PRACTICAL NO 7

a) Design a class that store the information of student and display the same.

Code :

```
class Student:
    def __init__ (self,name,age,gender):
        self.name=name
        self.age=age
        self.gender=gender
    def display(self):
        print(f"Student Name   : {self.name}")
        print(f"Student Age    : {self.age}")
        print(f"Student Gender : {self.gender}")
Name=input("Enter Name   : ")
Age=int(input("Enter Age    : "))
Gen=input("Enter Gender : ")
S=Student(Name,Age,Gen)
S.display()
```

Output :

```
Enter Name   : yashodip
Enter Age    : 20
Enter Gender : male
Student Name   : yashodip
Student Age    : 20
Student Gender : male
```

b) Implement the concept of inheritance using python.

1) Single Level Inheritance :

Code :

```
class Student:
    def __init__ (self,name,age):
        self.name=name;
        self.age=age;
    def Display(self):
        print(f"Name : {self.name}")
        print(f"Age  : {self.age}")
class Course(Student):
    def __init__ (self):
        n=input("enter name : ")
        a=int(input("enter age : "))
        super().__init__(n,a)
        self.cname=input("enter course name : ")
    def Display(self):
        super().Display()
        print(f"Course name : {self.cname}")
C=Course()
C.Display()
```

Output :

```
enter name : vaibhavi
enter age : 20
enter course name : FSQC
Name : vaibhavi
Age  : 20
Course name : FSQC
```

2) Multiple Inheritance :

Code :

```
class Internal:
    def __init__(self):
        self.Imarks = int(input("Enter Internal marks : "))

    def Display(self):
        print(f"Internal Marks : {self.Imarks}")

class External:
    def __init__(self):
        self.Emarks = int(input("Enter External marks : "))

    def Display(self):
        print(f"External Marks : {self.Emarks}")

class Total(Internal, External):
    def __init__(self):
        Internal.__init__(self)
        External.__init__(self)

    def Display(self):
        Internal.Display(self)
        External.Display(self)
        print(f"Total marks : {self.Imarks + self.Emarks}")
T = Total()
T.Display()
```

Output :

```
Enter Internal marks : 73
Enter External marks : 21
Internal Marks : 73
External Marks : 21
Total marks : 94
```

3) Multi-Level Inheritance :

Code :

```
class Student:
    def __init__ (self,name,age):
        self.name=name
        self.age=age
    def Display(self):
        print(f"Name \t\t: {self.name}")
        print(f"Age \t\t: {self.age}")

class Teacher(Student):
    def __init__ (self,name,age,sub,ex):
        Student.__init__(self,name,age)
        self.sub=sub
        self.exp=ex
    def Display(self):
        Student.Display(self)
        print(f"Subject \t: {self.sub}")
        print(f"Experience \t: {self.exp} Years")
```

```
class Hod(Teacher):
    def __init__ (self,name,age,sub,ex,dept):
        Teacher.__init__(self,name,age,sub,ex)
        self.dept=dept
    def Display(self):
        Teacher.Display(self)
        print(f"Department \t: {self.dept}")

H=Hod('K.A.Kubal',34,'Green It & PL/SQL',12,'IT')
H.Display()
```

Output :

```
Name       : K.A.Kubal
Age        : 34
Subject    : Green It & PL/SQL
Experience : 12 Years
Department : Information Technology
```

c) Create a class called Numbers, which has a single class attribute called MULTIPLIER, and a constructor which takes the parameters x and y (these should all be numbers).

1) **Write a method called add which returns the sum of the attributes x and y.**
2) **Write a class method called multiply, which takes a single number parameter a and returns the product of a and MULTIPLIER.**
3) **Write a static method called subtract, which takes two number parameters, b and c, and returns b - c.**
4) **Write a method called value which returns a tuple containing the values of x and y. Make this method into a property, and write a setter and a deleter for manipulating the values of x and y.**

Code :

```
class Numbers:
    Multiplier = 10

    def __init__(self,x,y):
        self.x=x
        self.y=y

    def Add(self):
        return self.x + self.y

    def Multiply(self,a):
        return self.Multiplier * a

    @staticmethod
    def Subtract(p,q):
        return p-q

    @property
    def Value(self):
        return self.x,self.y

    @Value.setter
    def Set_X(self,x):
        self.x = x

    @Value.setter
    def Set_Y(self,y):
        self.y = y

    @Value.deleter
    def Delete_X(self):
        self.x = None

    @Value.deleter
    def Delete_Y(self ):
        self.y = None
```

```
x,y=10,20
Num = Numbers(x,y)
print(f"The value of x                   : {x}")
print(f"The value of y                   : {y}")
print(f"Addition of numbers x & y        : {Num.Add()}")
print(f"Multiplication of a number 10    : {Num.Multiply(10)}")
print(subtraction of numbers 400 & 250   : {Num.Subtract(400,250)}")
print(f"Value Set by The Constructor     : {Num.Value}")
Num.Set_X = 22
Num.Set_Y = 25
print(f"Value Set by The Setter Property : {Num.Value}")
del Num.Delete_X
del Num.Delete_Y
print(f"Value Set by The Deleter Property : {Num.Value}")
```

Output :

```
The value of x                    : 10
The value of y                    : 20
Addition of numbers x & y         : 30
Multiplication of a number 10     : 100
Subtraction of numbers 400 & 250  : 150
Value Set by The Constructor      : (10, 20)
Value Set by The Setter Property  : (22, 25)
Value Set by The Deleter Property : (None, None)
```