Chapter six: Inheritance in python

6.1 Chapter objectives

By the end of this chapter, learners should be exhibit thorough understanding on how python implements the concept of inheritance

6.2 Introduction

One of the most important concepts in object-oriented programming is that of inheritance. It's a concept of OOP which allows an object to derive properties and characteristics (data and methods) from another object (class). In other terms, we can define inheritance as ability of a child class to acquire the properties (the data members) and functionality (the member methods) from parent class. A child class can also provide its specific implementation to the functions of the parent class.

Through the concept, we can achieve hierarchical classification. At the end, the programmer ends up building a hierarchy of classes, moving from the most general one to the most specific one.

6.3 Keys terms in inheritance

- **Child class**

  This is a class that inherits properties and characteristics from another class. Its also known as derived class or subclass.

- **parent class**

  This refers to a class other classes acquire properties from. It's also known as super class or base class.

6.4 Advantages of inheritance

The main advantages of inheritance are code reusability and readability. When child class inherits the properties and functionality of parent class, we need not to write the same code again in child class. This makes it easier to reuse the code, makes us write the less code and the code becomes much more readable. It also provides an opportunity to fast track implementation time.

6.5 Examples of inheritance in real life

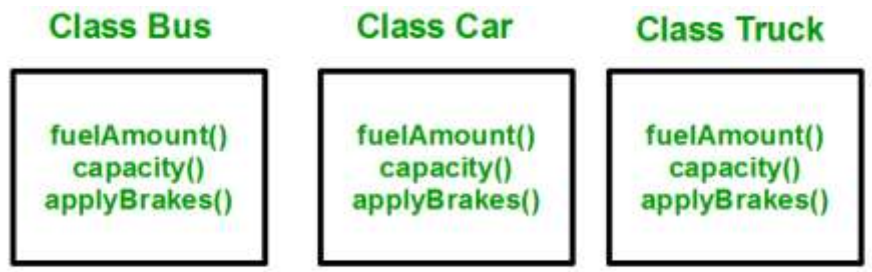
# example 1

Let's assuming we have a class known as Human with *height, weight, colour* etc as properties and *eating (), sleeping (), dreaming (), working()* e.t.c as functionality.

Now we want to create Male and Female class, these classes are different but since both Male and Female are humans, they share some common properties and behaviors (functionality) so they can inherit those properties and functionality from Human class and rest can be written in their class separately.

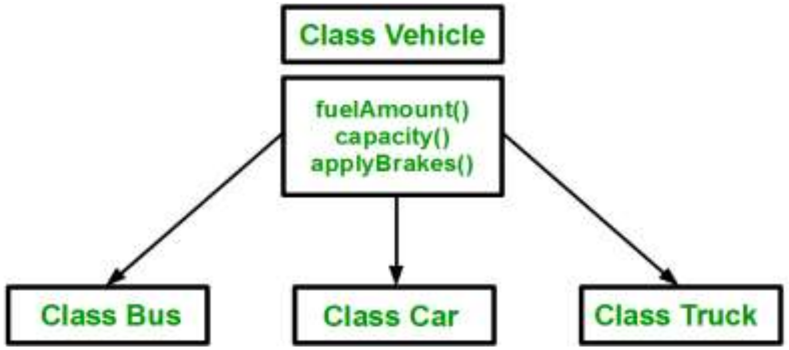
# example2

We consider a group of vehicles; Bus, Car and Truck. The methods/operations *fuelAmount (), capacity(), applyBrakes()* will be same for all of the three classes. If we create these classes avoiding inheritance then we have to write all of these functions in each of the three classes as shown in below figure:



Its vivid clear that the above process results in duplication of same code 3 times increasing chances of error and data redundancy. To avoid this type of situation, we apply inheritance by creating a parent class known as Vehicle which contains these three functions in it.

