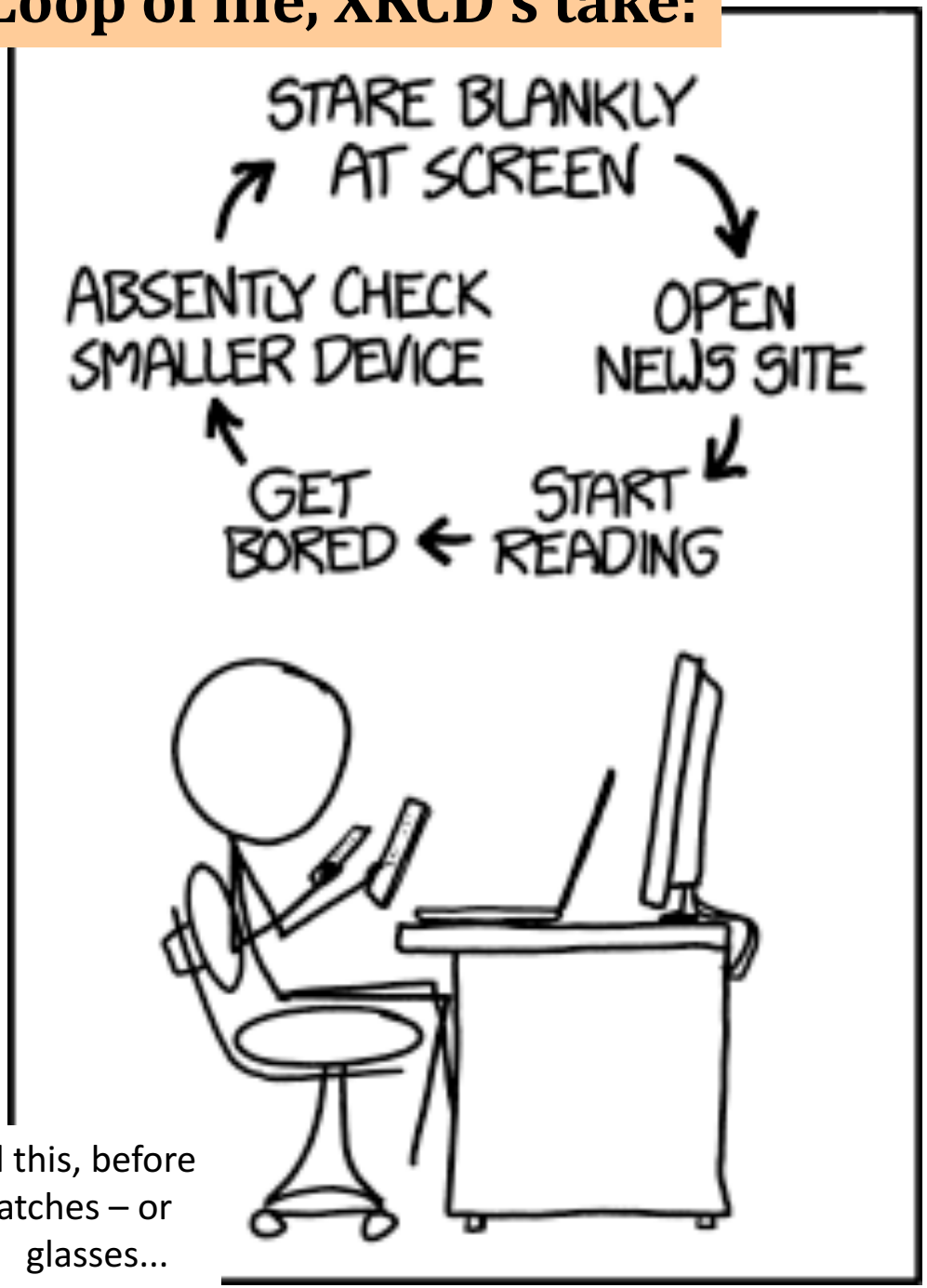
Loop of life, XKCD's take:



and this, before watches – or glasses...

Challenge:

use Hmmm code somewhere in your first **Writ1** or **SpecRel** assignment…

Next Tues

Guest Lecture ~ Tues.
Prof. Julie Medero!

Loop of life, Prof. Medero's take:

enge:

use Hmmm code somewhere in your rst **Writ1** or **SpecRel** assignment…

t Tues

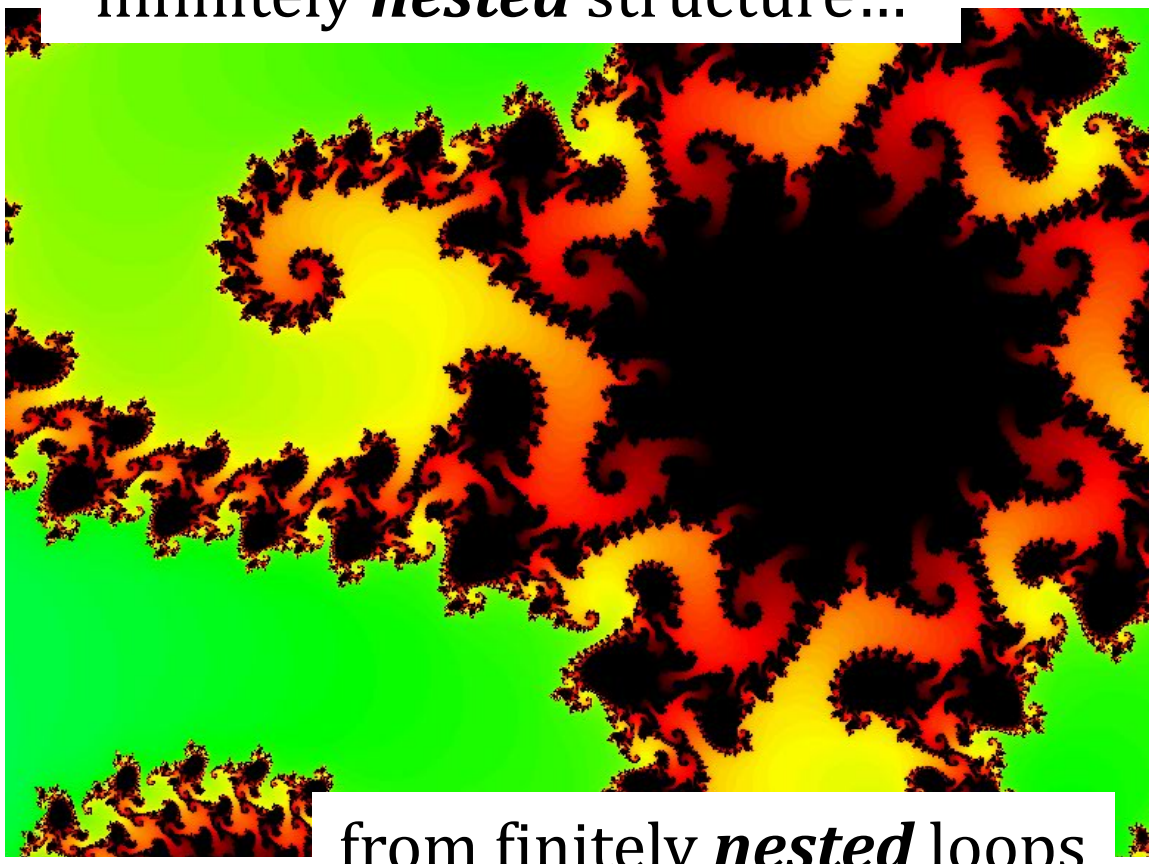
uest Lecture ~ Tues. Prof. Julie Medero!

and this, before watches – or glasses...

CS 5 Today

Seems more like two-thirds term to me...

infinitely ***nested*** structure…



from finitely ***nested*** loops

Homework 8 ***Loops!*** due Mon. 11/4

CS Midterm

Thursday, Nov. 7

In-class, written

Page of notes is OK

Topics

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

See online practice…

Accommodations...

Homework 8 preview

#0 *When Algorithms Discriminate...*

#1 ~ lab The Mandelbrot Set

#2 *Lots of loops!*

#3 *Pi from Pie*

#4 TTS Securities

(Extra) ASCII Art

Thinking in Loops...

Algorithms ~ better angels... ?

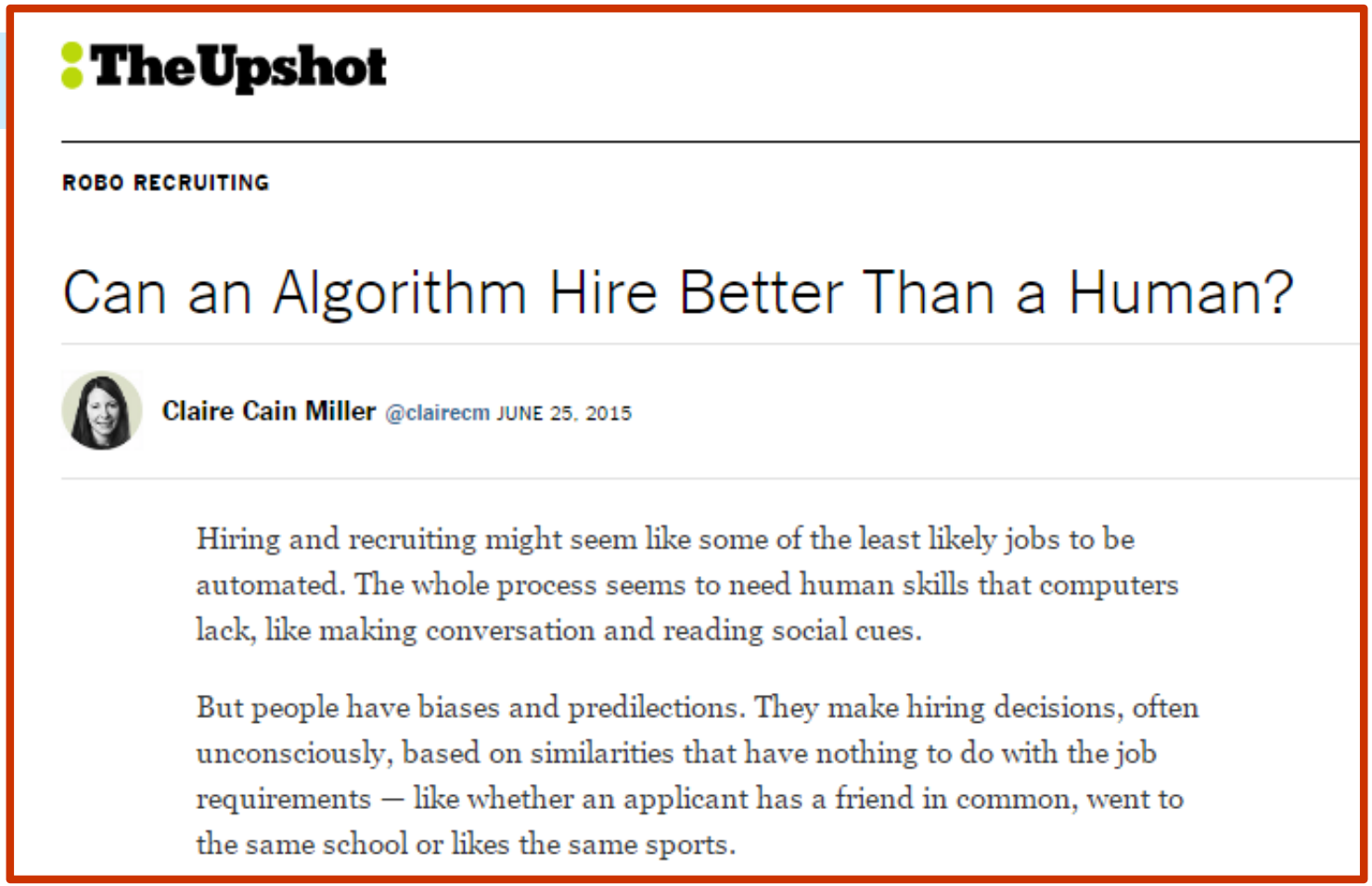
#0 *When Algorithms Discriminate...*

#1 ~ lab

#2

#3

(Extra)



The Upshot

ROBO RECRUITING

Can an Algorithm Hire Better Than a Human?

