CSC 110: Week 10, Lecture 1

Dr. Laura Bodine

Last week on CSC110…

- Graphics programming
- graphics.py library
  - (based on Tkinter)
- Graphics objects
  - GraphWin, Circle, Line, Point, …
  - Methods!
- Graphics programming as practice with objects and OOP

This week…

- Simulation
  - Random numbers
  - Using computers to solve math problems for us
- Basic design techniques
  - Top-down design
  - Bottom-up design
  - Prototyping and spiral design
- Object-oriented design

Organizational info

- HW: there will be a homework assignment due next week
- Class: there will be class next Tuesday (but not Thursday)
- Final: The exam is next Friday 1-3 pm. Half quiz-like questions, half programming assignment. Programming portion will be open book/note and you will be able to use the shell for testing.

Simulation

- Technique for solving real-world problems with computers

- Useful in situations where:

    1. the number of computations to be performed is too large to do by hand

    2. the equations are too complex to solve exactly and must be solved numerically

    3. large variance/wide range of values must be taken into account (see #1)

Random numbers

- Sources of random numbers?
  - Roll of the dice
  - Flipping a coin
  - *Monte Carlo*

Coin flipping…

- Odds of flipping and getting heads: ½

Coin flipping…

- Odds of flipping and getting heads: ½
- Odds of flipping twice in a row and getting heads both times: ¼

Coin flipping…

- Odds of flipping and getting heads: ½
- Odds of flipping twice in a row and getting heads both times: ¼
- Odds of flipping heads three times in a row: 1/8

Coin flipping…

- Odds of flipping and getting heads: ½
- Odds of flipping twice in a row and getting heads both times: ¼
- Odds of flipping heads three times in a row: 1/8
- Odds of flipping heads three times in a row then tails: 1/16

Coin flipping…

- Odds of flipping and getting heads: ½
- Odds of flipping twice in a row and getting heads both times: ¼
- Odds of flipping heads three times in a row: 1/8
- Odds of flipping heads three times in a row then tails: 1/16
- Does that seem right?

Random numbers

- Sources of random numbers?
  - Roll of the dice
  - Flipping a coin
  - *Monte Carlo*

- Why we want random numbers?
  - Mimic physical processes
  - Test out whether something is random

Random numbers

- Sources of random numbers?
  - Roll of the dice
  - Flipping a coin
  - *Monte Carlo*

- Why we want random numbers?
  - Mimic physical processes
  - Test out whether something is random

- Is it really possible to get random numbers on a computer?

Random numbers, some more ideas…

- Let probability of some outcome be 50%
- Does that something happen every other time?
- What is true about random numbers?
  - Fluctuations
  - Clustering

Pseudo-random number generators

- Python (and most languages) has built-in pseudo-random number generators in a library called **`random`**

- `random()`
  - Has no parameters, covers the interval [0,1)
  - Returns a float

- `randrange()`
  - Selects from a list set by using `range()`
  - `randrange(a)` vs `randrange(a,b)` vs `randrange(a,b,c)`
  - Returns an integer value from the list

Let’s try it out

- Simulate rolling a standard die (single of dice)
- What do you want to come out?
- Which random number generator do we want to select?

Let's try it out

- Simulate rolling a standard die (single of dice)
- What do you want to come out?
  - An integer between 1 and 6
- Which random number generator do we want to select?
  - `randrange`
  - What do we use as parameters?

Let's try it out

- Simulate rolling a standard die (single of dice)

- What do you want to come out?
  - An integer between 1 and 6

- Which random number generator do we want to select?
  - `randrange`
  - What do we use as parameters?
    - Want to start at 1 and count by 1s so we want `randrange(1,b)`
    - Remember that b is not included in the list we select from so b = 6+1 = 7

Let’s try it out

- Simulate rolling a die (single of dice).

