Loop of life, XKCD's take:

**Challenge:**
use Hmm code somewhere in your first *Writ1* or *SpecRel* assignment...

**Next Tues**
Guest Lecture ~ Tues.
Prof. Julie Medero!

and this, before watches – or glasses...
Loop of life, Prof. Medero's take:

and this, before watches – or glasses...

enge:
use Hmmm code somewhere in your rst Writ1 or SpecRel assignment...

Guest Lecture ~ Tues.
Prof. Julie Medero!
CS 5 Today

indefinitely nested structure...

from finitely nested loops

CS Midterm

Thursday, Nov. 7

In-class, written

Page of notes is OK

• Recursion in Python
• Function composition
• Circuit design
• Hmmm assembly code
• Loops in Python

Topics

See online practice...

Homework 8  Loops!  due Mon. 11/4

Accommodations...
Homework 8 preview

When Algorithms Discriminate...

#0

#1 ~ lab

#2

#3

#4

(Extra)

The Mandelbrot Set

Lots of loops!

Pi from Pie

TTS Securities

ASCII Art

Thinking in Loops...
Algorithms ~ better angels... ?

#0 When Algorithms Discriminate...

#1 ~ lab

#2

#3

(Extra)

Can an Algorithm Hire Better Than a Human?

Claire Cain Miller @clairecm JUNE 25, 2015

Hiring and recruiting might seem like some of the least likely jobs to be automated. The whole process seems to need human skills that computers lack, like making conversation and reading social cues.

But people have biases and predilections. They make hiring decisions, often unconsciously, based on similarities that have nothing to do with the job requirements — like whether an applicant has a friend in common, went to the same school or likes the same sports.
Or ~ louder angels of human nature...?

When Algorithms Discriminate...

The online world is shaped by forces beyond our control, determining the stories we read on Facebook, the people we meet on OkCupid and the search results we see on Google. Big data is used to make decisions about health care, employment, housing, education and policing.

But can computer programs be discriminatory?
Homework 8 preview

When Algorithms Discriminate...

#0

#1 ~ lab

#2

#3

#4

(Extra)

The Mandelbrot Set

Lots of loops!

Pi from Pie

TTS Securities

ASCII Art

Thinking in Loops!
PythonBat loop practice...

google for "PythonBat" then...

CodingBat code practice

String-2 > double_char

Given a string, return a string where for every char in the original, there are two chars.

def double_char(str):
    result = 
    for c in str:
        result += 2*c
    return result

Our Solution:

5 points required, up to 11 points available...
What are the design differences between these two types of Python loops?
**HOW TO BREATHE EASIER**

1. Inhale deeply through your nose.
2. Hold your breath for 5 long seconds counting “1-Mississippi, 2-Mississippi, etc.”
3. **DO NOT EXHALE.** Breathe in another short breath and hold for 5 long seconds.
4. **DO NOT EXHALE.** Repeat one more time.
5. Exhale SLOWLY to a slow count of 10 long seconds.
6. Repeat the whole sequence until you feel the stress, anger or frustration exit your body.

**MORE TIPS?**

www.cuc.claremont.edu/heo/balance

find your balance!

---

Loop design...

Is this a for or a while loop?

Careful here!
HOW TO BREATHE EASIER

1 Inhale deeply through your nose.
2 Hold your breath for 5 long seconds counting “1-Mississippi, 2-Mississippi, etc.”
3 DO NOT EXHALE. Breathe in another short breath and hold for 5 long seconds.
4 DO NOT EXHALE. Repeat one more time.
5 Exhale SLOWLY to a slow count of 10 long seconds.
6 Repeat the whole sequence until you feel the stress, anger or frustration exit your body.

MORE TIPS?
www.cuc.claremont.edu/hee/balance

find your balance!

Lather. Rinse. Repeat.

Loop design...

Is this a for or a while loop?
Thinking in *loops*

**for**
- *definite iteration*
  - For a **known** list or # of iterations

**while**
- *indefinite iteration*
  - For an **unknown** number of iterations
Homework 8 preview

- When Algorithms Discriminate...
- The Mandelbrot Set
- Lots of loops!
- Pi from Pie
- TTS Securities
- ASCII Art
- CSS: Cascading Style Sheets
- (Extra)
- (Web extra)

Loopy thinking
Pi from Pie?

Pizza is the universal constant, after all...
Pi from Pie?

This couldn't be just a coincidence!
Estimating $\pi$ from pie?

