Classes

March 30, 2016

1 Objektorientering

- Grupperar data och funktioner till en enhet
- Funktioner i klassen opererar på data i klassen
- Möjligör enklare och mer lättförstådd kod
- Kan utökas i framtiden utan att existerande kod behöver ändras
- Alla datatyper i python är klasser

2 Funktionsorienterad programmering

```
In [12]: def createPoint(x, y):
             return [x, y]

         def movePoint(point, dx, dy):
             point[0] += dx
             point[1] += dy

         def zeroPoint(point):
             point[0] = 0.0
             point[1] = 0.0

         def setPoint(point, x, y):
             point[0] = x
             point[1] = y

         def printPoint(point):
             print("x =",point[0], "y = ", point[1])
```

```
In [14]: p = createPoint(0.5, 0.0)
         print(p)
```

```
[0.5, 0.0]
```

```
In [15]: movePoint(p, 3.0, 2.0)
         print(p)
```

```
[3.5, 2.0]
```

```
In [16]: setPoint(p, -2.0, -1.0)
         print(p)
```

```
[-2.0, -1.0]
```

```
In [11]: printPoint(p)
```

```
x = -1.0 y =  -1.0
```

3 Motsvarande med objektorienterad kod

```
In [132]: import math

          class Point(object):
              n = 42
              def __init__(self):
                  self.x = 0.0
                  self.y = 0.0

              def move(self, dx, dy):
                  self.x += dx
                  self.y += dy

              def setPoint(self, x, y):
                  self.x = x
                  self.y = y

              def printPoint(self):
                  print(Point.n, self.x, self.y)

              def incN(self):
                  Point.n +=1

              def __str__(self):
                  return "Point("+str(self.x)+","+str(self.y)+")"

              def __call__(self, x):
                  return math.sin(x)
```

```
In [133]: p = Point()
```

```
In [134]: print(p.x)
```

```
0.0
```

```
In [135]: print(p)
```

```
Point(0.0,0.0)
```

```
In [136]: p2 = Point()
```

```
In [137]: print(p2)
```

```
Point(0.0,0.0)
```

```
In [138]: p2.x = 2.0
          print(p.x)
          print(p2.x)
```

```
0.0
2.0
```

```
In [139]: p.printPoint()
```

```
42 0.0 0.0
```

```
In [140]: p2.printPoint()
```

```
42 2.0 0.0
```

```
In [ ]:
```

```
In [141]: p.printPoint()
```

```
42 0.0 0.0
```

```
In [142]: p.incN()
```

```
In [143]: p2.printPoint()
```

```
43 2.0 0.0
```

```
In [144]: p.move(10.0, 10.0)
```

```
In [145]: p.printPoint()
```

```
43 10.0 10.0
```

```
In [146]: print(p)
```

```
Point(10.0,10.0)
```

```
In [148]: import math
          p(1)
```

```
Out[148]: 0.8414709848078965
```

4 Arv

```
In [149]: class Circle(Point):
              def __init__(self):
                  Point.__init__(self)
                  self.r = 0.5

              def setR(self, r):
                  self.r = r

              def area(self):
                  return math.pi*pow(self.r,2)
```

```
In [150]: c = Circle()
```

```
In [151]: print(c)
```

```
Point(0.0,0.0)
```

```
In [152]: print(c.area())
```

```
0.7853981633974483
```

```
In [153]: c.printPoint()
```

```
43 0.0 0.0
```

```
In [154]: c.move(10.0,10.0)
```

```
In [155]: c.printPoint()
```

```
43 10.0 10.0
```

```
In [ ]:
```