ICGE Programming Module—Python!

PYTHON! YOU'RE FLYING! HOW? https://xkcd.com/35 I DUNNO ... DYNAMIC TYPING? I JUST TYPED import antigravity WHITESPACE? THAT'S IT? COME JOIN US! PROGRAMMING ... I ALSO SAMPLED I LEARNED IT LAST 15 FUN AGAIN! EVERYTHING IN THE NIGHT! EVERYTHING IT'S A WHOLE MEDICINE CABINET 15 SO SIMPLE! NEW WORLD FOR COMPARISON. UP HERE! HELLO WORLD IS JUST print "Hello, world!" BUT I THINK THIS BUT HOW ARE IS THE PYTHON. ω YOU FLYING?

Why Python?

- Freely available for all types of computers
- Widely used grade schools to universities to industry
- Powerful object-oriented language, with great built-in types
- Easy to create GUIs and interface to internet
- Fun! Python does a lot of the hard work for you (unlike C++)

		Language Rank	Types	Spectrum Ranking
My toolkit		1. Python	\bigoplus \Box	100.0
		2. C		99.7
		3. Java		99.5
		4. C++		97.1
		5. C#		87.7
		6. R	-	87.7
		7. JavaScript	\oplus []	85.6
		8. PHP	\oplus	81.2
		9. Go	\bigoplus \Box	75.1
		10. Swift		73.7

If you're going to be using computers in your research take the time to find the toolkit that works right for you

Python is distributed with a simple integrated development environment (IDE) called *idle*

When you start idle it opens and interactive shell window:

Python 2.7.10 Shell		×
<u>File Edit Shell Debug Options Window H</u> elp		
<pre>Elle Edit Shell Debug Options Window Help Python 2.7.10 (default, May 23 2015, 09:40:32) [MSC v.1500 32 bit (Intel)] on Type "copyright", "credits" or "license()" for more information. >>></pre>	ı win32	
		-
	Ln: :	5 COI: 4

Just like R, Matlab or other interpreted languages, you can interactively run Python commands in the shell

```
print "Hello world"
                                 >>> math.
print("Hello world")
                                                      .......
                                           acos.
                                           acosh
7*1.3
                                           asin
                         Typing prompts
                                           lasinh
import math
                                           atan
                                           atan2
math.
                                           latanh
math.sqrt(16*144)
                                           ceil
                                           copysign
from math import sqrt
                                           COS
sqrt(1024)
                                    >>> math.sqrt(
                                                 sqrt(x)
for i in range(1,11):
      y = i*i
```

print y

```
completeness of its built-in data types (objects)
Lists are a general data container:
 alist=[1, 9, 3, 7, "a"]
                                     >>> alist.
 alist.count(3)
                                             append
                                             count
 alist.pop()
                                             extend
 alist.sort()
                                             index
                                             insert
 alist.reverse()
                                             pop
                                             remove
Lists can be "multimensional":
                                             reverse.
                                             sort
 blist=[[1,2],[3,4]]
 blist
 blist[0][1]
 blist[1]
 list2d=[[0 for i in range(10)] for j in range(10)]
```

A key strength of Python is the power and

The idle environment provides an editor to enter Python code without immediately executing it

Python 2.7.10 Shell		
<u>File Edit Shell Debug Options Window H</u> elp		
Python 2.7.10 (default, May 23 2015	5, 09:40:32) [MSC v.1500 32 bit (1	[ntel)] on win32 🔟
Type "copyright", "credits" or "lio >>>	cense()" for more information.	
	👌 Untitled	
=	<u>File Edit Format Run Options Window H</u> elp	
		<u>^</u>
		In:1 Col: 0

The idle environment provides an editor to enter Python code without immediately executing it

Enter the following code into the editor box

```
from random import choice
ndice=5
ngames=10
for i in range(ngames):
    dice=[]
    for j in range(ndice):
        dice.append(choice(range(1,7)))
    dice.sort()
    lowtwo=dice[0]+dice[1]
    print lowtwo
```

Then select the "Run/Run module" menu option Save the program as "dice.py"

There's a great online learning tool for Python and other languages lets you visualize code execution

- 1. Open up the web browser and go to the website: http://www.pythontutor.com/
- 2. Click on the link entitled "Start visualizing your code now!"
- 3. Copy and paste the **dice.py** program from your idle window
- 4. Click the button entitled "Visualize Execution" below the editor box
- 5. Click the "Forward" button for each step in the program to run
- 6. As each variable is created or updated, you'll see that space next to the program.
- 7. Output will appear in a box below the editor box
- 8. To run to completion, click the "Last" button"
- 9. To modify the program, click the "Edit Code" link

You can load existing Python programs into idle and run them. Let's look at a few simulation codes