Claire Cain Miller @clairecm JUNE 25, 2015

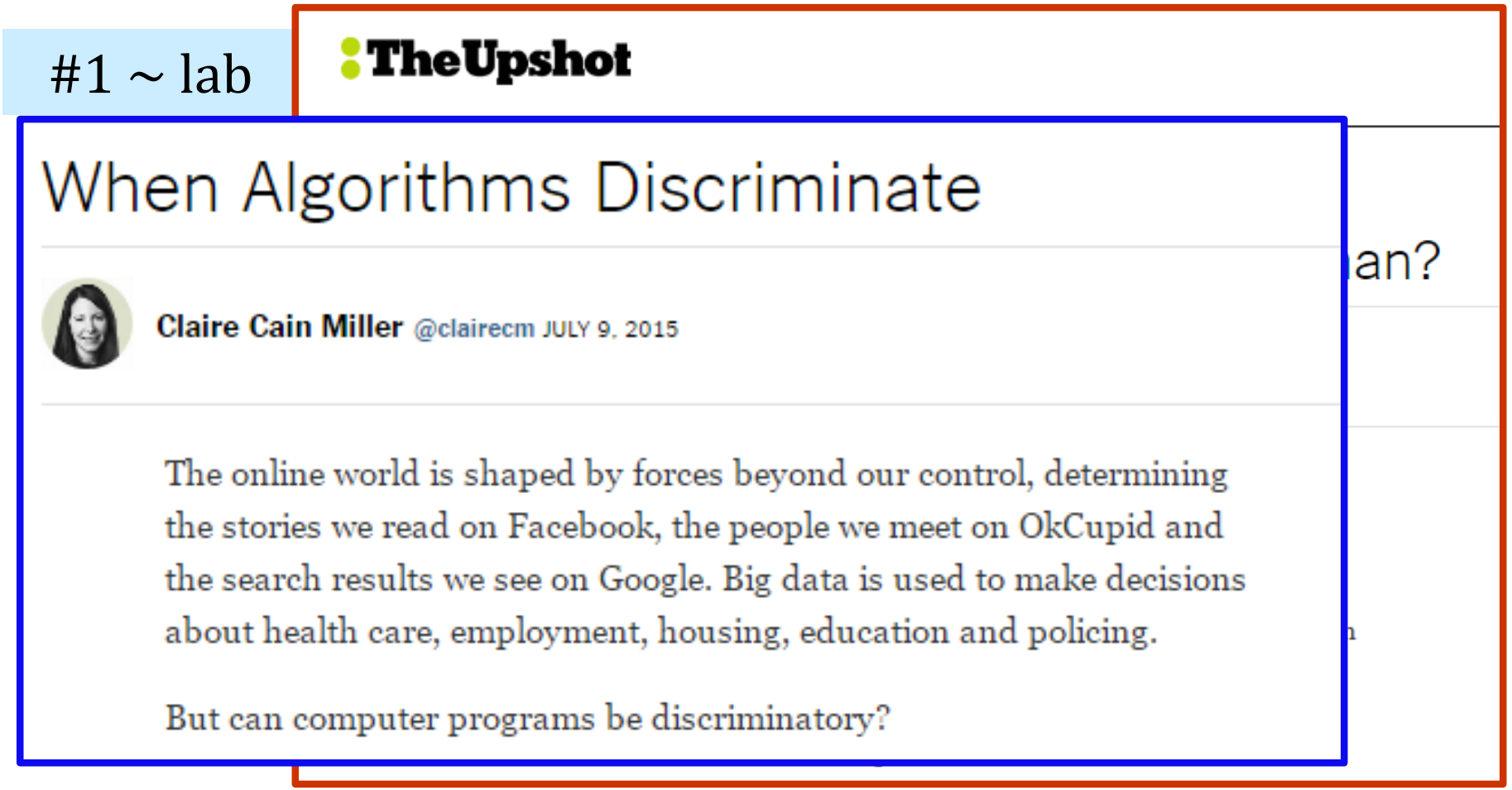
Hiring and recruiting might seem like some of the least likely jobs to be automated. The whole process seems to need human skills that computers lack, like making conversation and reading social cues.

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... ?

#0 *When Algorithms Discriminate...*

#1 ~ lab



Homework 8 preview

#0 *When Algorithms Discriminate...*

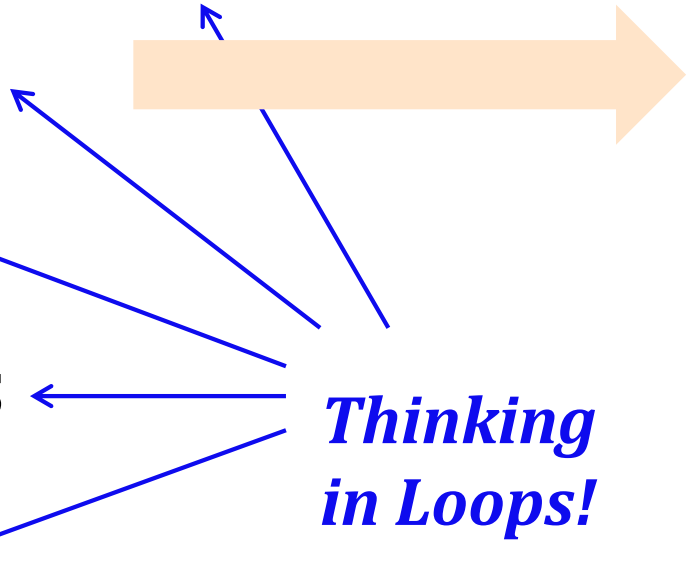
#1 ~ lab The Mandelbrot Set

#2 *Lots of loops!*

#3 *Pi from Pie*

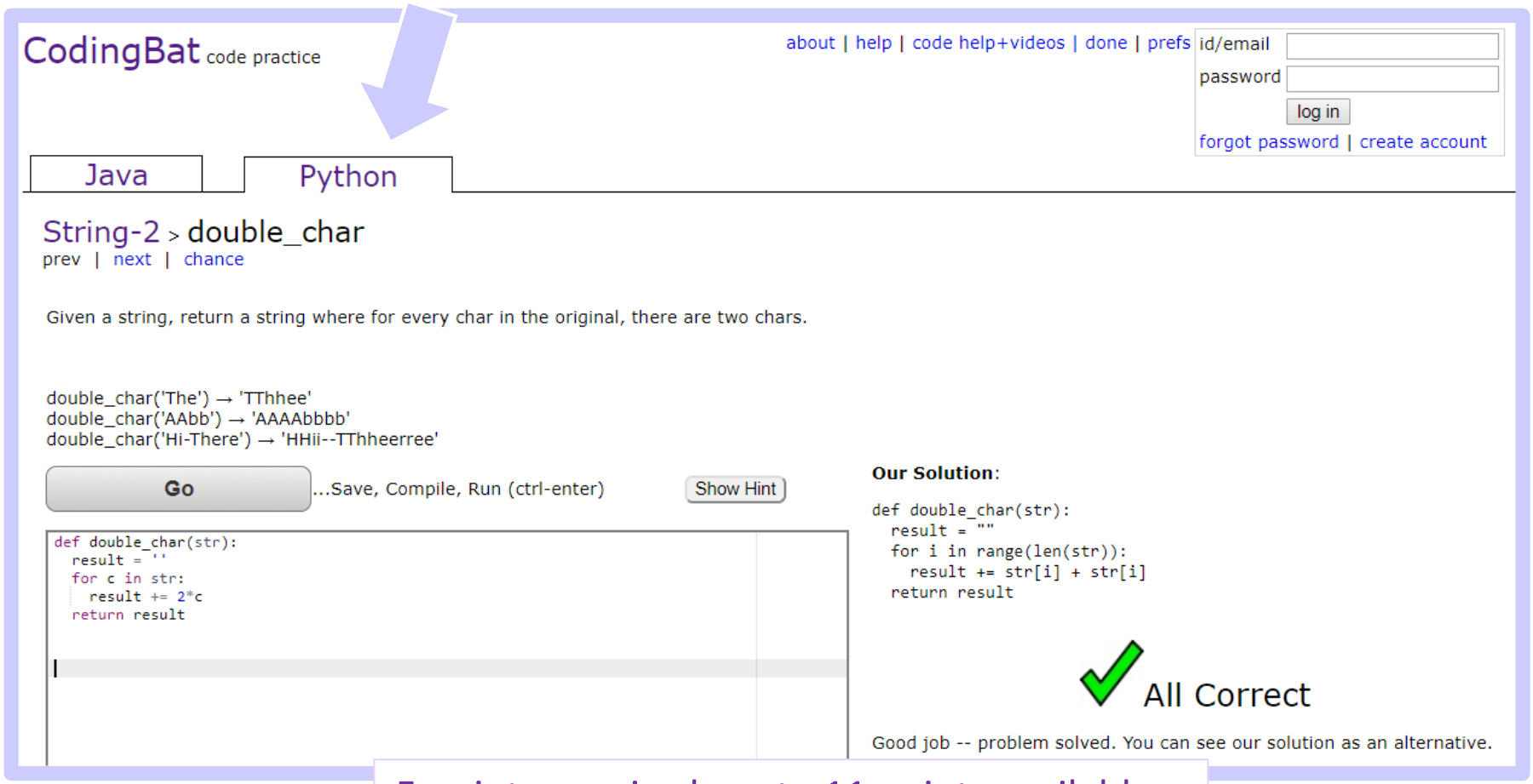
#4 TTS Securities

(Extra) ASCII Art



PythonBat loop practice...

google for *"PythonBat"* then...



5 points required, up to 11 points available...

