Module 4:
Inheritance

By Ninad Gaikwad

Reference - “Core Python Programming”
Dr. R. Nageshwara Rao
Dreamtech Press

14.1 Inheritance

14.1 Inheritance:

- Creating new class from existing class is called inheritance
- All the members of the base class are also the members of the subclass
- All classes inherit from base class “object”
- Syntax for Inheritance - class Subclass(Baseclass):
- Eg.

```
class Teacher:
        # Base class - save as teacher.py
        def setid(self, id):
                self.id = id

        def getid(self):
                return self.id

        def setname(self, name):
                self.name = name

        def getname(self):
                return self.name
```

14.1 Inheritance

```
# Derived Class - Save as Student.py
class Student(Teacher):
        def setmarks(self, marks):
                self.marks = marks

        def getmarks(self):
                return self.marks
```

14.2 Overriding super class constructors and methods:

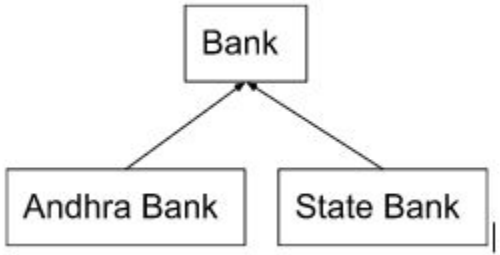
- Use super method to override the values of super class
- Eg

```
# accessing base class constructor in subclass
class Father:
        def __init__(self,property = 0):
                self.property = property

        def display_property(self):
                print("Fathers property is ", self.property)
class Son(Father):
        def __init__(self,property1 = 0, property=0 ):
                super().__init__(property)
                self.property1 = property1

        def display_property(self):
                print("Sons property is ", self.property1+self.property)
# Create sub class instance
s= Son(20000, 80000)
s.display_property()  # will give O/P 1,00,000
```

14.3 Types of Inheritance: Single Inheritance

14.3.1 Single Inheritance

- Deriving one or more classes from single class is called single inheritance



- Eg.

```
# Single Inheritance
class Bank(Object):
        cash = 1000
        @classmethod
        def available_cash(cls):
                print(cls.cash)
```

14.3 Types of Inheritance: Single Inheritance

```
class AndhraBank(Bank):
        pass

class StateBank(Bank):
        cash = 2000
        @classmethod
        def available_cash(cls):
                print(cls.cash+Bank.cash)

a = AndhraBank()
a.available_cash()

s= StateBank()
s.available_cash()

‘’’ Output:
1000
3000 ‘’’
```

14.3 Types of Inheritance: Multiple inheritance

14.3.2 Multiple inheritance

- Deriving base class from multiple base classes is called as multiple inheritance
- There are more than one super classes
- Eg

```
class Father:
        def height(self):
                print(“Height is 6 feet”)
class Mother:
        def color(self):
                print(“color is brown”)
class child(Father, Mother):
        pass

c= child()
c.height()
c.color()
“”” Output
        Height is 6 feet
        color is brown  “””
```

14.3 Types of Inheritance: Multiple inheritance

- Second super class constructor is not available to the subclass
- Eg B class constructor will not be available in below example

```
class A(object):
        def __init__(self):
                self.a = ‘a’
                print(self.a)
class B(object):
        def __init__(self):
                self.b = ‘b’
                print(self.b)
class C(A, B):
        def __init__(self):
                self.c = ‘c’
                print(self.c)
                super().__init__()
o=c()
“”” Output
c
a  “””
```

14.3 Types of Inheritance: Multiple inheritance

- To make available constructors of both class use super().__init__() in every class

```
class A(object):
        def __init__(self):
                self.a = ‘a’
                print(self.a)
                super().__init__()
class B(object):
        def __init__(self):
                self.b = ‘b’
                print(self.b)
                super().__init__()
class C(A, B):
        def __init__(self):
                self.c = ‘c’
                print(self.c)
                super().__init__()
o=c()
“”” Output
c
a
b  “””
```