Practical No 2

Aim : Write A Program To Implement Different Methods In Python.

1. Class Method In Python

```
class ClassMethod:
    # Creating Class Method
    @classmethod
    def call(cls):
        return "I Am Class Method Example"

# Creating Object
obj = ClassMethod()

# Calling
print(obj.call())
```

```
I Am Class Method Example
```

2.Static Method In Python

```
class Static:
    # Creating Static Method
    @staticmethod
    def call():
        return "I Am Static Method"

# Creating Object
obj = Static()

# Calling
print(obj.call())
```

```
I Am Static Method
```

3. Single Inheritance

```
class Parent:
    def func1(self):
        print("This function is in parent class.")

class Child(Parent):
    def func2(self):
        print("This function is in child class.")

object = Child()
object.func1()
object.func2()
```

```
This function is in parent class.
This function is in child class.
```

4. Multiple Inheritance

```
# Multiple Inheritance
# First Class
class Mother:
    motherName = ""
    def motherNameDisplay(self):
        return self.motherName

# Second Class
class Father:
    fatherName = ""
    def fatherNameDisplay(self):
        return self.fatherName

# Third Class
class Child(Mother,Father):
    def getParents(self):
        print("My Mother Is : ",self.motherNameDisplay())
        print("My Father Is : ",self.fatherNameDisplay())

# Calling Class

son = Child()

son.motherName = "Jane Doe"
son.fatherName = "John Doe"

son.getParents()
```

```
My Mother Is :  Jane Doe
My Father Is :  John Doe
```

5. Multi Level Inheritance

```
# Base Class
class Grandfather:
    def __init__(self,grandFatherName):
        self.grandFatherName = grandFatherName

# Intermidiate Class
class Father(Grandfather):
    def __init__(self,fatherName,grandFatherName):
        self.fatherName = fatherName
        # Invoking Constructor Of GrandFather
        Grandfather.__init__(self,grandFatherName)

# Derived Class
class Son(Father):
    def __init__(self,sonName,fatherName,grandFatherName):
        self.sonName = sonName
```

```
        # Invoking Constructor Of Father
        Father.__init__(self,fatherName,grandFatherName)

    def displayFamily(self):
        print("My Name Is : ",self.sonName)
        print("My Father Name Is : ",self.fatherName)
        print("My GrandFather Name Is : ",self.grandFatherName)

# Calling Class
family = Son("Himesh","Dipesh","Alpesh")

print(family.grandFatherName)

family.displayFamily()
```

```
Alpesh
My Name Is :  Himesh
My Father Name Is :  Dipesh
My GrandFather Name Is :  Alpesh
```

6. Calling Of Method in Different Class

```
class ClassA:
    @staticmethod
    def method_in_class_a():
        print("Method in classA")

class ClassB:
    def call_method_from_class_a(self):
        ClassA.method_in_class_a()
        print("Method in classB")

obj_b = ClassB()
obj_b.call_method_from_class_a()
```

```
Method in classA
Method in classB
```

7 . Hierarchical inheritance

```
# Base Class
class Parent():
    def callParent(self):
        print("This Is The Class Of Parent.")

# Child Class
class Child1(Parent):
    def callChild(self):
        print("This Is The Class Of Child 1.")

# Child Class
```

```
class Child2(Parent):
    def callChild(self):
        print("This Is The Class Of Child 2.")

# Child Class
class Child3(Parent):
    def callChild(self):
        print("This Is The Class Of Child 3.")

# Child Class
class Child4(Parent):
    def callChild(self):
        print("This Is The Class Of Child 4.")

# Creating The Object
obj1 = Child1()
obj2 = Child2()
obj3 = Child3()
obj4 = Child4()

# Calling The Object
obj1.callParent()
obj1.callChild()

obj2.callParent()
obj2.callChild()

obj3.callParent()
obj3.callChild()

obj4.callParent()
obj4.callChild()
```

```
This Is The Class Of Parent.
This Is The Class Of Child 1.
This Is The Class Of Parent.
This Is The Class Of Child 2.
This Is The Class Of Parent.
This Is The Class Of Child 3.
This Is The Class Of Parent.
This Is The Class Of Child 4.
```

8 . Hybrid inheritance

```
# Creating Parent Class
class School:
    def schoolName(self):
        print("This is a school")

# Creating Child Class
class Student1(School):
```

```
    def studentName(self):
        print("This is a student1")

class Student2(School):
    def studentName(self):
        print("This is a student2")

# Creating Object And Calling The Method
s1 = Student1()
s1.studentName()
s1.schoolName()

s2 = Student2()
s2.studentName()
s2.schoolName()
```

```
This is a student1
This is a school
This is a student2
This is a school
```

9 . Super Keyboard

```
class Parent:
    def show(self):
        print("Inside Parent")

class Child(Parent):
    def show(self):
        super().show()
        print("Inside Child")

obj = Child()
obj.show()
```

```
Inside Parent
Inside Child
```

9 . Method Overloading

```
class Example:
    def add(self,a,b,c=None):
        x=0
        if a and b and c != None :
            x = a+b+c
        elif a and b != None and c == None:
            x = a+b
        return x

obj = Example()
print(obj.add(10,20,30))
print(obj.add(10,20))
```

```
60
30
```