**Python Class Variables**

Updated on: February 7, 2022 | [7 Comments](https://pynative.com/python-class-variables/#llc_comments)

In [Object-oriented programming](https://pynative.com/python/object-oriented-programming/), when we design a class, we use instance variables and class variables.

In Class, attributes can be defined into two parts:

* [**Instance variables**](https://pynative.com/python-instance-variables/): If the value of a variable varies from object to object, then such variables are called instance variables.
* **Class Variables**: A class variable is a variable that is declared inside of class, but outside of any instance method or \_\_init\_\_() method.

**After reading this article, you’ll learn**:

* How to create and access class variables
* Modify values of a class variables
* Instance variable vs. class variables
* Behaviour of a class variable in inheritance

**Table of contents**

* [What is an Class Variable in Python?](https://pynative.com/python-class-variables/#h-what-is-an-class-variable-in-python)
* [Create Class Variables](https://pynative.com/python-class-variables/#h-create-class-variables)
* [Accessing Class Variables](https://pynative.com/python-class-variables/#h-accessing-class-variables)
* [Modify Class Variables](https://pynative.com/python-class-variables/#h-modify-class-variables)
* [Class Variable vs Instance variables](https://pynative.com/python-class-variables/#h-class-variable-vs-instance-variables)
* [Class Variables In Inheritance](https://pynative.com/python-class-variables/#h-class-variables-in-inheritance)
* [Wrong Use of Class Variables](https://pynative.com/python-class-variables/#h-wrong-use-of-class-variables)

**What is an Class Variable in Python?**

If the **value of a**[**variable**](https://pynative.com/python-variables/)**is not varied from object to object**, such types of variables are called class variables or static variables.

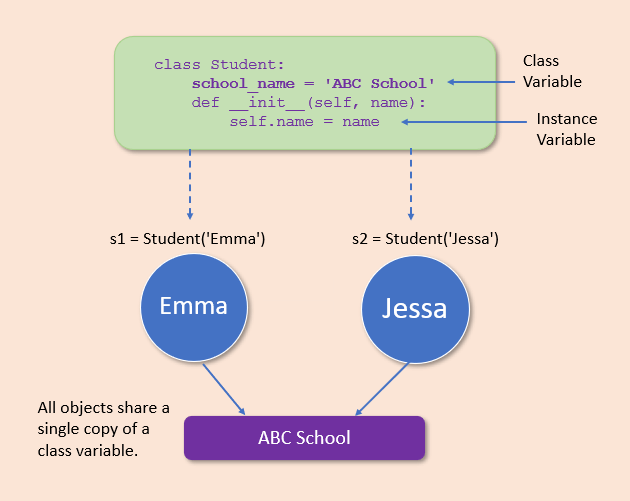
Class variables are **shared by all instances of a class**. Unlike instance variable, the value of a class variable is not varied from object to object,

In Python, Class variables are declared when a [class](https://pynative.com/python-classes-and-objects/) is being constructed. They are not defined inside any methods of a class because of this only one copy of the static variable will be created and shared between all objects of the class.

For example, in Student class, we can have different instance variables such as name and roll number because each student’s name and roll number are different.

But, if we want to include the school name in the student class, we must use the class variable instead of an instance variable as the school name is the same for all students. So instead of maintaining the separate copy in each object, we can create a class variable that will hold the school name so all students (objects) can share it.

We can add any number of class variables in a class.

Understand Class Variables

**Create Class Variables**

A class variable is declared inside of class, but outside of any instance method or \_\_init\_\_() method.

By convention, typically it is placed right below the class header and before the constructor method and other methods.

**Example**:

**class** Student:

# Class variable

school\_name = 'ABC School '

**def** \_\_init\_\_(self, name, roll\_no):

self.name = name

self.roll\_no = roll\_no

# create first object

s1 = Student('Emma', 10)

**print**(s1.name, s1.roll\_no, Student.school\_name)

# access class variable

# create second object

s2 = Student('Jessa', 20)

# access class variable

**print**(s2.name, s2.roll\_no, Student.school\_name)

**Output**

Emma 10 ABC School

Jessa 20 ABC School

In the above example, we created the class variable school\_name and accessed it using the object and class name.

