

Practical No 2

Aim: write a program to implement different method to do polymorphism , operator overloading and overriding in python

1. Polymorphisim

```
class VisualStudio:  
    def execute(self):  
        print("Compiling")  
        print("Running")  
        print("Spell Check")  
        print("Convention Check")
```

```
class Desktop:  
    def code(self,ide):  
        ide.execute()
```

```
ide = VisualStudio()  
desk = Desktop()  
desk.code(ide)
```

```
Compiling  
Running  
Spell Check  
Convention Check
```

```
class Dog:  
    def tellAboutYou(self):  
        print("I am a dog, I am from Kingdom Animalia and from sub-kingdom Mammalia")  
    def limbs(self):  
        print("I have 4 limbs")
```

```
class Lizard:  
    def tellAboutYou(self):  
        print("I am a lizard, I am from Kingdom Animalia and from sub-kingdom Reptilia")  
    def limbs(self):  
        print("I have 4 limbs")
```

```
def printObject(obj):  
    obj.tellAboutYou()  
    obj.limbs()
```

```
d = Dog()  
l = Lizard()
```

```
printObject(d)
printObject(l)
```

```
I am a dog, I am from Kingdom Animalia and from sub-kingdom
Mammalia
I have 4 limbs
I am a lizard, I am from Kingdom Animalia and from sub-kingdom
Reptilia
I have 4 limbs
```

2. Duck type

```
class Duck:
    def swim(self):
        print("I Am A Duck I Can Swim.")

class Sparrow:
    def swim(self):
        print("I Am A Sparrow I Can't Swim,But I Can Fly.")

class Crocodile:
    def swim(self):
        print("I Am A Crocodile I Can Swim And Walk.")

def callFunction(obj):
    obj.swim()

callFunction(Duck())
callFunction(Sparrow())
callFunction(Crocodile())
```

```
I Am A Duck I Can Swim.
I Am A Sparrow I Can't Swim,But I Can Fly.
I Am A Crocodile I Can Swim And Walk.
```

3. Method Overriding

```
class Parent:
    def __init__(self):
        self.name = "Parent"

    def call(self):
        print("I Am Inside ", self.name)

class Child(Parent):
    def __init__(self):
        super().__init__()
        self.name = "Child"

    def call(self):
        print("I Am Inside ", self.name)
```

```
parent = Parent()
parent.call()
child = Child()
child.call()
```

```
I Am Inside Parent
I Am Inside Child
```

4. Operator Overloading

```
class Base:
```

```
    def __init__(self,data):
        self.data = data
```

```
    def __add__(self,other):
        return self.data + other.data
```

```
obj = Base(10)
obj2 = Base(20)
print(obj+obj2)
```

```
fName = input("Enter the first name: ")
lName = input("Enter the last name: ")
print(fName + " " + lName)
```

```
30
Enter the first name: John
Enter the last name: Doe
John Doe
```