Random Walks and Simulation Models

Lecturer: John Guttag

Simulation Models

Simulation attempts to build an experimental device called a model

Kinds of Simulation Models

Deterministic simulations are completely defined by the model
    Rerunning the simulation will not change the result

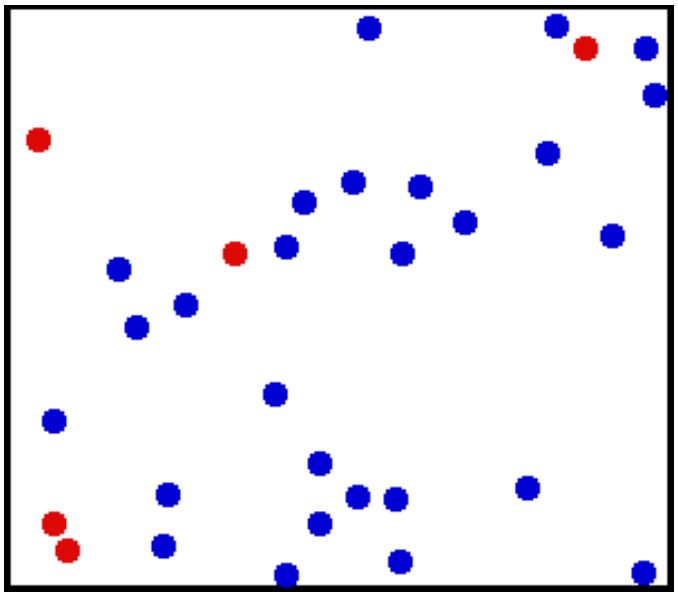
Stochastic simulations include randomness
    Different runs can generate different results

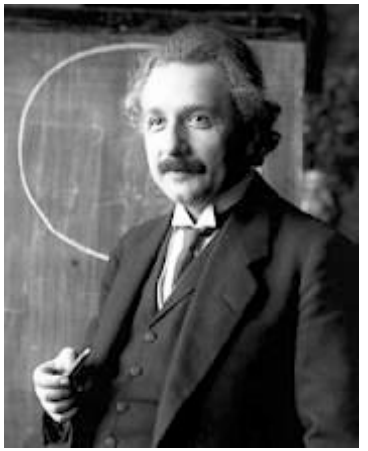
In a discrete model, values of variables are enumerable (e.g., integers). In a continuous model, they are not enumerable (e.g., real numbers).

