Python Programming

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

PYTHON'S OBJECT OROENTED PROGRAMMING - 4

Guido Van Rossum

Agenda:

I. Various places to declare Static Variables

II. How to access static variables?

III. Where we can modify the value of static variable?

IV. Example programs about Instance and Static Variables

V. How to delete static variables?

1. Various places to declare Static Variables

Recap:

- If the value of a variable is not varied from object to object, such type of variables we have to declare with in the class directly but outside of methods. Such type of variables are called Static variables.
- For total class only one copy of static variable will be created and shared by all objects of that class.
- We can access static variables either by class name or by object reference. But recommended to use class name.
- Most of the times static variables should be declared within the class directly.

Note:

- In the case of instance variables for every object a separate copy will be created, but in the case of static variables for total class only one copy will be created and shared by every object of that class.

Eg:

```
class student:
    college_name = "RGMCET"        # Static variable

    def __init__(self,name,rollno):
        self.name = name
        self.rollno = rollno        # Instance variable
```

I. Various places to declare static variables:

1. In general we can declare within the class directly but from out side of any method (or) constructor.
2. Inside constructor by using class name.
3. Inside instance method by using class name.
4. Inside class method by using either class name or cls variable.
5. Inside static method by using class name.
6. Outside of the class by using class name.

1. In general we can declare within the class directly but from out side of any method or constructor:

Eg:

```
class Test:
    x = 10
print(Test.__dict__)          # Class dictionary
```

```
{'__module__': '__main__', 'x': 10, '__dict__': <attribute '__dict__' of 'Te
st' objects>, '__weakref__': <attribute '__weakref__' of 'Test' objects>, '_
_doc__': None}
```

2. Inside constructor by using class name:

Eg:

```
class Test:
    x = 10
    def __init__(self):
        Test.y = 20           # Static Variable
t = Test()
print(Test.__dict__)
```

```
{'__module__': '__main__', 'x': 10, '__init__': <function Test.__init__ at 0
x00000157CE7E3B80>, '__dict__': <attribute '__dict__' of 'Test' objects>, '_
_weakref__': <attribute '__weakref__' of 'Test' objects>, '__doc__': None,
'y': 20}
```

3. Inside instance method by using class name:

Eg :

I:

```
class Test:
    x = 10

    def __init__(self):
        Test.y = 20          # Static Variable


    def m1(self):
        Test.z = 30

t = Test()
print(Test.__dict__)
```

```
{'__module__': '__main__', 'x': 10, '__init__': <function Test.__init__ at 0
x00000157CE81F040>, 'm1': <function Test.m1 at 0x00000157CE81F0D0>, '__dict_
_': <attribute '__dict__' of 'Test' objects>, '__weakref__': <attribute '__w
eakref__' of 'Test' objects>, '__doc__': None, 'y': 20}
```

II.

```
class Test:
    x = 10

    def __init__(self):
        Test.y = 20           # Static Variable


    def m1(self):
        Test.z = 30

t = Test()
# print(Test.__dict__)
t.m1()
print(Test.__dict__)
```

```
{'__module__': '__main__', 'x': 10, '__init__': <function Test.__init__ at 0
x00000157CE81F4C0>, 'm1': <function Test.m1 at 0x00000157CE81F430>, '__dict_
_': <attribute '__dict__' of 'Test' objects>, '__weakref__': <attribute '__w
eakref__' of 'Test' objects>, '__doc__': None, 'y': 20, 'z': 30}
```

4. Inside class method by using either class name or cls variable:

Eg:

```
class Test:
    x = 10

    def __init__(self):
        Test.y = 20          # Static Variable


    def m1(self):
        Test.z = 30

    @classmethod
    def m2(cls):
        cls.a = 40
        Test.b = 50

t = Test()
t.m1()           # Calling Instance method
Test.m2()        # Calling class method
print(Test.__dict__)
```

```
{'__module__': '__main__', 'x': 10, '__init__': <function Test.__init__ at 0
x00000157CE81F310>, 'm1': <function Test.m1 at 0x00000157CE81F3A0>, 'm2': <c
lassmethod object at 0x00000157CE822B50>, '__dict__': <attribute '__dict__'
of 'Test' objects>, '__weakref__': <attribute '__weakref__' of 'Test' object
s>, '__doc__': None, 'y': 20, 'z': 30, 'a': 40, 'b': 50}
```

