Random Walks and More Plotting, Segment 2

Last Segment

- Presented a related collection of data abstractions
  - Location
  - Field
  - Drunk
    - UsualDrunk
    - ColdDrunk
- Talked about structuring simulations
  - Simulate one walk of k steps
  - Simulate n such walks
  - Report aggregated results.

Simulating a Single Walk

```
def walk(f, d, numSteps):
    """Assumes: f a Field, d a Drunk in f, and
       numSteps an int >= 0.
       Moves d numSteps times; returns the distance
       between the final location and the location
       at the start of the  walk."""
    start = f.getLoc(d)
    for s in range(numSteps):
        f.moveDrunk(d)
    return start.distFrom(f.getLoc(d))
```

Simulating Multiple Walks

```
def simWalks(numSteps, numTrials, dClass):
    """Assumes numSteps an int >= 0, numTrials an
         int > 0, dClass a subclass of Drunk
       Simulates numTrials walks of numSteps steps
         each. Returns a list of the final distances
         for each trial"""
    Homer = dClass()
    origin = Location(0, 0)
    distances = []
    for t in range(numTrials):
        f = Field()
        f.addDrunk(Homer, origin)
        distances.append(round(walk(f, Homer,
                                    numSteps), 1))
    return distances
```

create <-

Putting It All Together

```
def drunkTest(walkLengths, numTrials, dClass):
    """Assumes walkLengths a sequence of ints >= 0
         numTrials an int > 0,
         dClass a subclass of Drunk
       For each number of steps in walkLengths,
         runs simWalks with numTrials walks and
         prints results"""
    for numSteps in walkLengths:
        distances = simWalks(numSteps, numTrials,
                             dClass)
        print(dClass.__name__, 'random walk of',
              numSteps, 'steps')
        print(' Mean =',
              round(sum(distances)/len(distances), 4))
        print(' Max =', max(distances),
              'Min =', min(distances))
```

Let's Try It

```
drunkTest((10, 100, 1000, 10000), 100,
          UsualDrunk)

UsualDrunk random walk of 10 steps
 Mean = 8.634
 Max = 21.6 Min = 1.4
UsualDrunk random walk of 100 steps
 Mean = 8.57
 Max = 22.0 Min = 0.0
UsualDrunk random walk of 1000 steps
 Mean = 9.206
 Max = 21.6 Min = 1.4
UsualDrunk random walk of 10000 steps
 Mean = 8.727
 Max = 23.5 Min = 1.4
```

not changed much

Let's Try a Sanity Check

- Try on cases where we think we know the answer
  - A very important precaution!

Sanity Check

```
drunkTest((0, 1, 2) 100, UsualDrunk)

UsualDrunk random walk of 0 steps
 Mean = 8.634
 Max = 21.6 Min = 1.4
UsualDrunk random walk of 1 steps
 Mean = 8.57
 Max = 22.0 Min = 0.0
UsualDrunk random walk of 2 steps
 Mean = 9.206
 Max = 21.6 Min = 1.4
```

not possible, debugging needed

And the Heat-seeking Drunk?

```
class ColdDrunk(Drunk):
    def takeStep(self):
        stepChoices = [(0.0,0.9), (0.0,-1.1),
                       (1.0, 0.0), (-1.0, 0.0)]
        return random.choice(stepChoices)

def simAll(drunkKinds, walkLengths, numTrials):
    for dClass in drunkKinds:
        drunkTest(walkLengths, numTrials, dClass)

random.seed(0)
simAll((UsualDrunk, ColdDrunk),
       (1, 10, 100, 1000, 10000), 100)
```