INHERITANCE

CSC542B

Dr. M. Prabu

Dr. M. Umme Salma

CONTENTS

- ➢ **Class**

- ➢ **Object**

- ➢ **Inheritance**

WHAT ARE CLASSES AND OBJECTS IN PYTHON?

Python is an object oriented programming language. Unlike procedure oriented programming, where the main emphasis is on functions, object oriented programming stress on objects.

Object is simply a collection of data (variables) and methods (functions) that act on those data. And, class is a blueprint for the object.

We can think of class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows etc. Based on these descriptions we build the house. House is the object.

As, many houses can be made from a description, we can create many objects from a class. An object is also called an instance of a class and the process of creating this object is called instantiation.

DEFINING A CLASS IN PYTHON

Like function definitions begin with the keyword <u>def</u>, in Python, we define a class using the keyword <u>class</u>.

Here is a simple class definition.

```
class MyNewClass:
    '''This is a docstring. I have created a new class'''
    pass
```

CREATING AN OBJECT IN PYTHON

We saw that the class object could be used to access different attributes.

It can also be used to create new object instances (instantiation) of that class. The procedure to create an object is similar to a function call.

```
>>> ob = MyClass()
```

File   Edit   Format   Run   Options   Window   Help

```python
class names:
    def basic(self):
        print('This is a class')


#class method nameofobject=nameofclass()
n=names()
n.basic()
```

File   Edit   Shell   Debug   Options   Window   Help

```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 17:00:18) [MSC v.1900 64 bit (AN
4)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: C:/Users/win10/Desktop/program/class1.py ==============
This is a class
>>> 
```

```
File   Edit   Format   Run   Options   Window   Help

class names:
    def basic(self):
        print('This is a class')
    def add(self,a):
        add=a+10
        print(add)


#class method nameofobject=nameofclass()
n=names()
n.basic()
n.add(10)
```

```
File   Edit   Shell   Debug   Options   Window   Help

Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018,
4)] on win32
Type "copyright", "credits" or "license()" fo
>>>
============== RESTART: C:/Users/win10/Desktop
This is a class
20
>>>
```

WHAT IS INHERITANCE?

Inheritance is a powerful feature in object oriented programming.

It refers to defining a new <u>class</u> with little or no modification to an existing class. The new class is called derived (or child) class and the one from which it inherits is called the base (or parent) class.
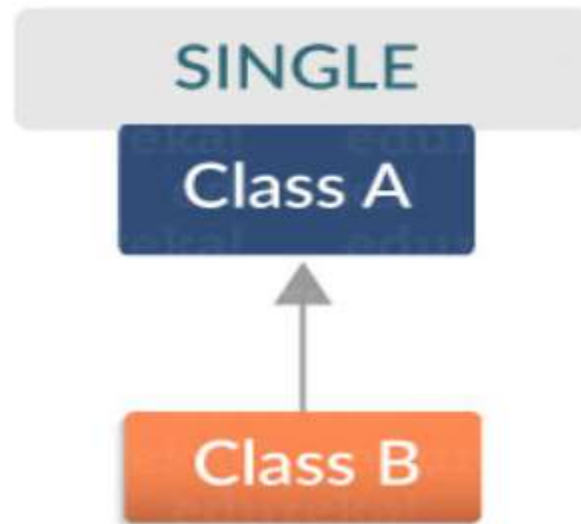
PYTHON INHERITANCE SYNTAX

```
class BaseClass:
    Body of base class
class DerivedClass(BaseClass):
    Body of derived class
```

