Chapter 10

Classes

Chapter 10

Classes

In This Chapter:

1. Dealing with Complex Numbers
2. Find the Torsional Angle

10.1 Dealing with Complex Numbers

You are given two complex numbers, and you have to print the result of their addition, subtraction, multiplication, division and modulus operations.

The real and imaginary precision part should be correct up to two decimal places.

Input Format

One line of input: The real and imaginary part of a number separated by a space.

Output Format

For two complex numbers C and D, the output should be in the following sequence on separate lines:

$$C + D$$

$$C - D$$

$$C * D$$

$$C/D$$

$$mod(C)$$

$$mod(D)$$

For complex numbers with non-zero real (A) and complex part (B), the output should be in the following format:

A + B*i*

Replace the plus symbol (+) with a minus symbol (-) when B<0.

For complex numbers with a zero complex part i.e. real numbers, the output should be:

A + 0.00*i*

For complex numbers where the real part is zero and the complex part is non-zero, the output should be:

0.00 + B*i*

Sample Input

```
2 1
5 6
```

Sample Output

```
7.00+7.00i
-3.00-5.00i
4.00+17.00i
0.26-0.11i
2.24+0.00i
7.81+0.00i
```

Concept

Python is a fully object-oriented language like C++, Java, etc. For reading about classes.

Methods with a double underscore before and after their name are considered as built-in methods. They are used by interpreters and are generally used in the implementation of overloaded operators or other built-in functionality.

```
__add__-> Can be overloaded for + operation
```

```
__sub__ -> Can be overloaded for - operation
```

```
__mul__ -> Can be overloaded for * operation
```

#Code

```
import math
from math import sqrt

class Complex:
    def __init__(self, real = 0, im = 0):
        self.r = real
        self.i = im
        self.rr = 0
        self.ri = 0

    def mprint(self):
        if self.ri >= 0:
            return ('{:.2f}+{:.2f}i'.format(self.rr, self.ri))
        elif self.ri < 0:
            return ('{:.2f}-{:.2f}i'.format(self.rr, abs(self.ri)))

    def __add__(self, d):
        self.rr = self.r+d.r
        self.ri = self.i+d.i
        return (self.mprint())

   def __sub__(self, d):
        self.rr = self.r-d.r
        self.ri = self.i-d.i
        return (self.mprint())

    def __mul__(self, d):
        self.rr = self.r*d.r-self.i*d.i
        self.ri = self.r*d.i+self.i*d.r
        return (self.mprint())

    def __truediv__(self, d):
        self.rr = (self.r*d.r+self.i*d.i)/(d.r**2+d.i**2)
        self.ri =(self.i*d.r-self.r*d.i)/(d.r**2+d.i**2)
        return (self.mprint())
```

```
    def mod(self):
        self.rr = 0
        self.ri = 0
        self.rr = sqrt(self.r**2+self.i**2)
        return (self.mprint())

if __name__ == '__main__':
    c = map(float, input().split())
    d = map(float, input().split())
    x = Complex(*c)
    y = Complex(*d)
    print(*map(str, [x+y, x-y, x*y, x/y, x.mod(), y.mod()]), sep='\n')
```

10.2 Find the Torsional Angle

You are given four points A, B, C and D in a 3-dimensional Cartesian coordinate system. You are required to print the angle between the plane made by the points A, B, C and B, C, D in degrees (**not radians**). Let the angle be PHI.

$$Cos(PHI) = (X.Y)/|X||Y|$$

where X= AB × BC and Y = BC × CD

Here, X.Y means the dot product of X and Y, and AB × BC means the cross product of vectors AB and BC. Also, AB= B-A.

Input Format

One line of input containing the space separated floating number values of the X, Y and Z coordinates of a point.

Output Format

Output the angle correct up to two decimal places.

Sample Input

```
0 4 5
1 7 6
0 5 9
1 7 2
```

Sample Output

```
8.19
```

#Code

```
import math

class Points(object):
    def __init__(self, x, y, z):
        self.x =x
        self.y = y
        self.z= z

    def __sub__(self, no):
        return Points(self.x-no.x, self.y-no.y, self.z -no.z)

    def dot(self, no):
        return self.x*no.x+ self.y*no.y+self.z *no.z

    def cross(self, no):
         return Points(self.y* no.z - self.z*no.y,
                       self.z*no.x - self.x*no.z,
                       self.x*no.y - self.y*no.x)

    def absolute(self):
        return pow((self.x ** 2 + self.y ** 2 + self.z ** 2), 0.5)

if __name__ == '__main__':
    points = list()
    for i in range(4):
        a = map(float, raw_input().split())
        points.append(a)

    a=Points( *points[0])
    b=Points(*points[1])
    c=Points(*points[2])
    d=Points(*points[3])

    x = (b - a).cross(c - b)
    y = (c - b).cross(d - c)
```

```
angle = math.acos(x.dot(y) / (x.absolute() * y.absolute()))
print "%.2f" % math.degrees(angle)
```