Thinking in *loops*

for

```
for x in range(42):
    print(x)
```

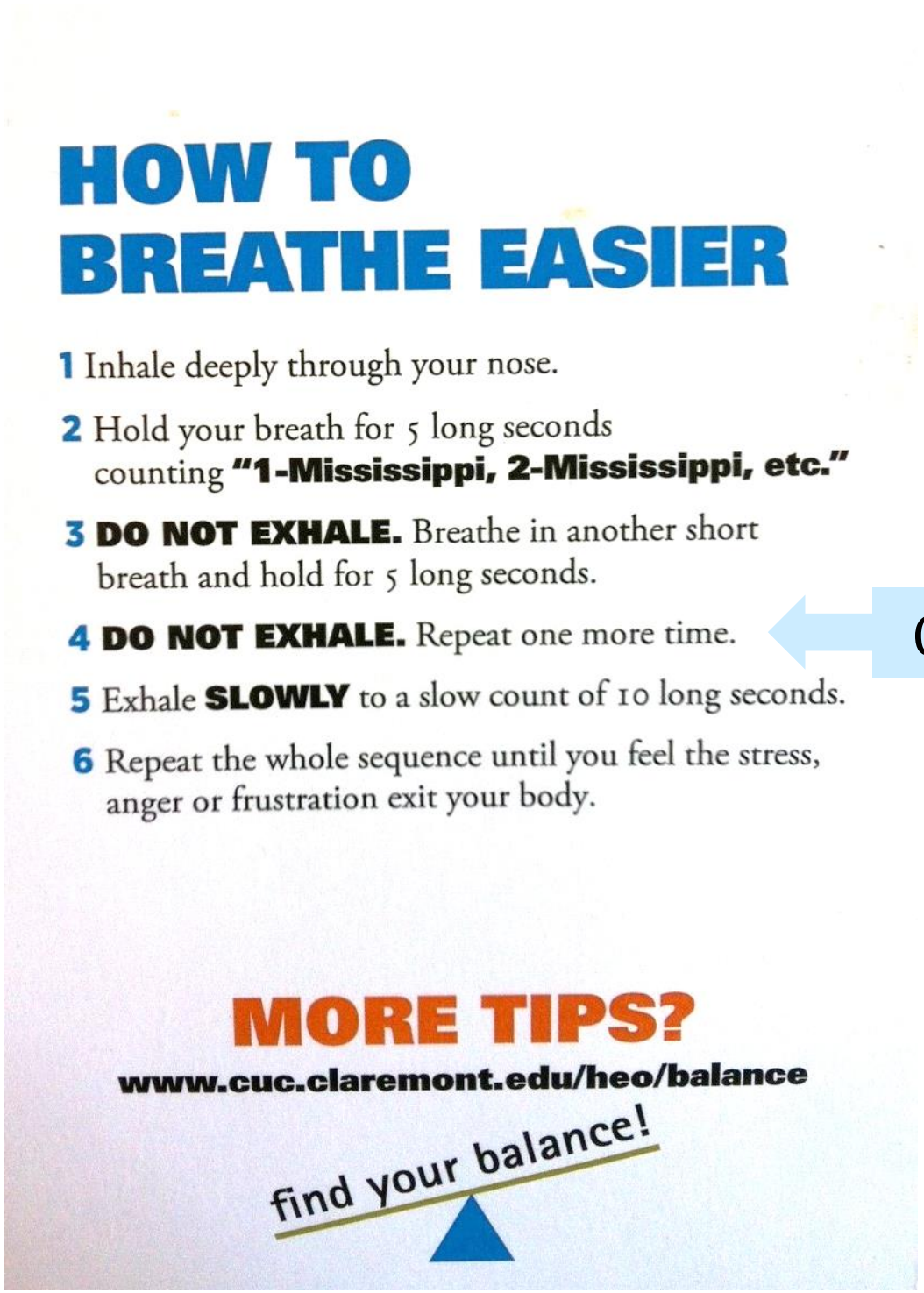
list

while

```
x = 1
while x < 42:
    print(x)
    x *= 2
```

What are the design differences between these two types of Python loops?

Loop design...



Careful here!

Is this a `for` or a `while` loop?

Table tent...

Loop design...

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!

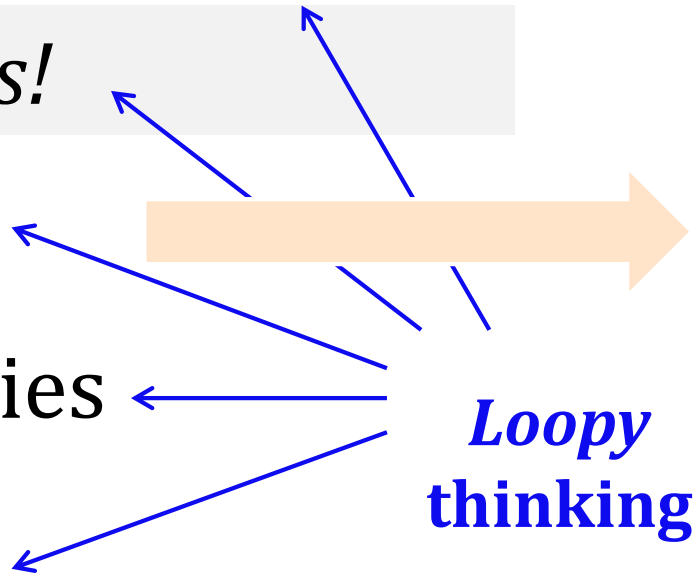
Is this a `for` or a `while` loop?

Thinking in *loops*

`for`	`while`
definite iteration	***indefinite iteration***
For a **known** list or # of iterations	For an **unknown** number of iterations

Homework 8 preview

#0	*When Algorithms Discriminate...*
#1 ~ lab	The Mandelbrot Set
#2	*Lots of loops!*
#3	*Pi from Pie*
#4	TTS Securities
(Extra)	ASCII Art
(Web extra)	CSS: Cascading Style Sheets



Hw8 Pr3

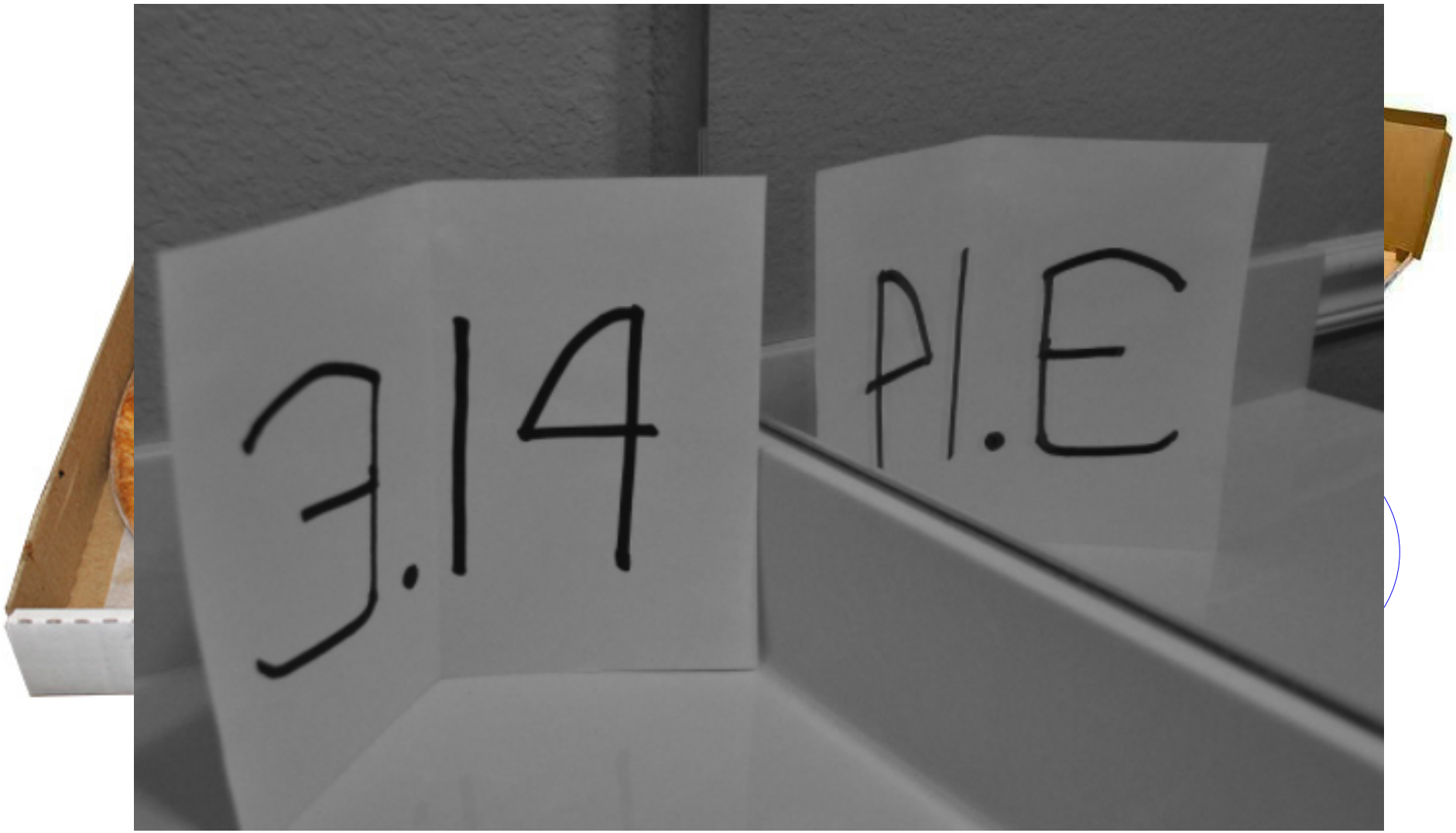
Pi from Pie?

π

Pizza is the universal constant, after all…

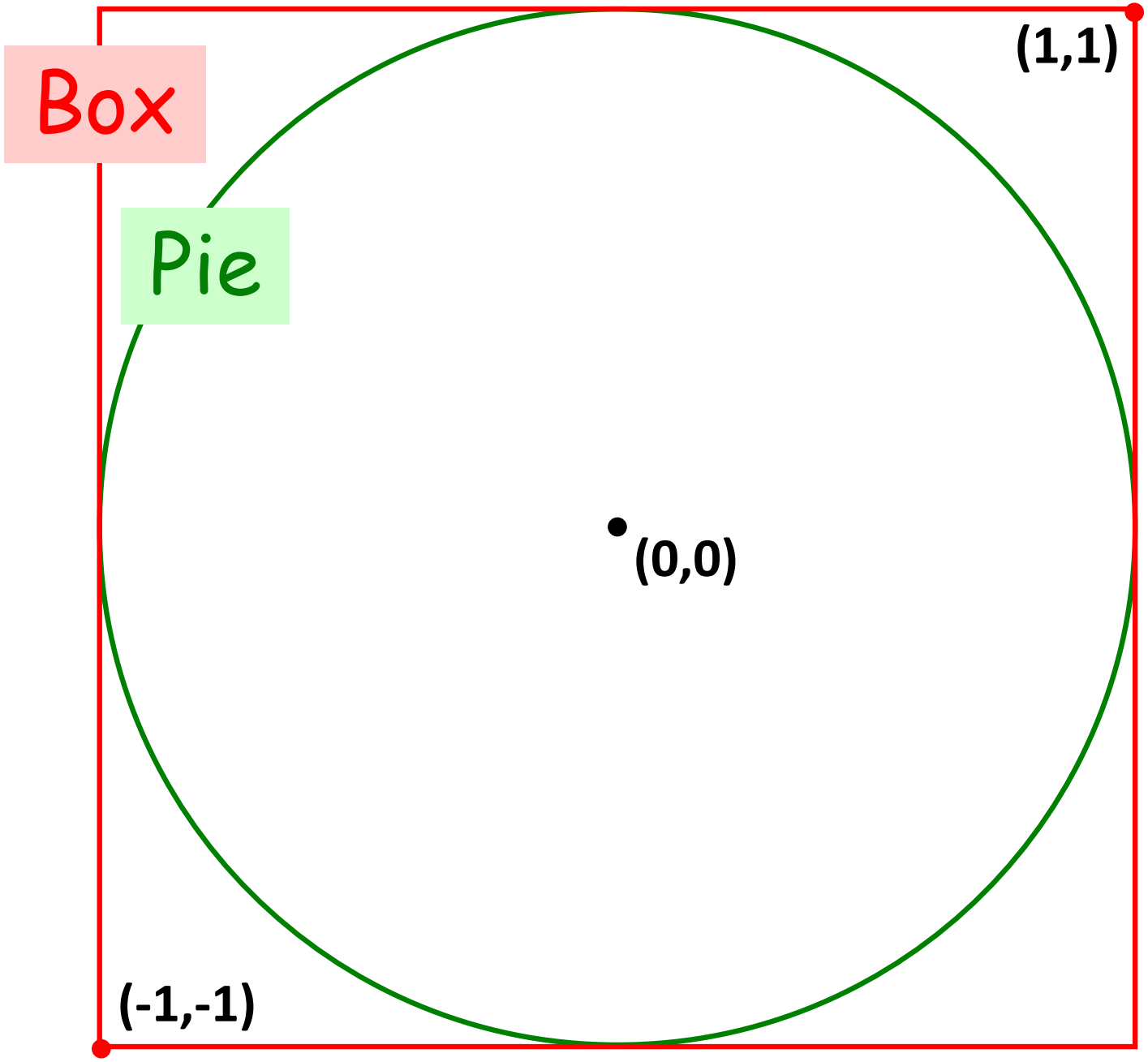
Hw8 Pr3

Pi from Pie?