TYPES OF INHERITANCE

➢ Single Inheritance

➢ Multiple Inheritance

➢ Multilevel Inheritance

➢ Hybrid Inheritance

➢ Hierarchical Inheritance
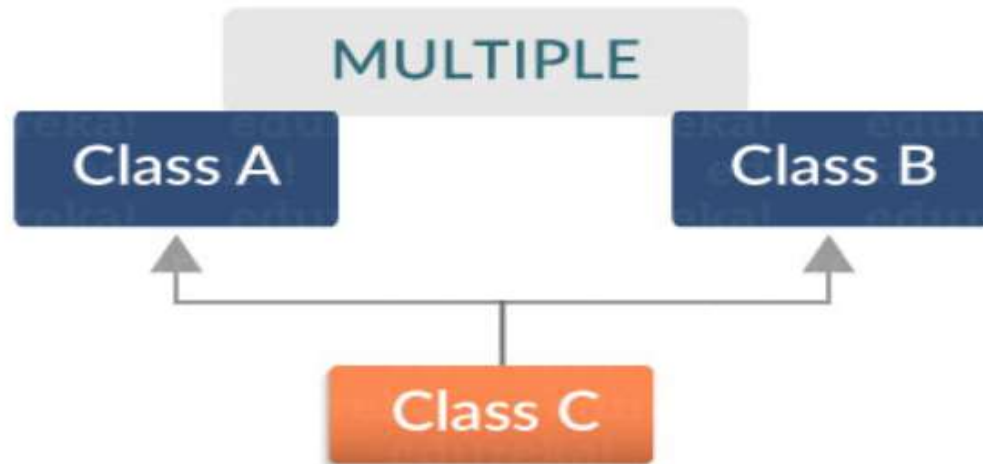
SINGLE INHERITANCE

➢ **In which there is one base class and one derived class**

MULTIPLE INHERITANCE

➢ Multiple inheritance is possible in python

➢ A class can be derived from more then one base classes. The syntax for multiple inheritance is similar to single inheritance

➢ Here is an example of multiple inheritance

```
class first():
    def sum1(self,a,b):
        c=a+b;
        return c;
class second():
    def sub1(self,x,y):
        z=x-y;
        return z;
class third(first,second):
    pass
obj1=third()
print(obj1.sum1(20,25))
print(obj1.sub1(50,25))
```

```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 201
4)] on win32
Type "copyright", "credits" or "license()"
>>>
============ RESTART: C:/Users/win10/Deskto
45
25
>>>
```
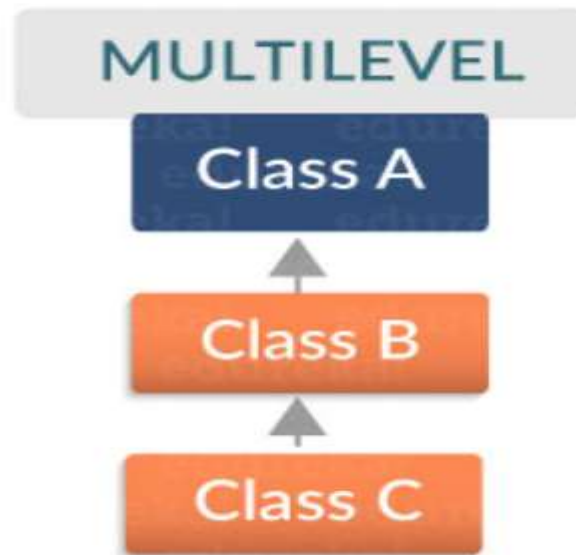
MULTILEVEL INHERITANCE