- With our randrange(1,7) we can simulate a die roll:

```
from random import random,randrange

roll = randrange(1,7)
```

Let's go a little further…

- How likely are we to roll a 5?
- Mathematically, the answer is 1/6
- Does our "random roll" have the same probability?
  - What does it mean for a finite set to have the same probability?

```
from random import random,randrange

def main(nrolls):
    roll = randrange(1,7)
    n5s = 0
    for i in range(nrolls):
        roll = randrange(1,7)
        if roll == 5: n5s+=1
    print("Probability of rolling a 5 is:", n5s/nrolls)

main(10)
main(10)
main(100)
main(100)
main(1000)
main(1000)
main(10000)
main(10000)
```

Think-pair-share

- What does it mean for a finite set of numbers to come from a certain probability?
- What are the impacts of this on a simulation?
- Are there things we can do to mitigate this?

Let’s look at this with our graphics!

- Start with an empty graphics window
  - Let’s make it 500 x 500 for visibility
- Plot random points into the window
  - Make them red for visibility
- Is there anything noticeable about them?
- What if we repeat this again?

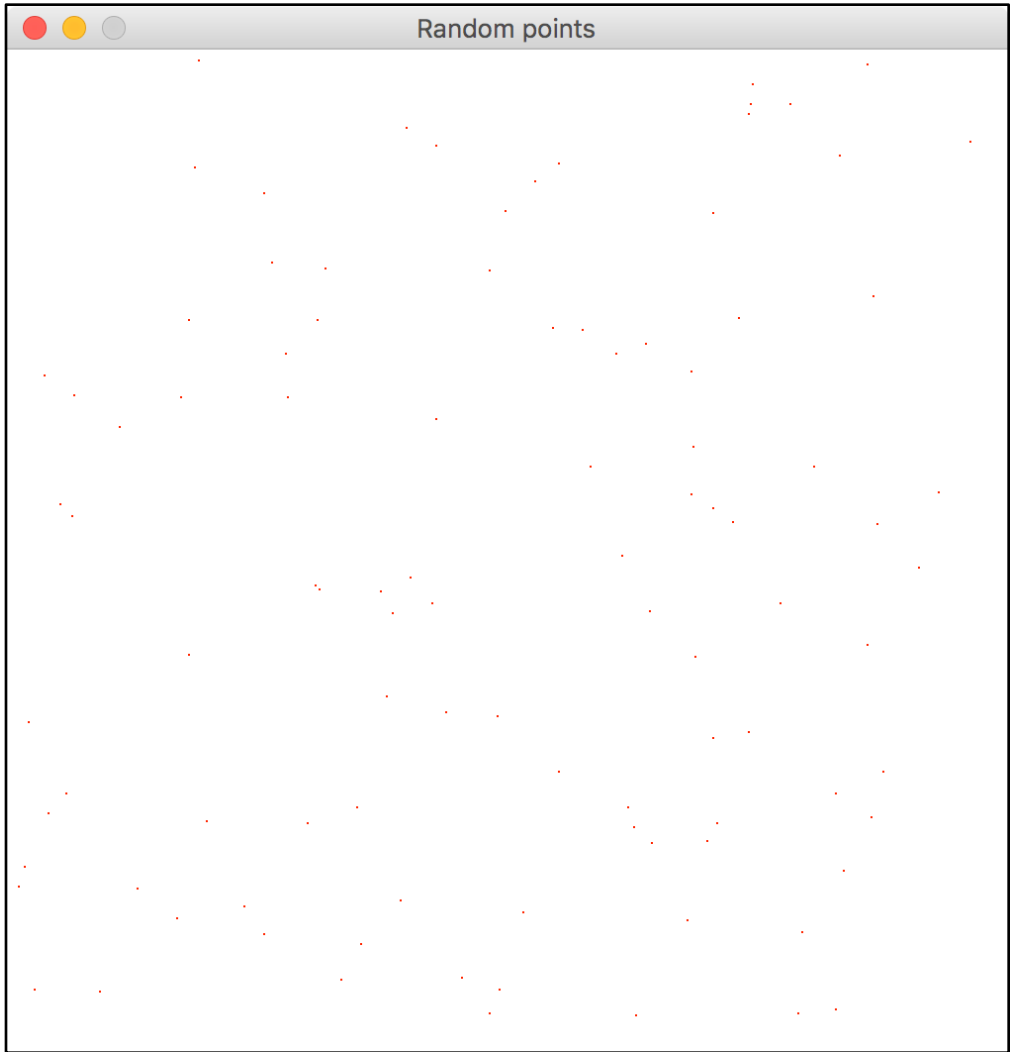
Sample code for random points

```
from graphics import *
import random

def randomplot(nPoints):
    win = GraphWin("Random points",500,500)

    for i in range(nPoints):
        x = random.randrange(500)
        y = random.randrange(500)
        win.plot(x,y,"red")

randomplot(100)
randomplot(1000)
```


