Object Oriented Programming using Python (I)

Lecture(9)

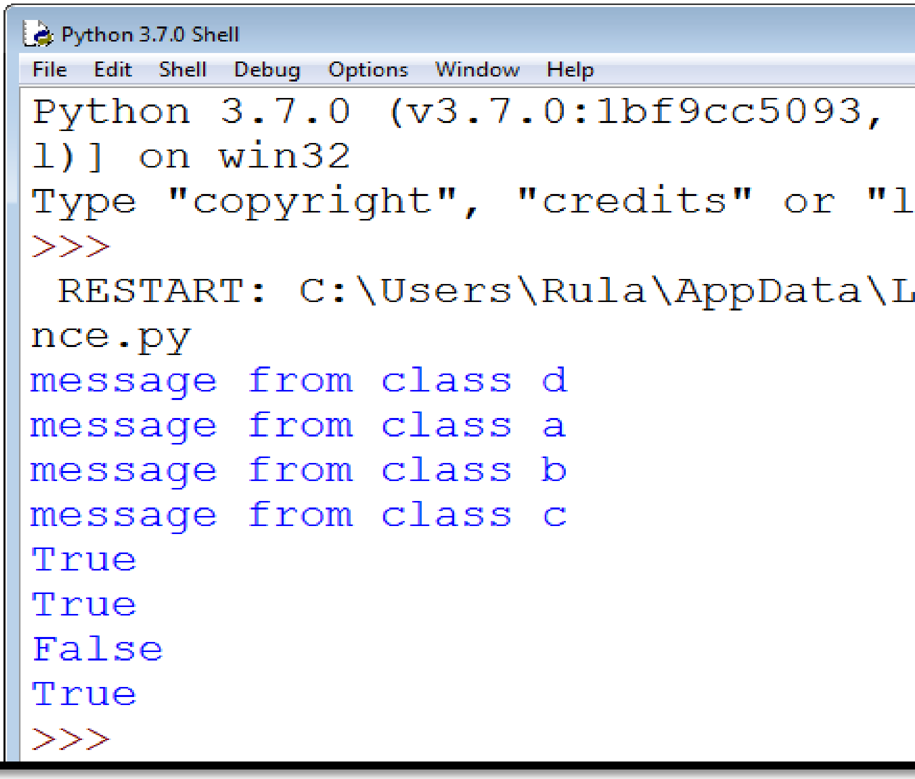
Class Inheritance
Overriding methods

Prepared by: ***Dr. Rula Amjed Hamid***
University of Information Technology and Communications
College of Business Informatics

Multi Inheritance (2)

There are 2 built-in functions in Python that are related to inheritance to check a relationships of two classes and instances.
They are:
1. **isinstance():** It checks the type of an object.
Its syntax is:
isinstance(object_name, class_name)
It would return **True** if the class of object_name is class_name else **False**.

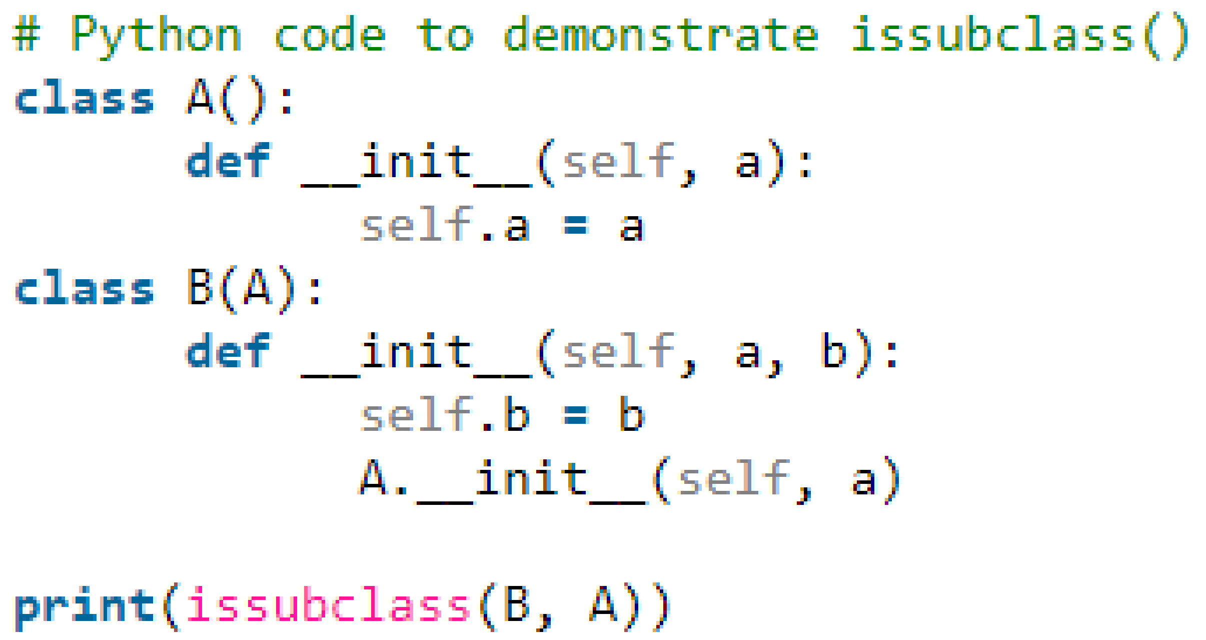
```
class a:
    def m(self):
        print("message from class a")
class b(a):
    def m(self):
        print("message from class b")
class c():
    def m(self):
        print("message from class c")
class d(b,c):
    def m(self):
        print("message from class d")
        a.m(self)
        b.m(self)
        c.m(self)
obj1=d()
obj1.m()
obj2=a()
print(isinstance(obj1,a))
print(isinstance(obj1,d))
print(isinstance(obj2,c))
print(isinstance(5, int))
```