The other three class will inherit from the vehicle class avoiding duplication of data as well as increasing code reusability. We now model the new class as shown below

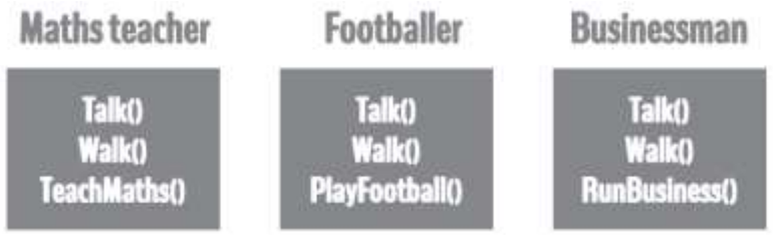


Employing inheritance enables us to write the functions only one time instead of three times as we have inherited rest of the three classes from base class (Vehicle).

# example 3

Let's assume we are working in solution consisting of three characters;maths teacher, footballer and a businessman.

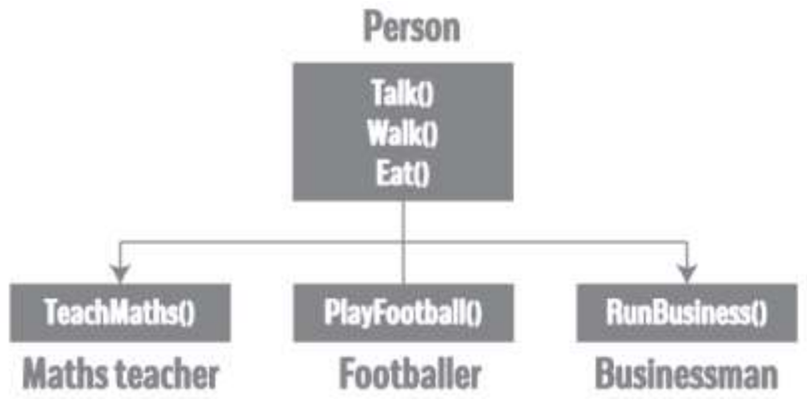
Since, all of the characters are persons, they can walk and talk. However, they also have some special skills. A maths teacher can teach maths, a footballer can play football and a businessman can run a business. We individually create three classes who can walk, talk and perform their special skill as shown in the figure below.



For three classes, we need the same code for walk and talk operations.

If we wish to add a new feature e.g., eat, the same code has to be implemented for each class making the program error prone as well as duplicating codes.

To simplify the program, we can introduce a super class known as person which captures the common characteristics like talk, walk, eat, sleep but add special skills to characteristics to the respective class as shown below.



Employing inheritance, we can have the same code. for walk and talk for each class. You just need to inherit them.

For example, the Maths teacher (derived class), we inherit all features of a Person (base class) and add a new feature TeachMaths. The same applies for footballer and businessman class.

The overall effect is having clean code which is understandable and extendable.

6.6 Implementing inheritance in python

We begin by creating the base class which is created just like normal methods in python

To create a derived class in python which is inherited from the base class we have to follow the below syntax

Example: We create a class named Person with name, height, width and gender as properties, and a view Person as method:

```
class Person:
    def __init__(self,name,w,h,g):
        self.name=name
        self.weight=w
        self.height=h
        self.gender=g
    def viewPerson(self):
        print(self.name,self.weight,self.height,self.gender)
p=Person("Joe","62.5 Kgs","6 FT","Male")
p.viewPerson()
```

6.6.1 Create a Child Class

To create a class that inherits the functionality from another class, send the parent class as a parameter when creating the child class. The following general syntax is used

```
class derived-class-name (base class):
    class body
```

If a class inherits multiple classes, we mention each of them by placing them inside the bracket. The following syntax is applied Consider the following syntax.

Syntax

```
class derive-class(<base class 1>, <base class 2>, ….. <base class n>):
class body
```

Example, to create a class known as student which inherits the above properties, we use the following statement in python

```
class Person:
    def __init__(self, name, w, h, g):
        self.name = name
        self.weight = w
        self.height = h
        self.gender = g
    def viewPerson(self):
        print(self.name, self.weight, self.height, self.gender)
class Student(Person):
    pass
s = Student("Joe", "62.5 Kgs", "6.2 FT", "Male")
s.viewPerson()
```

You will notice class student has acquired properties (attributes and behavior) from class person

