6.00 Introduction to Computer Science and Programming
Fall 2008

6.00 Handout, Lecture 15
Not intended to make sense outside of lecture

```
import math

## points as lists

def addPoints(p1,p2):
    r = []
    r.append(p1[0]+p2[0])
    r.append(p1[1]+p2[1])
    return r

p = [1,2]
q = [3,1]
r = addPoints(p,q)
print r

## points as classes

class cartesianPoint:
    pass

cp1 = cartesianPoint()
cp2 = cartesianPoint()
cp1.x = 1.0
cp1.y = 2.0
cp2.x = 1.0
cp2.y = 3.0

def samePoint(p1,p2):
    return (p1.x == p2.x) and (p1.y == p2.y)

def printPoint(p):
    print '(' + str(p.x) + ', ' + str(p.y) + ')'

class polarPoint:
    pass

pp1 = polarPoint()
pp2 = polarPoint()
pp1.radius = 1.0
pp1.angle = 0
pp2.radius = 2.0
pp2.angle = math.pi / 4.0

class cPoint:
    def __init__(self,x,y):
        self.x = x
        self.y = y
        self.radius = math.sqrt(self.x*self.x + self.y*self.y)
        self.angle = math.atan2(self.y,self.x)
    def cartesian(self):
        return (self.x, self.y)
    def polar(self):
        return (self.radius, self.angle)
    def __str__(self):
        return '(' + str(self.x) + ', ' + str(self.y) + ')'
```

```
    def __cmp__(self,other):
        return (self.x == other.x) and (self.y == other.y)

class pPoint:
    def __init__(self,r,a):
        self.radius = r
        self.angle = a
        self.x = r * math.cos(a)
        self.y = r * math.sin(a)
    def cartesian(self):
        return (self.x, self.y)
    def polar(self):
        return (self.radius, self.angle)
    def __str__(self):
        return '(' + str(self.x) + ', ' + str(self.y) + ')'
    def __cmp__(self,other):
        return (self.x == other.x) and (self.y == other.y)

class Segment:
    def __init__(self,start,end):
        self.start = start
        self.end = end
    def length(self):
        return math.sqrt( ((self.start.x - self.end.x) *
                           (self.start.x - self.end.x))
                        + ((self.start.y - self.end.y) *
                           (self.start.y - self.end.y)))

p1 = cPoint(3.0, 4.0)
p2 = cPoint(5.0, 7.0)
s1 = Segment(p1,p2)
print s1.length()

class Rectangle:
    def __init__(self,width,height,corner):
        self.width = width
        self.height = height
        self.corner = corner


def findCenter(box):
  p = cPoint(box.corner.x + box.width/2.0,
             box.corner.y - box.height/2.0)
  return p

box = Rectangle(100,200,p1)
print findCenter(box)

def growRect(box, dwidth, dheight):
  box.width = box.width + dwidth
  box.height = box.height + dheight

growRect(box,10,20)
print findCenter(box)


class newPoint:
    def __init__(self,x = 0,y = 0):
```

```
        self.x = x
        self.y = y
    def __str__(self):
        return '(' + str(self.x) + ', ' + str(self.y) + ')'
    def __eq__(self,other):
        return (self.x == other.x) and (self.y == other.y)
    def __add__(self, other):
        return newPoint(self.x + other.x, self.y + other.y)
    def __cmp__(self,other):
        return self.x < other.x and self.y < other.y

origin = newPoint()
p1 = newPoint(3.0,4.0)
p2 = newPoint()
p3 = p1+p2
print p3
print p1 < p2
```