**Note**: Like regular [variables](https://pynative.com/python-variables/), class variables can store data of any type. We can use [Python list](https://pynative.com/python-lists/), [Python tuple](https://pynative.com/python-tuples/), and [Python dictionary](https://pynative.com/python-dictionaries/) as a class variable.

**Accessing Class Variables**

We can access static variables either by class name or by object reference, but it is recommended to use the class name.

In Python, we can access the class variable in the following places

* Access inside the [constructor](https://pynative.com/python-constructors/) by using either self parameter or class name.
* Access class variable inside instance method by using either self of class name
* Access from outside of class by using either object reference or class name.

**Example 1**: Access Class Variable in the constructor

**class** Student:

# Class variable

school\_name = 'ABC School '

# constructor

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

self.name = name

# access class variable inside constructor using self

**print**(self.school\_name)

# access using class name

**print**(Student.school\_name)

# create Object

s1 = Student('Emma')

**Output**

ABC School

ABC School

**Example 2**: Access Class Variable in Instance method and outside class

**class** Student:

# Class variable

school\_name = 'ABC School '

# constructor

**def** \_\_init\_\_(self, name, roll\_no):

self.name = name

self.roll\_no = roll\_no

# Instance method

**def** show(self):

**print**('Inside instance method')

# access using self

**print**(self.name, self.roll\_no, self.school\_name)

# access using class name

**print**(Student.school\_name)

# create Object

s1 = Student('Emma', 10)

s1.show()

**print**('Outside class')

# access class variable outside class

# access using object reference

**print**(s1.school\_name)

# access using class name

**print**(Student.school\_name)

**Output**

Inside instance method

Emma 10 ABC School

ABC School

Outside class

ABC School

ABC School

In this example, we accessed the class variable school\_name using class name and a self keyword inside a method.

**Modify Class Variables**

Generally, we assign value to a class variable inside the class declaration. However, we can change the value of the class variable either in the class or outside of class.

**Note**: We should change the class variable’s value using the class name only.

**Example**

**class** Student:

# Class variable

school\_name = 'ABC School '

# constructor

**def** \_\_init\_\_(self, name, roll\_no):

self.name = name

self.roll\_no = roll\_no

# Instance method

**def** show(self):

**print**(self.name, self.roll\_no, Student.school\_name)

# create Object

s1 = Student('Emma', 10)

**print**('Before')

s1.show()

# Modify class variable

Student.school\_name = 'XYZ School'

**print**('After')

s1.show()

**Output**:

Before

Emma 10 ABC School

After

Emma 10 XYZ School

**Note**:

It is best practice to use a class name to change the value of a class variable. Because if we try to change the class variable’s value by using an object, a new instance variable is created for that particular object, which shadows the class variables.

**Example**:

**class** Student:

# Class variable

school\_name = 'ABC School '

# constructor

**def** \_\_init\_\_(self, name, roll\_no):

self.name = name

self.roll\_no = roll\_no

# create Objects

s1 = Student('Emma', 10)

s2 = Student('Jessa', 20)

**print**('Before')

**print**(s1.name, s1.roll\_no, s1.school\_name)

**print**(s2.name, s2.roll\_no, s2.school\_name)

# Modify class variable using object reference

s1.school\_name = 'PQR School'

**print**('After')

**print**(s1.name, s1.roll\_no, s1.school\_name)

**print**(s2.name, s2.roll\_no, s2.school\_name)

**Output**:

Before

Emma 10 ABC School

Jessa 20 ABC School

After

Emma 10 PQR School

Jessa 20 ABC School

A new instance variable is created for the s1 object, and this variable shadows the class variables. So always use the class name to modify the class variable.

**Class Variable vs Instance variables**

The following table shows the difference between the instance variable and the class variable.

In Python, properties can be defined into two parts:

* **Instance variables**: Instance variable’s value varies from object to object. Instance variables are not shared by objects. Every object has its own copy of the instance attribute
* **Class Variables**: A class variable is a variable that is declared inside of class, but outside of any instance method or \_\_init\_\_() method. Class variables are shared by all instances of a class.