6.6.2 Add the __init__() Function

We have now created a child class ***(student)*** that inherits the properties and methods from its parent ***(person).*** We shall now proceed to implement the *__init__()* function to the child class which will replace the pass keyword

Always remember, this function is called automatically every time the class is instantiated

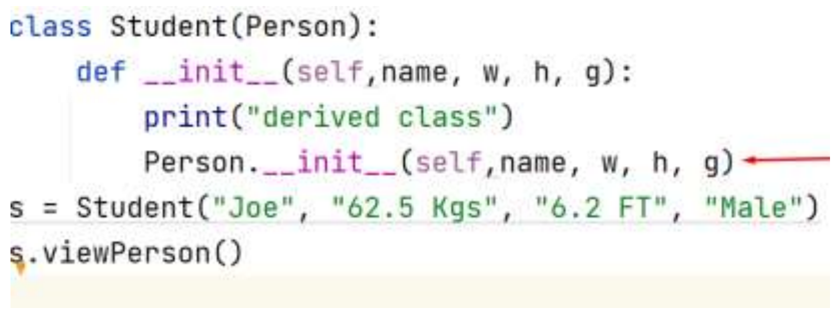
```
class Student(Person):

    def __init__(self,name, w, h, g):

        print("derived class")
```

When you add the __init__() function in the child class, we will no longer inherit the parent's __init__() function.

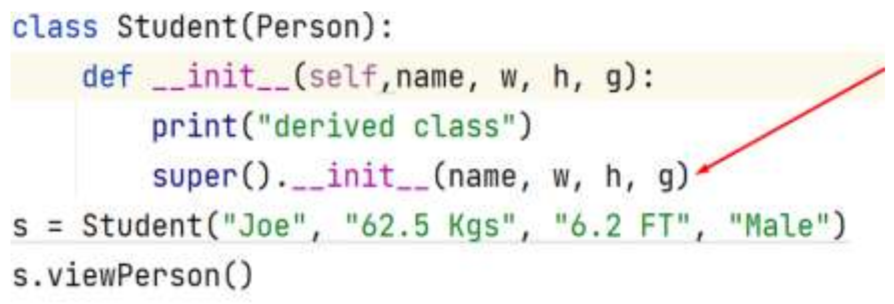
To inherit __init__() function in the base class, we can add the following python code.



```
class Student(Person):
    def __init__(self,name, w, h, g):
        print("derived class")
        Person.__init__(self,name, w, h, g)
s = Student("Joe", "62.5 Kgs", "6.2 FT", "Male")
s.viewPerson()
```

6.6.3 Use the super() Function

Python also has a super () function that will make the child class inherit all the methods and properties from its parent

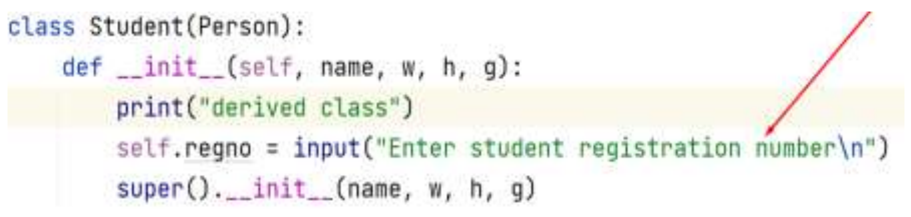


```
class Student(Person):
    def __init__(self,name, w, h, g):
        print("derived class")
        super().__init__(name, w, h, g)
s = Student("Joe", "62.5 Kgs", "6.2 FT", "Male")
s.viewPerson()
```

By using the function, we do not have to use the name of the parent element. All methods and properties shall be automatically inherited from the parent class.

