Full Stack Software Development- Assignment I

Inheritance in Python

Date: 09.11.2023

By-

Aakash LS

2020506001

Like any other OOP languages, Python also supports the concept of class inheritance.

Inheritance allows us to create a new class from an existing class.

The new class that is created is known as subclass (child or derived class) and the existing class from which the child class is derived is known as superclass (parent or base class).

```
# base class definition
class Vehicle:
   def disp():
      pass
# derived class definition
class Bus(Vehicle):
   pass
```

Code:

Example:

```
class Aakash:


   # attribute and method of the parent class
   name = ""
```

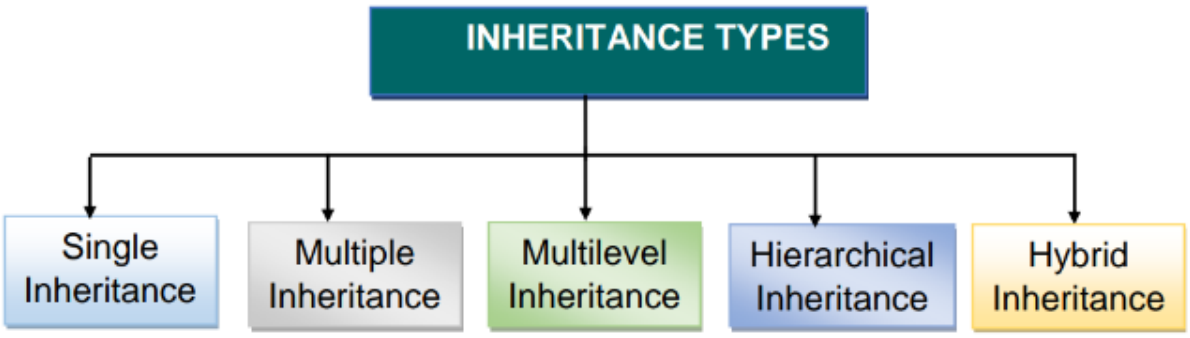
```
    def intro(self):
        print("Hey I am Aakash LS! This is Full stack ")


class Ls(Aakash):

    # new method in subclass
    def display(self):
        # access name attribute of superclass using self
        print("My name is ", self.name)

obj = Ls()

# access superclass attribute and method
obj.name = "Aakash LS"
obj.intro()

# call subclass method
obj.display()
```

Output:

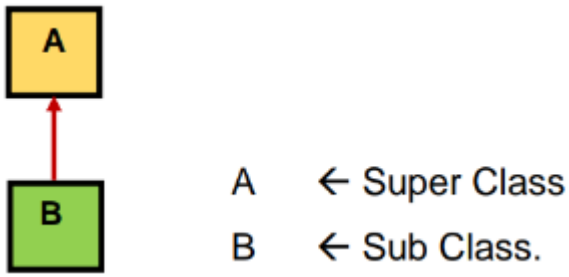
Hey I am Aakash LS! This is Full stack

My Name is Aakash LS

Inheritance Types in Python:



1. **Single Inheritance** Single inheritance allows a derivate class to inherit properties of one parent class, and this allows code reuse and the introduction of additional features in existing code.



Code:

```
# Here, we will create the base class or the Parent class
class Parent1:
    def func_1(self):
        print ("This function is defined inside the parent class.")

# now, we will create the Derived class
class Child1(Parent1):
```

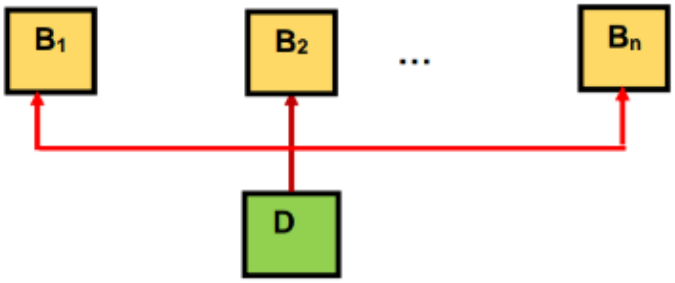
```
    def func_2(self):
        print ("This function is defined inside the child class.")


# Driver's code
object = Child1()
object.func_1()
object.func_2()
```

Output:

```
This function is defined inside the parent class.
This function is defined inside the child class.
```

2. MULTIPLE INHERITANCE

- Creating a new class (sub class) from more than one super class is called as multiple inheritance.
- Like 1-many mapping.



