LING 408/508: Programming for Linguists

Lecture 25
December 2\textsuperscript{nd}
Last Time

• Monte Carlo Simulation

Player A has an **unfair** advantage!

```python
41 def simOneGame(probA, probB):
42     # Simulates a single game of racquetball between two players whose
43     # abilities are represented by the probability of winning a serve.
44     # Returns final scores for A and B
45     serving = "A"
46     scoreA = 0
47     scoreB = 0
48     while not gameOver(scoreA, scoreB):
49         if serving == "A":
```
100,000 games

```
dhcp-10-132-141-49:ling508-15 sandiway$ python rball.py
This program simulates a game of racquetball between two players called "A" and "B". The abilities of each player is indicated by a probability (a number between 0 and 1) that the player wins the point when serving. Player A always has the first serve.

What is the prob. player A wins a serve? 0.5
What is the prob. player B wins a serve? 0.5
How many games to simulate? 1000000

Games simulated: 100000
Wins for A: 52695 (52.7%)
Wins for B: 47305 (47.3%)
```
1,000,000 games

dhcp-10-132-141-49:ling508-15 sandiway$ python rball.py
This program simulates a game of racquetball between two players called "A" and "B". The abilities of each player is indicated by a probability (a number between 0 and 1) that the player wins the point when serving. Player A always has the first serve.

What is the prob. player A wins a serve? 0.5
What is the prob. player B wins a serve? 0.5
How many games to simulate? 1000000

Games simulated: 1000000
Wins for A: 526524 (52.7%)
Wins for B: 473476 (47.3%)
• Let's make it fair:

def simOneGame(serving, probA, probB):
    # Simulates a single game of racquetball between two players whose
    # abilities are represented by the probability of winning a serve.
    # Returns final scores for A and B
    scoreA = 0
    scoreB = 0
    while not gameOver(scoreA, scoreB):
        if serving == "A":
            if random() < probA:
                scoreA = scoreA + 1
            else:
                serving = "B"
        else:
            if random() < probB:
                scoreB = scoreB + 1
            else:
                serving = "A"
    return scoreA, scoreB
• Let's make it fair:

```python
def simNGames(n, probA, probB):
    # Simulates n games of racquetball between players whose
    # abilities are represented by the probability of winning a serve.
    # Returns number of wins for A and B
    winsA = winsB = 0
    serving = "A"
    for i in range(n):
        scoreA, scoreB = simOneGame(serving, probA, probB)
        if scoreA > scoreB:
            winsA = winsA + 1
        else:
            winsB = winsB + 1
    if serving == "A":
        serving = "B"
    else:
        serving = "A"
    return winsA, winsB
```
This program simulates a game of racquetball between two players called "A" and "B". The abilities of each player is indicated by a probability (a number between 0 and 1) that the player wins the point when serving. Player A always has the first serve.

What is the prob. player A wins a serve? 0.5
What is the prob. player B wins a serve? 0.5
How many games to simulate? 100000

Games simulated: 100000
Wins for A: 49824 (49.8%)
Wins for B: 50176 (50.2%)
1,000,000 games

dhcp-10-142-185-144:ling508-15 sandiway$ python rball2.py
This program simulates a game of racquetball between two players called "A" and "B". The abilities of each player is indicated by a probability (a number between 0 and 1) that the player wins the point when serving. Player A always has the first serve.

What is the prob. player A wins a serve? 0.5
What is the prob. player B wins a serve? 0.5
How many games to simulate? 1000000

Games simulated: 1000000
Wins for A: 499696 (50.0%)
Wins for B: 500304 (50.0%)
Chapter 10: Defining Classes
Chapter 10: MSDie

**Multi-Sided Die:**

```python
# msdie.py
# Class definition for an n-sided die.
from random import randrange

class MSDie:
    def __init__(self, sides):
        self.sides = sides
        self.value = 1

    def roll(self):
        self.value = randrange(1, self.sides+1)

    def getValue(self):
        return self.value

    def setValue(self, value):
        self.value = value
```

---

`self` is just a conventional name for the implicit 1\textsuperscript{st} argument to all methods inside the class: *could call it anything ...*

---

`__init__` is the object constructor.
Chapter 10: MSDie

- From last lecture:
  - library module random

```python
>>> from random import randrange
>>> randrange(1,6)
3
>>> randrange(1,6)
3
>>> randrange(1,6)
5
>>> randrange(1,6)
5
>>> randrange(1,6)
5
>>> randrange(1,6)
1
>>> randrange(1,6)
5
>>> randrange(1,6)
4
>>> randrange(1,6)
2
```
Chapter 10: Projectile

• Cannonball example:
  – *how far does a cannonball fly?*
  – parameters:
    • launch angle
    • height above ground
    • initial velocity
  – gravity: $9.8 \text{ ms}^{-2}$
  – (without calculus) simulation: time interval