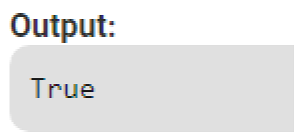
```
Python 3.7.0 (v3.7.0:1bf9cc5093,
1)] on win32
Type "copyright", "credits" or "l
>>>
 RESTART: C:\Users\Rula\AppData\L
nce.py
message from class d
message from class a
message from class b
message from class c
True
True
False
True
>>>
```

This is because 5 is an integer and hence belongs to the class of int.
NOTE: ‘int’ is both a type and a class in Python.

2. issubclass(): It checks whether a specific class is the child class of another class or not. Its syntax is:
issubclass(childclass_name, parentclass_name)
It would return **True** if the entered child class is actually derived from the entered parent class else, it returns **False**.
For example:

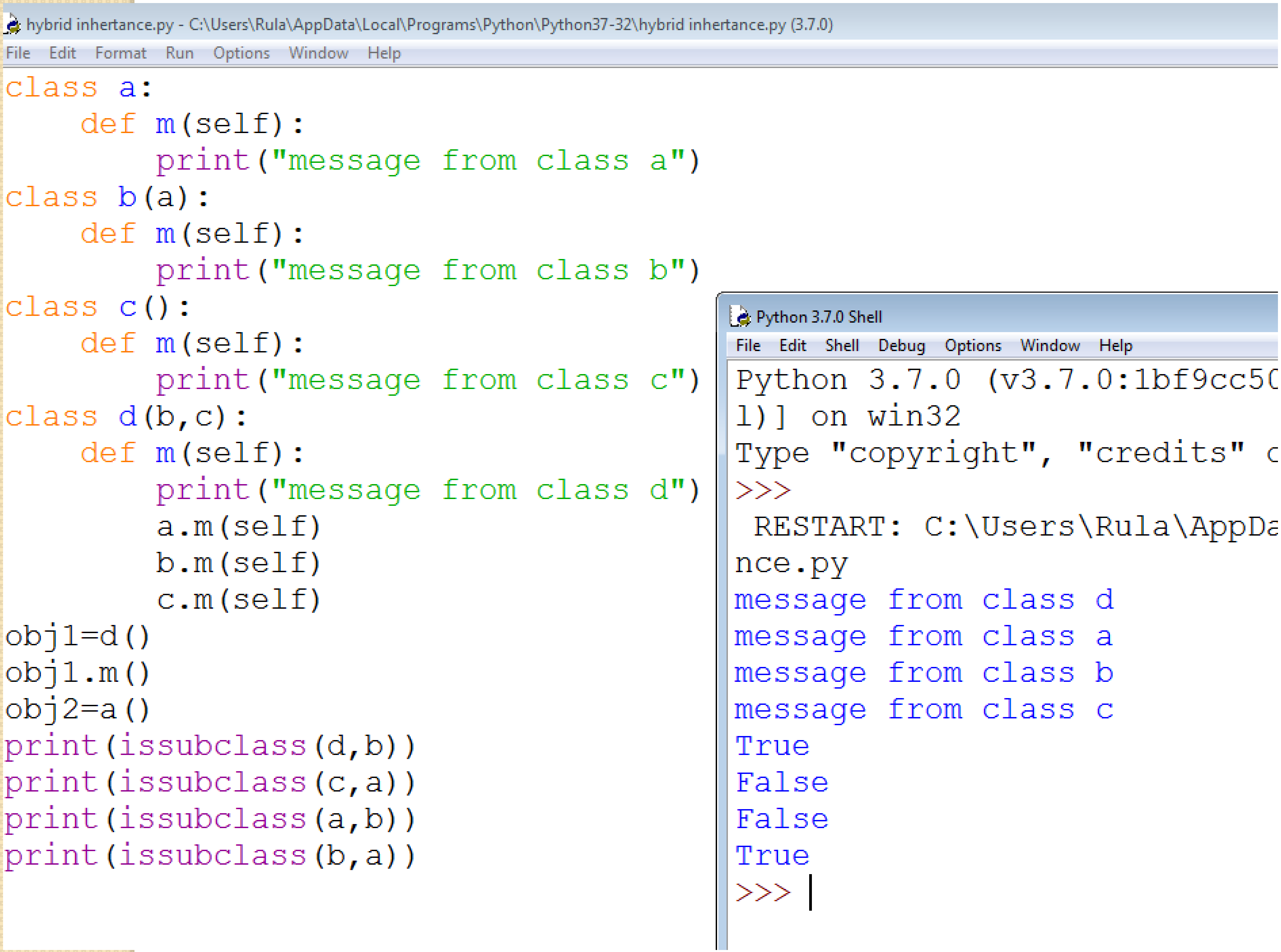


```
# Python code to demonstrate issubclass()
class A():
      def __init__(self, a):
            self.a = a
class B(A):
      def __init__(self, a, b):
            self.b = b
            A.__init__(self, a)

print(issubclass(B, A))
```



