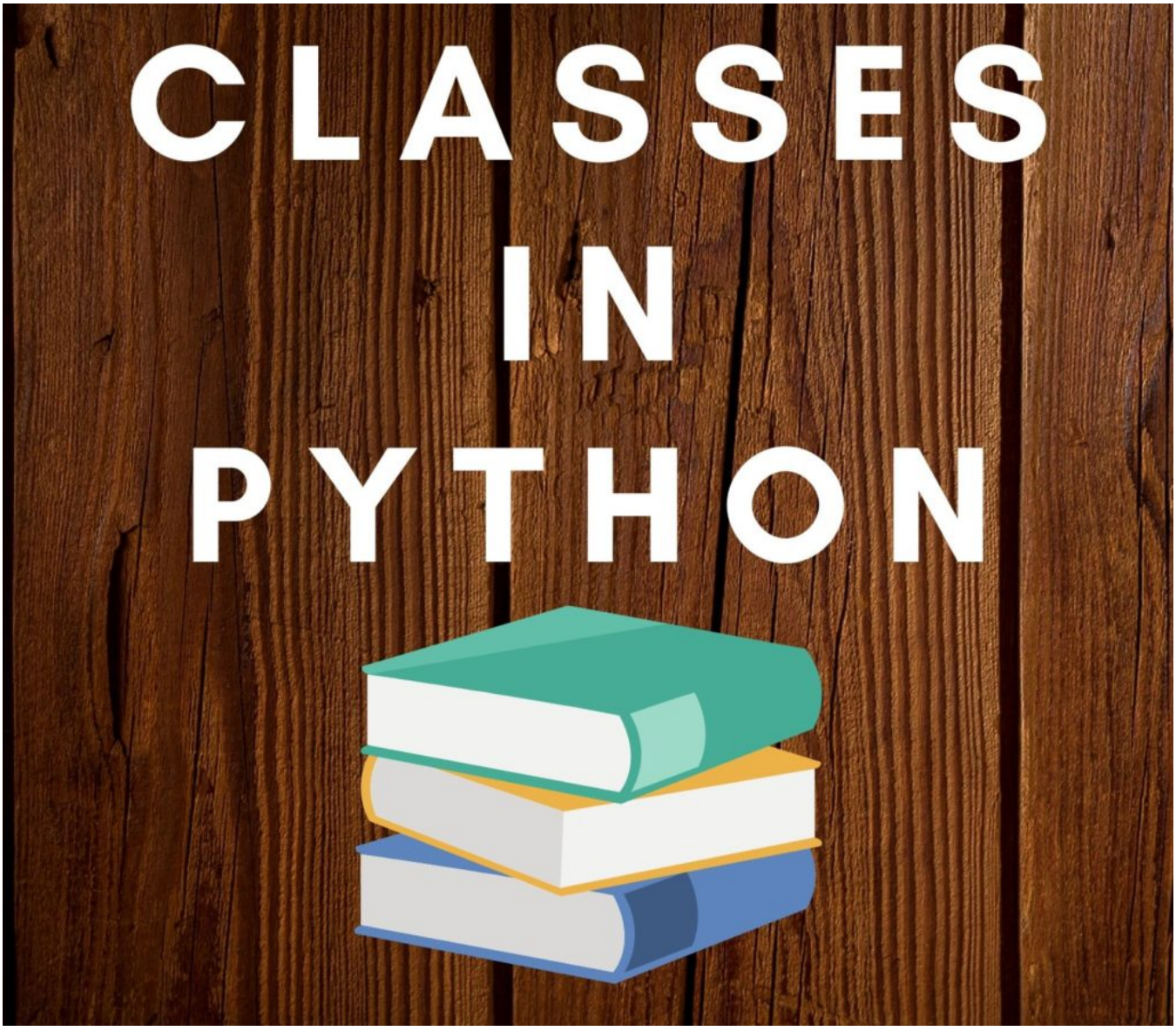
Classes in Python-Complete Tutorial

Python / March 22, 2020



This **Classes in Python-Complete Tutorial** is made for you to give complete knowledge of the **Classes in Python**. At the end of this tutorial, you will have full knowledge of the **Classes in Python**.

Hello, & Welcome! to the **Classes in Python-Complete Tutorial**.

Part 13- Classes in Python-Complete Tutorial

In this article of Python Tutorial, you will learn following-

1. What are Classes and Objects in Python?
2. How to Create a Class?
3. Methods and attributes in Class.
4. How to create an Object in Class?
5. How to Access attributes or Methods of Class?
6. The Self Parameter.
7. The __init__() Function in Class.

What are Classes and Objects in Python?

Python is an Object-Oriented Programming Language, which is based on objects. An object is basically a collection of functions and attributes. And Class shows the blueprint of an object.

You can take a class as a prototype or sketch of a car, which includes all the information about its machines, body, and design. Based on this information you can build a car. Here, the car is an object. Because you can make many cars from this prototype. Similarly, you can create multiple objects with one Class.

How to Create a Class?

For creating a Class in python, the 'class' keyword is used. After 'class' keyword use class name and at the end, use colon (:).

Syntax of Class declaration-

```
class class_name:
```

Let's have a look in the example, to create a class in Python-

```
class Car:
   //class functions and attributes
```

Methods and attributes in Class.

After declaring a class, you can define functions or methods inside the class. You can create multiple methods inside the class.