Random points


Random points

Notes on random numbers

- Random does not mean uniformly distributed
  - On the contrary for small numbers it is very nonuniform
  - Only tends toward evenness as nPoints-> infinity

- The pseudo-random number generators are used by computers and rely and a seed
  - Python use random seed by default
  - If you want a reproducible results you have specify where to start (seed)

Seeding pseudo-random number generators

- In python, you have to have imported the random module
  - Not from random import random
- The seed is set for the **module**
- `random.seed(<your seed>)`

```
import random
random.seed(10)
random.random()
random.random()
random.random()

random.seed(10)
random.random()
random.random()
random.random()
```

Design schemes

- How do we go about starting to write our program?
- How have you guys done it so far?
- Do we write the whole thing?
- Do we have a way to test it piece by piece?

Think-pair-share

- Think back to the functions assignment involving toNumbers, squareEach and sumList.
- How did you go about writing the program?
- Did you test the individual functions before adding them to main?
- If you had to do it over again, would you change anything about your approach to writing this program?

Multiple approaches to designing programs

- Top-down
- Bottom-up
- Prototype/spiral
- Object-oriented

Top-down design

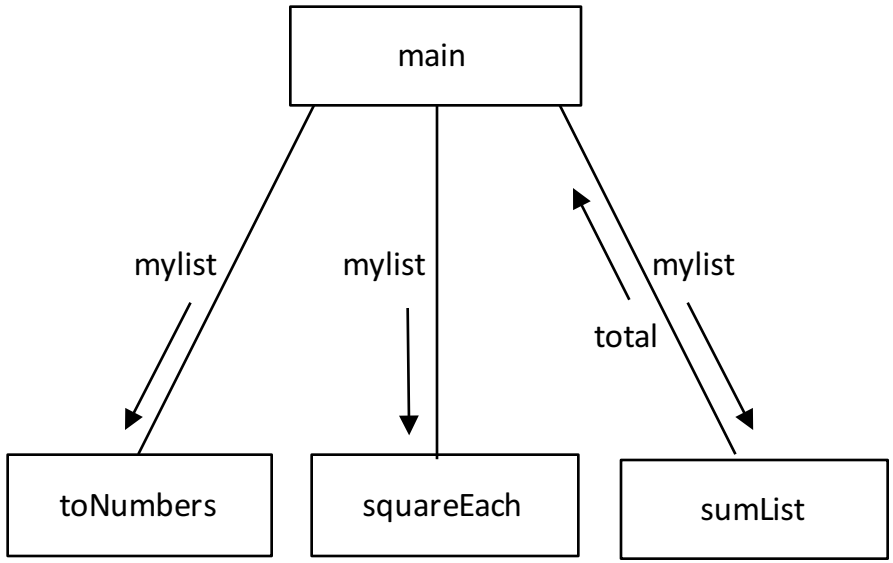
- Start with the inputs and outputs
- Break down tasks into blocks based on operations: input, output, major computations
- Look at each block and decide if you need to break it down further
- Rinse and repeat…

Top-down design

- Separation of concerns or *abstraction*
- Don’t have to worry about details of how something handles the information
- Similar to encapsulation
- Overarching theme of OOP

Top-down design example

- Let's look back at our homework example
- Break down the processes into blocks
- Stay focused on each blocks inputs and outputs
- Look at the structure chart



Design…what to keep in mind

- Multiple approaches to designing software
  - Not a one-size-fits-all kind of deal!
  - Best approach depends on the problem and the developer
- Don’t be afraid to start over if you run into a dead end
  - How do I know it is a dead-end? =
  - Armed with this knowledge you will make different design choices!
- Remember to follow the general idea of abstraction
  - Final product should be high-level overview of what is happening