Code:

```
# base class 1
class Person:
        def m1(self):
                self.pname="Rohit"
                self.pid=14
        def p1(self):
                print("Name\t\t: ",self.pname)
                print("Id\t\t: ",self.pid)
# base class 2
class Personal:
        def m2(self):
                self.pack=55000.75
                self.loc="Mumbai"
        def p2(self):
        print("Package\t\t: ",self.pack)
        print("Location\t: ", self.loc)
# derived class creation
class D(Person,Personal):
        def m3(self):
                self.cmp="Google"
                print("Company\t\t: ",self.cmp)
# object creation for derived class
obj=D()


print("\t Multiple Inheritance")
```

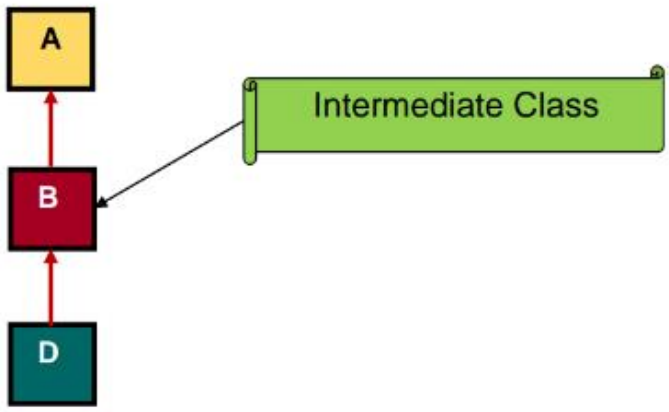
```
obj.m1()
obj.p1()
obj.m2()
obj.p2()
obj.m3()
```

Output:

```
----------------------------------------
        Multiple Inheritance
----------------------------------------
Name            :  Rohit
Id              :  14
Package         :  55000.75
Location        :  Mumbai
Company         :  Google
PS C:\Users\Ganesh\Documents\VSC Python Projects\inheritance>
```

3. MULTILEVEL INHERITANCE

- Creating a new class (sub class) from another sub class or derived class (already inherited class) is called as multilevel inheritance.
- Intermediate class
  - If a class acts as **base class on one side** and acts as **derived class on another side** is called as intermediate class.



```
class Grandfather1:


    def __init__(self, grandfathername1):
        self.grandfathername1 = grandfathername1


# here, we will create the Intermediate class
class Father1(Grandfather1):
    def __init__(self, fathername1, grandfathername1):
        self.fathername1 = fathername1


        # here, we will invoke the constructor of Grandfather class
        Grandfather1.__init__(self, grandfathername1)


# here, we will create the Derived class
class Son1(Father1):
    def __init__(self,sonname1, fathername1, grandfathername1):
        self.sonname1 = sonname1


        # here, we will invoke the constructor of Father class
        Father1.__init__(self, fathername1, grandfathername1)


    def print_name(self):
        print('Grandfather name is :', self.grandfathername1)
        print("Father name is :", self.fathername1)
        print("Son name is :", self.sonname1)
```

```
# Driver code
s1 = Son1('John', 'John Jr', 'John Jr Jr')
print (s1.grandfathername1)
s1.print_name()
```

Output:

John Jr Jr

Grandfather name is : John Jr Jr

Father name is : John Jr

Son name is : John

4. HIERARCHICAL INHERITANCE

- Creating several new classes (sub classes) from **ONLY ONE SUPER CLASS** is called as hierarchical inheritance.
- Like many-1 mapping.

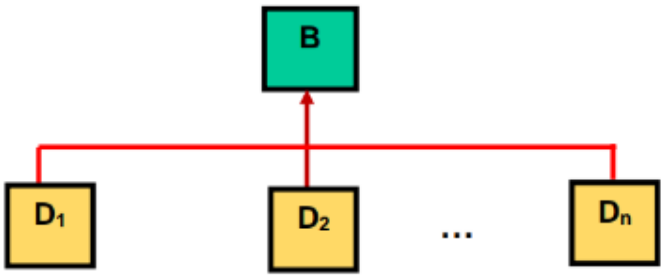


Figure 1.4 Hierarchical Inheritance

Where,

- **D_1, D_2, ...D_n** are list of sub classes
- **B** is a super class.

```
class Parent1:
    def func_1(self):
        print ("This function is defined inside the parent class.")

# Derived class1
class Child_1(Parent1):
    def func_2(self):
        print ("This function is defined inside the child 1.")

# Derivied class2
class Child_2(Parent1):
    def func_3(self):
        print ("This function is defined inside the child 2.")

# Driver's code
object1 = Child_1()
object2 = Child_2()
object1.func_1()
object1.func_2()
object2.func_1()
object2.func_3()
```

Output:

This function is defined inside the parent class.