This couldn't be just a coincidence!

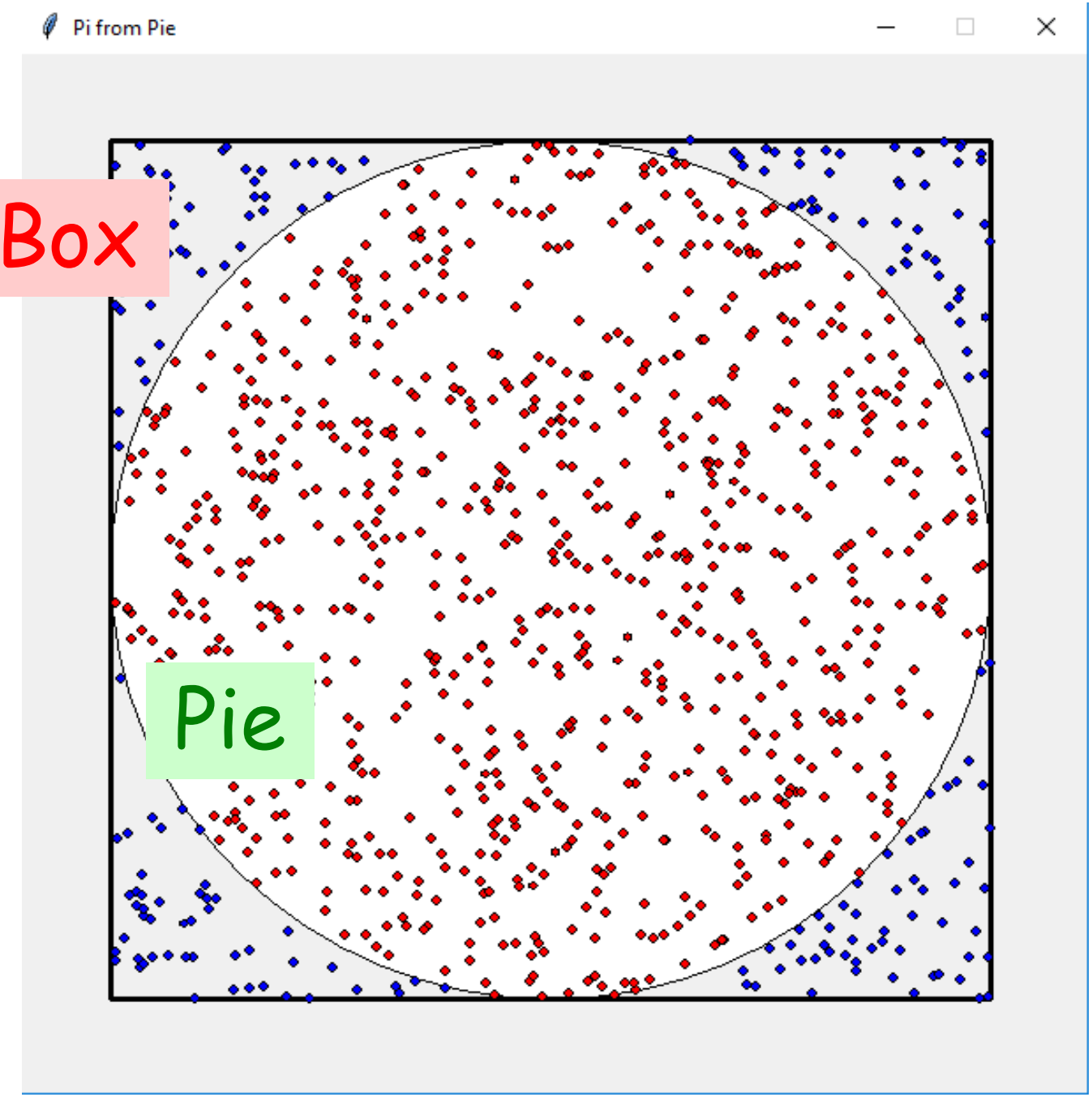
Hw8 Pr3



Estimating π from pie?

What if we just throw darts at this picture?

Pi-design challenge...



Estimating π 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 π from these throws? ⇦

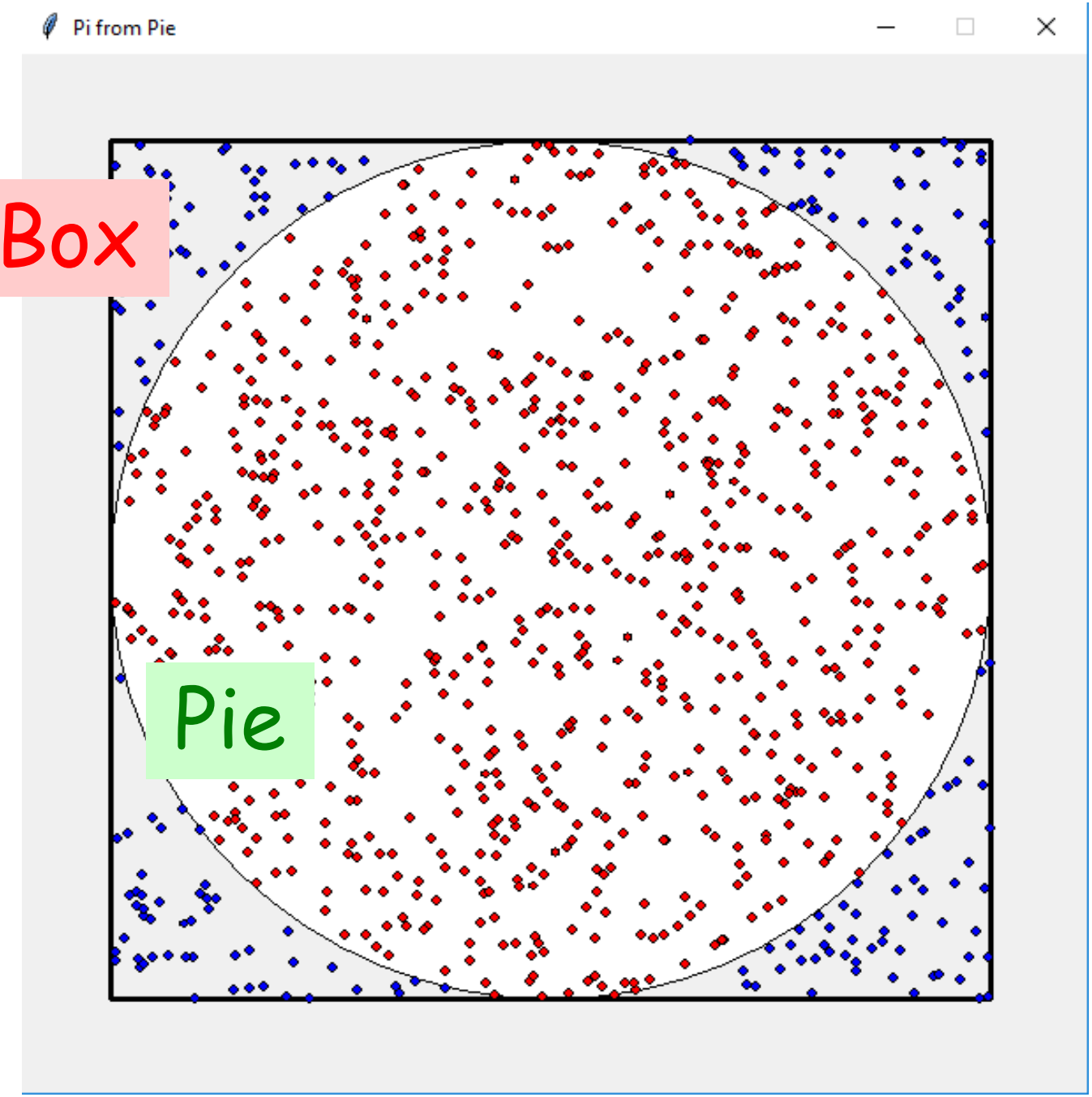
Hints

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

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

How do these help!?

Pi-design challenge...



Estimating π from pie?

$$\frac{\pi}{4} = \frac{\bigcirc \text{ area}}{\square \text{ area}}$$

$$\pi \sim \frac{4 * \bigcirc \text{ hits}}{\square \text{ hits}}$$

Hw8 Pr3

Loops: `for` or `while`?

`pi_one(e)`

e == how close to π we need to get

`pi_two(n)`

n == number of darts to throw

Which function will use which kind of loop?

Loops: `for` or `while`?

