Cyriak: *conceptually disruptive recursion...*

*Fractals and Turtles*

Baaa

CS 5 Today

hw due Monday...

Lots of tutoring...

How random!
My mom painted this and said no one would like it. It's her 2nd painting.

imgur.com/Y0Dg3l...
Recursive painting…

https://nubleh.github.io/i_painted/
Recursion's idea:

```python
def dot(L, K):
    if len(L) == 0 or len(K) == 0:
        return 0.0
    if len(L) != len(K):
        return 0.0
    else:
        return L[0]*K[0] + dot(L[1:], K[1:])
```

You handle the FIRST
Recursion handles the REST

Base Cases

1. The lists aren't empty
2. The lists have the same length

Combine

Handle the FIRST of L
Handle the FIRST of K

Firsts, as appropriate

Handle the REST of L
Handle the REST of K

Recursion w/the rest

First

Rest
```python
def dot(L, K):
    if len(L) == 0 or len(K) == 0:
        return 0.0
    if len(L) != len(K):
        return 0.0
    else:
        return L[0]*K[0] + dot(L[1:], K[1:])
```

```
L = [3, 2, 4] and K = [4, 7, 4]
L = [2, 4] and K = [7, 4]
L = [] and K = []
```

```
dot([[3, 2, 4], [4, 7, 4]])
3*4 + dot([[2, 4], [7, 4]])
2*7 + dot([[4], [4]])
4*4 + dot([], [])
16.0
30.0
42.0
```
There are four different values of L and four different values of K – all alive, simultaneously, in the stack.

Seeing the "stack" ...
elif name == 'Sun':
    print('What? Are you kidding me? Incorrect')
    my_choice = random.choice(['Earth',])
    print('I would choose', my_choice)

elif name == 'Earth':
    print('Well, that is a good choice. Congrats, you win!
    print('WOW', name)
    my_choice = random.choice(['Earth',])
    print('I would choose that too!', my_choice)

elif name == 'Mars':
    print('mmmm, still a bit too desert like. Incorrect')
    my_choice = random.choice(['Earth',])
    print('I would choose', my_choice)

choice2 = input('You decide to walk back to the dining hall you just had dinner at. Which dining hall did you eat at? [mudd/scripps/pomona/pitzer/cmc]
    print()
    if choice2 == 'mudd':
        print("As you enter the dining hall, you remember how traumatically long that pasta line was...at least you had your phone with you to keep you from dying of boredom")
        if choice2 == 'scripps':
            print("Walking past the Motley reminds you of how much you could use some caffeine right now...oh well, you have something more important to do right now")
            if choice2 == 'pomona':
                print("During your walk back to Frary you seriously question your decision to have eaten so far away from your dorm. The walk seems to take ages...")
                if choice2 == 'pitzer':
                    print("As you walk to the dining hall you wonder whether they would notice if you took another dessert on your way out")
                    print("but then you remember you have some cookies left in your dorm...you decide to eat those after you get back")
                    if choice2 == 'cmc':
                        print("As you walk into the dining hall you remember how sad you were because the cold brew dispenser was broken during dinner.")
                        print("You make a mental note to get coffee before you head back.")
                        print()
Some *random* asides...

```python
import random
from random import *
```

allows use of `dir(random)` and `help(random)`

all random functions are now available!

0
```
random.choice(---)
```

2
```
choice(---)
```
Some *random* asides...

```python
import random
from random import *
```

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

```python
choice(L)
```

chooses 1 element from the sequence L

```python
choice('mudd')
```

... or 1 character from a string

```python
choice(['cmc','scripps','pitzer','pomona'])
```

m 1/4
u 1/4
d 1/2
import random
from random import *

allows use of dir(random) and help(random)
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' ])
I want a list of these values

list(range(5))  \rightarrow  [0,1,2,3,4]

list(range(1,5))  \rightarrow  [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 16 places of precision

Aargh – so close!
from random import *

def guess( hidden ):
    #
    # try to guess our "hidden" #
    #
    compguess = choice( list(range(100)) )
    if compguess == hidden: # at last!
        print('I got it!')
    return 1
else:
    guess( hidden )
Recursive guess-counting

```python
from random import *
import time

def guess( hidden ):
    """ guessing game """
    compguess = choice( list(range(100)) )

    # print('I choose', compguess)
    # time.sleep(0.05)

    if compguess == hidden:  # at last!
        # print('I got it!')
        return 1
    else:
        return 1 + guess( hidden )
```

A few random thoughts...