This function is defined inside the child 1.

This function is defined inside the parent class.

This function is defined inside the child 2.

5. HYBRID INHERITANCE

- Process of creating new class involving **atleast one form of inheritance** (single or multiple or multilevel or hierarchical) and **one or more number of super classes** is called as hybrid inheritance.

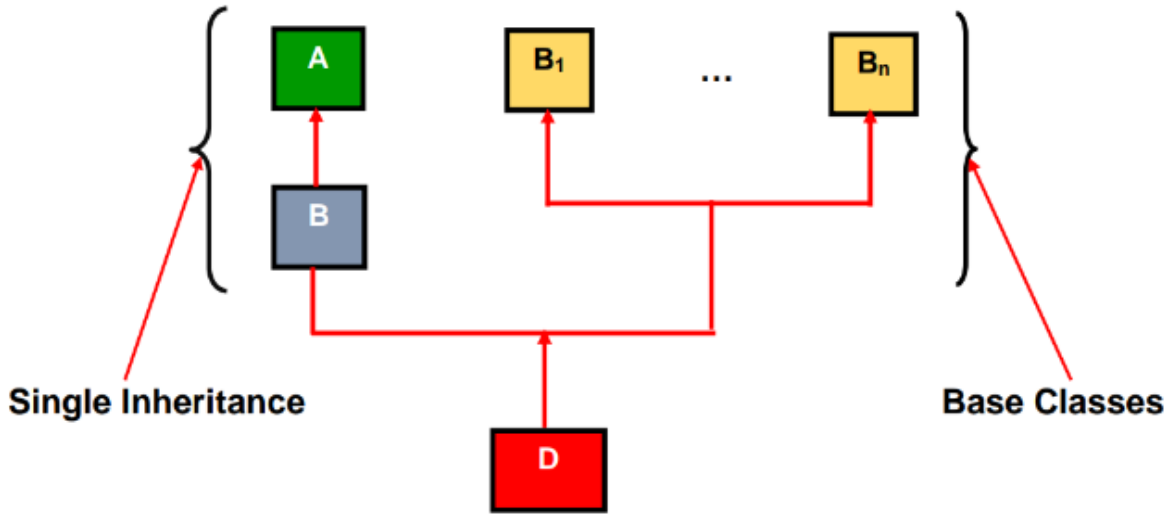


Figure 1.5 Hybrid Inheritance

Where,

- **A→ B is a single inheritance**
- **B_1, … B_n are super classes**
- **D is a sub class.**

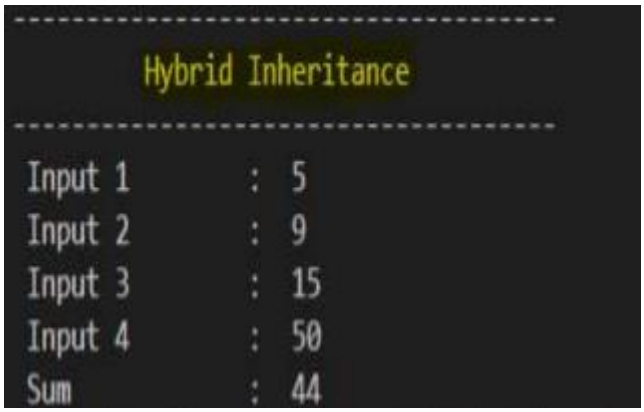
Code:

```
class B(A):
        def f2(self):
        self.j=9
# super class 2
class C:
```

```
        def f3(self):
        self.k=15
# super class 3
class D:
        def f4(self):
        self.z=50
# new derived class
class HI(B,C,D):
        def sum4(sel):
        sel.rs=sel.i+sel.j+sel.k+sel.k
        print(" Input 1 \t: ",sel.i)
        print(" Input 2 \t: ",sel.j)
        print(" Input 3 \t: ",sel.k)
        print(" Input 4 \t: ",sel.z)
        print(" Sum \t\t: ",sel.rs)
# derived class object creation
obj=HI()
print("-----------------------------------")
print("\t Hybrid Inheritance")
print("-----------------------------------")
# call base class methods and derived class methods using derived
class object
obj.f1()
obj.f2()
obj.f3()
obj.f4()
```

```
obj.sum4()
```

Output:



Thus concept of inheritance, it's types along with the code have been written.