04_07_class_inheritence

Unknown Author

April 1, 2014

Part I

Class Hierarchies

In [1]:

```
# Imports
import math
```

In this lecture we shall try to understand how to use functions defined in one class easily in other classes. For that, we shall define a class called Polygon which will define a polygon on the plane. It will have methods to find the area and circumference. Then we shall use this to have special subclasses : Rectangle, Triangle. We shall then have another subclass : RegularPolygon. Square, EquilateralTriangle will be subclasses of this class, as well as Rectangle and Triangle respectively.

1 The Class Polygon

The shoelace method says that if a polygon has vertices $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ in *counter clockwise* direction, then the area of the polygon is given by the formula

$$\frac{1}{2}\left(\det\begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} + \det\begin{pmatrix} x_2 & x_3 \\ y_2 & y_3 \end{pmatrix} + \cdots + \det\begin{pmatrix} x_{n-1} & x_n \\ y_{n-1} & y_n \end{pmatrix} + \det\begin{pmatrix} x_n & x_1 \\ y_n & y_1 \end{pmatrix}\right). \tag{1}$$

In [2]:

```
class Polygon :
    """Computer representation of a polygon.

    Attributes :
        lop : A list of coordinate pairs which list the vertices of
        the polygon in the counter clockwise order.

    Method :
        __init__ : Initializes the polygon
        area() : returns the area of the polygon
        __str__ : Returns the string "n-gon with area A"
        __repr__ : returns lop in string format
    """

    def __init__(self, lop) :
        self.lop = lop

    def area(self) :
        l = self.lop # list of vertices
        n = len(l) # number of vertices
        # Find area using the shoelace method
        s = 0 # To store the value of the sum in shoelace
        for i in range(n) :
            ip1mn = (i + 1) % n
            s += l[i][0] * l[ip1mn][1] - l[ip1mn][0]*l[i][1]
        return float(s)/2.0

    def __str__(self) :
        return "%d-gon with area %g" % (len(self.lop), self.area())

    def __repr__(self) :
        return "Polygon(" + str(self.lop) + ")"
```

In [3]:

```
if __name__ == '__main__' :
    print "Testing class Polygon"
    poly = Polygon([(0, 0), (1, 0), (1, 1), (0, 1)])
    print "Our polygon is a", poly, "."
    print "It was defined by", poly.__repr__(), "."
```

```
Testing class Polygon
Our polygon is a 4-gon with area 1 .
It was defined by Polygon([(0, 0), (1, 0), (1, 1), (0, 1)]) .
```

In [4]:

```
class Rectangle(Polygon) :
    """A subclass of Polygon which defines rectangle by the two ends of the base and h
    Positive height is in the direction 90 deg counter clockwise from v1 -> v2.
    """
    def __init__(self, v1, v2, height) :
        """v1 : first vertex of the base
        v2 : second vertex
        height : height
        """

        (x1, y1) = v1
        (x2, y2) = v2
        perpdir = (y1 - y2, x2 - x1)
        lenperp = math.sqrt((y1 - y2)**2 + (x2 - x1)**2)

        x3 = float(height) * (y1 - y2) / lenperp + x1 # Why?
        y3 = float(height) * (x2 - x1) / lenperp + y1 # Why?
        v3 = (x3, y3)

        x4 = float(height) * (y1 - y2) / lenperp + x2 # Why?
        y4 = float(height) * (x2 - x1) / lenperp + y2 # Why?
        v4 = (x4, y4)

        Polygon.__init__(self, [v1, v2, v4, v3])

    def __str__(self) :
        return "A rectangle of area %g" % self.area()
```

In [5]:

```
if __name__ == '__main__' :
    rect = Rectangle((0,0), (1, 1), math.sqrt(1.0/2))
    print rect
    print rect.__repr__()
```

```
A rectangle of area 1
Polygon([(0, 0), (1, 1), (0.5, 1.5), (-0.5, 0.5)])
```

2 Class Triangle

In [6]:

```
class Triangle(Polygon) :
    """Create a triangle given 3 vertices."""

    def __init__(self, v1, v2, v3) :
        Polygon.__init__(self, [v1, v2, v3])

    def __str__(self) :
        return "A triangle with area %g" % self.area()
```

In [7]:

```
if __name__ == ’__main__’ :
    tr = Triangle((0,0), (1,0), (0, 2))
    print tr
    print tr.__repr__()
```

```
A triangle with area 1
Polygon([(0, 0), (1, 0), (0, 2)])
```

3 Class RegularPolygon

In [8]:

```
class RegularPolygon(Polygon) :
    """Given the end points of a side an number of sides,
    this creates an internal representation of a regular
    polygon."""

    def __init__(self, v1, v2, n) :
        internalangle = float(n-2) * math.pi / float(n)
        # print internalangle * 180/ math.pi
        (x1, y1) = v1
        (x2, y2) = v2
        lenside = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)

        lop = [v1, v2]
        for i in range(3, n+1) :
            vm2 = lop[-2]
            vm1 = lop[-1]
            (x, y) = vm2
            (z, w) = vm1

            # Why does the following work?
            p = z + (x - z) * math.cos(internalangle) \
            + (y - w) * math.sin(internalangle)
            q = w + (y - w) * math.cos(internalangle) \
            + (z - x) * math.sin(internalangle)
            lop.append((p, q))
            # print x, y, z, w, p,q

        Polygon.__init__(self, lop)
```

In [9]:

```
if __name__ == ’__main__’ :
    pt = RegularPolygon((4, 3), (4 + 1.0/math.sqrt(2), 3 + 1.0/math.sqrt(2)), 5)
    print pt
    print pt.__repr__()
```

```
5-gon with area 1.72048
Polygon([(4, 3), (4.707106781186548, 3.7071067811865475),
(4.253116281447001, 4.598113305374915), (3.2654279408518634,
4.441678840334684), (3.108993475811633, 3.4539904997395463)])
```

4 Class EquilateralTriangle and Square.

I use multiple inheritence just to show how it works. However it is better not to use multiple inheritence and some languages do not even support it.

In [10]:

```
class EquilateralTriangle(Triangle, RegularPolygon) :
    def __init__(self, v1, v2) :
        RegularPolygon.__init__(self, v1, v2, 3)
```

In [11]:

```
if __name__ == '__main__' :
    eqt = EquilateralTriangle((0, 0), (1, 0))
    print eqt
    print eqt.__repr__()
```

```
A triangle with area 0.433013
Polygon([(0, 0), (1, 0), (0.4999999999999999, 0.8660254037844386)])
```

In [12]:

```
class Square(Rectangle, RegularPolygon) :
    def __init__(self, v1, v2) :
        RegularPolygon.__init__(self, v1, v2, 4)
```

In [13]:

```
if __name__ == '__main__' :
    sq = Square((0, 0), (1, 0))
    print sq
    print sq.__repr__()
```

```
A rectangle of area 1
Polygon([(0, 0), (1, 0), (0.9999999999999999, 1.0),
(-1.1102230246251565e-16, 0.9999999999999998)])
```