CS 5 Today

Fractals and Turtles

How random!

Read Sections 3.1-3.5

https://www.youtube.com/watch?v=TfrAf1a9Qhs 1:00-1:30

Photograph of CS 5 co-worker accused of burnination.

CS 5 alien on strike!

CS 5's three-eyed spokesalien has walked off the job, according to an AFL-CIO (Alien Federation of Labor and Congress of Interplanetary Organizations) life-form. "I can't work under these conditions—when I arrived this morning, I was immediately—and indecorously—burninated by a new co-worker," sources reported hearing as the trinocular terrestrial stormed off. No word yet on who this reputed co-worker might be, though... picket lines consumed by flames, p. 42

Some *random* asides...

```
import random                   # allows use of dir(random) and help(random)
from random import *            # all random functions are now available!

choice(L)                       # chooses 1 element from the sequence L
choice('mudd')                  # ...or 1 character from a string
choice(['cmc','scripps','pitzer','pomona'])

list(range(1, 5)) → [1, 2, 3, 4]
```

How would you get a random integer from 0 to 99 inclusive?

```
uniform(low, hi)                chooses a random float from low to hi

>>> uniform(41.9, 42.1)
42.08010107642389
```

floats have about **16** places of precision *Aargh—so close!*

A *random* function...

```
from random import *

def guess(hidden):
    """Tries to guess our "hidden" number
    """
    compguess = choice(list(range(100)))

    if compguess == hidden:   # at last!
        print('I got it!')

    else:
        guess(hidden)
```

Remember, this is [0,1,...,99]

Suspicious? I am!

print the guesses ?
slow down...
return the number of guesses ?
investigate **expected** # of guesses?!??

Monte Carlo in action

How many doubles will you get in **N** rolls of 2 dice?

N is the total number of rolls

```
def countDoubles(N):
    """Argument: the number of dice rolls to make
       Result: the number of doubles seen"""
    if N == 0:
        return 0      # zero rolls, zero doubles...
    else:
        d1 = choice([1, 2, 3, 4, 5, 6])      } two dice from
        d2 = choice(list(range(1, 7)))       } 1-6 inclusive

        if d1 == d2:
            return 1 + countDoubles(N - 1)   # COUNT IT! t
        else:
            return 0 + countDoubles(N - 1)   # don't count it
```

where and how is the check for doubles?

Monte Carlo Monty Hall

Your initial choice!

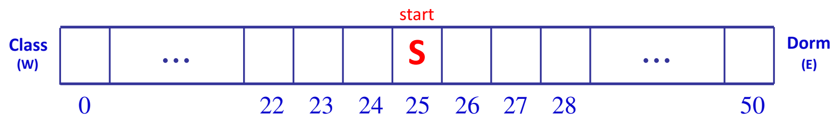
'switch' or 'stay'

number of times to play

```
def MCMH(init, sors, N):
   """Plays the "Let's make a deal" game N times
      Returns the number of times you win the *Spam!*
   """
   if N == 0: return 0           # don't play, can't win
   przDoor = choice([1, 2, 3])  # where the spam (prize) is…

   if   init == przDoor and sors == 'stay':   result = 'Spam!'
   elif init == przDoor and sors == 'switch': result = 'pmfp.'
   elif init != przDoor and sors == 'switch': result = 'Spam!'
   else:                                       result = 'pmfp.'

   print 'You get the', result

   if result == 'Spam!':  return 1 + MCMH(init, sors, N - 1)
   else:                  return 0 + MCMH(init, sors, N - 1)
```

An example *closer to home*

hw2pr2



An overworked 5C student (**S**) leaves H/S after their "late-night" breakfast—or lunch. Each moment, they randomly stumble toward class (**W**) or the dorm (**E**)

Once the student arrives at the dorm or classroom, the trip is complete.
The program should then print the total number of steps taken.

Write a program to model ***and analyze!*** this scenario...

`rwpos(s, nsteps)`

take `nsteps` random steps starting at `s`

`rwsteps(s, low, hi)`

take random steps starting at `s` until you reach either `low` or `hi`

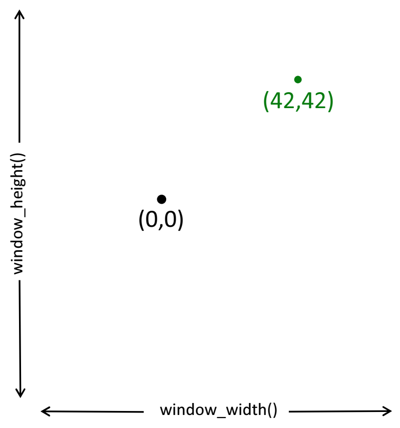
Python's Etch-a-Sketch

```
import time
from turtle import *

def draw():    # define it!
    shape('turtle')
    # pause
    time.sleep(2)
    # drawing...
    width(5)
    left(90)
    forward(50)     pixels!
    right(90)       degrees!
    backward(50)
    down() or up()  is the pen on the "paper"?
    color('darkgreen')
    tracer(1) or tracer(0)
```

sets if the pen animates or not

```
# run it!
reset()
draw()
```

the turtle's canvas



http://docs.python.org/library/turtle.html

Single-path recursion

```
def tri():   # define it!
    """A triangle!
    """
    forward(100)
    left(120)
    forward(100)
    left(120)
    forward(100)
    left(120)

# run now
tri()
```

I don't know about **tri**, but there sure is NO **return** ... !