Output:

```
True
```


hybrid inhertance.py - C:\Users\Rula\AppData\Local\Programs\Python\Python37-32\hybrid inhertance.py (3.7.0)
File Edit Format Run Options Window Help

```python
class a:
    def m(self):
        print("message from class a")
class b(a):
    def m(self):
        print("message from class b")
class c():
    def m(self):
        print("message from class c")
class d(b,c):
    def m(self):
        print("message from class d")
        a.m(self)
        b.m(self)
        c.m(self)
obj1=d()
obj1.m()
obj2=a()
print(issubclass(d,b))
print(issubclass(c,a))
print(issubclass(a,b))
print(issubclass(b,a))
```

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help

```
Python 3.7.0 (v3.7.0:1bf9cc50
1)] on win32
Type "copyright", "credits" o
>>>
 RESTART: C:\Users\Rula\AppDa
nce.py
message from class d
message from class a
message from class b
message from class c
True
False
False
True
>>> |
```

Overriding Methods:

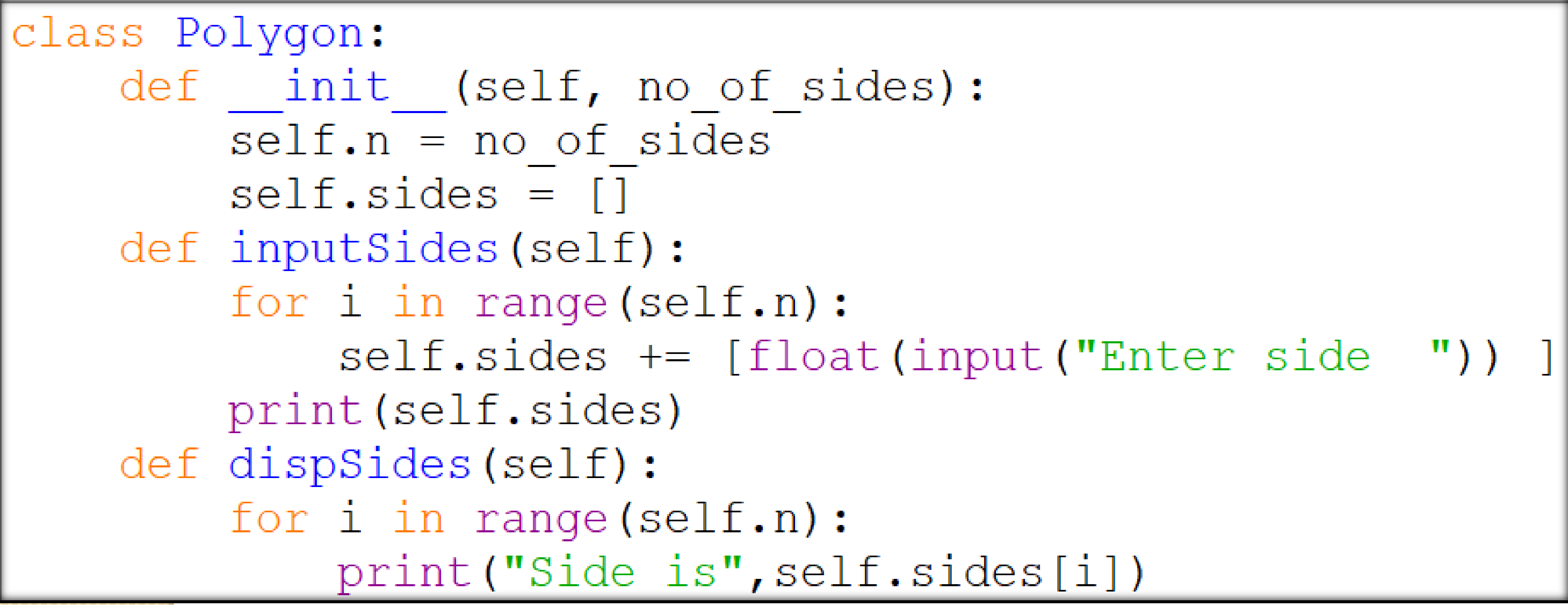
Overriding a method means redefining a method in the subclass when it has already been defined in some other class.
A method in the subclass would be called as overridden only when there exists another method with the same name and same set of parameters in the superclass.