6.6.4 Adding properties to a child class

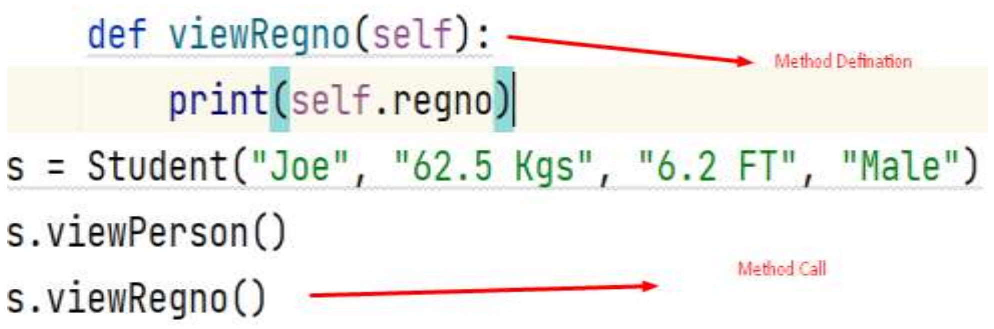
Let's add registration number as a parameter



```
class Student(Person):
    def __init__(self, name, w, h, g):
        print("derived class")
        self.regno = input("Enter student registration number\n")
        super().__init__(name, w, h, g)
```

6.6.5 Adding method to a child class

Let's add new method known as *viewRegno()*



```
    def viewRegno(self):
        print(self.regno)
s = Student("Joe", "62.5 Kgs", "6.2 FT", "Male")
s.viewPerson()
s.viewRegno()
```

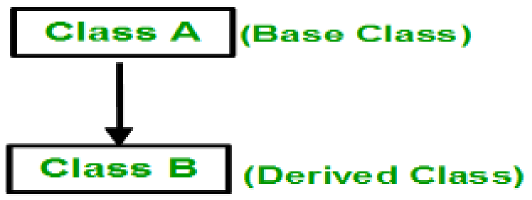
If you add a method in the child class with the same name as a function in the parent class, the inheritance of the parent method will be overridden.

6.7 Types of inheritance Python

Python supports five types of inheritance ***Single inheritance (1), Multiple inheritance (2), Hierarchical inheritance (3), Multilevel inheritance (4) and Hybrid inheritance***

6.8.1 Single inheritance

In this type of inheritance, a class is allowed to inherit from only one class. i.e., one sub class is inherited by one base class only.



The syntax for single inheritance is as follows

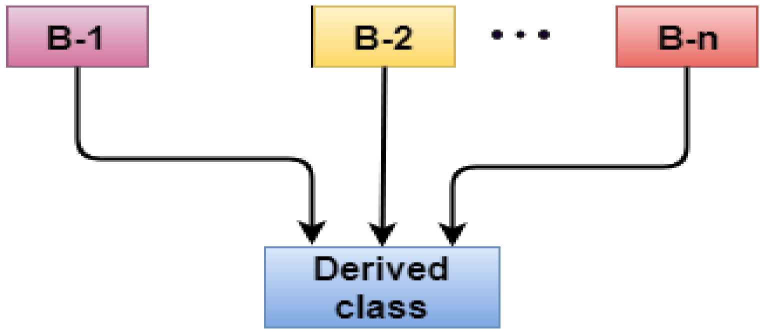
```
class subclass_name(base_class):
        Class-body
```

The following python program demonstrate the concept of single inheritance

```
class person:
    def __init__(self):
        self.name=input("Enter the person name\n")
class staff(person):
    def __init__(self):
        self.staffid=input("Enter staff ID\n")
        super().__init__()
    def printStaff(self):
        print("Your name is\t"+ self.name+ "and staff
Id\t" +self.staffid)
s=staff()
s.printStaff()
```

6.8.2 Multiple Inheritance

Multiple Inheritance is a feature of python where a class can inherit from more than one classes. i.e. one sub class inherits from more than one base class.



