobile2**

class cl_mobile2:
  def __init__(self, model, processor, ram, intmem, extmem):
    self.mob_model = model
    self.mob_processor = processor
    self.mob_ram = ram
    self.mob_intmem = intmem
    self.mob_extmem = extmem

  def my_mobile(self):
    print("I need a new mobile", self.mob_model, self.mob_processor,
      self.mob_ram, self.mob_intmem, self.mob_extmem, "configuration")

obj_cl_mob2 = cl_mobile2('dell','i9','64 GB','512 GB','1 TB')

obj_cl_mob2.my_mobile()
```

```
I need a new mobile dell i9 64 GB 512 GB 1 TB configuration
```