For example:

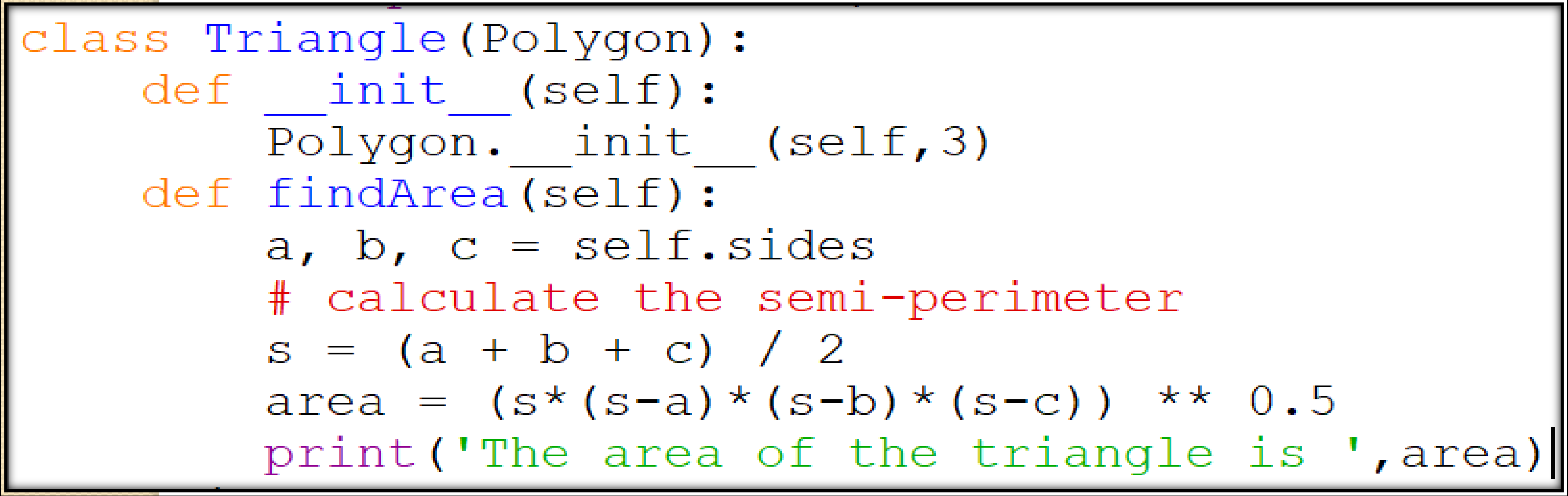
```
# Base Class
class A(object):
        def __init__(self):
                constant1 = 1
        def method1(self):
                print('method1 of class A')

class B(A):
        def __init__(self):
                constant2 = 2
                self.calling1()
                A. __init__(self)
        def method1(self):
                print('method1 of class B')
        def calling1(self):
                self.method1()
                A.method1(self)
b = B()
```

Output:

