[Hindi] Inheritance In Python oops | Object Oriented Programming In Python Tutorial #7

Inheritance In Python oops

In the previous article, we have discussed classes, instance variables, and methods. In this article, we are going to study Inheritance. Inheritance is considered one of the four pillars of object-oriented programming. So, let us, deep-dive into this concept.

What is Inheritance?

- Inheritance allows us to define a class that **inherits** all the methods and properties from another class. The parent class is the class being **inherited** from, also called the base class. The child class is the class that **inherits** from another class, also called the **derived class**.
- The question is, why we want to inherit a class? Why can't we simply copy and paste the same code for a different class? The straightforward answer to this question is that we do not want to copy and paste the code because it violates the first rule of programming, i.e., Do Not Repeat Yourself.
- The concept of inheritance was introduced to deal with the scenarios where we want to use the methods and variables of a class in some other class.
- Inheritance increases the **reusability** of a code so that we don’t have to write the same code again and again.

In the previous tutorial, we created a class Employee with class methods, static methods, and instance variables. Below is the code for the same:

```
class Employee:
    increment = 1.5
    no_of_employe= 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

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

    @classmethod
    def change_increment(cls,amount):
        cls.increment = amount

#shubham =Employee("shubham", "jackson", "88000")
#rohan = Employee("rohan","das",4400)
```

To understand the concept of inheritance, we will create a new class named Programmer that will inherit the Employee class. So, let's get started.

Creating a new class:

```
class Programmer(Employee):
    pass
```

a. Here, the class Programmer will *inherit* all the methods and functions of class Employee as we have passed Employee as the **parameter** in **Programmer class.**
b. Let us create an object of class Programmer:

```
harry = Programmer("harry","jackson",99000)
```

Printing the output:

```
print(harry.fname)
print(harry.lname)
print(harry.salary)
```

Output:

```
harry
jackson
99000
```

Here, the **Programmer** class has inherited/copied all the methods, functions, and working of a class **Employee.** Like any other class, we can also create constructors, methods, and attributes for the **Programmer** class. So, let's do it.

Creating a constructor:

```
def __init__(self,lname,fname,salary,proglang,exp):
        super.__init__(fname, lname, salary)
        self.proglang = proglang
        self.exp = exp
```

- In the above code, we've used super.__init__ to call the __init__ of the super class.
- We've also created two instance variables named **prolang** and **exp** for the **Programmer** class.

Creating an object of the programmer class:

```
abhishek = Programmer("abhishek","Pathak",10000,"python","1 yrs")
```

In the above code:

- fname = “abhishek”
- lname = “Pathak”
- Salary = 10000
- proglang = “python”
- exp = “1 yrs”

Getting the output:

```
print(abhishek.fname)
print(abhishek.lname)
print(abhishek.salary)
print(abhishek.proglang)
print(abhishek.exp)
```

Output:

```
abhishek
Pathak
10000
python
1 yrs
```

This tutorial ends here, and I will see you in the next tutorial. Till then, keep coding and keep learning.

Previous Next