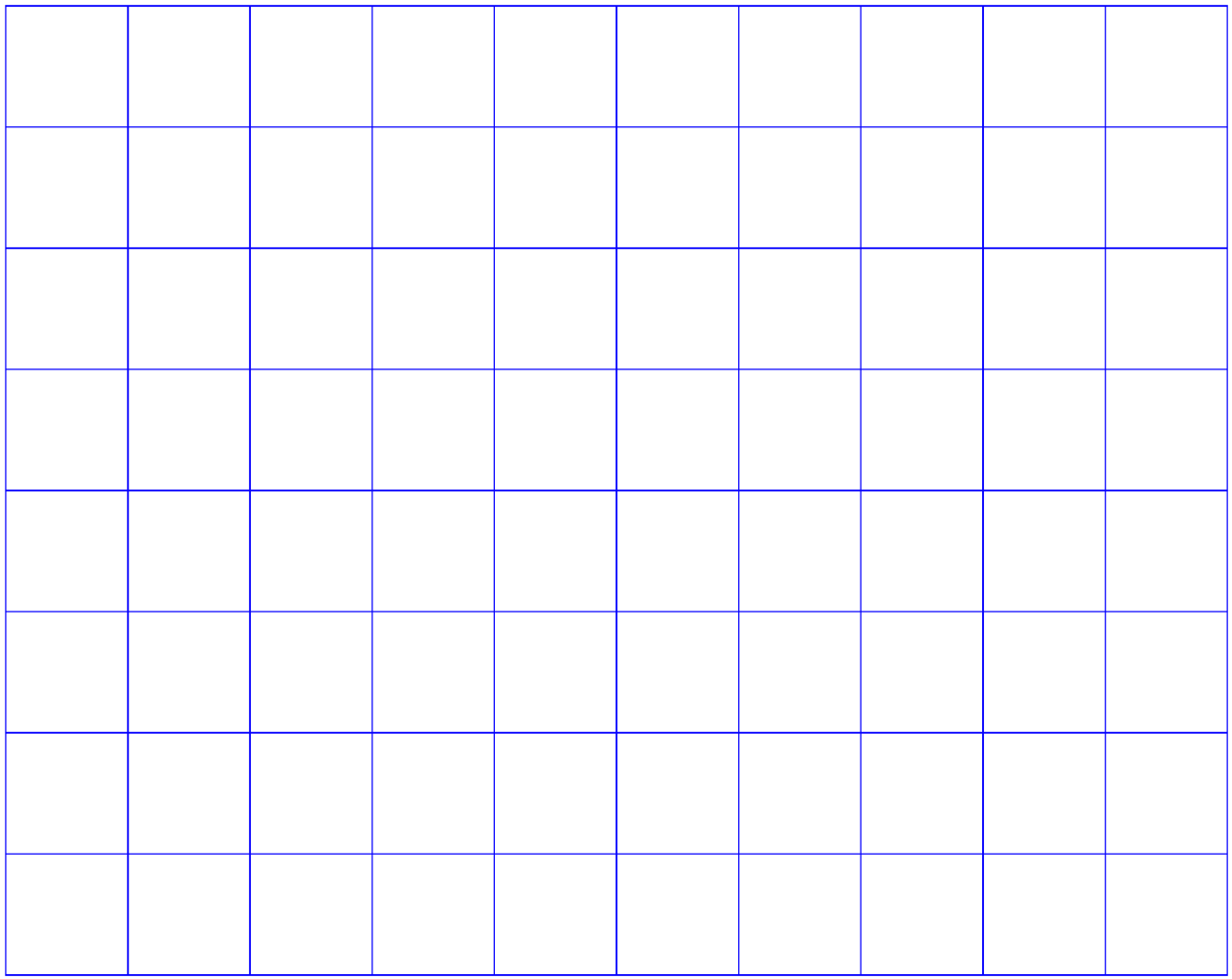
Random Walks and Simulation Models

Lecturer: John Guttag

Brownian Motion







Notable Aspects of Class Location

Notable Aspects of Class Field

```
class Drunk(object):
    def __init__(self, name):
        self.name = name
    def __str__(self):
        return 'This drunk is named ' + self.name

import random

class UsualDrunk(Drunk):
    def takeStep(self):
        stepChoices =\
            [(0.0,1.0),(0.0,-1.0),(1.0, 0.0),(-1.0, 0.0)]
        return random.choice(stepChoices)
```