5.Inside static method by using class name:

Eg:

```
class Test:
    x = 10

    def __init__(self):
        Test.y = 20          # Static Variable


    def m1(self):
        Test.z = 30

    @classmethod
    def m2(cls):
        cls.a = 40
        Test.b = 50

    @staticmethod
    def m3():
        Test.c = 60

t = Test()
t.m1()          # Calling Instance method
Test.m2()       # Calling Class method
Test.m3()       # Calling Static method
print(Test.__dict__)
```

```
{'__module__': '__main__', 'x': 10, '__init__': <function Test.__init__ at 0
x00000157CE81F700>, 'm1': <function Test.m1 at 0x00000157CE81F790>, 'm2': <c
lassmethod object at 0x00000157CE822880>, 'm3': <staticmethod object at 0x00
000157CE822520>, '__dict__': <attribute '__dict__' of 'Test' objects>, '__we
akref__': <attribute '__weakref__' of 'Test' objects>, '__doc__': None, 'y':
20, 'z': 30, 'a': 40, 'b': 50, 'c': 60}
```

6. Outside of the class by using class name:

Eg:

```
class Test:
    x = 10                  # 1

    def __init__(self):
        Test.y = 20          # 2


    def m1(self):
        Test.z = 30         # 3

    @classmethod
    def m2(cls):
        cls.a = 40
        Test.b = 50         # 4

    @staticmethod
    def m3():
        Test.c = 60         # 5

Test.d = 70                 # 6

t = Test()
t.m1()          # Calling Instance method
Test.m2()       # Calling Class method
Test.m3()       # Calling Static method
print(Test.__dict__)
```

```
{'__module__': '__main__', 'x': 10, '__init__': <function Test.__init__ at 0
x00000157CE81F5E0>, 'm1': <function Test.m1 at 0x00000157CE81F670>, 'm2': <c
lassmethod object at 0x00000157CE7E6EB0>, 'm3': <staticmethod object at 0x00
000157CE7E6CA0>, '__dict__': <attribute '__dict__' of 'Test' objects>, '__we
akref__': <attribute '__weakref__' of 'Test' objects>, '__doc__': None, 'd':
70, 'y': 20, 'z': 30, 'a': 40, 'b': 50, 'c': 60}
```

II. How to access static variables?

1. **Inside constructor:** By using either self or class name
2. **Inside instance method:** By using either self or class name
3. **Inside class method:** By using either cls variable or class name
4. **Inside static method:** By using class name
5. **From outside of class:** By using either object reference or class name

1. Inside constructor: By using either self or class name

Eg:

```
class Test:
    a= 10
    def __init__(self):
        print(self.a)
        print(Test.a)
t = Test()
```

```
10
10
```

2. Inside instance method: By using either self or class name

Eg:

```
class Test:
    a= 10
    def __init__(self):
        print(self.a)
        print(Test.a)

    def m1(self):
        print(self.a)
        print(Test.a)
t = Test()
t.m1()
```

```
10
10
10
10
```

3. Inside class method: By using either cls variable or class name

Eg:

```
class Test:
    a= 10
    def __init__(self):
        print(self.a)
        print(Test.a)

    def m1(self):
        print(self.a)
        print(Test.a)

    @classmethod
    def m2(cls):
        print(cls.a)
        print(Test.a)
t = Test()
t.m1()
t.m2()
```

```
10
10
10
10
10
10
```