	_										A Python 2.7.10 Shell	-	. Passificati		
Python 2.7.10 Shell											<u>File Edit Shell Debug Options Wi</u>	ndow <u>H</u> elp			
File Edit Shell Del	bug Options	Window	v Help								<pre>susp_tit_for_tat:</pre>	10	16.1330	10	
		 []+	Marr	22	2015	00.40	. 221	IMCC	1 E	00	grudger:	0	0.0000	0	• •
New File Ctu	U.V.CHUN	uiu,	May	23	2015,	09:40:	32)	IMSC	V.15	00	Concration 27	Dop	Saoro	Novi Dop	ind hy
New File Cu	IN+X CUN+IN	cred	its"	or	"lice	nse()"	for	more	info	rma	defect.	0	0 0000	New Pop	
Open Ctr	rl+X Ctrl+F										cooperate:	ő	0.0000	0	
Recent Files	•	I									tit for tat:	26	47.4600	26	• • •
Open Module Ctr	rl+X Ctrl+M	I									coin_flip:	15	27.1616	15	
Class Browser Ctr	rl+X Ctrl+B										<pre>biased_random:</pre>	24	42.9736	24	
Path Browser											<pre>susp_tit_for_tat:</pre>	10	16.1544	10	
		1									grudger:	0	0.0000	0	
Save Ctr	rl+X Ctrl+S	1									Concration- 28	Dop	Caoro	Note Dop	
Save As Ctr	rl+X Ctrl+W)									defect.	0	0 0000	New Pop	
Save Copy As Ctr	rl+X Ctrl+V	1									cooperate:	ŏ	0.0000	ő	
											tit for tat:	26	47.4897	26	
Print Window Ctr	rl+X Ctrl+P	1									coin_flip:	15	27.2124	15	
											biased_random:	24	42.9816	24	
Close Ctr	rl+X Ctrl+0	1									<pre>susp_tit_for_tat:</pre>	10	16.1070	10	
Exit Ctr	rl+X Ctrl+C	I									grudger:	0	0.0000	0	
		1									Concration= 39	Pop	Score	Now Dop	
											defect.	0	0 0000	New Pop	
											cooperate:	ŏ	0.0000	ŏ	
											tit for tat:	26	47.4456	26	
									ſ	ED an at	coin flin:	15	27,1649	15	
🔏 2D Molecular Dynamics Simulato	or							- 0	×	er ryth	on 2.7.10 Shell				
Run Potential Form Dynamics Se	Setup Analysis									File Ed	int Shell Debug Options Windo	v нер	2 0015 00	. 40 . 20) 5	100 - 1500 20 hit (Tot)
Properties		_						Number of a	atoms	Pyth	on 2.7.10 (default,	May 2	3 2015, 09	:40:32) []	$ASC V.1500 32 BIC (Inc \square$
		<u> </u>		($\overline{}$)			ET)]	Un win32			()	na information
Zam	n n					\sim		Box Size		туре	"Copyright", "Cred	uts" c	r "license	()" IOF mc	
LUII		r y	\bigcirc			<u> </u>		4.0		>>>			RE:	START ====	
	J		\smile			\circ	1	Interaction et	reporth	=====	===				
			6	_	00	\cup \cup)	1.0	engui	>>>			• •	•	
Distribution		_)	00	_				Boar	a		inh	$+ \land$	aut by
24-)	\sim		Mass		0	0 0 0 0			15	
								20		0	1 1 1 0	•		· · · _	
16-								Temperatur	e	0	1 1 1 0				
		\sim		<u></u>	0			1.0		0	1 1 1 0				
8-		$\bigcirc \bigcirc$	\bigcirc)	\bigcirc (\sim	_			0	0 0 0 0				
						-	\bigcirc	Time Step		No s	olutions found for	1 flip	s		
0.0 0.5 1.0 1.5	5 2.0		$\bigcirc \bigcirc$		(2.0		No s	olutions found for	2 flip	S		
			ŏ	0		<u> </u>		Periodic Bo	oundary	No s	olutions found for	3 flip	S		
0.5-				VX V				I Trace Sel	lected	No s	olutions found for	4 flip	S		
		~			-	0 0				No s	olutions found for	5 flip	s		
					\sim)					-			
-					(
			11.1		Outformin 14										
0.0			Centr	ersity of er for Cor	california, Merc mputational Bio	cea logy		RUN Simul	auon						
0.0 0.5 1.0 1.5	5 2:0							STOP Simu	lation						

Monte Carlo simulations use random numbers to statistically sample different outcomes

A simple use of Monte Carlo simulation is to calculate the relative area of a region:



What's the ratio of the area of the circle to the area of the square?

πR^2	_π_	Darts in circle
(2R) ²	4	Darts in square

 $\pi = 4 \times \frac{\text{Darts in circle}}{\text{Darts in square}}$

Here's a simple Python program to simulate this process using virtual darts

```
import random
import math
inside=0
trials=1000
for i in range(trials):
                            Indentation matters!
    x=random.random()
    y=random.random()
    if (x*x+y*y)<1.0:
        inside+=1
pi=4.*float(inside)/float(trials)
print "N=%d Error=%8.5f "%(trials,pi-math.pi)
```

Enter into an idle editing window and then save as "**pi.py**"

Many problems in physics, chemistry and biology essentially boil down to "random walks"



Simplest question to ask—does the walker ever get back home?