The Syntax is as follows

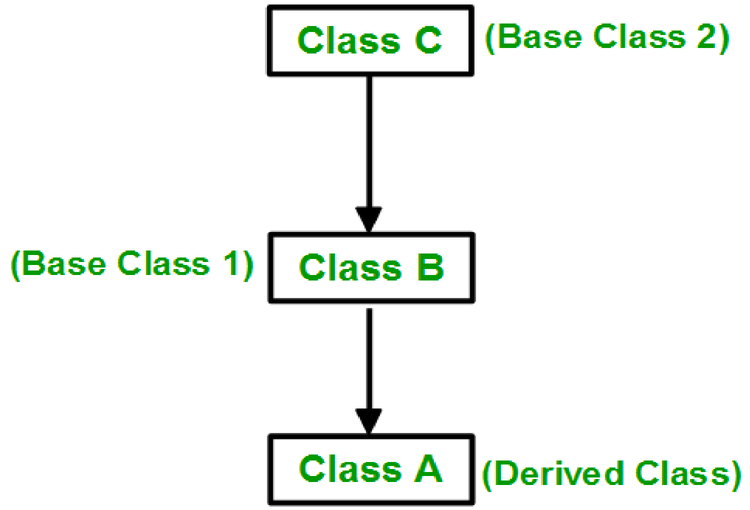
```
class subclass_name(base_class1, base_class2…base classn):
      Class-body
```

The number of base classes will be separated by a comma .A simple program that demonstrates multilevel inheritance is given below

```
class person:
   def __init__(self):
      self.name=input("Enter the person name\n")
class Department:
   def __init__(self):
      self.depname=input("Enter staff department\n")
class staff(person,Department):
   def __init__(self):
      self.staffno=input("Enter staff number\n")
      person.__init__(self)
      Department.__init__(self)
   def viewStaff(self):
      print(self.staffno,self.depname,self.name)
s=staff()
s.viewStaff()
```

6.8.3 Multilevel inheritance

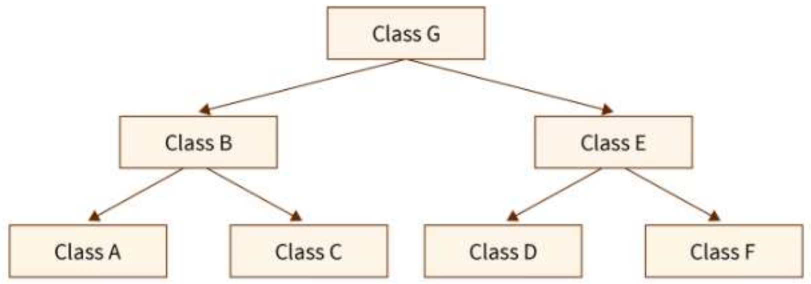
In this type of inheritance, a derived class is created from another derived class.



In the above example, class A inherits members of class B which itself is a derived class (derived from base class C).

6.8.4 Hierarchical Inheritance

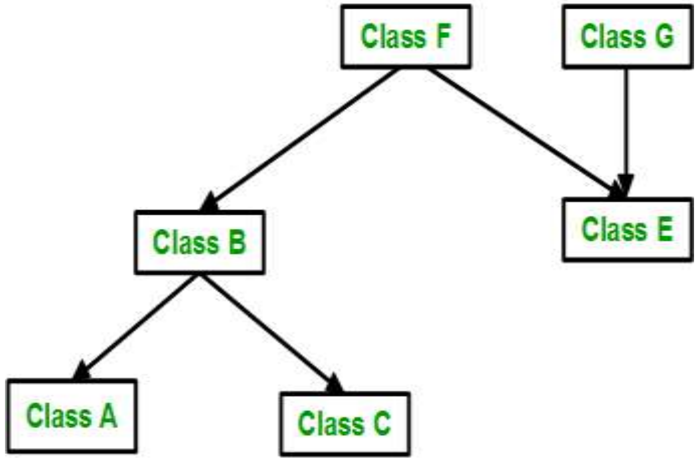
In hierarchical inheritance, we have one base class and more than one child class. These child classes may or may not have multiple child classes. All the child classes will acquire/inherit the properties of the base class. This forms a tree-like hierarchical structure as shown in the figure below



6.8.5 Hybrid (Virtual) Inheritance:

This type of inheritance is implemented by combining more than one type of inheritance e.g. Combining multilevel inheritance and Multiple Inheritance.

Below image shows the combination of hierarchical and multiple inheritance:



6.8.6 Additional programs

Consider the following example of a program that exhibits multiple inheritance. The program has two base classes i.e. person and grading. The properties of the two classes are inherited by a third class know as student.

6.8 Chapter summary

6.9 Further reading suggestions