4. Inside static method: By using class name

Eg:

```
class Test:
    a= 10
    def __init__(self):
        print(self.a)
        print(Test.a)

    def m1(self):
        print(self.a)
        print(Test.a)

    @classmethod
    def m2(cls):
        print(cls.a)
        print(Test.a)

    @staticmethod
    def m3():
        print(Test.a)
t = Test()
t.m1()
t.m2()
t.m3()
```

```
10
10
10
10
10
10
10
```

```
class Test:
    a= 10
    def __init__(self):
        print(self.a)
        print(Test.a)

    def m1(self):
        print(self.a)
        print(Test.a)

    @classmethod
    def m2(cls):
        print(cls.a)
        print(Test.a)

    @staticmethod
    def m3():
        print(Test.a)
        print(self.a)
t = Test()
t.m1()
t.m2()
t.m3()
```

```
10
10
10
10
10
10
10
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-17-6600e2ff061e> in <module>
     21 t.m1()
     22 t.m2()
---> 23 t.m3()

<ipython-input-17-6600e2ff061e> in m3()
     17     def m3():
     18         print(Test.a)
---> 19         print(self.a)
     20 t = Test()
     21 t.m1()

NameError: name 'self' is not defined
```

5. From outside of class: By using either object reference or class name

Eg:

```
class Test:
    a= 10

    def __init__(self):
        print("1.Inside constructor:  ")
        print(self.a)
        print(Test.a)


    def m1(self):
        print("2. Inside instance method:
        print(self.a)
        print(Test.a)


    @classmethod
    def m2(cls):
        print("3. Inside class method:  ")
        print(cls.a)
        print(Test.a)


    @staticmethod
    def m3():
        print("4. Inside static method:  ")
        print(Test.a)

t = Test()

t.m1()
t.m2()
t.m3()
print("5. Outside of the class :  ")
print(t.a)
print(Test.a)
```

```
1.Inside constructor:
10
10
2. Inside instance method:
10
10
3. Inside class method:
10
10
4. Inside static method:
10
5. Outside of the class :
10
10
```

III. Where we can modify the value of static variable?

- We can modify the value of static variable anywhere (either within the class or outside the class) by using class name.
- But inside class method, we can also use cls variable.
- We can not modify the value of static variable by using self or object reference variable.

Eg 1:

```
class Test:
    a = 10

print(Test.a)
```

```
10
```

Eg 2:

```
class Test:
    a = 10
    def __init__(self):
        self.a = 20

t = Test()
print(Test.a)        # VALUE is not modified
```

```
10
```

Eg 3:

```
class Test:
    a = 10
    def __init__(self):
        Test.a = 20

t = Test()
print(Test.a)        # VALUE is modified
```

```
20
```

Eg 4:

```
class Test:
    a = 10
    def __init__(self):
        self.a = 20

    def m1(self):
        Test.a = 30

    @classmethod
    def m2(cls):
        cls.a = 40
        Test.a = 50

    @staticmethod
    def m3():
        Test.a = 60

t = Test()
t.m1()

print(Test.a)
```

```
30
```

Eg 5:

```
class Test:
    a = 10
    def __init__(self):
        self.a = 20

    def m1(self):
        Test.a = 30

    @classmethod
    def m2(cls):
        cls.a = 40
        Test.a = 50

    @staticmethod
    def m3():
        Test.a = 60

t = Test()
t.m1()
t.m2()

print(Test.a)
```

```
50
```

Eg 6:

```
class Test:
    a = 10
    def __init__(self):
        self.a = 20

    def m1(self):
        Test.a = 30

    @classmethod
    def m2(cls):
        cls.a = 40
        Test.a = 50

    @staticmethod
    def m3():
        Test.a = 60

t = Test()
t.m1()
Test.m2()

print(Test.a)
```

```
50
```

Eg 7:

```
class Test:
    a = 10
    def __init__(self):
        self.a = 20

    def m1(self):
        Test.a = 30

    @classmethod
    def m2(cls):
        cls.a = 40
        Test.a = 50

    @staticmethod
    def m3():
        Test.a = 60

t = Test()
t.m1()
t.m2()
t.m3()

print(Test.a)
```

60

Eg 8:

```
class Test:
    a = 10
    def __init__(self):
        self.a = 20

    def m1(self):
        Test.a = 30

    @classmethod
    def m2(cls):
        cls.a = 40
        Test.a = 50

    @staticmethod
    def m3():
        Test.a = 60

t = Test()
t.m1()
t.m2()
Test.m3()

print(Test.a)
```