Random Walks and Simulation Models

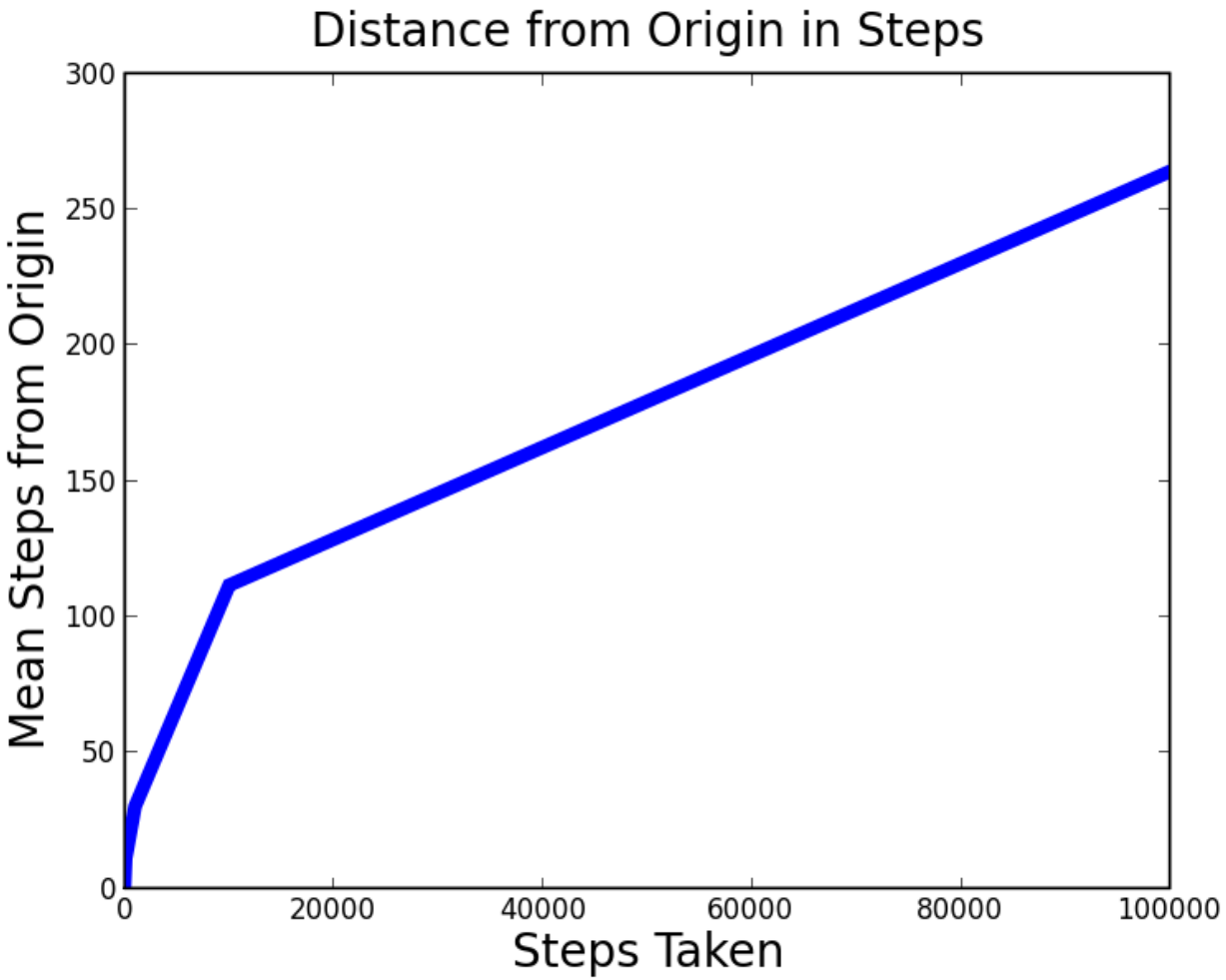
Lecturer: John Guttag

