Object Oriented Programming (Python l) Lab 8

ASST.LEC Fatima Mohammed & Adnan Habeeb

Introduction to Method Overriding in Python

a very important aspect of object-oriented programming -- Method Overriding in Python. It is an essential part of the inheritance mechanism.

```
class Shape:                  ←  Create a Parent Class
    data1="abc"               ←  Properties
    def no_of_sides(self):    ←  Function no_of_sides
        print("My sides need to be defined. I am from shape class. ")
    def tow_dimensional(self):    ←  Function tow_dimensional
        print("I am a 2D object. I am from shape class ")
class Square(Shape):          ←  Sub-class
    data2="xyz"               ←  Properties
    def no_of_sides(self):
        print("I have 4 sides. I am from Square class")
    def color(self):
        print("I have teal color. I am from square class")
sq=Square()                   ←  Create an object of square class
sq.no_of_sides()              ←  Override the no_of_sides of parent class
sq.tow_dimensional()          ←  Will inherit this method from the parent class
sq.color()                    ←  Its own method – color
print("Old value of date1 =", sq.data1)
sq.data1="New value"          ←  Override property of the parent class
print("the value of data1 in Shape class overridden by the Square class =",sq.data1)
```

Output :

```
I have 4 sides. I am from squuare class
I am a 2D object. I am from shape class
I have teal color. I am from square class
Old value of date1 = abc
the value of data1 in Shape class
overridden by the Square class = New
value
```

Overriding Method in Python

```
class A():              ←        Create a Parent Class
    def __init__(self):
        constant=1      ←        Properties
    def method1(self):  ←        Function method1
        print("method1 of class A")
class B(A):             ←        Sub-class
    def __init__(self):
        A.__init__(self)  ←      Will inherit this method from the parent class
        constant2=2     ←        Properties
        self.calling1()

    def method1(self):  ←        Override the method1 of parent class
        print("method1 of class B")
    def calling1(self): ←        Its own method – calling1
        self.method1()
        A.method1(self)
z=B()                   ←        Create an object of B class
```

Output :

```
method1 of class B
method1 of class A
```

Overriding Method in Python

```
class polygon:
    def __init__(self,no_of_sides):
        self.n=no_of_sides
        self.sides=[]
    def inputsides(self):
        for i in range(self.n):
            self.sides +=[float(input("Enter side"))]
        print(self.sides)
    def dispside(self):
        for i in range(self.n):
            print("side is ",self.sides[i])
class Triangle (polygon):
    def __init__(self):
        polygon.__init__(self,3)
    def findarea(self):
        a,b,c=self.sides
        s=(a+b+c)/2
        area=(s*(s-a)*(s-b)*(s-c))**0.5
        print("The area of the triangle is =",area)
```

```
t=Triangle()
t.inputsides()
t.dispside()
t.findarea()
```

Output :

```
Enter side 4
Enter side 7
Enter side 9
[4.0, 7.0, 9.0]
side is  4.0
side is  7.0
side is  9.0
The area of the triangle is = 13.416407869
```

Thank you