Cyriak: conceptually disruptive recursion...

Baaa

Fractals and Turtles

CS 5 alien on strike!

CS 5 green mascot representing today's terrestrial theme

CS 5 Today

hw3 due Monday...

Lots of tutoring...

How random!
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

You handle the FIRST
Recursion handles the REST

Firsts, as appropriate

Handle the FIRST of L
Handle the FIRST of K

Recursion w/the rest

Handle the REST of L
Handle the REST of K

First

Rest
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:])

dot([[3, 2, 4], [4, 7, 4]])
3*4 + dot([[2, 4], [7, 4]])
2*7 + dot([[4], [4]])
4*4 + dot([[]], [[]])
16.0 + 0.0
30.0
42.0
There are four different values of L and four different values of K – all alive, simultaneously, in the stack.
Using a Euro, Yuan the deadline comes next time I’ll try to make the Mark. I’m Pounding it into my brain, but I have to Peso much attention to other deadlines in the classes I Currency have - but if this keeps up, my grade will be demolished to Ruble.

```python
if user == comp:
    print('TIE!')
elif user == 'flight' and comp == 'invisibility':
    print('It\'s harder for villains to find me when I\'m invisible, but they saw you try to fly away. I win this one.')
elif user == 'invisibility' and comp == 'flight':
    print('It\'s harder for villains to find you when you\'re invisible, but they saw me try to fly away. You win this one.')
elif user == 'flight' and comp == 'super-strength':
    print('You\'re much more agile when you fly then I am with my super-strength. You win this one.')
elif user == 'super-strength' and comp == 'flight':
    print('I\'m much more agile when I fly then you are with your super-strength. I win this one.')
elif user == 'super-strength' and comp == 'invisibility':
    print('Your super-strength overpowers my invisibility! You win this one.')
elif user == 'invisibility' and comp == 'super-strength':
    print('My super-strength overpowers your invisibility! I win this one.')

user = input("Pick your CMS player!")
comp = random.choice(['CMC', 'Harvey Mudd', 'Scripps'])

username = input("Welcome to Scripps, what is your name? ")

print()
print("Welcome," + username + ",")
print()"It's your first week here at Scripps College, and you must navigate the core without getting lost. Good luck."
print()
print("You wake up and it's time for you to head over to the first Core. You hear many people say that "had morning classes. You don't exactly know anybody yet so you wonder if they have a lab today."
print("You walk down the stairs of Browning and manage to find yourself at the Humanities Building."
print("towards the Humanities Building or walk towards Toll...")
print()
reponse = input("So, where do you go? [humanities/toll] ")
```
import random
from random import *

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

```python
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' ])
```
Some *random* asides...

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

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

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

chooses 1 element from the sequence L

... or 1 character from a string

```python
list(range(5))  # [0,1,2,3,4]
list(range(1,5))  # [1,2,3,4]
```

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

```python
choice(range(100))
choice(list(range(100)))
```

chooses a random float from low to hi

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

floats have 16 places of precision

_Aargh – so close!_
from random import *

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

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

# remember, this is [0,1,...,99]
Recursive guess-counting

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 )
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)) )
choice( list(range(5)) )
choice(0, 1, 2, 3, 4)
choice([list(range(5))])
choice[list(range(5))]
uniform( -20.5, 0.5 )
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] )  # 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)  # syntax error: needs [...] or '...'

choice([[list(range(5))]])  # correct: always returns [0,1,2,3,4]

choice[list(range(5))]  # syntax error: needs choice( ...)
from random import *

c = [1, 2, 3, 2]
choice(c)  # What are the chances this returns a 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([1, 2, 3, 2])
choice(list(range(5)))

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

Is this more likely to be even or odd?

What are the chances of this returning a 4?

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 casino, *Monaco*

Bond, James Bond

Monte Carlo methods, *Math/CS*

Ulam, Stan Ulam
def countDoubles( N):
    """ input: the # of dice rolls to make
    output: the # of doubles seen """
    if N == 0:
        return 0  # zero rolls, zero doubles...
    else:
        d1 = choice( [1,2,3,4,5,6] )
        d2 = choice( list(range(1,7)) )
        if d1 != d2:
            return 0+countDoubles( N-1 )  # not doubles
        else:
            return 1+countDoubles( N-1 )  # DOUBLES! Add 1

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

inspiring the Monty Hall paradox
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!
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 )
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 \textit{and analyze!} this scenario...

\begin{align*}
\text{rwpos}(s, \text{nsteps}) & \quad \text{rwsteps}(s, \text{low}, \text{hi}) \\
\text{take nsteps random steps starting at } s & \quad \text{take random steps starting at } s \text{ until you reach either low or hi}
\end{align*}
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 \((S)\)

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

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

Your task: To create this as an "ASCII" animation...
Lab!  *Python's Etch-a-Sketch*
Lab!  Python's Etch-a-Sketch

No way this is real... but it is!
more *usual* etch-a-sketch work...
Python's ability? It varies...
Python's ability? It varies...
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 `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

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

(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):
    """ 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` ...!
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)

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

(1) What would chai(100) draw?

Be the turtle!

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

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.
**lab ~ hw3pr1**

fractal art

```
spiral(100, 90, 0.8)
```

```
spiral(initLength, angle, multiplier)
```
lab ~ hw2pr1

fractal art

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

\text{spiral}(80, 90, 0.8)

\text{spiral}(\text{initLength}, \text{angle}, \text{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 )

svtree( 75, 4 )

What steps does the turtle need to take before recursing?

levels == 5

levels == 4

levels == 3

levels == 2

levels == 1

levels == 0
(no drawing)
svtree( trunkLength, levels )

Be sure the turtle always returns to its starting position!

step #1: go forward...

step #2: turn a bit...

step #3: draw a smaller svtree!

step #4: turn to another heading

step #5: draw another smaller svtree!

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

svtree( 100, 5 )
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...

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!