**6.00 Handout, Lecture 13**
**(Not intended to make sense outside of lecture)**

```
import random

class Location(object):
    def __init__(self, x, y):
        """x and y are floats"""
        self.x = x
        self.y = y
    def move(self, deltaX, deltaY):
        """deltaX and deltaY are floats"""
        return Location(self.x + deltaX, self.y + deltaY)
    def getX(self):
        return self.x
    def getY(self):
        return self.y
    def distFrom(self, other):
        ox = other.x
        oy = other.y
        xDist = self.x - ox
        yDist = self.y - oy
        return (xDist**2 + yDist**2)**0.5
    def __str__(self):
        return '<' + str(self.x) + ', ' + str(self.y) + '>'

class Field(object):
    def __init__(self):
        self.drunks = {}
    def addDrunk(self, drunk, loc):
        if drunk in self.drunks:
            raise ValueError('Duplicate drunk')
        else:
            self.drunks[drunk] = loc
    def moveDrunk(self, drunk):
        if not drunk in self.drunks:
            raise ValueError('Drunk not in field')
        xDist, yDist = drunk.takeStep()
        self.drunks[drunk] = self.drunks[drunk].move(xDist, yDist)
    def getLoc(self, drunk):
        if not drunk in self.drunks:
            raise ValueError('Drunk not in field')
        return self.drunks[drunk]

class Drunk(object):
    def __init__(self, name):
        self.name = name
    def takeStep(self):
        stepChoices = [(0,1), (0,-1), (1, 0), (-1, 0)]
        return random.choice(stepChoices)
    def __str__(self):
        return 'This drunk is named ' + self.name
```

```
def walk(f, d, numSteps):
    start = f.getLoc(d)
    for s in range(numSteps):
        f.moveDrunk(d)
    return(start.distFrom(f.getLoc(d)))

def simWalks(numSteps, numTrials):
    homer = Drunk('Homer')
    origin = Location(0, 0)
    distances = []
    for t in range(numTrials):
        f = Field()
        f.addDrunk(homer, origin)
        distances.append(walk(f, homer, numTrials))
    return distances

def drunkTest(numTrials):
    for numSteps in [0, 10, 100, 1000, 10000]:
        distances = simWalks(numSteps, numTrials)
        print 'Random walk of ' + str(numSteps) + ' steps'
        print '  Mean =', sum(distances)/len(distances)
        print '  Max =', max(distances), 'Min =', min(distances)

def rollDie():
    """returns a random int between 1 and 6"""
    return random.choice([1,2,3,4,5,6])

def testRoll(n = 10):
    result = ''
    for i in range(n):
        result = result + str(rollDie())
    print result

import pylab

pylab.plot([1,2,3,4], [1,2,3,4])
pylab.plot([1,4,2,3], [5,6,7,8])
pylab.show()

pylab.figure(1)
pylab.plot([1,2,3,4], [1,2,3,4])
pylab.figure(2)
pylab.plot([1,4,2,3], [5,6,7,8])
pylab.savefig('firstSaved')
pylab.figure(1)
pylab.plot([5,6,7,10])
pylab.savefig('secondSaved')
pylab.show()
```

```
principal = 10000  #initial investment
interestRate = 0.05
years = 20
values = []
for i in range(years+1):
    values.append(principal)
    principal += principal*interestRate
pylab.plot(values)

pylab.title('5% Growth, Compounded Annually')
pylab.xlabel('Years of Compounding')
pylab.ylabel('Value of Principal ($)')

pylab.show()
```

6.00SC Introduction to Computer Science and Programming
Spring 2011