[Hindi] Instance and Class Variables | Object Oriented Programming Using Python Tutorial #3

Instance and Class Variables

In the previous article, we have discussed Classes, objects, and constructors. In this article, we are going to study Instance and class variables. Take a look at the code given below:

```
class Employee:
    def __init__(self,fname,lname,salary):
        self.fname=fname
        self.lname=lname
        self.salary=salary

# Initializing the object
harry = Employee("harry","jackson",4400)
rohan = Employee("rohan","das",4400)


print(harry.fname,rohan.fname)
```

- Here rohan and harry are the object of class Employee with attributes such as fname,lname, and salary.
- These mentioned attributes are associated with the particular object and may differ in their properties.
- But, there are many attributes which are common for all the object of the class—Eg. salary increment, total holidays, total annual race, etc.

To deal with this situation, a **class variable** is taken into consideration.

Class variable :

Suppose we want to increase the salary of all the Employees of the class.

1. We will create a method which will increase the salary of all the Employees.
2. Let the method name be *increase ()*.

```
def increase(self):
    pass
```

3. Let *increment* be the name of the variable which is responsible for the change in salary.

```
increment = 1.5
```

4. Why Self?

   Here, You might be thinking about why the *self* is used, or the self is passed as a parameter to the function: Just remember, the Self is the object. In class, everything is associated with the instance variable, i.e., Object. Here, self represents the instance of the Class. Take a look at the example given below to get a better understanding:

```
harry.increase()
```

   In the above code, the increase() function is called on the harry object. Note that we have not passed any arguments in the method call, but by default, self is passed, and self means the object itself, i.e., harry. Now, let's what happens if we do not pass *self* as an argument to the function.

```
def increase():
    pass

harry.increase()
```

The above code will prompt the following error:

```
TypeError: increase() takes 0 positional arguments but 1 was given
```

When we call any method inside a class, self is passed as the default argument. But, in the increase() function, we have not accepted *self* as an argument. Function increase() takes 0 argument means we have not provided any arguments, but while function call, self is passed by default. Now, let's get back to the increase() function, which will help us to increase the salary of the employees. There are 2 ways to work with the increase method and increment variable.

a. With the help of instance:

```
self.salary = self.salary * self.increment
```

b. With class :

```
def __init__(self,fname,lname,salary):
    self.fname=fname
    self.lname=lname
    self.salary=salary
    self.incriment=1.4 #instance variable

self.salary = self.salary * Employee.increment
```

Variable Self.salary will not search for increment in the instance and will directly use the increment of Employee class.
Now, let's create one more class variable, which will increase the total number of employees by 1 when a new employee is added to the Employee class, or a new object is created.

```
class Employee:

    increment = 1.5
    no_of_employe= 0 #Initilizing the varaible with 0
    def __init__(self,fname,lname,salary):
        self.fname=fname
```

```
        self.lname=lname
        self.salary=salary
        self.increment=1.4
        Employee.no_of_employe +=1 #incrementing values

    def increase(self):
        self.salary = int(self.salary * Employee.increment)
```

Object creation :

Copy

```
print(Employee.no_of_employee)  #Till now, no object is created, so 0 will be printed
harry = Employee("harry","jackson",4400) #one object created, value of no_of_employee will be incremented by 1.
print(Employee.no_of_employe)  #1 will be printed
rohan = Employee("rohan","das",4400) #second object created
print(Employee.no_of_employe)  #2 will be printed
```

Output :

```
0
1
2
```

So, this is all for this tutorial. I hope you've enjoyed the tutorial, and all the concepts discussed in the article are crystal clear to you. In the next article, we will study class methods. Till then, keep coding!

Previous Next

CodeWithHarry