What if we just throw darts at this picture?
**Pi-design challenge...**

**Estimating \( \pi \) from pie?**

1. Suppose you throw 100 darts at the square *(All of them hit the square)*

2. Suppose 80 of the 100 hit inside the circle.

3. How could you estimate \( \pi \) from these throws?

**Hints**

How big is a *side* of the square?  Its area?

How big is the *radius* of the circle?  Its area?

*How do these help!?*
Estimating $\pi$ from pie?

$\frac{\pi}{4} = \frac{\text{area}}{\text{box area}}$

$\pi \sim \frac{4 \times \text{hits}}{\text{box hits}}$

Pi-design challenge...
Loops: *for* or *while*?

\[ \text{pi\_one}(e) \quad \text{e == how close to } \pi \text{ we need to get} \]

\[ \text{pi\_two}(n) \quad \text{n == number of darts to throw} \]

*Which function will use which kind of loop?*
Loops: **for** or **while**?

`pi_one(e)` **while**

`pi_two(n)` **for**

\( e == \text{how close to } \pi \text{ we need to get} \)

\( n == \text{number of darts to throw} \)
π day!

3/14/15 9:26:53
Homework 8 preview

- #0: When Algorithms Discriminate...
- #1 ~ lab
- #2
- #3
- #4
- (Extra)

- The Mandelbrot Set
- Lots of loops!
- Pi from Pie
- TTS Securities
- ASCII Art

Not just loops... Nested loops
Nested loops are familiar, too!

```python
for mn in range(60):
    for s in range(60):
        tick()
```
Nested loops are familiar, too!

So close!
Nested loops

for y in range(84):
    for m in range(12):
        for d in range(f(m,y)):
            for h in range(24):
                for mn in range(60):
                    for s in range(60):
                        tick()
Nested loops!

How nested loops can feel...

```python
for mm in range(60):
    for s in list
        for h in range(24):
            for d in range(f(m,y)):
                for m in range(12):
                    for y in range(84):
                        tick()
```
for mn in range(60):
    for s in range(60):
        tick()

hour()
Creating 2d structure ~ in ASCII

```python
for row in range(3):
    for col in range(4):
        print("#")
```

Wait! this needs something more...
Creating 2d structure

for row in range(3):
    for col in range(4):
        print("#", end='')

Hmmm...
Creating 2d structure

```
[0,1,2]
for row in range(3):
    [0,1,2,3]
    for col in range(4):
        print('#', end='')
    print()
```

row =
    col =
    col =
    col =

row =
    col =
    col =
    col =

row =
    col =
    col =
    col =

Creating 2d structure

```python
for row in range(3):
    for col in range(4):
        if col == row:
            print('#', end='')
        else:
            print(' ', end='')
        print()
```

Let's take an alien's-eye view!

```
row = 0
col = 0
col = 1
col = 2
col = 3

row = 1
col = 0
col = 1
col = 2
col = 3

row = 2
col = 0
col = 1
col = 2
col = 3
```
**Match!**

What code creates the fourth one?

* and ** are extra!

**Name(s) __________________________**
for r in range(3):
    for c in range(6):
        if c >= r:
            print('#', end='')
        else:
            print(' ', end='')
    print()
Match!

What code creates the fourth one?

* and ** are extra!

Pass those...

ashington-ward

houston-ward

for r in range(3):
    for c in range(6):
        if c+r<=4 and c>=r:
            print('#', end='')
        else:
            print(' ', end='')
    print()
for d in range(f(m)):
    for m in range(1, 13):
        num_bdays(m, d)

What trends appear in this birthday data?

How might we be suspicious of the fairness of this data?!

Data represents the # of babies born in the United States between 1973 and 1999.

how many shared birthdays are in CS5?

vizwiz.blogspot.com/2012/05/how-common-is-your-birthday-find-out.html
Nested loops: from ASCII Art

That's my *type* of alien!

... to "*real*" images!
Python and images

```python
from cs5png import *

inputs are width and height

im = PNGImage( 300, 200 )

im.plotPixel( 10, 100 )
```

(0,0) is in the usual place!
Python and images

```
from cs5PNG import *

im = PNGImage(300, 200)

im.plotPixel(10, 100)

im.plotPixel(42, 42, (255, 0, 0))

im.saveFile()
```

**objects** are variables that can contain their own functions, often called **methods**

These functions are clearly **plotting something** – if only I knew what they were up to...
from cs5png import *

def testImage():
    """ image demonstration """
    WD = 300
    HT = 200
    im = PNGImage( WD, HT )

    for row in range(HT):
        for col in range(WD):
            if col == row:
                im.plotPoint( col, row )

    im.saveFile()
Complex #s!

\[ \sqrt{-1} = i \]

i can’t believe this!

