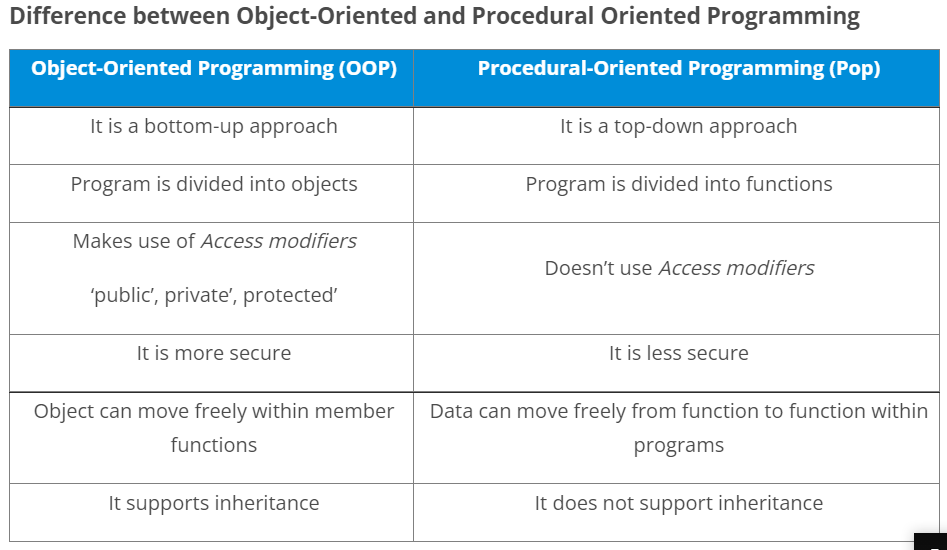
**OOPs**

**What is OOP?**

Object Oriented Programming is a way of computer programming using the idea of “[objects](https://www.edureka.co/blog/python-class/#Objects)” to represent 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 itself.



In Python, object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming. The main concept of OOPs is to bind the data and the functions that work on that together as a single unit so that no other part of the code can access this data.

**Python is a high-level language, we don’t need to manage memory in this, Python does it for us!!**

**What are Python OOPs Concepts?**

Major OOP (object-oriented programming) concepts in Python include **Class, Object, Method, Inheritance, Polymorphism, Data Abstraction, and Encapsulation.**

**What are Classes and Objects?**

A class is a collection of objects or you can say it is a blueprint of objects defining the common attributes and behavior. Now the question arises, how do you do that?

Well, it logically groups the data in such a way that code reusability becomes easy. I can give you a real-life example- think of an office going ’employee’ as a class and all the attributes related to it like ’emp\_name’, ’emp\_age’, ’emp\_salary’, ’emp\_id’ as the objects in [Python](https://www.edureka.co/blog/python-programming-language). Let us see from the coding perspective that how do you instantiate a class and an object.

|  |  |
| --- | --- |
| 1 | class class1(): // class 1 is the name of the class |

Class is defined under a “Class” Keyword.  
**Example:**

**Note:**Python is not case-sensitive.

### **Objects:**

Objects are an instance of a class. It is an entity that has a state and behavior. In a nutshell, it is an instance of a class that can access the data.

**Syntax:**obj = class1()

Here obj is the “object “ of class1.