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

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

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

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

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

choice( list(range(5)) )

uniform( -20.5, 0.5 )

choice(0,1,2,3,4)
choice( [list(range(5))] )
choice( list(range(5)) )

What are the chances this returns a 2?
$\frac{2}{4} = \frac{1}{2}$

What are the chances of this returning a 4?
$\frac{3}{9} = \frac{1}{3}$

What's the most likely return value here?

What's the most likely return value here?

What's the most likely return value here?

Is this more likely to be even or odd?

Even

What're the chances of this being > 0? Extra!

Which two of these 3 are syntax errors?

Also, what does the third one – the one syntactically correct – actually do?
```python
from random import *

choice( [1,2,3,2] )  # What are the chances this returns a 2? 2/4 or 50%

choice( list(range(5))+[4,2,4,2] )  # What are the chances of this returning a 4? 3/9

choice( '1,2,3,4' )  # What's the most likely return value here? ', ' 3/7

choice( ['1,2,3,4'] )  # What's the most likely return value here? '1,2,3,4' 1/1

choice( '[1,2,3,4]' )  # What's the most likely return value here? ', ' 3/9

choice( list(range(5)) )  # Is this more likely to be even or odd? even 3/5

uniform( -20.5, 0.5 )  # What're the chances of this being > 0? 1/42

choice([0,1,2,3,4])
choice([list(range(5))])
choice(list(range(5)))
```
from random import *

choice([1,2,3,2])  # What are the chances this returns a 2? 2/4 or 50%

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

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

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

choice([1,2,3,2])

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

choice(list(range(5)))

choice(list(range(5)))+[4,2,4,2]  # What's the most likely return value there?

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

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

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

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

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

choice(list(range(5)))]

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

syntax error: needs [...] or'...

correct: always returns [0,1,2,3,4]

syntax error: needs choice( ...)

1/1 chance

Data is in black. Probabilities are in blue.

Team up and try this on the backpage first…

Pass these in and up!
The two *Monte Carlos* and their denizens...

Monte Carlo casino, *Monaco*

Insights via *random trials*

Monte Carlo methods, *Math/CS*
The two *Monte Carlos* and their denizens...
Monte Carlo in action

How many doubles will you get in \( N \) rolls of 2 dice?

\[
\text{def } \text{countDoubles}( N ):\n\begin{align*}
\text{""" input: the \# of dice rolls to make} \\
\text{output: the \# of doubles seen """} \\
\text{if } N == 0: \\
\quad \text{return } 0 \quad \# \text{ zero rolls, zero doubles...} \\
\text{else:} \\
\quad d1 = \text{choice}( [1,2,3,4,5,6] ) \\
\quad d2 = \text{choice}( \text{list}(\text{range}(1,7)) ) \\
\quad \text{if } d1 \neq d2: \\
\quad \quad \text{return } 0+\text{countDoubles}(N-1) \quad \# \text{ not doubles} \\
\quad \text{else:} \\
\quad \quad \text{return } 1+\text{countDoubles}(N-1) \quad \# \text{ DOUBLES! Add 1}
\end{align*}
\]

\( N \) is the total number of rolls

\[\text{https://repl.it/@JulieMedero/CountDoubles}\]

where and how is the check for doubles being done?
Another Monty... ?
Let's make a deal...

inspiring the Monty Hall paradox

Monty

'63-'86
Monte Carlo Monty Hall

Suppose you always switch to the other door...
What are the chances that you will win the prize?

Let's play (randomly) 300 times and see!
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 )

https://repl.it/@JulieMedero/MCMH
Let's make a deal: XKCD's take...

...what if you considered the goat the grand prize!?
If you win some **SPAM**...? or **pmfp**...?
If you win some **SPAM**... ? or pmfp... ?
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...

```python
rwpos(s, nsteps)
```

* take `nsteps` random steps starting at `s`

```python
rwsteps(s, low, hi)
```

* take random steps starting at `s` until you reach either `low` or `hi`
An example closer to home

An overworked 5C student (S) leaves H/S after their "late-night" breakfast–or lunch. Each moment, they randomly stumble toward class (N) or the Dorm (S). Once the student arrives at the dorm or classroom, the trip is complete. Write a program to model and analyze! this scenario...

Your task: To create this as an "ASCII" animation

```
rwpos(s, nsteps)
```

`rwpos(s, nsteps)`

- `take nsteps random steps starting at s`

```
rwsteps(s, low, hi)
```

`rwsteps(s, low, hi)`

- `take random steps starting at s until you reach either low or hi`
Lab!  *Python's Etch-a-Sketch*
Lab! Python's Etch-a-Sketch

No way this is real... but it is!

www.gvetchedintime.com
more *usual* etch-a-sketch work...
Python's ability? It varies...
Python's ability? It varies...
```python
import time
from turtle import *

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

# run it!
draw(); done()
```

http://docs.python.org/library/turtle.html
Turtle happiness?

