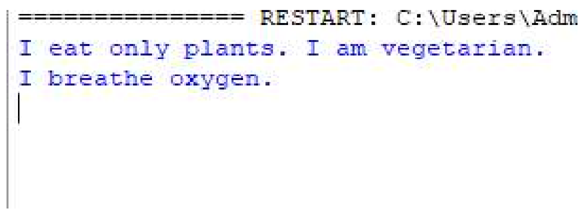
Module 4
OOP

Name : Mandar Awatade **Grade:**
Class : Sy Comps
Roll No :25 **Sign:**
Div : B

Experiment 4A

Aim: Write a python program to implement Method Overriding

Program:
```
# parent class

class Animal:
 # properties
multicellular = True
eukaryotic = True
# function breath

def breathe(self):
 print("I breathe oxygen.")
 # function feed

def feed(self):
 print("I eat food.")

 # child class
 class Herbivorous(Animal):
 # function feed
def feed(self):
 print("I eat only plants. I am vegetarian.")

herbi = Herbivorous()
herbi.feed()
# calling some other function
herbi.breathe()
```
Output:

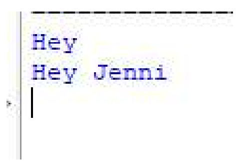

```
=============== RESTART: C:\Users\Adm
I eat only plants. I am vegetarian.
I breathe oxygen.
```

Experiment 4B

Aim: Write a python program to implement Method Overloading

Program:
```
class Student:
 def hello(self, name=None):
        if name is not None:
              print('Hey ' + name)
        else:
             print('Hey ')

# Creating a class instance
std = Student()
# Call the method
std.hello()
# Call the method and pass a parameter
std.hello('Jenni')
```

Output:


```
--------------
Hey
Hey Jenni
```

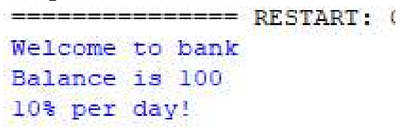
Experiment 4C

Aim: Demonstrate abstract class in python

Program:

```
from abc import ABC, abstractmethod
#Abstract Class
class Bank(ABC):
   def bank_info(self):
        print("Welcome to bank")
@abstractmethod
def interest(self):

    pass
#Sub class/ child class of abstract class
class SBI(Bank):
    def balance(self):
        print("Balance is 100")
    def interest(self):
        print("10% per day!")
s= SBI()
s.bank_info ()
s.balance()
s.interest()
```

Output:



```
=============== RESTART: (
Welcome to bank
Balance is 100
10% per day!
```

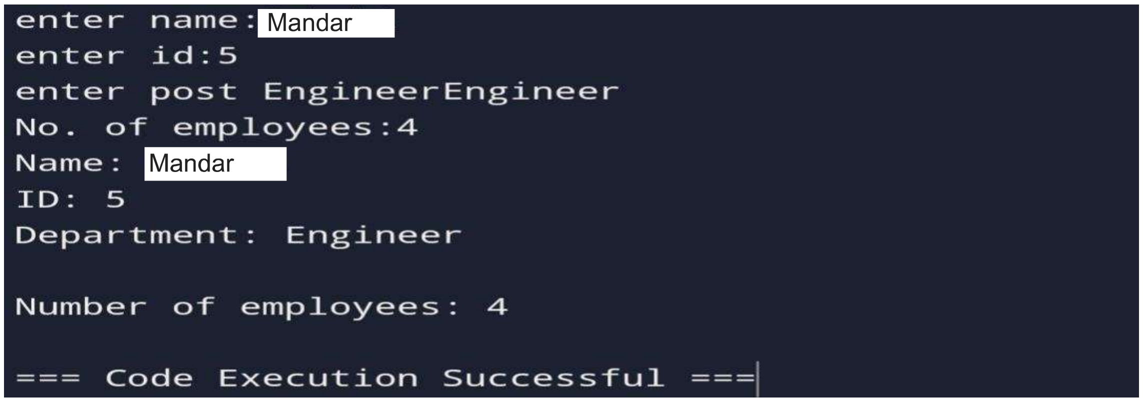
Exercise

1.Create a class Manager which will inherit the Employee class with data members:Post,No_of_Employee. Create suitable methods for reading and printing Manager information(Single Inheritance,use Super method)