**PFB**

### **Creating an Object and Class in python:**

**Example:**

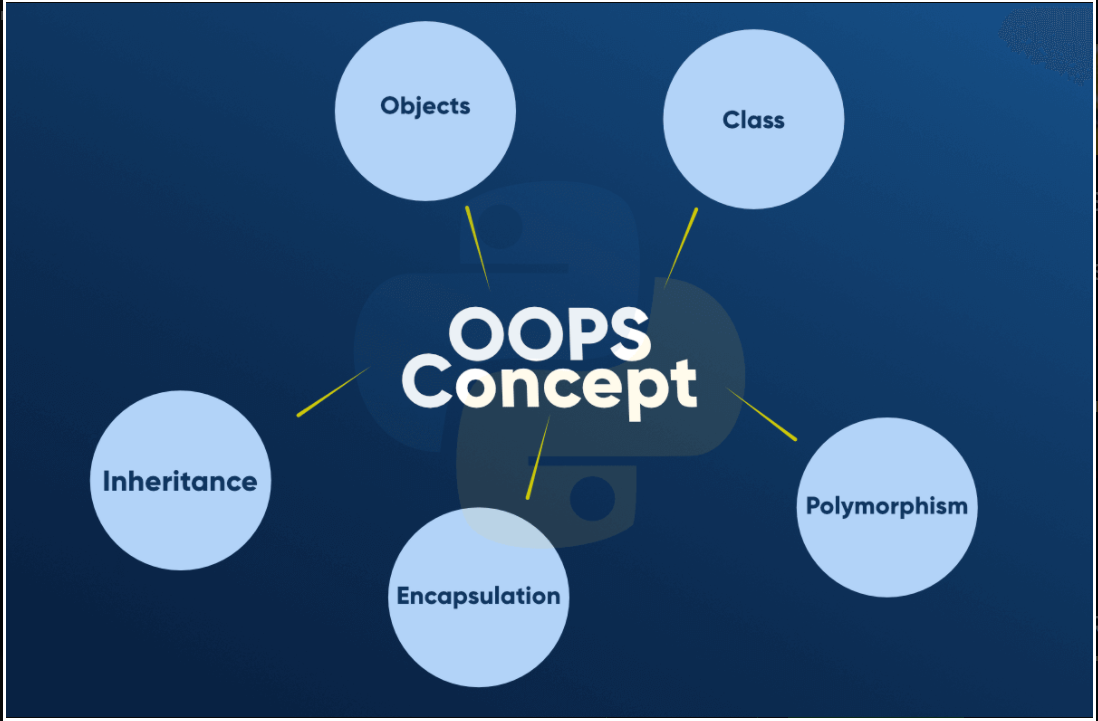
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | class employee():  def \_\_init\_\_(self,name,age,id,salary):          self.name = name // self is an instance of a class  self.age = age  self.salary = salary  self.id = id  emp1 = employee("harshit",22,1000,1234) //creating objects  emp2 = employee("arjun",23,2000,2234)  print(emp1.\_\_dict\_\_)//Prints dictionary |

**Explanation:**’emp1′ and ’emp2′ are the objects that are instantiated against the class ’employee’.Here, the word (\_\_dict\_\_) is a “dictionary” which prints all the values of object ‘emp1’ against the given parameter (name, age, salary).(\_\_init\_\_) acts like a constructor that is invoked whenever an object is created.

## ****Object-Oriented Programming methodologies:****

**Main Concepts of Object-Oriented Programming (OOPs)**

* Class
* Objects
* Polymorphism
* Encapsulation
* Inheritance



**Class**

A class is a collection of objects. A class contains the blueprints or the prototype from which the objects are being created. It is a logical entity that contains some attributes and methods.

To understand the need for creating a class let’s consider an example, let’s say you wanted to track the number of dogs that may have different attributes like breed, age. If a list is used, the first element could be the dog’s breed while the second element could represent its age. Let’s suppose there are 100 different dogs, then how would you know which element is supposed to be which? What if you wanted to add other properties to these dogs? This lacks organization and it’s the exact need for classes.

**Some points on Python class:**

* Classes are created by keyword class.
* Attributes are the variables that belong to a class.
* Attributes are always public and can be accessed using the dot (.) operator. Eg.: Myclass.Myattribute

**Objects**

The object is an entity that has a state and behavior associated with it. It may be any real-world object like a mouse, keyboard, chair, table, pen, etc. Integers, strings, floating-point numbers, even arrays, and dictionaries, are all objects. More specifically, any single integer or any single string is an object. The number 12 is an object, the string “Hello, world” is an object, a list is an object that can hold other objects, and so on. You’ve been using objects all along and may not even realize it.

**An object consists of :**

* **State:** It is represented by the attributes of an object. It also reflects the properties of an object.
* **Behavior:** 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.

To understand the state, behavior, and identity let us take the example of the class dog (explained above).

* The identity can be considered as the name of the dog.
* State or Attributes can be considered as the breed, age, or color of the dog.
* The behavior can be considered as to whether the dog is eating or sleeping.

**Inheritance**

Inheritance is the capability of one class to derive or inherit the properties from another class. The class that derives properties is called the derived class or child class and the class from which the properties are being derived is called the base class or parent class. The benefits of inheritance are:

* It represents real-world relationships well.
* It provides the reusability of a code. We don’t have to write the same code again and again. Also, it allows us to add more features to a class without modifying it.
* It is transitive in nature, which means that if class B inherits from another class A, then all the subclasses of B would automatically inherit from class A.

## Polymorphism (GFG)

Polymorphism simply means having many forms. For example, we need to determine if the given species of birds fly or not, using polymorphism we can do this using a single function.

**What is Polymorphism:** The word polymorphism means having many forms. In programming, polymorphism means the same function name (but different signatures) being used for different types.