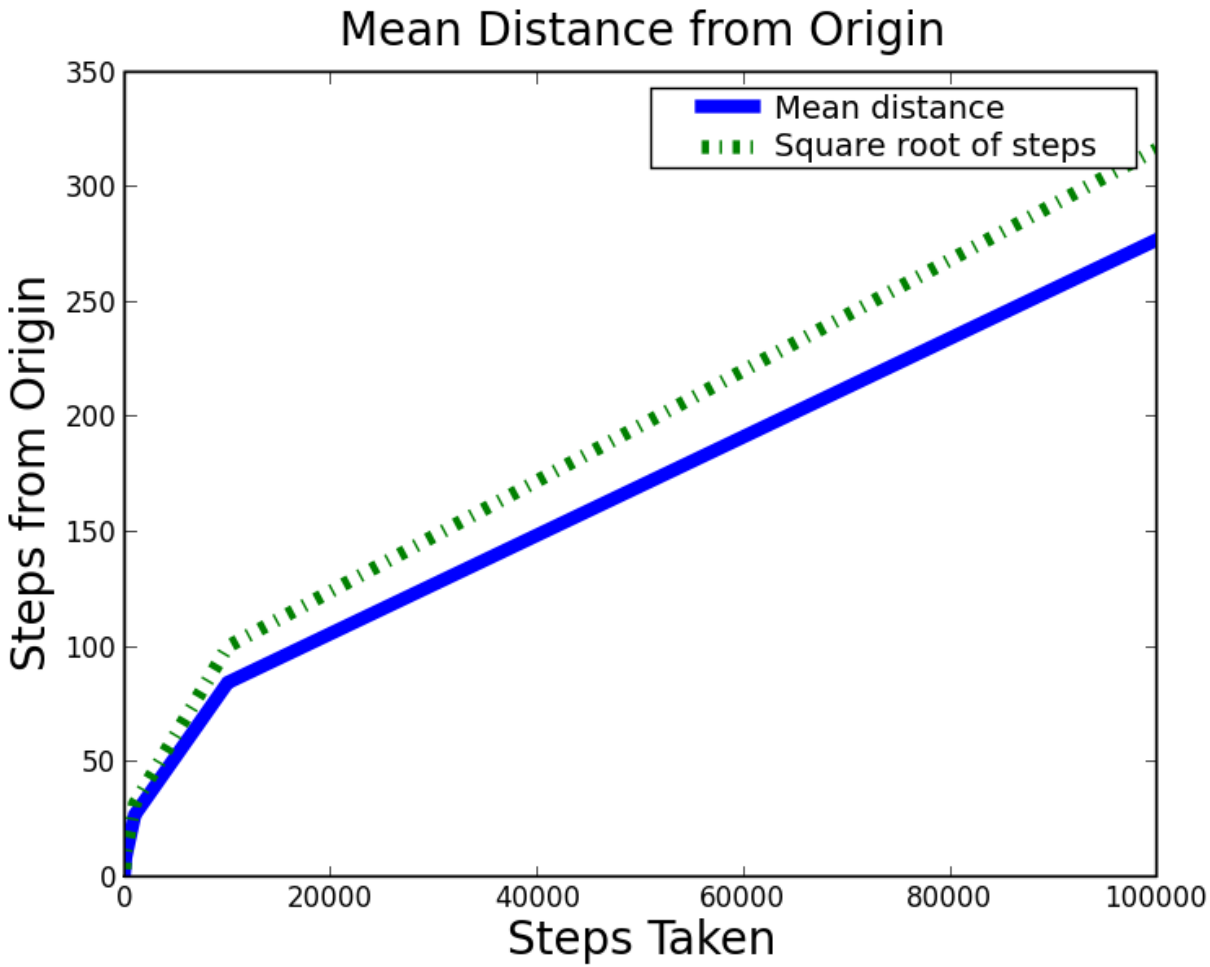
```
import random

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
```