*some* Pythons need

```python
done()
```

after turtle drawing!

This releases control of the turtle window to the computer (the operating system)
Terminator error!

Problem:  Terminator Error

Solution:  Just run it again!
**Single-path recursion**

Let's **tri** this with recursion:

```python
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?

```python
def poly(n, N):
    """ n sides of an N-gon""
    if n == 0: return
    else:
        forward(100)  # one side
        left(360.0/N) # turn 360/N
        poly(n-1, N)  # draw rest
```

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

1. What would `chai(100)` draw?

   ```python
def chai(dist):
    """ mystery fn! """
    if dist < 5: return
    forward(dist)
    left(90)
    forward(dist/2.0)
    right(90)
    right(90)
    forward(dist)
    left(90)
    left(90)
    forward(dist/2.0)
    right(90)
    backward(dist)
```

2. Have `rwalk` draw a "stock-market" path of \(N\) steps of 10 pixels each. *Use recursion.*

   ```python
   from random import *

   def rwalk(N):
    """ make \(N\) 10-pixel steps, NE or SE ""
    if N == 0: return
    elif choice(['left','right']) == 'left':
        left(45)
        forward(10)
    else: # this handles 'right'
        #

   Extra! How could you make this a bull (or a bear) market?
   
   Extra #2! What if the line `chai(dist/2)` were placed between the two right(90) lines? And/or between the two left(90) lines?
from random import *

def rwalk(N):
    """ make N 10-px steps, NE or SE ""
    if N == 0:    return

elif choice(['left','right'])=='left':
    left(45)
    forward(10)
    right(45)
    rwalk( N-1 )

else:    # 'right'
    right(45)
    forward(10)
    left(45)
    rwalk( N-1 )

rwalk(N) is a random "stock market" walk...

What if we *didn't* turn back to face east each time?

"Single-path" (or *counting*) recursion
def chai(dist):
    """ mystery! ""
    if dist < 5:
        return
    forward(dist)
    left(90)
    forward(dist / 2.0)
    right(90)
    right(90)
    forward(dist)
    left(90)
    left(90)
    forward(dist / 2.0)
    right(90)
    backward(dist)

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

Why are there two identical commands in a row ~ twice!?
def chai(dist):
    """ mystery! ""
    if dist < 5:
        return
    forward(dist)
    left(90)
    forward(dist/2.0)
    right(90)
    chai(dist/2)
    right(90)
    forward(dist)
    left(90)
    left(90)
    forward(dist/2.0)
    right(90)
    backward(dist)

Now, what does chai(100) do?
Cyriak: *conceptually disruptive* recursion...

is the *branching*, not the *single-path* variety.
fractal art

$\text{spiral}(100, 90, 0.8)$

$\text{spiral}(\text{initLength, angle, multiplier})$
lab ~ hw2pr1

fractal art

spiral\((100,90,0.8)\)

spiral\((80,90,0.8)\)

spiral\((100,90,0.8)\)

spiral\((initLength, angle, multiplier)\)
svtree( trunkLength, levels )

svtree( 100, 5 )

levels == 5
levels == 4
levels == 3
levels == 2
levels == 1
levels == 0
(no drawing)
svtree( trunkLength, levels )

svtree( 100, 5 )

levels == 5

svtree( 75, 4 )

levels == 4

levels == 3

levels == 2

levels == 1

levels == 0
(no drawing)

What steps does the turtle need to take before recursing?
Be sure the turtle always returns to its starting position!

svtree( trunkLength, levels )

svtree( 100, 5 )

step #1: go forward...

levels == 5

levels == 4

levels == 3

levels == 2

levels == 1

step #4: turn to another heading

levels == 0
(no drawing)

step #3: draw a smaller svtree!

step #6: get back to the start by turning and moving!

step #2: turn a bit...

step #5: draw another smaller svtree!
Be sure the turtle always returns to its starting position!

That means it will finish the **recursive call** right here! so that it can change heading and draw another one...

svtree( trunkLength, levels )

svtree( 100, 5 )

svtree( 75, 4 )

levels == 5

levels == 4

levels == 3

levels == 2

levels == 1

levels == 0

(no drawing)
The Koch curve

snowflake(100, 0)  snowflake(100, 1)  snowflake(100, 2)
snowflake(100, 3)  snowflake(100, 4)  snowflake(100, 5)
Recursive art? Create your own...

Happy turtling in lab!

seven-cornered confetti

What? This is too happy to be art... My recursive compositions burninate even Cyriak’s brain!