**Python - Class Attributes**

The properties or variables defined inside a class are called as **Attributes**. An attribute provides information about the type of data a class contains. There are two types of attributes in Python namely **instance attribute** and **class attribute**.

The instance attribute is defined within the constructor of a Python class and is unique to each instance of the class. And, a class attribute is declared and initialized outside the constructor of the class.

Class Attributes (Variables)

**Class attributes** are those variables that belong to a class and whose value is shared among all the instances of that class. A class attribute remains the same for every instance of the class.

**Class attributes** are defined in the class but outside any method. They cannot be initialized inside **\_\_init\_\_()** constructor. They can be accessed by the name of the class in addition to the object. In other words, a class attribute is available to the class as well as its object.

class Employee:

name = "Bhavesh Aggarwal"

age = "30"

# instance of the class

emp = Employee()

# accessing class attributes

print("Name of the Employee:", emp.name)

print("Age of the Employee:", emp.age)

Instance Attributes Vs Class Attributes

The below table shows the difference between instance attributes and class attributes −

|  |  |  |
| --- | --- | --- |
| **SNo.** | **Instance Attribute** | **Class Attribute** |
| 1 | It is defined directly inside the \_\_init\_\_() function. | It is defined inside the class but outside the \_\_init\_\_() function. |
| 2 | Instance attribute is accessed using the object name followed by dot notation. | Class attributes can be accessed by both class name and object name. |
| 3 | The value of this attribute cannot be shared among other objects. | Its value is shared among other objects of the class. |
| 4 | Changes made to the instance attribute affect only the object within which it is defined. | Changes made to the class attribute affect all the objects of the given class. |

Print Page

Previou

A Python **class method** is a method that is bound to the class and not to the instance of the class. It can be called on the class itself, rather than on an instance of the class.

Most of us often get class methods confused with static methods. Always remember, while both are called on the class, **static methods** do not have access to the "cls" parameter and therefore it cannot modify the class state.

Unlike class method, the **instance method** can access the instance [variables](https://www.tutorialspoint.com/python/python_variables.htm) of the an object. It can also access the class variable as it is common to all the objects.

What is Python Static Method?

In Python, a **static method** is a type of method that does not require any instance to be called. It is very similar to the class method but the difference is that the static method doesn't have a mandatory argument like reference to the object **− self** or reference to the class **− cls**.

Static methods are used to access static fields of a given class. They cannot modify the state of a class since they are bound to the class, not instance.

AD

How to Create Static Method in Python?

There are two ways to create Python static methods −

* Using staticmethod() Function
* Using @staticmethod Decorator

The **Python access modifiers** are used to restrict access to class members (i.e., variables and methods) from outside the class. There are three types of access modifiers namely public, protected, and private.

* **Public members** − A class member is said to be public if it can be accessed from anywhere in the program.
* **Protected members** − They are accessible from within the class as well as by classes derived from that class.

**\_name (single underscore)**

* **Private members** − They can be accessed from within the class only.

**\_\_name (double underscore)**

Inheritance:

What is Inheritance in Python?

**Inheritance** is one of the most important features of object-oriented programming languages like Python. It is used to inherit the properties and behaviours of one class to another. The class that inherits another class is called a **child class** and the class that gets inherited is called a **base class or parent class**.

If you have to design a new class whose most of the attributes are already well defined in an existing class, then why redefine them? Inheritance allows capabilities of existing class to be reused and if required extended to design a new class.

Inheritance comes into picture when a new class possesses 'IS A' relationship with an existing class. For example, Car IS a vehicle, Bus IS a vehicle, Bike IS also a vehicle. Here, Vehicle is the parent class, whereas car, bus and bike are the child classes.

AD



* Single Inheritance
* Multiple Inheritance
* Multilevel Inheritance
* Hierarchical Inheritance
* Hybrid Inheritance