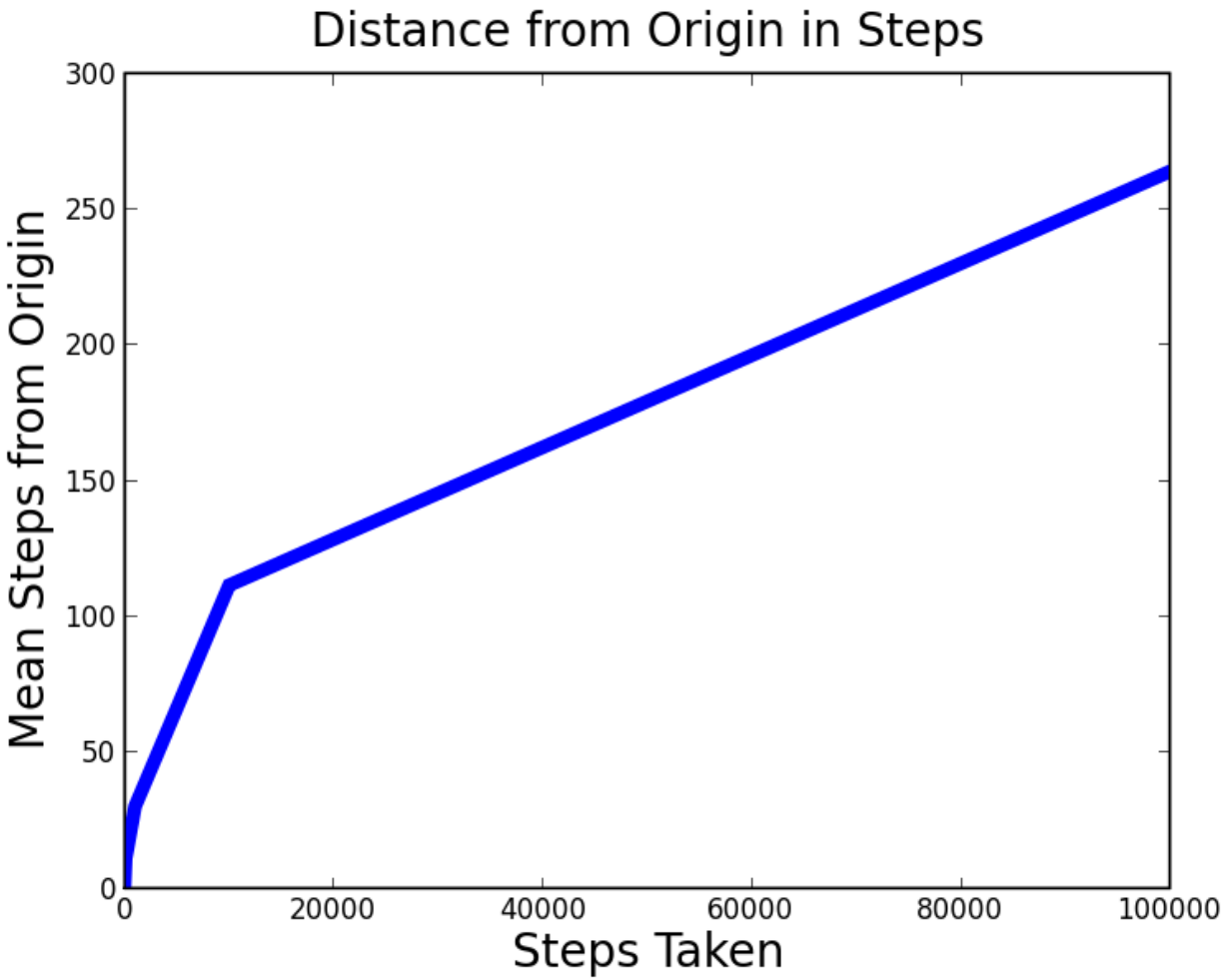
```
def drunkTest(numTrials):
    for numSteps in [10, 100, 1000, 10000, 100000]:
        distances = simWalks(numSteps, numTrials)
        print 'Random walk of ' + str(numSteps) + ' steps'
        print '  Mean =', sum(distances)/len(distances)
        print '  Max =', max(distances), 'Min =', min(distances)
```