1j * 1j == -1

(-1+0j)
Complex #s!

\[ \sqrt{-1} = i \]

```
In[]: c = -2+1j
```

```
In[]: c**2
```

```
In[]: c**2
```

```
\((-2+1j) * (-2+1j)\)
```

```
1j * 1j == -1
```

i can't believe this!
Complex #s!

In[]: `c = -2+1j`

In[]: `c**2 (3-4j)`
Consider an *update rule* for all complex numbers $c$

\[
z_0 = 0 \\
z_{n+1} = z_n^2 + c
\]

\[
c = 0.3 + 0.4j
\]

```python
z = z**2 + c ; print(z)
```
Consider an **update rule** for all complex numbers $c$

$$z_0 = 0$$

$$z_{n+1} = z_n^2 + c$$

Small values of $c$ keep the sequence near the origin, $0+0j$.

Some "stick around" ~ oscillate or converge

\[ z = z^2 + c ; \text{print}(z) \]
Consider an *update rule* for all complex numbers $c$

$$z_0 = 0$$

$$z_{n+1} = z_n^2 + c$$

Small values of $c$ keep the sequence near the origin, 0+0j.

Other values of $c$ make the sequence head to infinity.

- $c = 3 - 4j$
- $c = 0.3 + 0.4j$
Mandelbrot Definition

Consider an **update rule** for all complex numbers \(c\)

\[
\begin{align*}
    z_0 &= 0 \\
    z_{n+1} &= z_n^2 + c
\end{align*}
\]

Small values of \(c\) keep the sequence near the origin, 0+0j.

\(c = 3 - 4j\)

Which \(c\)'s stick around?

Other values of \(c\) make the sequence head to infinity.

\(c = .3 + .4j\)
Lab 8: the Mandelbrot Set

Consider an update rule for all complex numbers \( c \)

\[
    z_0 = 0 \\
    z_{n+1} = z_n^2 + c
\]

Click to choose \( c \).

\( c \) is \(-1.21368948247 + -0.16290726817 * 1j\)

- iter # 0 : \( z = 0.0 + 0.0 * 1j \)
- iter # 1 : \( z = -1.21368948247 + -0.16290726817 * 1j \)
- iter # 2 : \( z = 0.23281389367 + 0.232530407823 * 1j \)
- iter # 3 : \( z = -1.21355756129 + -0.0546346462374 * 1j \)
- iter # 4 : \( z = 0.256047527535 + -0.0303026920702 * 1j \)
- iter # 5 : \( z = -1.14904739926 + -0.17842516935 * 1j \)
- iter # 6 : \( z = 0.0747849173552 + -0.1377964 * 1j \)
- iter # 7 : \( z = -1.26917670088 * 1j \)
- iter # 8 : \( z = 0.075956431 * 1j \)
- iter # 9 : \( z = 0.05608 * 1j \)
- iter # 10 : \( z = -0.11606194429 * 1j \)
- iter # 11 : \( z = -0.12555732355 * 1j \)
- iter # 12 : \( z = 0.0493841573866 + -0.019832902025 * 1j \)
- iter # 13 : \( z = -1.21164403147 + -0.160948405863 * 1j \)
- iter # 14 : \( z = 0.228487387181 + 0.227117082506 * 1j \)

some \( c \)’s stick around
Lab 8: the Mandelbrot Set

Consider an update rule for all complex numbers $c$

$$z_0 = 0$$
$$z_{n+1} = z_n^2 + c$$
Mandelbrot Set ~ points that stick around

The shaded area are points that do not diverge for \( z = z^2 + c \)
Higher-resolution M. Set

The black pixels are points that do not diverge for \( z = z^{**2} + c \)

connected

finite area

\( \infty \) perimeter!
Complex things always consisted of simple parts...
Before the M. Set, complex things were made of simple parts:

Chaos!

This was a "naturally occurring" object where zooming uncovers more detail, not less:

not self-similar but quasi-self-similar

http://www.youtube.com/watch?v=0jGaio87u3A
The black pixels are points that do not diverge for $z = z^{**2} + c$.
Numbers in yellow indicate the number of dendrites or spiral arms found in each region, and in the corresponding Julia fractals for each region.
In the Seahorse Valley....
Happy Mandelbrotting!

www.cs.hmc.edu/~jgrasel/projects

http://www.youtube.com/watch?v=0jGaio87u3A