Program:

```
class Employee:


    def accept(self):
        self.name=input("enter name:")
        self.id=int(input("enter id:"))
        self.post=input("enter post")
    def display(self):
        print("Name:",self.name,"\nID:",self.id,"\nDepartment:",self.post)
class Manager(Employee):
    def accept(self):
        super().accept()
        self.noemp=int(input("No. of employees:"))
    def display(self):
        super().display()
        print("\nNumber of employees:",self.noemp)
obj=Manager()
obj.accept()
obj.display()
```

Output:



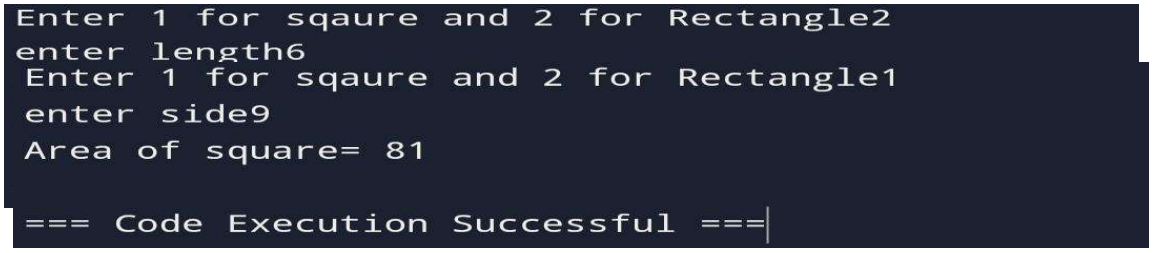
```
enter name: Mandar
enter id:5
enter post EngineerEngineer
No. of employees:4
Name: Mandar
ID: 5
Department: Engineer

Number of employees: 4

=== Code Execution Successful ===
```

2.Write a Python program to create a class to print the area of a square and a rectangle.Use Method overloading to find area.

Program:

```
class Area:
    def calc(self,l,b=None):
        if b is not None:
            print("Area of rectangle =",l*b)
        else:
            print("Area of square=",l*l)
obj=Area()
ch=int(input("Enter 1 for sqaure and 2 for Rectangle"))
if(ch==1):
    x=int(input("enter side"))
    obj.calc(x)
elif (ch==2):
    x=int(input("enter length"))
    y=int(input("enter breadth:"))
    obj.calc(x,y)
```

Output:



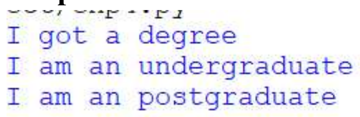
```
Enter 1 for sqaure and 2 for Rectangle2
enter length6
Enter 1 for sqaure and 2 for Rectangle1
enter side9
Area of square= 81

=== Code Execution Successful ===
```

3.Write a Python program to create a class 'Degree' having a method 'getDegree' that prints "I got a degree". It has two subclasses namely 'Undergraduate' and 'Postgraduate' each having a method with the same name that prints "I am an Undergraduate" and "I am a Postgraduate" respectively. Call the method by creating an object of each of the three classes.(Method Overriding)

Program:

```
class degree:
    def getdegree(self):
        print("I got a degree")

class Undergraduate(degree):
    def samemethod(self):
        print("I am an undergraduate")
class postgraduate(degree):
    def samemethod(self):
        print("I am an postgraduate")
obj1=degree()
obj2=Undergraduate()
obj3=postgraduate()
obj1.getdegree()
obj2.samemethod()
obj3.samemethod()
```

Output:



```
I got a degree
I am an undergraduate
I am an postgraduate
```

Conclusion:

In this module, we explored key OOP concepts such as method overriding, method overloading, and abstract classes. These concepts help in creating more flexible and reusable code by allowing multiple implementations and hierarchical class structures.