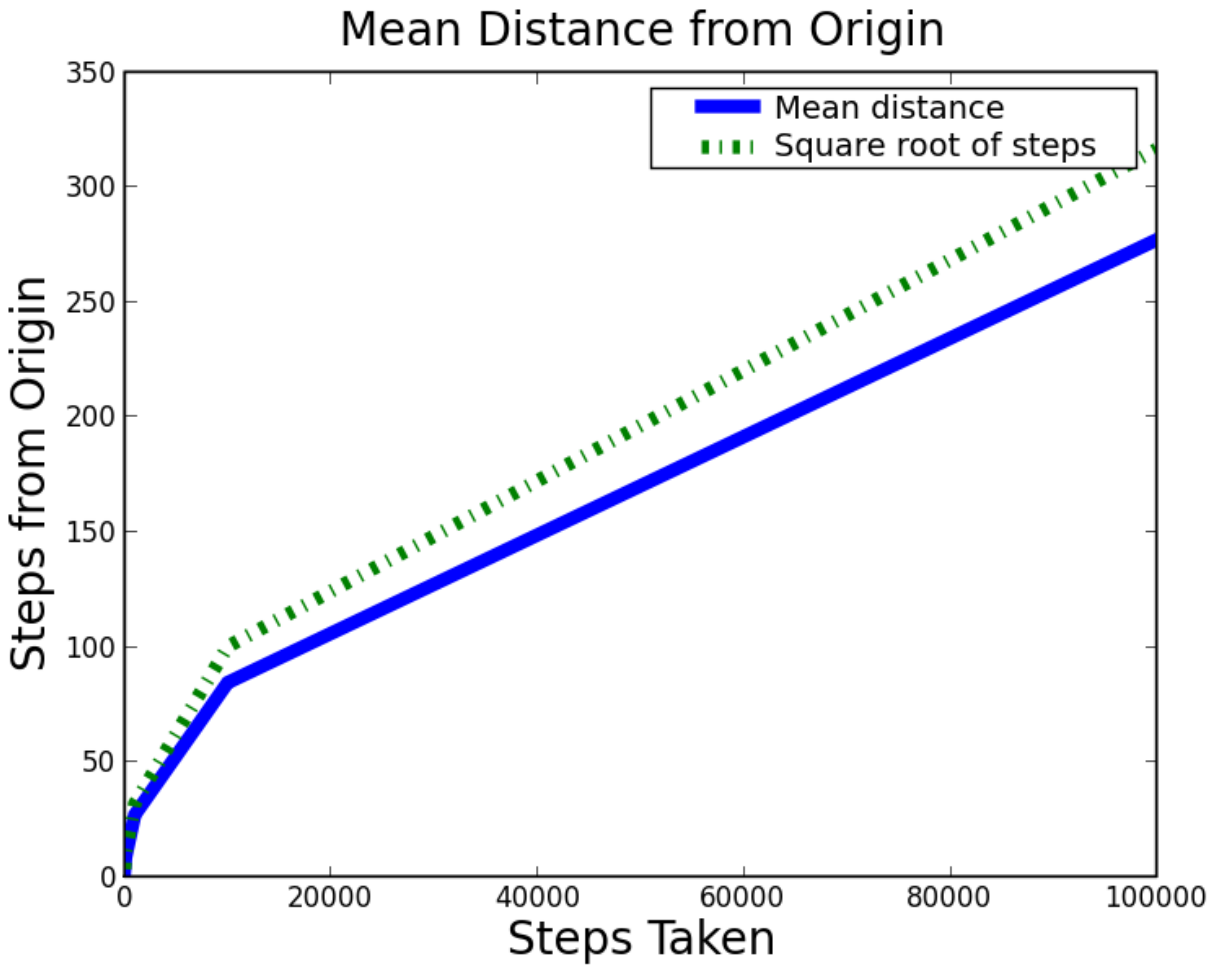

Distance from Origin in Steps
Mean Steps from Origin
Steps Taken
0
50
100
150
200
250
300
0
20000
40000
60000
80000
100000


Mean Distance from Origin
Mean distance
Square root of steps
Steps from Origin
Steps Taken
0
50
100
150
200
250
300
350
0
20000
40000
60000
80000
100000

Random Walks and Simulation Models

Lecturer: John Guttag


Distance from Origin in Steps
Mean Steps from Origin
Steps Taken
0
50
100
150
200
250
300
0
20000
40000
60000
80000
100000


Mean Distance from Origin
Mean distance
Square root of steps
Steps from Origin
Steps Taken
0
50
100
150
200
250
300
350
0
20000
40000
60000
80000
100000

```
class UsualDrunk(Drunk):
    def takeStep(self):
        stepChoices =\
            [(0.0,1.0),(0.0,-1.0),(1.0,0.0),(-1.0,0.0)]
        return random.choice(stepChoices)

class ColdDrunk(Drunk):
    def takeStep(self):
        stepChoices =\
            [(0.0,0.95),(0.0,-1.0),(1.0,0.0),(-1.0,0.0)]
        return random.choice(stepChoices)
```

```
class EDrunk(Drunk):
    def takeStep(self):
        deltaX = random.random()
        if random.random() < 0.5:
            deltaX = -deltaX
        deltaY = random.random()
        if random.random() < 0.5:
            deltaY = -deltaY
        return (deltaX, deltaY)
```

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

```
def drunkTestP(numTrials = 100):
    stepsTaken = [10, 100, 1000, 10000]
    for dClass in (UsualDrunk, ColdDrunk, EDrunk):
        meanDistances = []
        for numSteps in stepsTaken:
            distances = simWalks(numSteps, numTrials, dClass)
            meanDistances.append(sum(distances)/len(distances))
        pylab.plot(stepsTaken, meanDistances,
                           label = dClass.__name__)
    . . .
```