`pi_one(e)`

`while`

e == how close to π we need to get

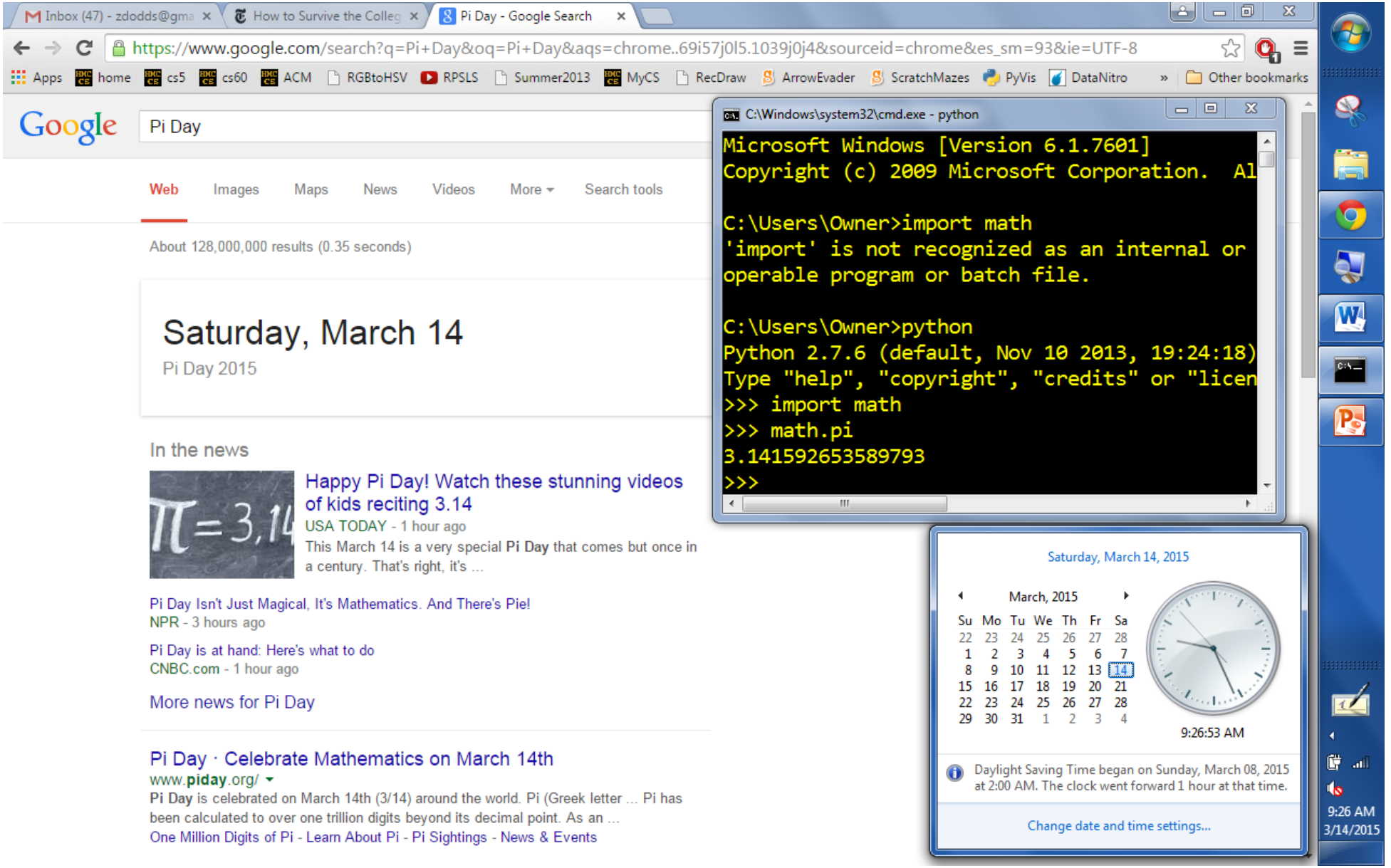
`pi_two(n)`

`for`

n == number of darts to throw

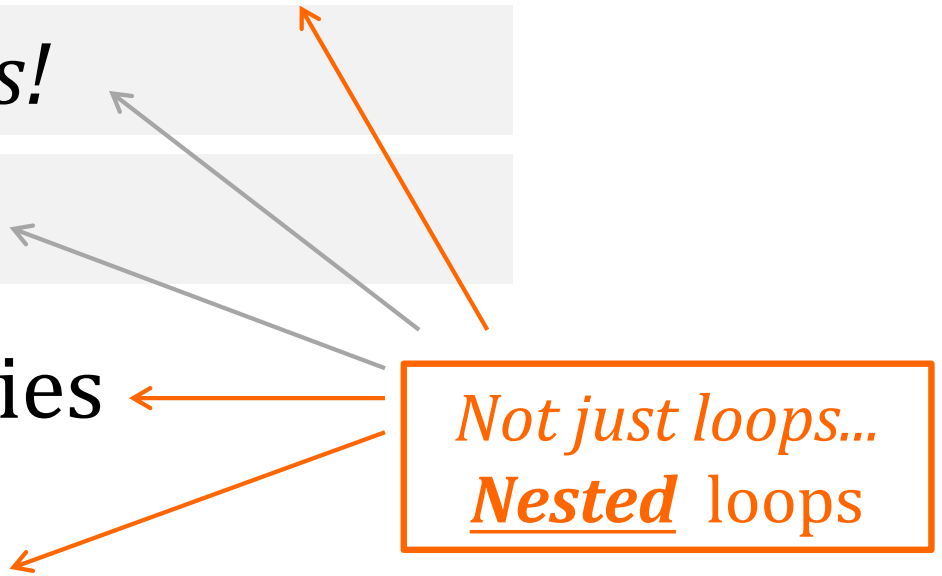
π day!

3/14/15 9:26:53



Homework 8 preview

#0	*When Algorithms Discriminate...*
#1 ~ lab	The Mandelbrot Set
#2	*Lots of loops!*
#3	Pi from Pie
#4	TTS Securities
(Extra)	ASCII Art



Nested loops are familiar, too!

```
for mn in range(60):
    for s in range(60):
        tick()
```

list

list

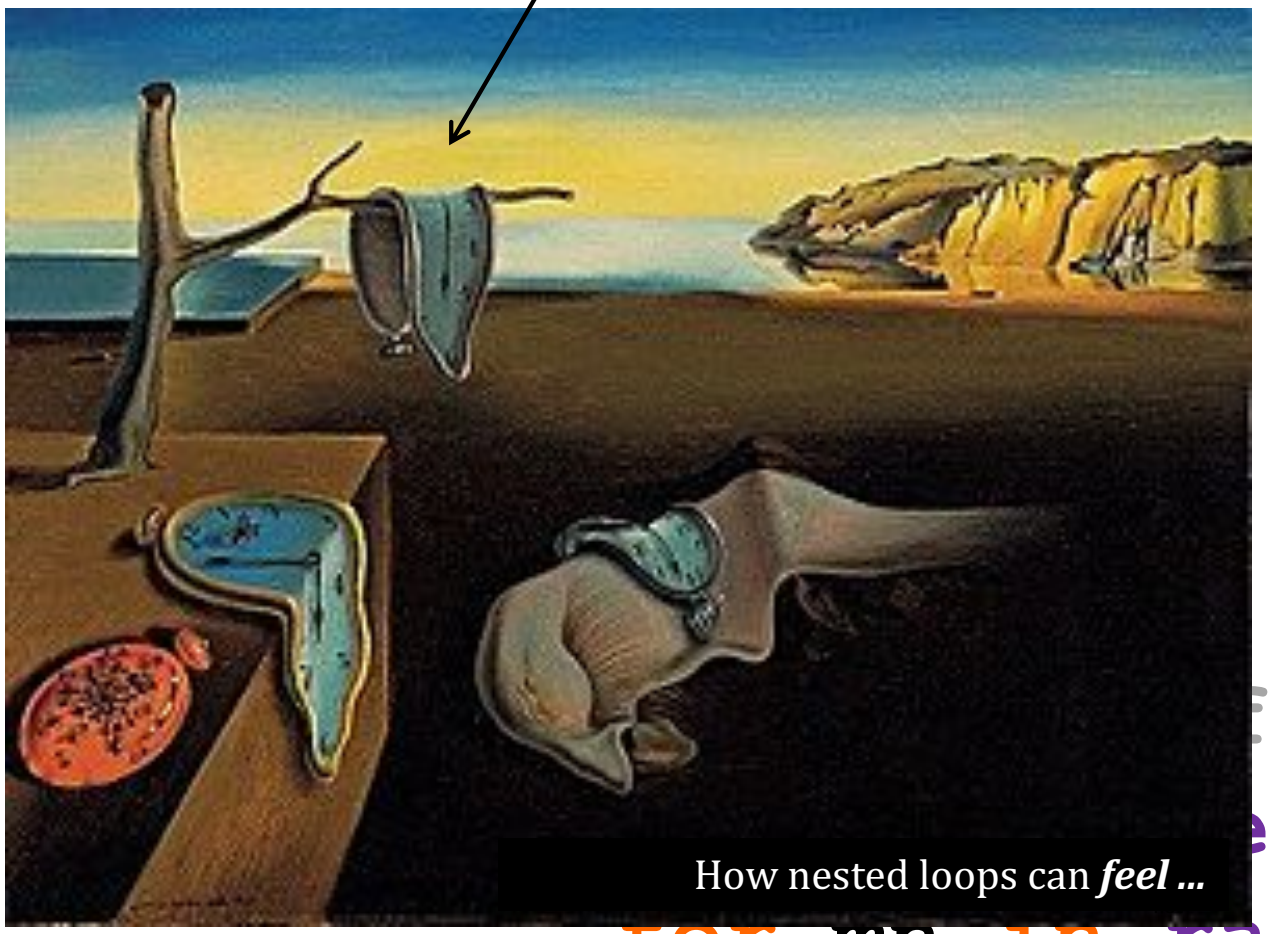
Nested loops are familiar, too!

Nested loops

Life clock


```
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()
```

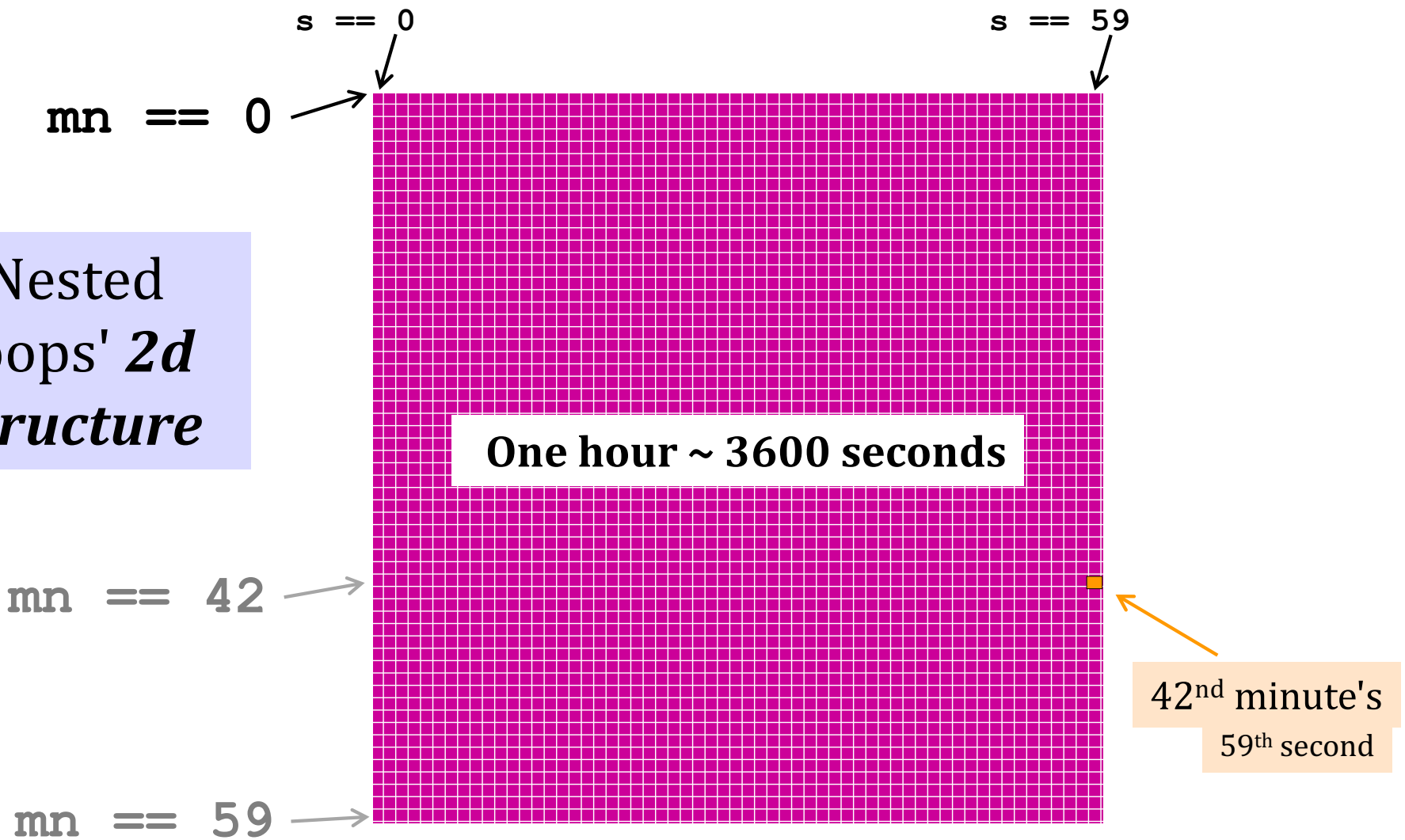
Nested loops!



