Python Class Variables

Updated on: February 7, 2022 | + 7 Comments

In 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**: If the value of a variable varies from object to object, then such variables are called

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

class variables

- Behaviour of a class variable in inheritance

Table of contents

Variable in Python?

If the **value of a variable 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 is being constructed. They are not defined inside any methods of a class because of this

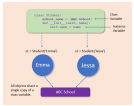
shared between all objects of the class.

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

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:

```
    def __init
        self.n
        self.r

# create first
s1 = Student(
print(s1.name,
# access class

# create secor
s2 = Student(
# access class
print(s2.name,
```

Run

Output

```
Emma 10 ABC Sch
Jessa 20 ABC Sc
```

In the above example, we

accessed it using the object and class name.

Note: Like regular variables, class variables can store data of any type. We can use Python list, Python tuple, and Python dictionary 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 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

```
    # construc
    def __init
        self.r
        # acce
        print(
        # acce
        print(

# create Objec
s1 = Student(
```

Run

Output

```
ABC School
ABC School
```

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

```
class Student
    # Class va
    school_nar
```

```
        self.r

    # Instance
    def show(s
        print(
        # acce
        print(
        # acce
        print(

# create Objec
s1 = Student(
s1.show()

print('Outside
# access class
# access using
print(s1.schoo

# access using
print(Student.
```

Run

Output

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 va
    school_nar

    # construc
    def __init
        self.r
        self.r

    # Instance
    def show(s
        print(

# create Objec
s1 = Student(
print('Before
s1.show()

# Modify class
Student.schoo
print('After'
s1.show()
```

Run

Output:

```
Before
Emma 10 ABC Sch

After
Emma 10 XYZ Sch
```

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

created for that particular object, which shadows the class variables.

Example:

```
clas

# cr
s1 =
s2 =

prin
prin
prin

# Mc
s1.s
prin
prin
prin
```

```
n
```

Output:

```
Before
Emma 1
Jessa

After
Emma 1
Jessa
```

A new instance variable is created for the s1 object, and this variable shadows the class variable. 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

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

Instance variables are not shared by objects. Every object has its own copy of the instance attribute	Cla var sha ins
Instance variables are declared inside the constructor i.e., the `__init__()` method.	Cla var dec ins cla def but any ins me and cor
It is gets created when an instance of	It is wh prc

the class is created.	beg exe
Changes made to these variables through one object will not reflect in another object.	Cha ma cla var refl obj

Class Variables vs. Instance Variables

Example:

Let's see the example to create a class variable

```
class Car:
    # Class va
    manufactur

    def __init
        # inst
        self.r
        self.p

# create Objec
car = Car('x1
print(car.mode
```

Run

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.

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 va
    course = '

class Studentl
```

```
        # Acc
        print
        print
        # cha
        print
        Stude
        print

# creating ob
stud = Studen
stud.show_stu
```

Run

Output

```
Before
Student name: E

Now
Student name: E
```

What if both **child class and parent class has the same class variable name**. In

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 va
    course = '

class Student(
    # class va
    course = '

    def __init
        self.n

    def show_s
        # Acce
        print(
        print(
        # char
        print(
        Studen
        print(

# creating obj
stud = Student
stud.show_stu

# parent class
print('Parent
```

Run

Output:

```
Before
Student name: Er
Now
Student name: Er

Parent Class Cou
```

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

For example,

Example

```
class Player:
    # class va
    club = 'Ch
    sport = 'F

    def __init
        # Inst
        self.n

    def show(s
        print(

p1 = Player('J

# wrong use of
p1.club = 'FC
p1.show()

p2 = Player('P
p2.sport = 'Te
p2.show()

# actual class
print('Club:',
```

Run

Output

```
Player : Name: 
Player : Name: 
Club: Chelsea S
```

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.

`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 =
Player.sport =
```

Filed Under: Python, Python Object-Oriented Programming (OOP)

Did you find this page helpful? Let others know about it. **Sharing helps me** continue to create free Python resources.

About Vishal

Founder of PYnative.com I

articles to help developers. Follow me on Twitter. All the best for your future Python endeavors!

Related Tutorial Topics:

Python Python Object-Oriented Programming (OOP)

Python Exercises and Quizzes

Free coding exercises and quizzes cover Python basics, data structure, data analytics, and more.

Quizzes

Each Exercise contains 10 questions

Each Quiz contains 12-15 MCQ

Exercises

Quizzes

About PYnative

PYnative.com is for Python lovers. Here, You can get Tutorials, Exercises, and Quizzes to practice and **improve your Python skills**.

Explore Python

- Learn Python
- Python Basics
- Python Databases
- Python Exercises
- Python Quizzes
- Online Python Code Editor
- Python Tricks

Follow Us

To get New Python Tutorials, Exercises, and Quizzes

- Twitter
- Facebook
- Sitemap

Legal Stuff

- About Us
- Contact Us

We use cookies to improve your experience. While using PYnative, you agree to have read and accepted our **Terms Of Use**, **Cookie Policy**, and **Privacy Policy**.