60

Eg 9:

```
class Test:
    a = 10
    def __init__(self):
        self.a = 20

    def m1(self):
        Test.a = 30

    @classmethod
    def m2(cls):
        cls.a = 40
        Test.a = 50

    @staticmethod
    def m3():
        Test.a = 60

t = Test()
t.m1()
t.m2()
t.m3()      # You can use class name
Test.a = 70
print(Test.a)
```

```
70
```

IV. Example programs about Instance and Static Variables

I.

```
class Test:
    a=10
    def m1(self):
        self.a=888
t1=Test()
t1.m1()
print(Test.a)
print(t1.a)
```

```
10
888
```

Conclusion:

- If we change the value of static variable by using either self or object reference variable, then the value of static variable won't be changed, just a new instance variable with that name will be added to that particular object.

II.

```
class Test:
    a=10
    def m1(self):
        Test.a=888
t1=Test()
t1.m1()
print(Test.a)
print(t1.a)
```

```
888
888
```

III.

```
class Test:
    a=10
    def m1(self):
        Test.b=888
t1=Test()
t1.m1()
print(Test.a)
print(t1.a)
```

```
10
10
```

IV.

```
class Test:
    a=10
    def m1(self):
        Test.b=888
t1=Test()
t1.m1()
print(Test.b)
print(t1.b)
print(t1.a)
```

```
888
888
10
```

V.

```
class Test:
    a=10
    def m1(self):
        Test.b=888
        self.b=999
t1=Test()
t1.m1()
print(Test.b)
print(t1.b)
print(t1.a)
```

```
888
999
10
```

VI.

```
class Test:
    a=10
    def m1(self):
        Test.a=888
        self.b=999
t1=Test()
t1.m1()
print(Test.a)
print(t1.b)
print(t1.a)
```

```
888
999
888
```

VII.

```
class Test:
    a=10                    # static variable
    def __init__(self):
        self.b=20           # instance variable

t1=Test()                   # objects t1,t2 created
t2=Test()

print('t1:',t1.a,t1.b)      # 10,20
print('t2:',t2.a,t2.b)      # 10,20

t1.a=888
t1.b=999

print('t1:',t1.a,t1.b)      # 888,999
print('t2:',t2.a,t2.b)      #10,20
```

```
t1: 10 20
t2: 10 20
t1: 888 999
t2: 10 20
```

VIII.

```
class Test:
    a=10                    # static variable
    def __init__(self):
        self.b=20           # instance variable

t1=Test()                   # objects t1,t2 created
t2=Test()

print('t1:',t1.a,t1.b)      # 10,20
print('t2:',t2.a,t2.b)      # 10,20

Test.a=888
Test.b=999    # Both instance & static variables may have same name in Python.

print('t1:',t1.a,t1.b)    # 888,20
print('t2:',t2.a,t2.b)    #888,20
print("Test :",Test.a,Test.b)     # 888,999
```

```
t1: 10 20
t2: 10 20
t1: 888 20
t2: 888 20
Test : 888 999
```

IX.

```
class Test:
    a=10                    # static variable
    def __init__(self):
        self.b=20           # instance variable

t1=Test()                  # objects t1,t2 created
t2=Test()


Test.a=888
t1.b=999    # Both instance & static variables may have same name in Python.

print('t1:',t1.a,t1.b)   # 888,999
print('t2:',t2.a,t2.b)   #888,20
```

```
t1: 888 999
t2: 888 20
```

x.

```
class Test:
    a=10
    def __init__(self):
        self.b=20

    def m1(self):
        self.a=888
        self.b=999

t1=Test()
t2=Test()

t1.m1()

print(t1.a,t1.b)      # 888,999
print(t2.a,t2.b)      # 10,20
```

```
888 999
10 20
```

XI.

```
class Test:
    a=10
    def __init__(self):
        self.b=20

    def m1(self):
        self.a=888
        self.b=999

t1=Test()
t2=Test()

t1.m1()
t2.m1()

print(t1.a,t1.b)      # 888,999
print(t2.a,t2.b)      # 888,999
```

```
888 999
888 999
```

XII.

```
class Test:
    a=10
    def __init__(self):
        self.b=20

    @classmethod
    def m1(cls):
        cls.a=888
        cls.b=999

t1=Test()
t2=Test()

t1.m1()

print(t1.a,t1.b)        # 888 20
print(t2.a,t2.b)        # 888 20
print(Test.a,Test.b)    # 888 999
```

```
888 20
888 20
888 999
```