Persistence of Memory, S. Dali (MoMA)


```
                    :
                (m,y)):
                (24):
for mn in range(60):
    for s in range(60):
        tick()
```

list



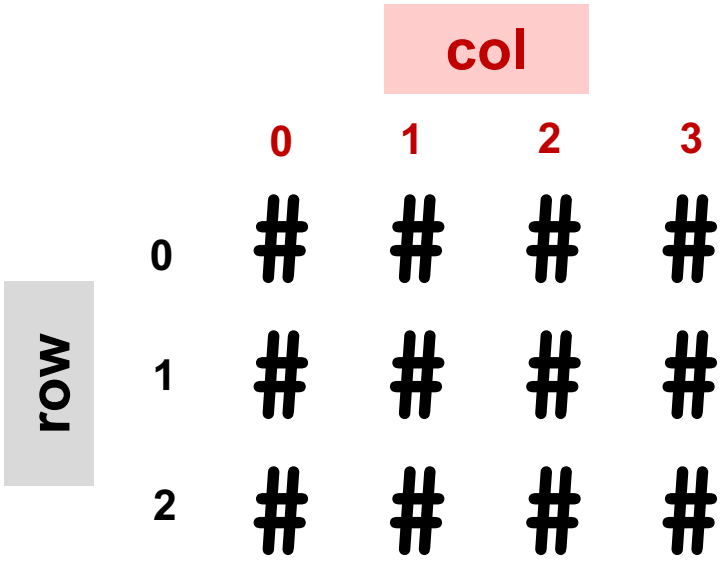
Nested loops' ***2d structure***

```
for mn in range(60):
    for s in range(60):
        tick()
```

(list)

hour()

Creating 2d structure ~ in ASCII



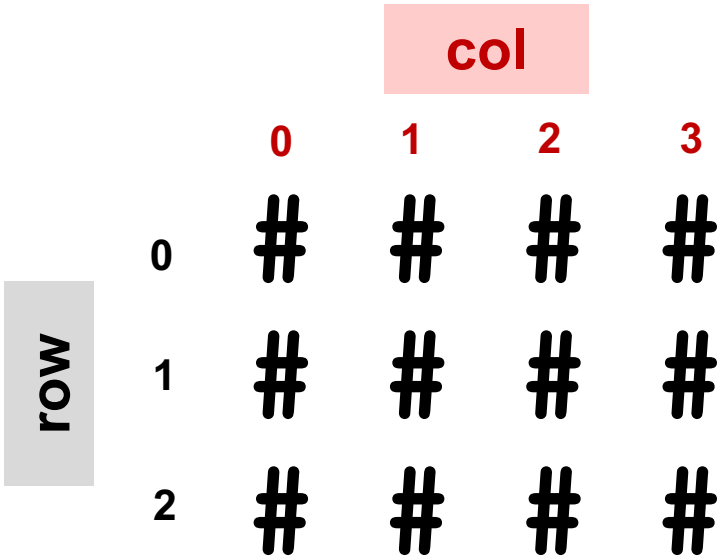

```
for row in range(3):
    for col in range(4):
        print("#")
```

list

list

Wait! this needs something more...

Creating 2d structure




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

list

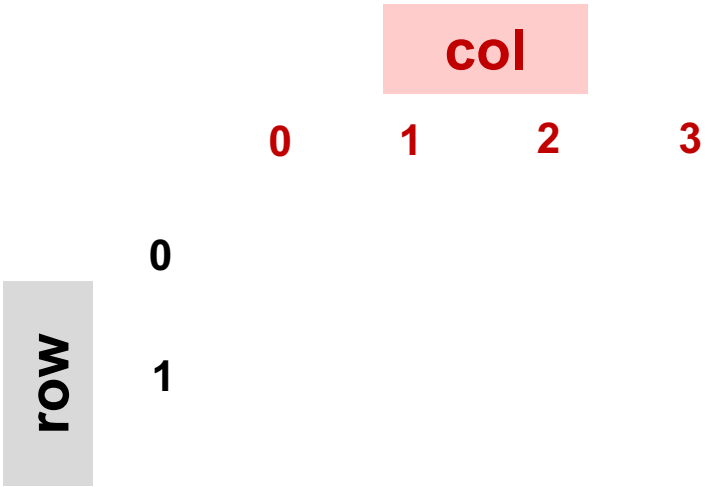
list

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 =
    col =

row =
    col =
    col =
    col =
    col =

row =
    col =
    col =
    col =
    col =
```

Creating 2d structure

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

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

col

0 1 2 3

row

0

1

2

```
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) _____

A

```
for r in range(3):          [0,1,2]
  for c in range(6):
    if c >= r:              [0,1,2,3,4,5]
      print('#',end='')
    else:
      print(' ',end='')
  print()
```

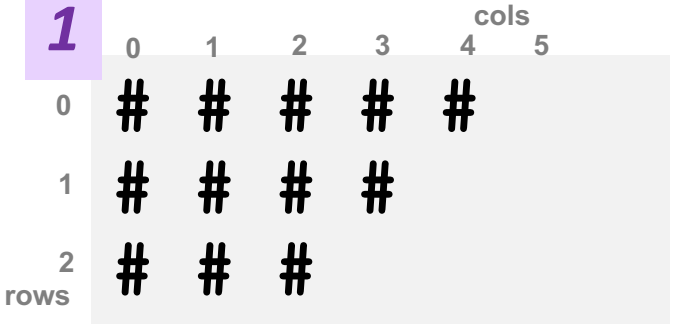
B

```
for r in range(3):
  for c in range(6):
    if c%2 == 1:
      print('#',end='')
    else:
      print(' ',end='')
  print()
```

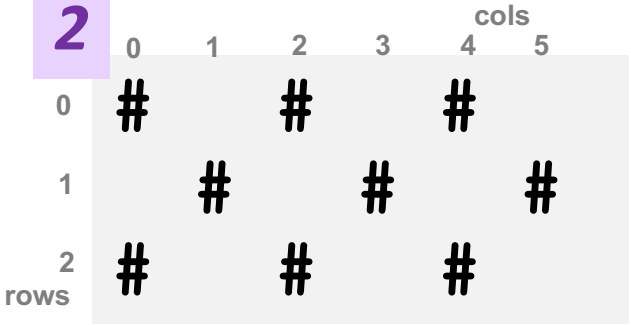
C

```
for r in range(3):
  for c in range(6):
    if c%2 == r%2:
      print('#',end='')
    else:
      print(' ',end='')
  print()
```

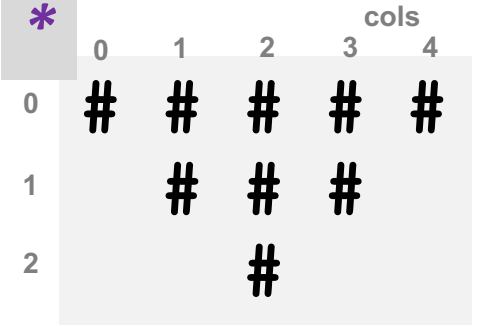
1



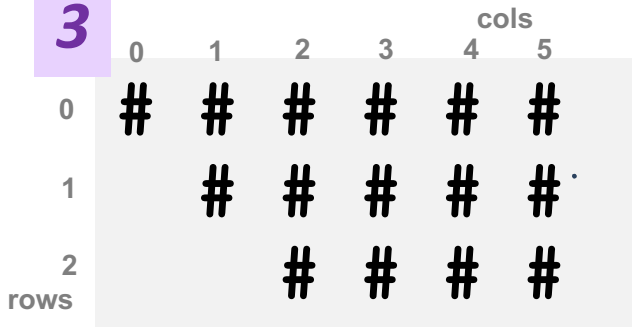
2



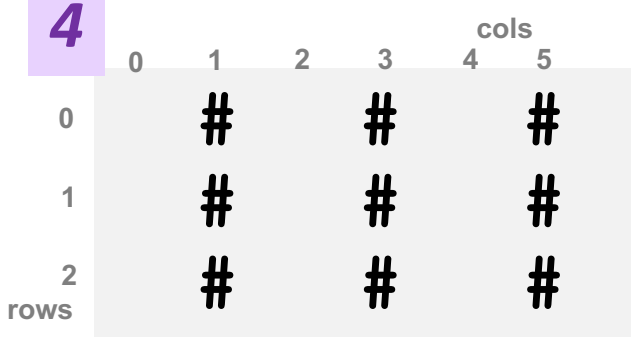
***



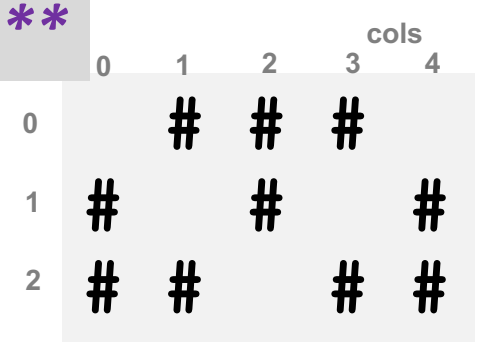
3



4



****



Match!

What code creates the fourth one?

* and ** are extra!

answers...