➤ Multilevel inheritance is also possible in Python like other Object Oriented programming languages. We can inherit a derived class from another derived class, this process is known as multilevel inheritance. In Python, multilevel inheritance can be done at any depth.
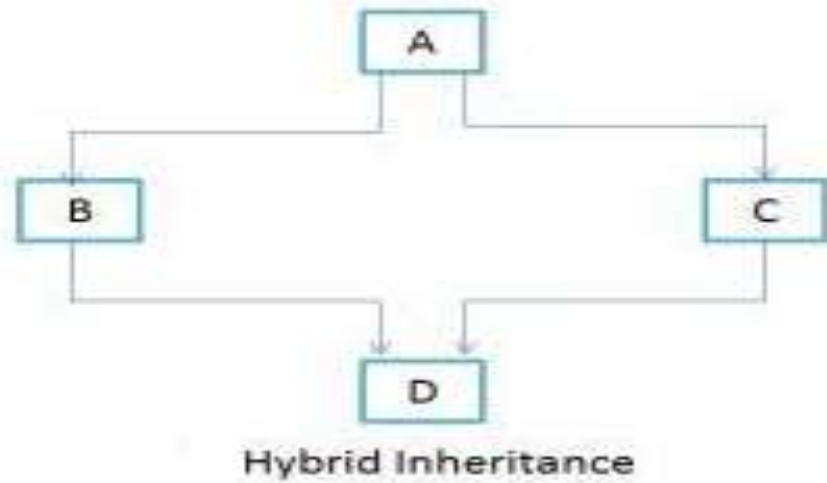
```
class first():
    def m1(self,a,b):
        c=a+b;
        return c;
class second(first):
    def m2(self):
        print('M2 method is called')
class third(second):
    def m3(self):
        print('M3 method is called')


thirdobj=third()
print(thirdobj.m1(20,25))
```

```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28
4)] on win32
Type "copyright", "credits" or "license
>>>
============== RESTART: C:/Users/win10/D
45
>>> |
```

HYBRID INHERITANCE

Hybrid Inheritance

```python
class a:
    def m(self):
        print("m() from class a...")


class b(a):
    def m(self):
        print("m() from class b...")


class c(a):
    def m(self):
        print("m() from class b...")
class d(b, c):
    def m(self):
        print("m() from class d...")
        b.m(self)
        c.m(self)
        a.m(self)
def main():
    obj1=d()
    obj1.m()


if __name__=="__main__":
    main()
```

```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28
4)] on win32
Type "copyright", "credits" or "license
>>>
============= RESTART: C:/Users/win10/De
m() from class d...
m() from class b...
m() from class b...
m() from class a...
```
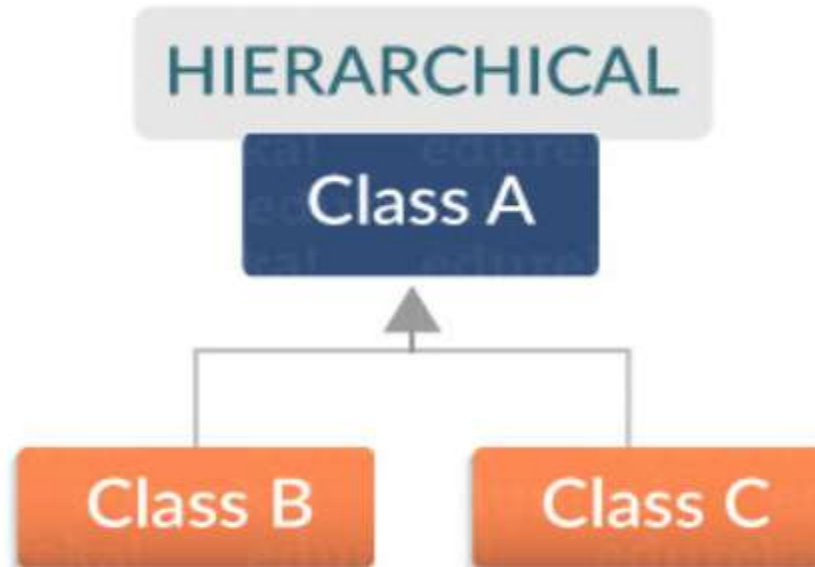
HIERARCHICAL INHERITANCE

➢ In which there is single base class and multiple derived class

```python
class animal():
    def move(self):
        print("I move therefore i am..")
class human(animal):
    def move(self):
        print("humans can walk and run..")
class fish(animal):
    def move(self):
        print("fishes can swim and dive..")


def main():
    hi=human()
    hi.move()
    fi=fish()
    fi.move()
if __name__=="__main__":
    main()
```

```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018,
4)] on win32
Type "copyright", "credits" or "license()" for
>>>
============== RESTART: C:/Users/win10/Desktop/
humans can walk and run..
fishes can swim and dive..
```