V. How to delete static variables?

- We can delete static variables from anywhere by using the following syntax:

  del classname.variablename

- But inside class method we can also use cls variable to delete static variable.

  del cls.variablename

Eg:

I.

```
class Test:
    a = 10

    @classmethod
    def m1(cls):
        #del Test.a
        del cls.a

print(Test.__dict__)
```

```
{'__module__': '__main__', 'a': 10, 'm1': <classmethod object at 0x00000157D
0550B50>, '__dict__': <attribute '__dict__' of 'Test' objects>, '__weakref_
_': <attribute '__weakref__' of 'Test' objects>, '__doc__': None}
```

II.

```
class Test:
    a = 10

    @classmethod
    def m1(cls):
        #del Test.a
        del cls.a

Test.m1()     # By using class name we are calling 'm1' method
print(Test.__dict__)   # static variable 'a' is deleted
```

```
{'__module__': '__main__', 'm1': <classmethod object at 0x00000157CF76DEE0>,
'__dict__': <attribute '__dict__' of 'Test' objects>, '__weakref__': <attrib
ute '__weakref__' of 'Test' objects>, '__doc__': None}
```

III.

```
class Test:
    a = 10

    @classmethod
    def m1(cls):
        #del Test.a
        del cls.a

del Test.a    # By using class name we are deleting 'a'
print(Test.__dict__)   # static variable 'a' is deleted
```

```
{'__module__': '__main__', 'm1': <classmethod object at 0x00000171455776A0>,
'__dict__': <attribute '__dict__' of 'Test' objects>, '__weakref__': <attrib
ute '__weakref__' of 'Test' objects>, '__doc__': None}
```

IV.

```
class Test:
    a=10
    def __init__(self):
        Test.b=20
        del Test.a

    def m1(self):
        Test.c=30
        del Test.b

    @classmethod
    def m2(cls):
        cls.d=40
        del Test.c

    @staticmethod
    def m3():
        Test.e=50
        del Test.d
print(Test.__dict__)    # only one static variable 'a' present.
```

```
{'__module__': '__main__', 'a': 10, '__init__': <function Test.__init__ at 0
x00000171455B5040>, 'm1': <function Test.m1 at 0x00000171455B50D0>, 'm2': <c
lassmethod object at 0x00000171455B6310>, 'm3': <staticmethod object at 0x00
000171455B6340>, '__dict__': <attribute '__dict__' of 'Test' objects>, '__we
akref__': <attribute '__weakref__' of 'Test' objects>, '__doc__': None}
```

V.

```
class Test:
    a=10
    def __init__(self):
        Test.b=20
        del Test.a

    def m1(self):
        Test.c=30
        del Test.b

    @classmethod
    def m2(cls):
        cls.d=40
        del Test.c

    @staticmethod
    def m3():
        Test.e=50
        del Test.d

t = Test()
print(Test.__dict__)
```

```
{'__module__': '__main__', '__init__': <function Test.__init__ at 0x00000171
455B5550>, 'm1': <function Test.m1 at 0x00000171455B54C0>, 'm2': <classmetho
d object at 0x00000171455B6E50>, 'm3': <staticmethod object at 0x00000171455
B6E80>, '__dict__': <attribute '__dict__' of 'Test' objects>, '__weakref__':
<attribute '__weakref__' of 'Test' objects>, '__doc__': None, 'b': 20}
```

VI.

```
class Test:
    a=10
    def __init__(self):
        Test.b=20
        del Test.a

    def m1(self):
        Test.c=30
        del Test.b

    @classmethod
    def m2(cls):
        cls.d=40
        del Test.c

    @staticmethod
    def m3():
        Test.e=50
        del Test.d

t = Test()
t.m1()
print(Test.__dict__)
```

```
{'__module__': '__main__', '__init__': <function Test.__init__ at 0x00000171
455B5670>, 'm1': <function Test.m1 at 0x00000171455B55E0>, 'm2': <classmetho
d object at 0x00000171455B6670>, 'm3': <staticmethod object at 0x00000171455
B62B0>, '__dict__': <attribute '__dict__' of 'Test' objects>, '__weakref__':
<attribute '__weakref__' of 'Test' objects>, '__doc__': None, 'c': 30}
```

