OOPs | 02-Variables

Class and Instance variables

Posted on October 17, 2021
Last updated on February 6, 2023

2. Variables/Attribute

- Variables are also term as Attribute

There are 3 types of variables are allowed.

1. `Instance` /Regular Variables (Object Level Variables)
2. `Static` Variables (`Class` Level Variables)
3. `Local` variables (Method Level Variables)

- Attributes are looked first in the instance and then in the class

```python
class Student:
    # class variable
    college = "XYZ-College"

    def __init__(self, stream):
        # instance variables
        self.stream = stream

    def data(self):
        # local variable
        name = 'Amrit'

        # access of all variables
        print(self.stream, self.college, name)


if __name__ == '__main__':
    pw1 = Student('Arts')
    pw1.data()
    # Arts XYZ-College Amrit XYZ-College
    print(pw1.stream, pw1.college)
    # Arts XYZ-College

    # Can't access local variable outside the block
```

2.1 Instance/Regular Variables (Object Level Variables)

If the value of a `variable is varied from object to object`, then such type of variables are called instance variables.

For every object a `separate copy` of instance variables will be created.

`self or obj name` is used to access instance variable of current object

- Where we can declare Instance variables:
  1. Inside Constructor by using self variable
  2. Inside Instance Method by using self variable
  3. Outside of the class by using object reference variable

```
class Student:
    def __init__(self, stream):
        # Declare instance variables
        self.stream = stream

    def var2(self):
        # Declare instance variables
        self.stream2 = 'Arts'

if __name__ == '__main__':
    s1 = Student('Science')
    print(s1.__dict__)
    # {'stream': 'Science'}
    s1.var2()
    print(s1.__dict__)
    # {'stream': 'Science', 'stream2': 'Arts'}

    # Declare instance variables
    s1.stream3 = 'Commerce'
    print(s1.__dict__)
    # {'stream': 'Science', 'stream2': 'Arts',
    # 'stream3': 'Commerce'}
```

2.2 Static Variables (Class Level Variables)

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.

Access static variables by using `class name or cls variable or object reference`. But `object reference not recommended`.

Declare static variable anywhere using `class name or cls variable`

```
class Student:
    # class variable
    stream = "Science"

    def change_class_var1(self):
        Student.stream = 'Commerce'

    # Update cls var
    @classmethod
    def change_class_var2(cls):
        cls.stream = 'Arts'


if __name__ == '__main__':
    s1 = Student()
    print(s1.stream)  # Science

    s1.change_class_var1()
    print(Student.stream)  # Commerce

    s1.change_class_var2()
    print(Student.stream)  # Arts

    # Update cls var
    Student.stream = 'Arts2'
    print(s1.stream)  # Arts2

    print(s1.__dict__)
    # {}

    # Don't use obj to update cls var
    # It creates ins var
    s1.stream = 'Commerce'
    print(s1.__dict__)
    # {'stream': 'Commerce'}
```

2.3 Local variables (Method Level Variables)

Sometimes to meet `temporary requirements` of programmer, we can declare `variables inside a method directly`, such type of variables are called local variable or temporary variables.

Local variables will be created at the time of method execution and destroyed once method completes.

Local variables of a method `cannot be accessed from outside of method.`

```
class Geek:
    cls_var = "Class var"

    # print(in_var) # Error
    def access_method(self):
        # in_var: can't be accessed outside this fxn
        in_var = 'inside_method'
        print(in_var)

g1 = Geek()
g1.access_method()
# print(g1.in_var)  # Error
```

2.4 Detail Variables/Attribute example

- Attributes are `looked first in the instance and then in the class`
- To `change class var use class name`
- Changing cls var with obj `- creates ins var`

```python
class Student:
    # class variable
    stream = "class-var"
    stream2 = 'class-var2'

    def data(self):
        # instance variables
        self.stream = 'ins-var'


if __name__ == '__main__':
    s1 = Student()
    s1.data()
    # look first in ins and then in cls
    print(s1.__dict__)
    # {'stream': 'ins-var'}

    # creates ins var
    s1.stream2 = 'ins-var2'
    print(s1.__dict__)
    # {'stream': 'ins-var', 'stream2': 'ins-var2'}

    # Change Class var  --> Need to use class name
    Student.stream = "m-cls-var"
    print(Student.stream2, Student.stream)
    # class-var2 m-cls-var
```

- Example 2

```python
class Employee:
    raise_amount = 1.04
    no_of_emp = 0

    def __init__(self, first, pay):
        self.first = first
        self.pay = pay

        # Don't use self. Use class name.
        # Else ins var will be created
        Employee.no_of_emp += 1

    def apply_raise(self):
        # Used class variable
        # self.pay=int(self.pay*Employee.raise_amount)

        # look first in ins and then in cls
        self.pay = int(self.pay * self.raise_amount)


if __name__ == '__main__':
    emp1 = Employee('Amrit', 5000)
    emp2 = Employee("Prasad", 2000)

    emp1.apply_raise()
    print(emp1.__dict__)
    # {'first': 'Amrit', 'pay': 5200}
    print(Employee.__dict__)
    # {'no_of_emp': 2, 'raise_amount': 1.04, ...}

    # look first in ins and then in cls
    print(emp1.raise_amount, emp2.raise_amount)
    # 1.04 1.04

    # Changing Class Value
    Employee.raise_amount = 2.1

    # look first in ins and then in cls
    print(Employee.raise_amount, emp1.raise_amount, emp2.raise_amount
    # 2.1 2.1 2.1

    # Changing Instance Value
    # creates a new instance var 'raise_amount'
    emp1.raise_amount = 2.5
    print(Employee.raise_amount, emp1.raise_amount, emp2.raise_amount
    # 2.1 2.5 2.1

    # Instance variable can't be accessed using Class name
    # print(Employee.first)   # Error
```

.

.

.

.

.

.

.

.

.

.

.

.

.

.