A

```
for r in range(3):          [0,1,2]
  for c in range(6):
    if c >= r:              [0,1,2,3,4,5]
      print('#',end='')
    else:
      print(' ',end='')
  print()
```

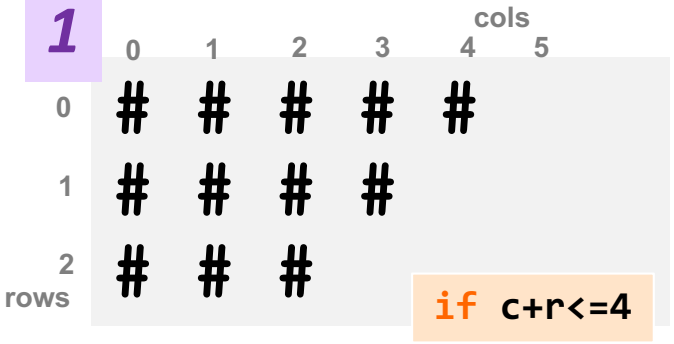
B

```
for r in range(3):
  for c in range(6):
    if c%2 == 1:
      print('#',end='')
    else:
      print(' ',end='')
  print()
```

C

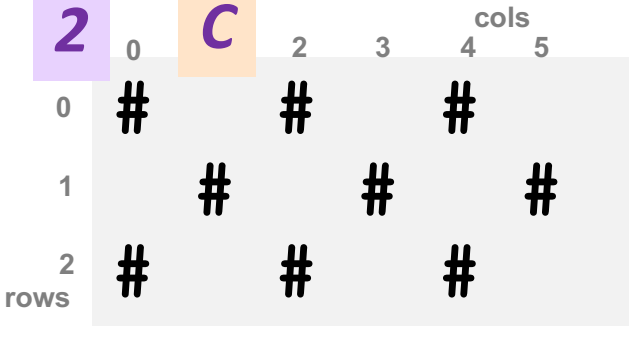
```
for r in range(3):
  for c in range(6):
    if c%2 == r%2:
      print('#',end='')
    else:
      print(' ',end='')
  print()
```

1

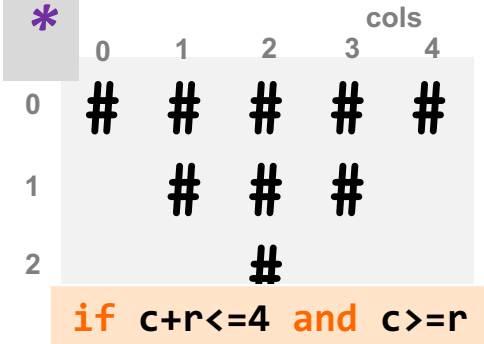


`if c+r<=4`

2 **C**

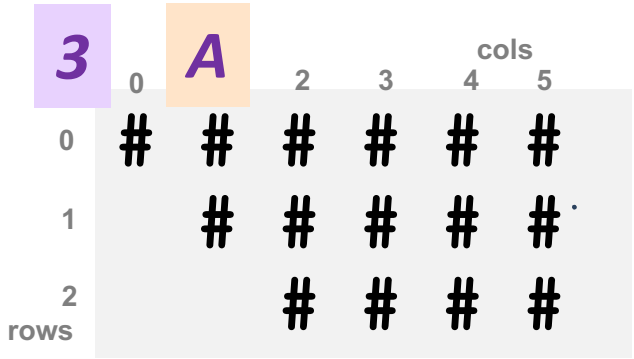


*****

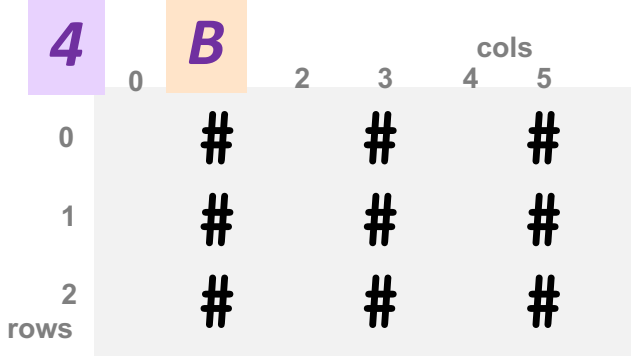


`if c+r<=4 and c>=r`

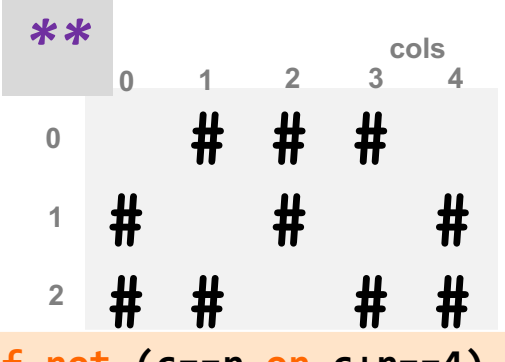
3 **A**



4 **B**



******



`if not (c==r or c+r==4)`

Match!

What code creates the fourth one?

* and ** are extra!

answers...

Pass those...

[0,1,2]

```
for r in range(3):
  for c i
    if c
      pri
    else:
      prin
  print()
```

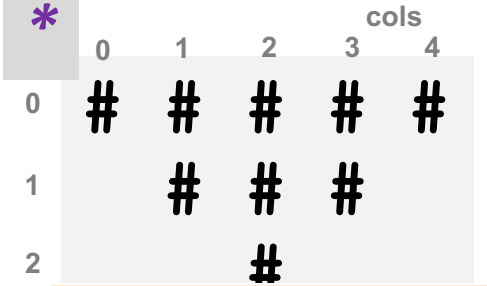
A

```
ange(3):
 range(6):
 == r%2:
 ('#',end='')
 print('  ',end='')
print()
```

