**6 OOPS CONCEPT**

1. **Class:**

* A class is a collection of objects.  Unlike the primitive data structures, classes are data structures that the user defines. They make the code more manageable.
* We define a class with a keyword “class” following the class\_name and colon. And we consider everything you write under this after using indentation as its body. To make this more understandable let’s see an example.
* For Example

**Class parent():**

**Pass**

1. **Object:**

* The object is an entity that has a state and behavior associated with it. It may be any real-world object like a chair, Table, Classroom, etc. Integers, string, floating-point numbers, even arrays, and dictionaries are all object. An single integer or any single string is an object. For example “Hello World!” is an object.
* An Object consists of:
  + State: It is represented by the attribute of an object. It also reflects the properties of an object.
  + Behavior: It is represented by the method 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 t interact with other objects.

For Example:

Class parent():

Def intro(self):

Return “Hello world”

Obj1 = parent()

Print(Obj1.intro())

1. **Encapsulation:**

* [Encapsulation](https://en.wikipedia.org/wiki/Encapsulation_(computer_programming)) in Python describes the concept of **bundling data and**[methods](https://pynative.com/python-instance-methods/)**within a single unit**. So, for example, when you create a [class](https://pynative.com/python-classes-and-objects/), it means you are implementing encapsulation. A class is an example of encapsulation as it binds all the data members ([instance variables](https://pynative.com/python-instance-variables/)) and methods into a single unit.
* Using encapsulation, we can hide an object’s internal representation from the outside. This is called information hiding

1. **Inheritance:**

* Inheritance enables us to define a class that takes all the functionality from a parent class and allows us to add more. In this tutorial, you will learn to use inheritance in Python.
* Inheritance is a powerful feature in object oriented programming.
* It refers to defining a new [class](https://www.programiz.com/python-programming/class) with little or no modification to an existing class. The new class is called **derived (or child) class** and the one from which it inherits is called the **base (or parent) class.**
* For Example:

Class parent():

Def intro(self):

return “Hello am Parent Class”

class child(parent):

pass

obj1 = child()

print(obj1.intro())

#OUTPUT: “Hello am Parent Class”

* Types of Inheritance :

**1).** Single inheritance  
**2).** Multiple inheritances  
**3).** Multilevel inheritance  
**4).** Hierarchical inheritance  
**5).** Hybrid inheritance

1. **Polymorphism:**

* The literal meaning of polymorphism is the condition of occurrence in different forms.
* Polymorphism is a very important concept in programming. It refers to the use of a single type entity (method, operator or object) to represent different types in different scenarios.
* For Example: The operator ( + ) is works differently for the integers and string.

Print(2 + 3)

#OUTPUT: 4

Print(“Hello”+”World”)

#OUTPUT: “Hello World”

1. **Data Abstraction:**

* Abstraction in OOP is a process of hiding the real implementation of the method by only showing a method signature. In Python, we can achieve abstraction using ABC (abstraction class) or abstract method.
* ABC is a class from the abc module in Python. If we extend any class with ABC and include any abstraction methods, then the classes inherited from this class will have to mandatorily implement those abstract methods.
* When we annotate any method with an abstract method keyword, then it is an abstract method in Python (it won’t have any method implementation). If the parent class has abstract method and not inherited from an abstract class, then it is optional to implement the abstract method .