Unit 7: Object Oriented Programming

Difference between Object-Oriented and Procedural Oriented Programming

Object-Oriented Programming (OOP)	**Procedural-Oriented Programming (Pop)**
It is a bottom-up approach	It is a top-down approach
Program is divided into objects	Program is divided into functions
Makes use of *Access modifiers* 'public', private', protected'	Doesn't use *Access modifiers*
It is more secure	It is less secure
Object can move freely within member functions	Data can move freely from function to function within programs
It supports inheritance	It does not support inheritance

Introduction to OOP:-

- **Object Oriented Programming** is a way of computer programming using the idea of **"objects"** to represents data and methods.
- It is also, an approach used for creating neat and reusable code instead of a redundant one.
- The program is divided into self-contained objects or several mini-programs.
- Every Individual object represents a different part of the application having its own logic and data to communicate within themselves.

What are Classes and Objects?

- Python is an object oriented programming language. Unlike procedure oriented programming, where the main emphasis is on functions, object oriented programming stresses on objects.
- An object is simply a collection of data (variables) and methods (functions) that act on those data. Similarly, a class is a blueprint for that object.
- A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together.

What are Classes and Objects?

- 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.

Defining a Class in Python

- Class definitions begin with a ***class keyword.***
- The first string inside the class is called ***docstring*** and has a brief description about the class. Although not mandatory, this is highly recommended.
- **Syntax:-**
  - class MyNewClass:

    '''This is a docstring. I have created a new class''‘

      pass
- A class creates a new local **namespace** where all its attributes are defined. Attributes may be data or functions.

Defining a Class in Python

- There are also special attributes in it that begins with double underscores __. For example, __doc__ gives us the docstring of that class.
- As soon as we define a class, a new class object is created with the same name. This class object allows us to access the different attributes as well as to instantiate new objects of that class.

Creating an Object in Python

- An Object is an instance of a Class. A class is like a blueprint while an instance is a copy of the class with *actual values*.
- An object consists of :
  - **State:** It is represented by the attributes of an object. It also reflects the properties of an object.
  - **Behaviour:** It is represented by the methods of an object. It also reflects the response of an object to other objects.
  - **Identity:** It gives a unique name to an object and enables one object to interact with other objects.

Creating an Object in Python

- The procedure to create an object is similar to a **function** call.

car1 = car() #car is classname, car1 is object name

Object name = class name()

- This will create a new object instance named car1. We can access the attributes of objects using the object name prefix.
- Attributes may be data or method. Methods of an object are corresponding functions of that class.
- car1.color
- car1.milage
- Objectname.attribute/method

Creating an Object in Python

- notice the **self parameter** in function definition inside the class but we called the method simply as car1.brand() without any arguments. It still worked.
- This is because, whenever an object calls its method, the object itself is passed as the first argument. So, car1.brand() translates into car.brand(car1).
- In general, calling a method with a list of n arguments is equivalent to calling the corresponding function with an argument list that is created by inserting the method's object before the first argument.
- For these reasons, the first argument of the function in class must be the object itself.
- This is conventionally called **self.**

Constructors in Python

- Class functions that begin with double underscore __ are called special functions as they have special meaning.
- Of one particular interest is the __init__() function. This special function gets called whenever a new object of that class is instantiated.
- This type of function is also called **constructors** in Object Oriented Programming (OOP). We normally use it to initialize all the variables
- **Constructors** are generally used for instantiating an object.
- The task of **constructors** is to **initialize(assign values)** to the data members of the class when an object of class is created.

Constructors in Python

- **Syntax of constructor declaration :**

```
def __init__(self):
    # body of the constructor
```

- **Types of constructors :**
- **Default constructor :**
- The default constructor is simple constructor which doesn't accept any arguments.
- It's definition has only one argument which is a reference to the instance being constructed.

Constructors in Python

- **Parameterized constructor :**
- constructor with parameters is known as parameterized constructor.
- The parameterized constructor take its first argument as a reference to the instance being constructed known as self and the rest of the arguments are provided by the programmer.