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

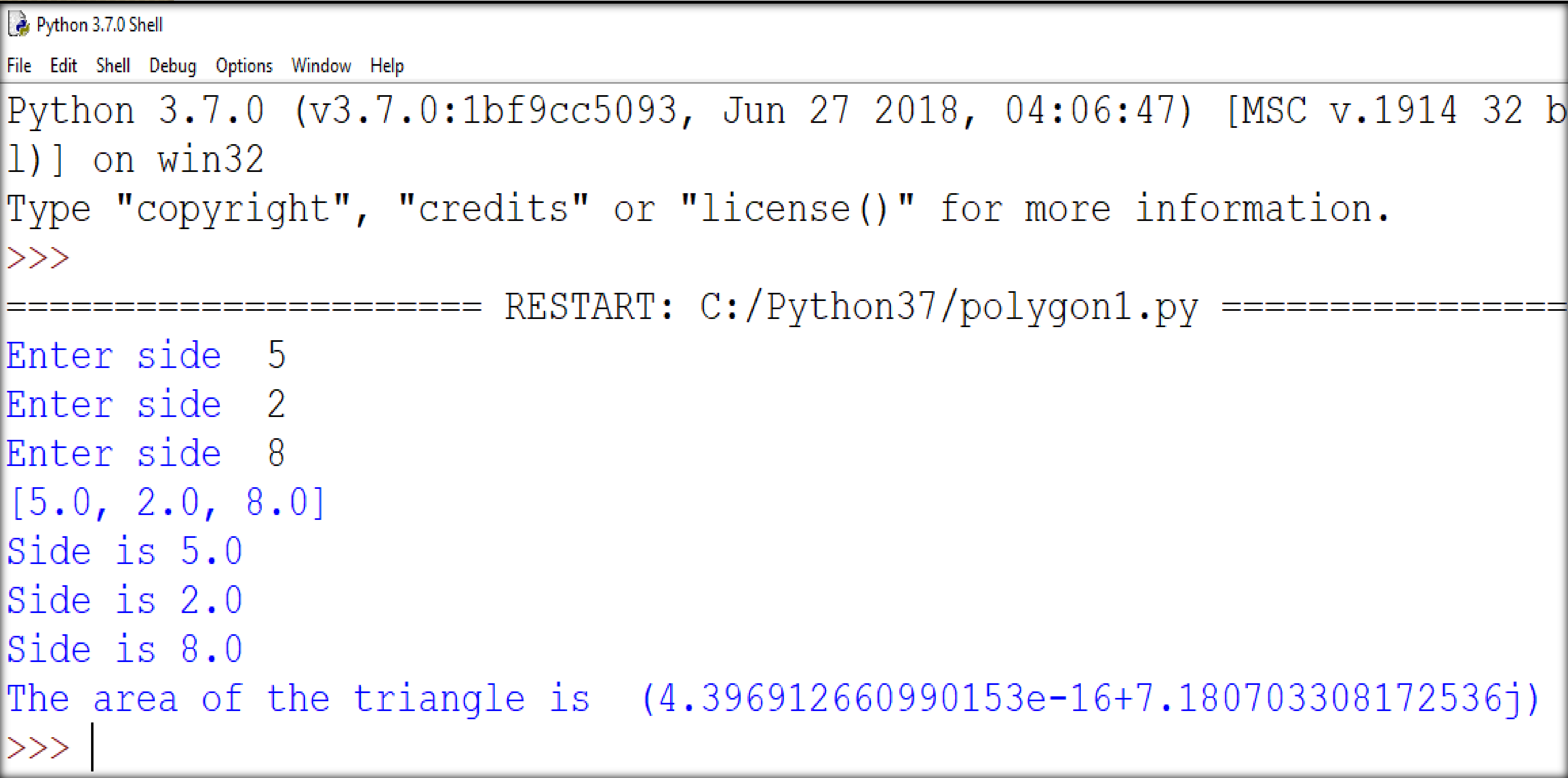


```
class Polygon:
    def __init__(self, no_of_sides):
        self.n = no_of_sides
        self.sides = []
    def inputSides(self):
        for i in range(self.n):
            self.sides += [float(input("Enter side  ")) ]
        print(self.sides)
    def dispSides(self):
        for i in range(self.n):
            print("Side is",self.sides[i])
```



```
class Triangle(Polygon):
    def __init__(self):
        Polygon.__init__(self,3)
    def findArea(self):
        a, b, c = self.sides
        # calculate the semi-perimeter
        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.dispSides()
t.findArea()
```



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 b
l)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
====================== RESTART: C:/Python37/polygon1.py ================
Enter side  5
Enter side  2
Enter side  8
[5.0, 2.0, 8.0]
Side is 5.0
Side is 2.0
Side is 8.0
The area of the triangle is  (4.396912660990153e-16+7.180703308172536j)
>>>
```