VII.

```
class Test:
    a=10
    def __init__(self):
        Test.b=20
        del Test.a

    def m1(self):
        Test.c=30
        del Test.b

    @classmethod
    def m2(cls):
        cls.d=40
        del Test.c

    @staticmethod
    def m3():
        Test.e=50
        del Test.d

t = Test()
t.m1()
t.m2()       # You can also call class method with class name
print(Test.__dict__)
```

```
{'__module__': '__main__', '__init__': <function Test.__init__ at 0x00000171
455B58B0>, 'm1': <function Test.m1 at 0x00000171455B5820>, 'm2': <classmetho
d object at 0x00000171455B6B20>, 'm3': <staticmethod object at 0x00000171455
B6970>, '__dict__': <attribute '__dict__' of 'Test' objects>, '__weakref__':
<attribute '__weakref__' of 'Test' objects>, '__doc__': None, 'd': 40}
```

IX.

```
class Test:
    a=10
    def __init__(self):
        Test.b=20
        del Test.a

    def m1(self):
        Test.c=30
        del Test.b

    @classmethod
    def m2(cls):
        cls.d=40
        del Test.c

    @staticmethod
    def m3():
        Test.e=50
        del Test.d

t = Test()
t.m1()
t.m2()
t.m3()
print(Test.__dict__)
```

```
{'__module__': '__main__', '__init__': <function Test.__init__ at 0x00000171
455B5AF0>, 'm1': <function Test.m1 at 0x00000171455B5A60>, 'm2': <classmetho
d object at 0x00000171455B4610>, 'm3': <staticmethod object at 0x00000171455
B4640>, '__dict__': <attribute '__dict__' of 'Test' objects>, '__weakref__':
<attribute '__weakref__' of 'Test' objects>, '__doc__': None, 'e': 50}
```

X.

```
class Test:
    a=10
    def __init__(self):
        Test.b=20
        del Test.a

    def m1(self):
        Test.c=30
        del Test.b

    @classmethod
    def m2(cls):
        cls.d=40
        del Test.c

    @staticmethod
    def m3():
        Test.e=50
        del Test.d

t = Test()
t.m1()
t.m2()
t.m3()
del Test.e
print(Test.__dict__)
```

```
{'__module__': '__main__', '__init__': <function Test.__init__ at 0x00000171
455B5DC0>, 'm1': <function Test.m1 at 0x00000171455B5E50>, 'm2': <classmetho
d object at 0x0000017145577FD0>, 'm3': <staticmethod object at 0x00000171455
77F70>, '__dict__': <attribute '__dict__' of 'Test' objects>, '__weakref__':
<attribute '__weakref__' of 'Test' objects>, '__doc__': None}
```

Key Points to Ponder:

1. We can delete static variables either by class name or cls variable.

2. We can not modify or delete static variables by using object reference variables /self. But we can access static variables by using object reference variables /self.

3. If we are trying to modify a static variable by using object reference variables /self, then a new instance variable will be added to that particular object.

4. If we are trying to delete static variable by using object reference variables /self, then we will get error.

Examples:

I.

```
class Test:
    a=10

t1=Test()
print(t1.a)
```

```
10
```

II.

```
class Test:
    a=10

t1=Test()
del t1.a    # AttributeError: a
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-12-151e37ef32df> in <module>
      3
      4 t1=Test()
----> 5 del t1.a    # AttributeError: a

AttributeError: a
```

- We can modify or delete static variables only by using class name or cls variable.

Eg:

```
class Test:
    a=10

t1=Test()
del Test.a
print(Test.__dict__)
```

```
{'__module__': '__main__', '__dict__': <attribute '__dict__' of 'Test' objec
ts>, '__weakref__': <attribute '__weakref__' of 'Test' objects>, '__doc__':
None}
```

Any question?

If you try to practice programs yourself, then you will learn many things automatically

Spend few minutes and then enjoy the study

Thank You