*Figure 10.1: Finding the x and y components of velocity.*
Chapter 10: Projectile

```python
# cball1.py
# Simulation of the flight of a cannon ball (or other projectile)
# This version is not modularized.

from math import pi, sin, cos

def main():
    angle = input("Enter the launch angle (in degrees): ")
    vel = input("Enter the initial velocity (in meters/sec): ")
    h0 = input("Enter the initial height (in meters): ")
    time = input("Enter the time interval between position calculations: ")

    radians = (angle * pi)/180.0
    xpos = 0
    ypos = h0
    xvel = vel * cos(radians)
    yvel = vel * sin(radians)
    while ypos >= 0:
        # calculate position and velocity in time seconds
        xpos = xpos + time * xvel
        yvel1 = yvel - time * 9.8
        ypos = ypos + time * (yvel + yvel1)/2.0
        yvel = yvel1

        print "\nDistance traveled: %0.1f meters." % (xpos)

main()
```
Chapter 10: Projectile

- Object-oriented style:
  - assume a Projectile class: method update()

```python
def main():
    angle, vel, h0, time = getInputs()
    cball = Projectile(angle, vel, h0)
    while cball.getY() >= 0:
        cball.update(time)
    print "\nDistance traveled: %0.1f meters." % (cball.getX())
```
Documentation

- Triple quoted documentation strings
  - *can span multiple lines*

```python
# projectile.py

"""projectile.py
Provides a simple class for modeling the flight of projectiles."""

from math import pi, sin, cos

class Projectile:

  """Simulates the flight of simple projectiles near the earth's surface, ignoring wind resistance. Tracking is done in two dimensions, height (y) and distance (x)."""

  def __init__(self, angle, velocity, height):
    """Create a projectile with given launch angle, initial velocity and height."""
```

```python
help(module)
help(module.method)
print(module.method.__doc__)
```
Help on module projectile:

NAME
   projectile

FILE
   /Users/sandhiway/courses/ling508-15/ppitcs_code/chapter10/projectile.py

DESCRIPTION
   projectile.py
   Provides a simple class for modeling the flight of projectiles.

CLASSES
   Projectile

   class Projectile
       Simulates the flight of simple projectiles near the earth's surface, ignoring wind resistance. Tracking is done in two dimensions, height (y) and distance (x).
       Methods defined here:
       __init__(self, angle, velocity, height)
Chapter 10: Projectile

def __init__(self, angle, velocity, height):
    """Create a projectile with given launch angle, initial velocity and height."""
    self.xpos = 0.0
    self.ypos = height
    theta = pi * angle / 180.0
    self.xvel = velocity * cos(theta)
    self.yvel = velocity * sin(theta)
def update(self, time):
    """Update the state of this projectile to move it time seconds
    farther into its flight"
    self.xpos = self.xpos + time * self.xvel
    yvel1 = self.yvel - 9.8 * time
    self.ypos = self.ypos + time * (self.yvel + yvel1) / 2.0
    self.yvel = yvel1

def getY(self):
    """Returns the y position (height) of this projectile."
    return self.ypos

def getX(self):
    """Returns the x position (distance) of this projectile."
    return self.xpos
Chapter 10: Projectile

```python
>>> from projectile import *
>>> p = Projectile(60, 50, 20)
>>> p.update(1)
>>> p.getX()
25.000000000000007
>>> p.getY()
58.40127018922193
>>> p.update(1)
>>> p.getX()
50.000000000000014
>>> p.getY()
87.00254037844385
>>> p.update(1)
>>> p.getX()
75.00000000000003
>>> p.getY()
105.80381056766578
>>> p.update(1)
>>> p.getY()
114.8050807568877
>>> p.update(1)
>>> p.getY()
114.00635094610963
```
Chapter 10: Projectile

```python
from graphics import *
from projectile import *

p = Projectile(60,50,20)  # angle, v, height
listX = []
listY = []

while p.getY() > 0:
    listX.append(p.getX())
    listY.append(p.getY())
    p.update(1)

w = GraphWin()
w.setCoords(0,0,max(listX) + 20,max(listY) + 20)
t = 0
for x,y in zip(listX,listY):
    Point(x+10,y+10).draw(w)
    Text(Point(x+10,5),t).draw(w)
    t = t+1
raw_input(""")
```
```python
import sys
from graphics import *
from projectile import *

if len(sys.argv) != 4:
    print "usage: angle (in degrees), initial velocity, height"
    sys.exit(1)

# angle, v, height
p = Projectile(eval(sys.argv[1]),eval(sys.argv[2]),eval(sys.argv[3]))

listX = []
listY = []
while p.getY() > 0:
    listX.append(p.getX())
    listY.append(p.getY())
    p.update(1)

w = GraphWin()
w.setCoords(0,0,max(listX) + 20,max(listY) + 20)
t = 0
for x,y in zip(listX,listY):
    Point(x+10,y+10).draw(w)
    Text(Point(x+10,5),t).draw(w)
    t = t+1
raw_input(""")
```

sys.argv
- list of command line arguments

sys.exit(1)
- quit with exit code 1