(1) Let's **tri** this with recursion:

```
def tri(n):
    """Draws a triangle"""
    if n == 0: return
    else:
      forward(100)  # one side
      left(120)     # turn 360/3
      tri(n-1)      # draw rest
```

(2) How about ***any*** regular N-gon?

```
def poly(n, N):
    """Draws a polygon"""
    if n == 0: return
    else:
      forward(100)  # one side
      left(360 / N) # turn 360/N
      poly(n-1, N)  # draw rest
```

Single-path recursion

What does **chai(50)** do here?

```
def chai(dist):
    """Mystery!"""
    if dist < 5:
        return

    forward(dist)
    left(90)
    forward(dist / 2)
    right(90)

    right(90)
    forward(dist)
    left(90)

    left(90)
    forward(dist / 2)
    right(90)
    backward(dist)
```

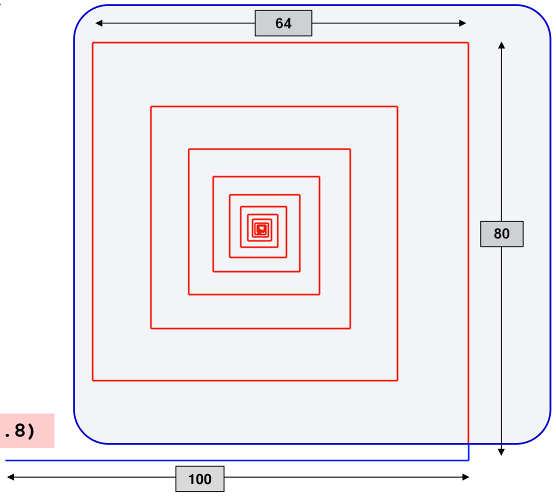
How could you add more to each T's tips?

Why are there two identical commands in a row ~ twice!?

lab ~ hw2pr1

fractal art

spiral(80, 90, 0.8)

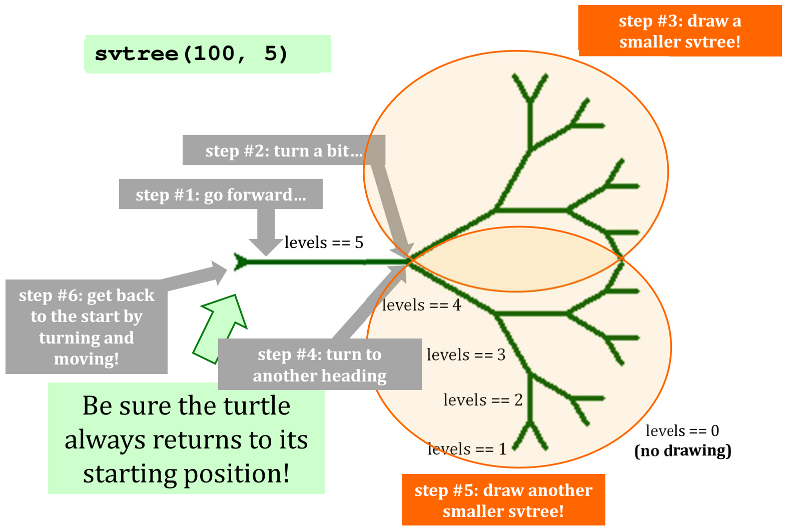


spiral(100, 90, 0.8)

spiral(initLength, angle, multiplier)

svtree(trunkLength, levels)

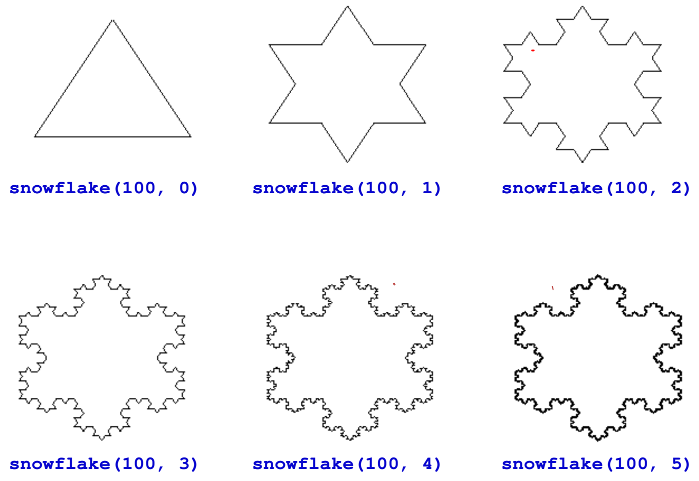
svtree(100, 5)



step #3: draw a smaller svtree!
step #2: turn a bit...
step #1: go forward...
levels == 5
step #6: get back to the start by turning and moving!
levels == 4
step #4: turn to another heading
levels == 3
levels == 2
levels == 1
levels == 0 (no drawing)
step #5: draw another smaller svtree!

Be sure the turtle always returns to its starting position!

The Koch curve



Quiz

Name(s): ____________________

A few *random* thoughts...

```
from random import *
choice([1, 2, 3, 2])
```

What are the chances this returns a 2 ?

```
choice(list(range(1, 5)) + [4, 2, 4, 2])
```

[1, 2, 3, 4]

What are the chances of this returning a 4 ?

Careful!

```
choice('1,2,3,4')
```

What's the most likely return value here?

```
choice(['1,2,3,4'])
```

What's the most likely return value here?

```
choice('[1,2,3,4]')
```

What's the most likely return value here?

```
choice(list(range(5)))
```

[0, 1, 2, 3, 4]

Is this more likely to be even or odd (or same)?

```
uniform(-20.5, 0.5)
```

What're the chances of this being >= 0?

Extra!

and ***how*** likely is each of these?

```
choice(0, 1, 2, 3, 4)

choice([list(range(5))])

choice[list(range(5))]
```

Which **two** of these 3 are *syntax errors*?

Also, what does the ***third*** one—the one syntactically correct—actually *do*?

Syntax corner...