Cyriak: *conceptually disruptive* recursion...

Fractals and Turtles

CS 5 alien on strike!

CS 5 green mascot representing today’s terrestrial theme

Baaa

CS 5 Today

hw3 due Monday...

Lots of tutoring...

COWS & COWS & COWS

How random!

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

Handle the FIRST of L
Handle the FIRST of K
Combine

Handle the FIRST of L
Handle the FIRST of K
Recursion w/the rest

Base Cases

There are four different values of L and four different values of K - all alive, simultaneously, in the stack

Seeing the "stack" ...

```
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]

3*4 + dot ([2,4], [7,4])

2*7 + dot ([4], [4])

4*4 + dot ([], [])

16.0

30.0

42.0

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

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


L = [] and K = []
import random
from random import *

Some random asides…

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

A random function…

from random import *

def guess( hidden ):
    """ tries to guess our "hidden" # ""
    compguess = choice( list(range(100)) )

    if compguess == hidden: # at last!
        print('I got it!')
    else:
        guess( hidden )
The two *Monte Carlos* and their denizens...

Monte Carlo casino, Monaco

Monte Carlo methods, Math/CS

Insights via random trials

Monte Carlo in action

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

\[
\text{N} \text{ is the total number of rolls}
\]

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

How are these the two dice?

where and how is the check for doubles being done?

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

\[
\text{def MCMH( init, sors, N, ) :}
\]
\["""
plays the "Let's make a deal" game \( N \) times
returns the number of times you win the *Spam!*
""
\]
\[
\text{if } N == 0: \text{return 0} \quad \# \text{ don't play, can't win}
\]
\[
\text{przDoor} = \text{choice([1,2,3])} \quad \# \text{ where the spam (prize) is...}
\]
\[
\begin{cases}
\text{if } \text{init} == \text{przDoor} \text{ and } \text{sors} == "'\text{stay}'": \text{result} = "'\text{Spam}'!
\text{elif } \text{init} == \text{przDoor} \text{ and } \text{sors} == "'\text{switch}'": \text{result} = "'\text{pmfp}'
\text{elif } \text{init} != \text{przDoor} \text{ and } \text{sors} == "'\text{switch}'": \text{result} = "'\text{Spam}'
\text{else:}
\end{cases}
\]
\[
\text{print 'You get the', result}
\]
\[
\text{if result == 'Spam!' :} \text{return 1 + MCMH( init, sors, N-1 )}
\]
\[
\text{else:} \text{return 0 + MCMH( init, sors, N-1 )}
\]
Let's make a deal: XKCD's take...

**Monty Hall**

...and my yard has so much grass, and I'll teach you tricks, and...

A few minutes later, the goat from behind door C drives away in the car.

...what if you considered the goat the grand prize!??
Lab! Python's Etch-a-Sketch

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 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

(1) Let's tri this with recursion:

```python
def tri(n):
    """ draws a triangle """
    if n == 0:
        return
    else:
        forward(100)
        left(120)
        tri(n-1)
```

(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)
        left(360.0/N)
        poly(n-1, N)
```

Be the turtle!

(1) What would chai(100) draw?

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

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

"Single-path" (or counting) recursion

Single-path 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?

Branching recursion

Now, what does chai(100) do?

lab ~ hw3pr1

fractal art

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)

spiral(100,90,0.8)

call( initLength, angle, multiplier )
**lab** ~ hw2pr1

- fractal art

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

```
snowflake(100, k)
```

W-ACM!

Meetings beginning now ~ (Julia, Daksha, Autumn, ARIELLE!)
Recursive art?  Create your own...

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

Happy turtling in lab!
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([1,2,3,2])
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?
What are the chances of this returning a 4?
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?
What're the chances of this being > 0?

Which two of these 3 are syntax errors?
Also, what does the third one – the one syntactically correct – actually do?