**Read More**: [Instance variables in Python with Examples](https://pynative.com/python-instance-variables/)

| **Instance Variable** | **Class Variable** |
| --- | --- |
| Instance variables are not shared by objects. Every object has its own copy of the instance attribute | Class variables are shared by all instances. |
| Instance variables are declared inside the constructor i.e., the \_\_init\_\_() method. | Class variables are declared inside the class definition but outside any of the instance methods and constructors. |
| It is gets created when an instance of the class is created. | It is created when the program begins to execute. |
| Changes made to these variables through one object will not reflect in another object. | Changes made in the class variable will reflect in all objects. |

Class Variables vs. Instance Variables

**Example**:

Let’s see the example to create a class variable and instance variable.

**class** Car:

# Class variable

manufacturer = 'BMW'

**def** \_\_init\_\_(self, model, price):

# instance variable

self.model = model

self.price = price

# create Object

car = Car('x1', 2500)

**print**(car.model, car.price, Car.manufacturer)

**Output**:

x1 2500 BMW

**Class Variables In Inheritance**

As you know, only one copy of the class variable will be created and shared between all objects of that class.

When we use [inheritance](https://pynative.com/python-inheritance/), all variables and methods of the base class are available to the child class. In such cases, We can also change the value of the parent class’s class variable in the child class.

We can use the parent class or child class name to change the value of a parent class’s class variable in the child class.

**Example**

**class** Course:

# class variable

course = "Python"

**class** Student(Course):

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

self.name = name

**def** show\_student(self):

# Accessing class variable of parent class

**print**('Before')

**print**("Student name:", self.name, "Course Name:", Student.course)

# changing class variable value of base class

**print**('Now')

Student.course = "Machine Learning"

**print**("Student name:", self.name, "Course Name:", Student.course)

# creating object of Student class

stud = Student("Emma")

stud.show\_student()

**Output**

Before

Student name: Emma Course Name: Python

Now

Student name: Emma Course Name: Machine Learning

What if both **child class and parent class has the same class variable name**. In this case, the child class will not inherit the class variable of a base class. So it is recommended to create a separate class variable for child class instead of inheriting the base class variable.

**Example**:

**class** Course:

# class variable

course = "Python"

**class** Student(Course):

# class variable

course = "SQL"

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

self.name = name

**def** show\_student(self):

# Accessing class variable

**print**('Before')

**print**("Student name:", self.name, "Course Name:", Student.course)

# changing class variable's value

**print**('Now')

Student.course = "Machine Learning"

**print**("Student name:", self.name, "Course Name:", Student.course)

# creating object of Student class

stud = Student("Emma")

stud.show\_student()

# parent class course name

**print**('Parent Class Course Name:', Course.course)

**Output**:

Before

Student name: Emma Course Name: SQL

Now

Student name: Emma Course Name: Machine Learning

Parent Class Course Name: Python

**Wrong Use of Class Variables**

In Python, we should properly use the class variable because all objects share the same copy. Thus, if one of the objects modifies the value of a class variable, then all objects start referring to the fresh copy.

For example,

**Example**

**class** Player:

# class variables

club = 'Chelsea'

sport = 'Football'

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

# Instance variable

self.name = name

**def** show(self):

**print**("Player :", 'Name:', self.name, 'Club:', self.club, 'Sports:', self.sport)

p1 = Player('John')

# wrong use of class variable

p1.club = 'FC'

p1.show()

p2 = Player('Emma')

p2.sport = 'Tennis'

p2.show()

# actual class variable value

**print**('Club:', Player.club, 'Sport:', Player.sport)

**Output**

Player : Name: John Club: FC Sports: Football

Player : Name: Emma Club: Chelsea Sports: Tennis

Club: Chelsea Sport: Football

In the above example, the instance variable name is unique for each player. The class variable team and sport can be accessed and modified by any object.

Because both objects modified the class variable, a new instance variable is created for that particular object with the same name as the class variable, which shadows the class variables.

In our case, for object p1 new instance variable club gets created, and for object p2 new instance variable sport gets created.

So when you try to access the class variable using the p1 or p2 object, it will not return the actual class variable value.

To avoid this, always modify the class variable value using the class name so that all objects gets the updated value. Like this

Player.club = 'FC'

Player.sport = 'Tennis'