CodeWithHarry

[Hindi] Classes, Objects and Constructors | Object Oriented Programming Using Python Tutorial #2

Classes, Objects and Constructors

In the previous article, I gave you a rough idea about the concept of objects and classes in object-oriented programming. In this article, we will be discussing Classes, Objects, and Constructors in detail.

First, let's discuss what are classes and objects?

Classes are just like the templates or blueprints, which hold all the functions of objects and objects are the actual product or instance of the class. Objects and classes helps us to simplify our work by increasing the resuability of the code.

Basically, before creating an object, we need to create a class. Talking with an example, before creating anything, we gather all the information on how to start, what all things are required, and from where to start to get the possible results. Similarly, before creating any object we design a blueprint of the object i.e., class. Let's see how we can create a class in Python :

Creating a class :

Classes in python are created using the keyword class. The class is followed by the class name.

Syntax :

```
class Employee:
    pass
```

In the above code, we've created a class named Employee.

Note: 'pass' means in the current scenario we are performing nothing, later we will proceed with something. 'pass' is used as a placeholder for future code.

Creating objects of the class :

Suppose, **harry** and **rohan** are the workers, means the object of the class Employee. Here, all the properties of rohan and harry will be the same as they both are from the same company or from the same **template (class)**. Properties like working timings, office location, etc., will be the same. Both harry and rohan (object) will be placed in different locations in memory.

```
harry = Employee()
rohan = Employee()
```

Adding methods and attributes :

- **Attributes** mean data of the Employees such as their names, email, passwords, salary.
- **Methods** are functions written inside a class. Suppose we want to increase the salary of the Employees so we will create a method or function to carry on this process.

There are two ways to add methods and attributes to a class.

Method 1 :

1. Here we are initializing the object of class Employee with the variables fname, lname, salary **outside the class.** rohan and harry are the object of the class.

```
harry.fname ="harry"
harry.lname = 'jackson'
harry.salary = 4400

rohan.fname = "rohan"
rohan.lname = "das"
rohan.salary = 4400
```

Printing the values:

```
#Retrieve the first name of both employees.
print(harry.fname)
print(rohan.fname)
```

```
#Retrieve the last name of both employees.
print(harry.lname)
print(rohan.lname)

#Retrieve the salary of both employees.
print(harry.salary)
print(rohan.salary)
```

Output :

Copy

```
harry
rohan
jackson
das
4400
4400
```

These steps can also be followed to practice class and object, but this nullifies the meaning of **class.**

Method 2 :

- We will start with creating a constructor inside the Employee class.
- Don't worry constructor is nothing but a default function which is invoked automatically everytime an object is created.
- In Python, __init__() method is called everytime an object is created.

```
def __init__(self):
    pass
```

The __init__() method accepts self as the first argument which helps us to access the methods or attributes of the class. We can pass multiple arguments to __init__() as shown in the code given below :

```
def __init__(self,fname,lname,salary):
        pass
```

Let's create a constructor inside the Employee class :

```
class Employee:
    def __init__(self,fname,lname,salary):
        self.fname=fname
        self.lname=lname
        self.salary=salary
```

Creating objects of the Employee class :

```
harry = Employee("harry","jackson",4400)
rohan = Employee("rohan","das",4400)
```

In the above code :

- “harry” will go in the position of fname.
- “Jackson” will go in the position of name.
- 4400 will go in the position of salary.
- Self is our object means harry. The same applies to the second object (rohan) also.

Printing values :

```
print(harry.fname)
print(rohan.fname)
print(harry.lname)
print(rohan.lname)
print(harry.salary)
print(rohan.salary)
```

Output:

```
harry
rohan
jackson
das
```

```
4400
4400
```

Both the outputs are the same but Type 2 is the actual implementation of the class.
This is just the basic implementation of class and objects. In the next article, we will be discussing instances of class and variables.
Thank you for reading this article. I hope this has cleared your doubts. If you haven't checked out the whole playlist yet, move on to codewithharry.com or my YouTube channel to access it. Looking forward to seeing you all next time. Till then, keep coding.

Previous Next

CodeWithHarry | Copyright © 2022 CodeWithHarry.com