Posted In

Python | Python Object-Oriented Programming (OOP)

Python OOP

All Python Topics

Python Class Variables

Updated on: February 7, 2022 | + 5 Comments

In Object-oriented programming, when we design a class, we use instance 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 instance variables.

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:

and access class variables

- Modify values of a class variables
- Instance variable vs. class variables
- Behaviour of a class variable in inheritance

conte nts

What is an Class Variable in Python?

If the **value of a variable is not varied from**

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

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

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

(objects) can share it.

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

Understand Class Variable

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

constructor method and other methods.

Example:

```
class S
    # C
    sch

    def


# creat
s1 = St
print(s
# acces

# creat
s2 = St
# acces
print(s
```

Run

Output

```
Emma 10 A
Jessa 20
```

we created the class variable `school_name` and 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.

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

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

nce or class name.

Example 1: Access Class Variable in the constructor

```
class S
    # C
    sch

    # c
    def
```

```
# creat
s1 = St
```

Run

Output

```
ABC Schoo
ABC Schoo
```

Access Class Variable in Instance method and outside class

```
class S
    # C
    sch

    # c
    def

    # I
    def

# creat
s1 = St
s1.show

print('
# acces
# acces
print(s

# acces
print(S
```

Output

```
Inside in
Emma 10 A
ABC Schoo

Outside c
ABC Schoo
ABC Schoo
```

In this example, we accessed the class variable `school_name` using class name and a `self` keyword inside a method.

Modify Class

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

```
    # c
    def

    # I
    def

# creat
s1 = St
print('
s1.show

# Modif
Student
print('
s1.show
```

Run

Output:

```
Before
Emma 10 A

After
Emma 10 X
```

N

o

It is best practice to use a class name

h
a
n
g
e
t
h
e
v
a
l
u
e
o
f
a
c
l
a
s
s
v
a
ri
a
b
l
e
.

a
u
s
e
if
w
e
t
r
y
t
o
c
h
a
n
g
e
t
h
e
c
l
a
s
s
v
a
ri

e
,

s
v
a
l
u
e
b
y
u
s
i
n
g
a
n
o
b
j
e
c
t,
a
n
e
w
i
n

n
c
e
v
a
ri
a
b
l
e
i
s
c
r
e
a
t
e
d
f
o
r
t
h
a
t
p
a
r

l
a
r
o
b
j
e
c
t,
w
h
i
c
h
s
h
a
d
o
w
s
t
h
e
c
l
a
s
s

a
b
l
e
s.

E
x
a
m
p
l
e
:

Run

Output:

```
Be
Em
Je

Af
Em
Je
```

A new instance

```
v
a
ri
a
b
l
e
i
s
c
r
e
a
t
e
d
f
o
r
t
h
e
s
1
o
b
j
e
c
```

```
d
t
h
i
s
v
a
ri
a
b
l
e
s
h
a
d
o
w
s
t
h
e
c
l
a
s
s
v
a
```

l
e
s.
S
o
a
l
w
a
y
s
u
s
e
t
h
e
c
l
a
s
s
n
a
m
e
t
o
m

y
t
h
e
c
l
a
s
s
v
a
ri
a
b
l
e
.

Class Variable vs Instance

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

ce 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

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 Variable

Instance variables are not shared by objects. Every object has its own copy of the instance attribute	
	Instance variables are declared inside the constructor i.e., the `__init__()` method.

It is gets created when an instance of the class is created.

Changes made to these variables through one object will not reflect in another object.

Class Variables vs. Instance Variables

Example:

create a class variable and instance variable.

```
class C
    # C
    man

    def


# creat
car = C
print(c
```

Run

Output:

```
x1 2500 B
```

Class Variables In

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, 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

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 C
    # c
    cou

class S

    def

    def
```

Run

Output

```
Before
Student n

Now
Student n
```

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 recommend

variable for child class instead of inheriting the base class variable.

Example:

```
class C
    # c
    cou

class S
    # c
    cou

    def

    def

# creat
stud = 
stud.sh
```

Run

Output:

```
Before
Student n
Now
Student n

Parent Cl
```

Wrong Use of Class Variables

In Python, we should properly use the class variable because all

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 P
    # c
    clu
    spo

    def

    def

p1 = Pl

# wrong
```

```
p2.spor
p2.show

# actua
print('
```

Run

Output

```
Player :
Player :
Club: Che
```

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

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

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

```
Player.
Player.
```


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

Did you find this page helpful? Let others know about it. **Sharing**

Python resources.

About Vishal

Founder of PYnative.com I am a Python developer and I love to write articles to help

me on Twitter. All the best for your future Python endeav ors!

Related Tutorial Topics:

Python
Python Object-Oriented Programming (OOP)

Pytho

es and Quizzes

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

15+ **Topic-specific Exercises and Quizzes**

Each Exercis

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.