# **5 Examples to Jumpstart Oriented Programming in Python**

OOP stands for Object Oriented Programming. This concept is a style of solving programming problems where properties and behavior of a real-life object is packaged as a single entity in the code.

This style of coding enables modularizing and scaling with least amount of issues.

Python is a dynamically typed, high level interpreted programming language. Python supports several OOP features including the following:

* Classes and Objects
* Encapsulation
* Inheritance
* Polymorphism

### **1. Classes in Python**

Class is a blueprint of the real-life entity. In Python, it is created using the class keyword as shown in the following code snippet.

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

### **2. Objects in Python**

Once a Person class is defined you can use it to create an instance by passing values as shown below.

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

if \_\_name\_\_ == "\_\_main\_\_":

p = Person("ranjeeta", 23)

print(p.name)

### **3. Inheritance in Python**

As the name suggest, this concept is about inheriting properties from an existing entity. This increases reusability of code. Single, Multiple and Multi-level inheritances are few of the many types supported by Python.

The following example shows how to use inheritance in python:

class Person:

def \_\_init\_\_(self):

pass

# Single level inheritance

class Employee(Person):

def \_\_init\_\_(self):

pass

# Multi-level inheritance

class Manager(Employee):

def \_\_init\_\_(self):

pass

# Multiple Inheritance

class Enterprenaur(Person, Employee):

def \_\_init\_\_(self):

pass

In multiple inheritance, classes are inherited from left to right inside parenthesis, depending on Method Resolution Order (MRO) algorithm of Python.

### **4. Encapsulation in Python**

It is the concept of wrapping data such that the outer world has access only to exposed properties. Some properties can be hidden to reduce vulnerability. This is an implementation of data hiding. For example, you want buy a pair of trousers from an online site. The data that you want is its cost and availability. The number of items present and their location is information that you are not bothered about. Hence it is hidden.

In Python this is implemented by creating private, protected and public instance variables and methods.

### **5. Polymorphism in Python**

This is a concept where a function can take multiple forms depending on the number of arguments or type of arguments given to the function.

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

def show\_salary(self):

print("Salary is unknown")

class Employee(Person):

def \_\_init\_\_(self, name, age, salary):

super().\_\_init\_\_(name, age)

self.salary = salary

def show\_salary(self):

print("Salary is", self.salary)

if \_\_name\_\_ == "\_\_main\_\_":

p = Person("y", 23)

x = Employee("x", 20, 100000)

p.show\_salary() # Salary is unknown

x.show\_salary() # Salary is 100000

In the above example, super keyword is used to call a method of parent class. Both classes have the method show\_salary. Depending on the object type that makes a call to this function, the output varies.