You can define attributes inside the class as many as you want.

Let's understand with the help of an example-

```
Class Car:
   car_name="BMW"
   def change_name(self,new_name):
       self.name=new_name
```

Here, we have created a class **'car'**, inside the class 'car', we have created an attribute name as **car_name** and one method **change_name**. Similarly, you can create any method and attributes inside the class.

You can also create a constructor function, like the __init__() function. We will discuss the __init__() function later in detail, but here I am gonna tell you how you can create an __init__() function.

Let's have a look at the example-

```
class Planet:
    def __init__(self):
       self.name="Hoth"
       self.radius=200000
       self.gravity=5.5
       self.system="Hoth system"
```

Here, an __init__() function is **constructor function**. self is an instance. We will discuss this later in detail.

How to create an Object in Class?

After class declaration and function declaration, next, we need to create an object of a class. Object creation is very easy in Python.

You can create an object by class name and object name. Let's have a look at the syntax of object creation.

```
object_name=class_name()
```

Look 🙂 it is as simple as that.

Let's see in the example, how we can create an object of our class **Planet**.

```
class Planet:
    def __init__(self):
        self.name="Hoth"
        self.radius=200000
        self.gravity=5.5
        self.system="Hoth system"
hoth=Planet()
```

Here, '**hoth**' is an object of **Planet** class. **One important thing** you should keep in mind that an object of a class is created from outside of the class, which means start writing from the corner, without any indentation. It is considered as outside of the class.

How to Access attributes or Methods of Class?

For accessing any attribute or method of a class, you need to use an object name and dot operator(.). Let's see its syntax-

```
object_name.attribute_name    #for attribute
object_name.function_name()   #for function or method
```

After looking at syntax, now let's understand with our Planet class example, how you can call a function or attributes.

```
class Planet:
    def __init__(self):
       self.name="Hoth"
       self.radius=200000
       self.gravity=5.5
       self.system="Hoth system"
hoth=Planet()
print(f'Name is:{hoth.name})     #attribute calling
```

Here, in 'hoth.name' 'hoth' is an object of a class and name is an attribute of a class.

You can also access the method or functions of a class. First, we will create one method then we will call this method outside of the class.

Let's see in the example of our **Planet** class.

```
class Planet:
    def __init__(self):
       self.name="Hoth"
       self.radius=200000
       self.gravity=5.5
       self.system="Hoth system"
    def orbit(self):
       return f'{self.name} is orbitting in {self.system}'
hoth=Planet()
print(f'Name is:{hoth.name})
print(hoth.orbit())          # method calling
```

Here, we are calling the **orbit** method by object name **hoth**.

Now, I hope you have learned, how to call the method and attribute of a class.

The Self Parameter.

Self is a pointer to the class instance, which means the Self shows an object of a class. The self parameter is used to access attributes which belong to the class.

Whenever you create a function inside a class, you must write Self first, because it represents the class.

For example-

```
class person
    def name(self, first_name, last_name):
        self.first_name=first_name
        self.last_name=last_name
```

To call any attribute or variable of a class, always use Self, because it represents that it is a member of a class.

Let's understand with the help of an example-

```
def print_name(self):
    print(self.first_name,"hi",self.last_name)
```

Here, we are calling attribute **first_name** and **last_name** with the help of the Self parameter.

Always pass the Self in every member function or attribute of a class, then pass other arguments. Suppose if member function doesn't take any arguments, which means function with no arguments, then also pass the Self.

For Example-

```
def print_name(self):
    print(self.first_name,"hi",self.last_name)
```

Here, function print_name doesn't have any argument, but we have passed the Self.

The __init__() Function in Class.

A function with start with a double underscore (__) is known as special functions. An __init__() function is built-in function. All classes have an __init__() function, which executes when the class initiates.

By using an __init__() function, you can create a class with different values for arguments. An __init__() function is known as special functions.

You can create multiple instances with the help of an __init__() function.

Let's understand with the help of an example-

```
class Planet:
     def __init__(self,name,age):
          self.name=name
          self.age=age
#1st Object
planet1=Planet('Hoth',22)
print(f'Name:{planet1.name}')
print(f'Age:{planet1.age}')

#2nd Object
planet2=Planet('Naboo',25)
print(f'Name:{planet2.name}')
print(f'Age:{planet2.age}')
```

Here, we have created multiple objects of a class by __init__() function, and we have provided different values to different objects.

That's all for the **Classes in Python**. I hope now you have a better understanding of the **Classes in Python**.

Congratulations! You successfully learned the **Classes in** Python.

In the next tutorial, we will start learning **Methods & Attributes in Python**.

Till then, Enjoy Learning Python!

All the Best!

Are you ML Beginner and confused, from where to start ML, then read my BLOG – How do I learn Machine Learning?

If you are looking for Machine Learning Algorithms, then read my Blog – Top 5 Machine Learning Algorithm.

If you are wondering about Machine Learning, read this Blog- What is Machine Learning?

Thank YOU!

Though of the Day...

> *"'Live as if you were to die tomorrow. Learn as if you were to live forever."*
>
> *– Mahatma Gandhi*

← Previous Post Next Post →

Leave a Comment

Your email address will not be published. Required fields are marked *

Type here..

Name*

Email*

Website

Save my name, email, and website in this browser for the next time I comment.

Post Comment »

Follow Us on Twitter-

Follow us on Pinterest-

Our Facebook Group-