Python program for 1-D random walk

Note double underscores " "

```
from _______ import division
from random import choice
trials=1000
steps=1000
gothome=0
for i in range(trials):
    point=0
    for step in range(steps):
        point+=choice((-1,1))
        if point==0:
            gothome+=1
            break
print "Fraction that got home=%f" % (gothome/trials)
```

Save this as "rwalk1d.py" and run for different numbers of steps

The problem gets more interesting if the walker moves in 2 or more dimensions



Note that if we randomly change both x & y coordinates by -1 or +1, the walker moves diagonally like a checkers piece.

Python program for arbitrary-D random walk making diagonal moves at every step

```
from future import division
from random import choice
dim=3
trials=1000
steps=1000
gothome=0
for i in range(trials):
    point=[0]*dim
    for step in range(steps):
        for j in range(dim):
            point[j]+=choice((-1,1))
        if point.count(0)==dim:
            gothome+=1
            break
print "Fract that got home=%f in %d dims" % (gothome/trials,dim)
```

Save this as "rwalknd.py" and run for different dimensions

For what #'s of dimensions does the walker make it home?

Exact results for an infinite number of steps moving in only one dimension at a time

Dimensions	Prob(get home)
1	1.000
2	1.000
3	0.341
4	0.193
5	0.135
6	0.105
7	0.086
8	0.073

(2,-2)	(2,-1)	(2,0)	(1,1)	(2,2)
(1,-2)	(1,-1)	(1,0)	(1,1)	(1,2)
(0,-2)	(0,-1)	於 (0,0)	(0,1)	(0,2)
(-1,-2)	(-1,-1)	(-1,0)	(1,1)	2,2)
(-2,-2)	(-2,-1)	(-2,0)	-2,1)	(-2,2)

Possible questions to explore:

- 1. How well does rwalknd.py agree with the exact results?
- 2. Do your results match better if you modify the program to step in only one dimension at a time?

Exact results from http://mathworld.wolfram.com/PolyasRandomWalkConstants.html

Another type of random walk simulation tests the effect of barriers on trapping a walker in space



Simulating percolation is different from random walks because we have to store the barrier locations

The 2D list object we saw previously is a good data structure for storing the barrier locations



grid =
$$[[1, 0, 0, 1, 0, 1, 0],$$

 $[0, 1, 0, 0, 1, 0, 1],$
 $[0, 0, 0, 1, 0, 0, 0],$
 $[1, 0, 0, 0, 0, 0, 0],$
 $[0, 0, 1, 0, 0, 0, 1],$
 $[1, 0, 0, 0, 1, 0, 0],$

Our random walker will move around "inside" this grid

The first half of the program sets up the simulation and builds a grid filled with barriers at a set density

Open up the program perc.py inside idle

```
from future import division
from random import choice, random
density=0.5 #Should be a number between 0 and 1
side=21
           #Should be an odd number
perc=0
trials=1000
maxtime=1000
steps = [(1,0), (-1,0), (0,1), (0,-1)]
for trial in range(trials):
    count=0
    for x in range(side):
        for y in range(side):
                                         "Non Pythonic"
            grid[x][y]=0
            if (random()<density):
                count+=1
                grid[x][y]=1
```

grid=[[0 if random() > density else 1 for x in range(side)] for y in range(side)]

The second half is similar to the random walk, but we need to test for barriers and when the walker leaves

```
# Start particle at center
    x,y = int(side/2), int(side/2)
    for time in range(maxtime):
        # Randomly move particle
        sx,sy = choice(steps)
        nx = x + sx
        ny = y + sy
        # if new position is occupied try again
        if grid[nx][ny]==1:
            continue
        if nx==0 or ny==0 or nx==(side-1) or ny==(side-1):
            perc+=1
            break
        grid[x][y]=0 #Remove particle from current spot
        grid[nx][ny]=1  #Put particle in new location
        x=nx
        y=ny
print "%4d %5.3f %5.3f"%(side,density,perc/trials)
```

Some additional Python learning resources

Posted to CatCourses:

- ThinkPython (book)
- Python_quick_reference
- Python_refcard
- Python2.7 Reference

Python classes available at all online course sites:

- Edx
- Coursera
- Software carpentry
- Udacity
- Datacamp

Other WWW resources:

- http://www.codecademy.com/tracks/python
- http://www.afterhoursprogramming.com/tutorial/Python/Overview/
- http://www.stavros.io/tutorials/python/ ("Learn Python in 10 Minutes")
- http://learnpython.org
- http://www.linuxtopia.org/online_books/programming_books/introduction_to_python/
- http://wiki.python.org/moin/BeginnersGuide

If you want a more intensive programming introduction, consider the Scientific Programming Workshop Sponsored

Instructors: Mike Colvin and David Quint Schedule: 8:30-12:30 for 10 days (June 4-15) Location: UC Merced campus, SE1-100

Topics:

- Linux/Bash scripting
- Using remote compute servers
- R programming and data analysis
- Python programming
- C/C++ [Optional dep. on interest]

Sponsored by the UC Merced Center for Cellular and Biomolecular Machines