1

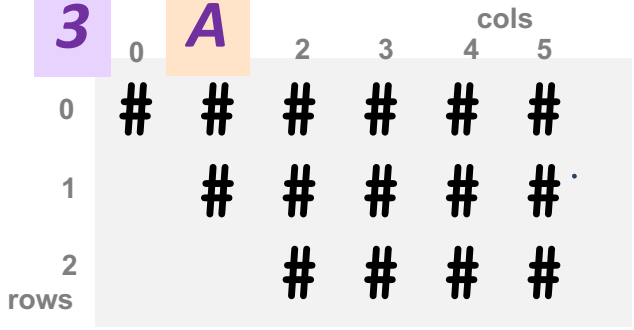
2 C

*




```
if c+r<=4 and c>=r
```

3 A



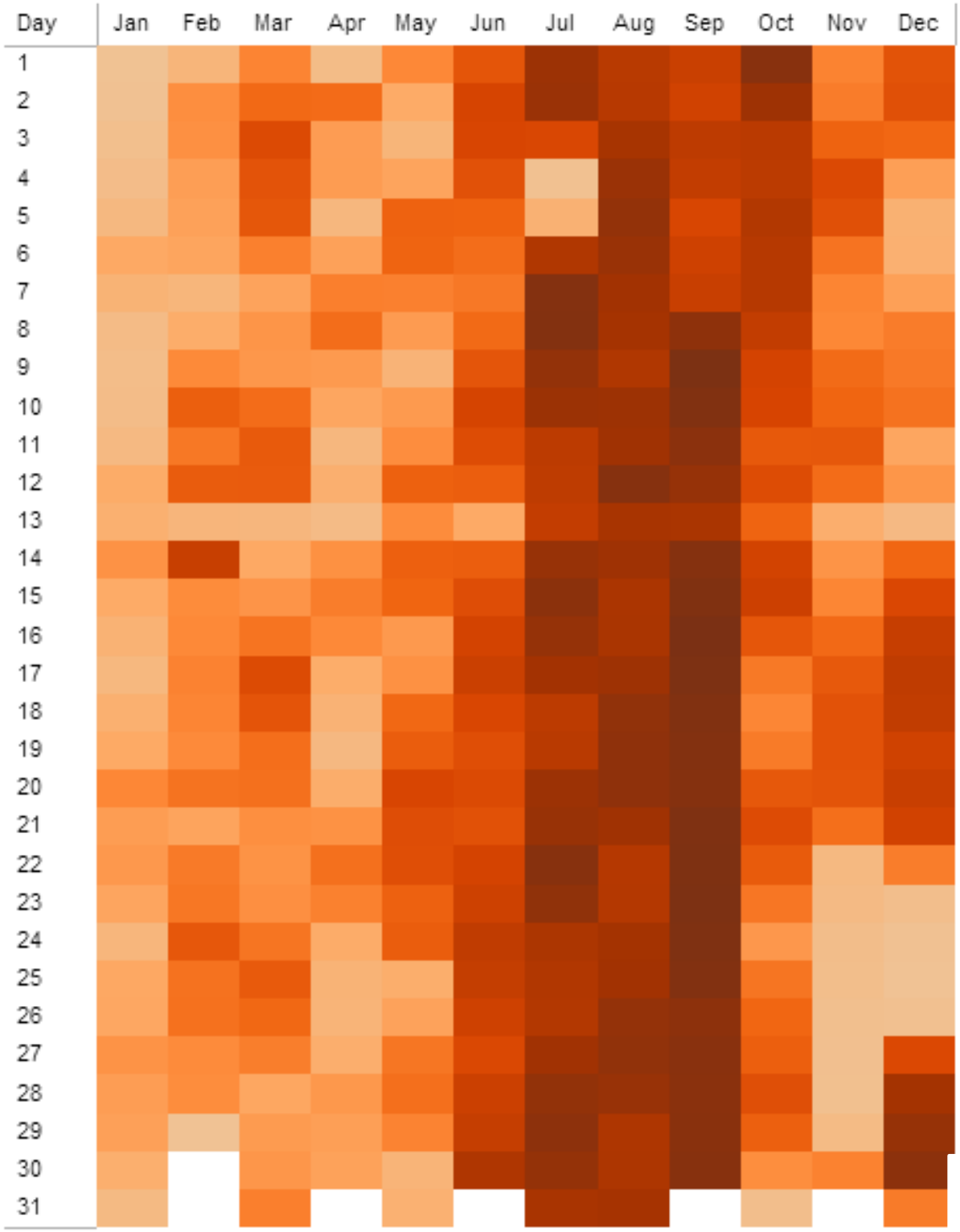
4

**


```
if not (c==r or c+r==4)
```

Washington-ward

Houston-ward



Nested loops' *2d structure*

```
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?!

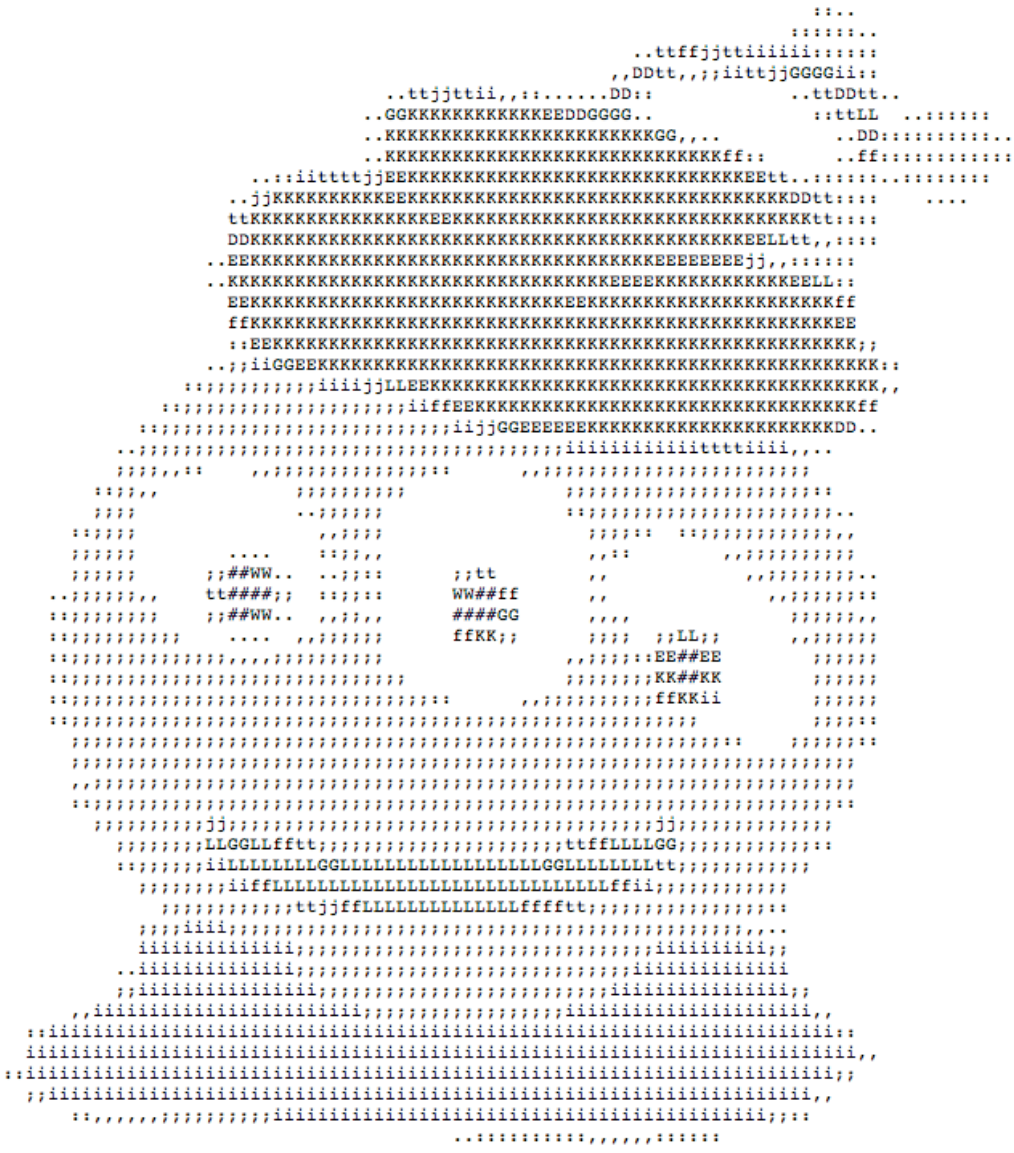
Data represents the # of babies born in the United States between 1973 and 1999

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!

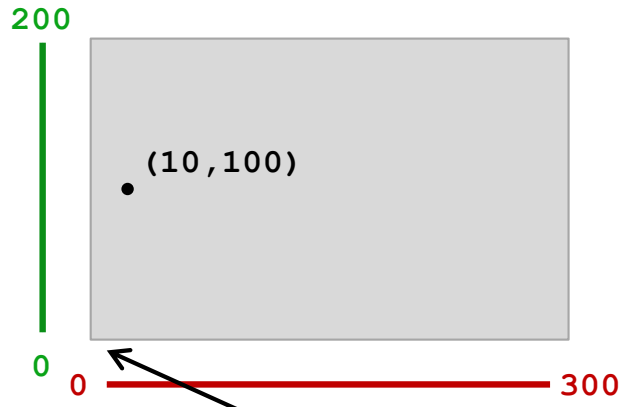
Python and images

```
from cs5png import *

im = PNGImage( 300, 200 )

im.plotPixel( 10, 100 )
```

*inputs are **width** and **height***



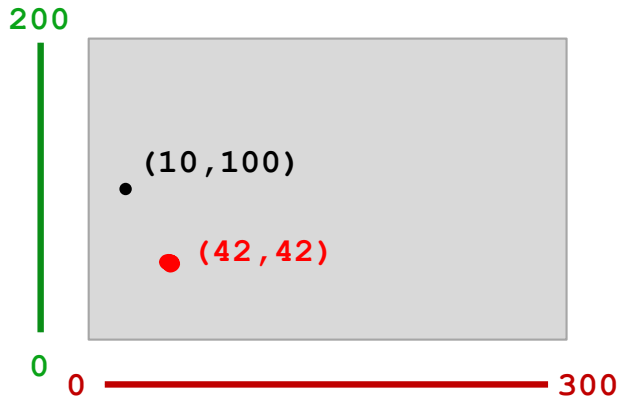
(0,0) is in the usual place!

Python and images

```
from cs5png import *
```

*inputs are **width** and **height***

```
im = PNGImage( 300, 200 )
```



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

```
im.plotPixel( 10, 100 )
im.plotPixel( 42, 42, (255,0,0) )
```

col x row y red green blue

```
im.saveFile( )
```

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

Imagining Images

```
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()
```

thicker line?

other diagonal?

stripes ?

thicker stripes?

thatching?

Complex #s !

Nothing's too ***complex for Python!***

$$\sqrt{-1} = i$$

`i` *can't believe this!*

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

```
(-1+0j)
```

Complex #s !

Nothing's too *complex for Python!*

$$\sqrt{-1} = i$$

`i` *can't believe this!*

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

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

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

```
In[]: c**2
(3-4j)
```

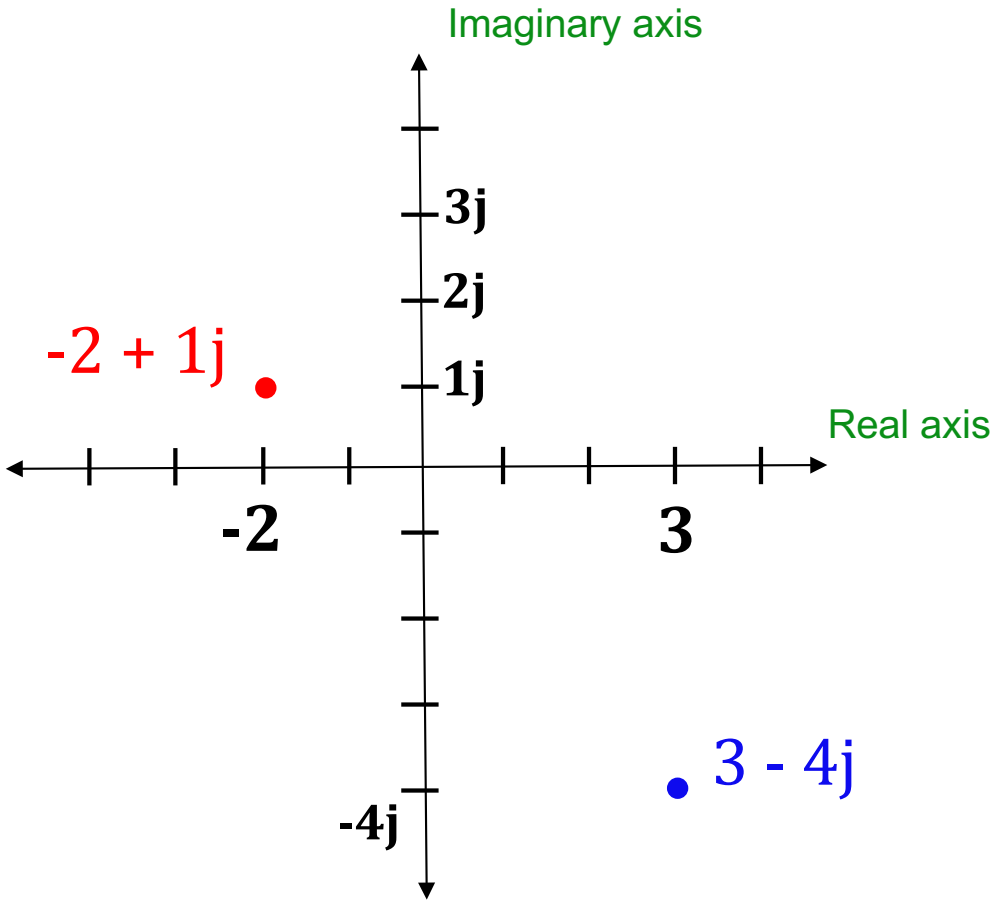
Complex #s !

Nothing's too *complex for Python!*

`i` *can't believe this!*

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

```
In[]: c**2
(3-4j)
```

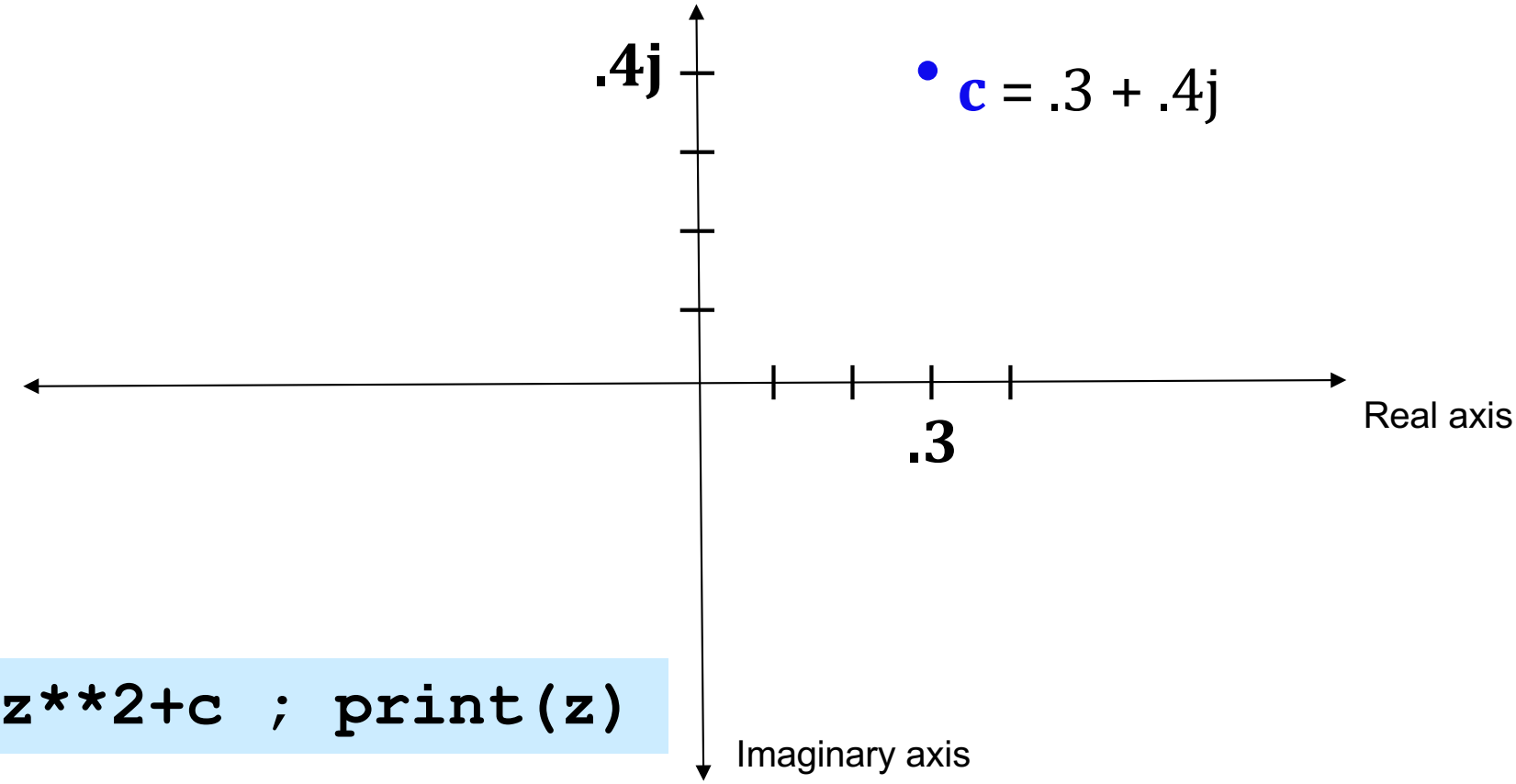


Lab 8: *the Mandelbrot Set*

Consider an ***update rule***
for all complex numbers ***c***

$$z_0 = 0$$

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




```
z = z**2+c ; print(z)
```

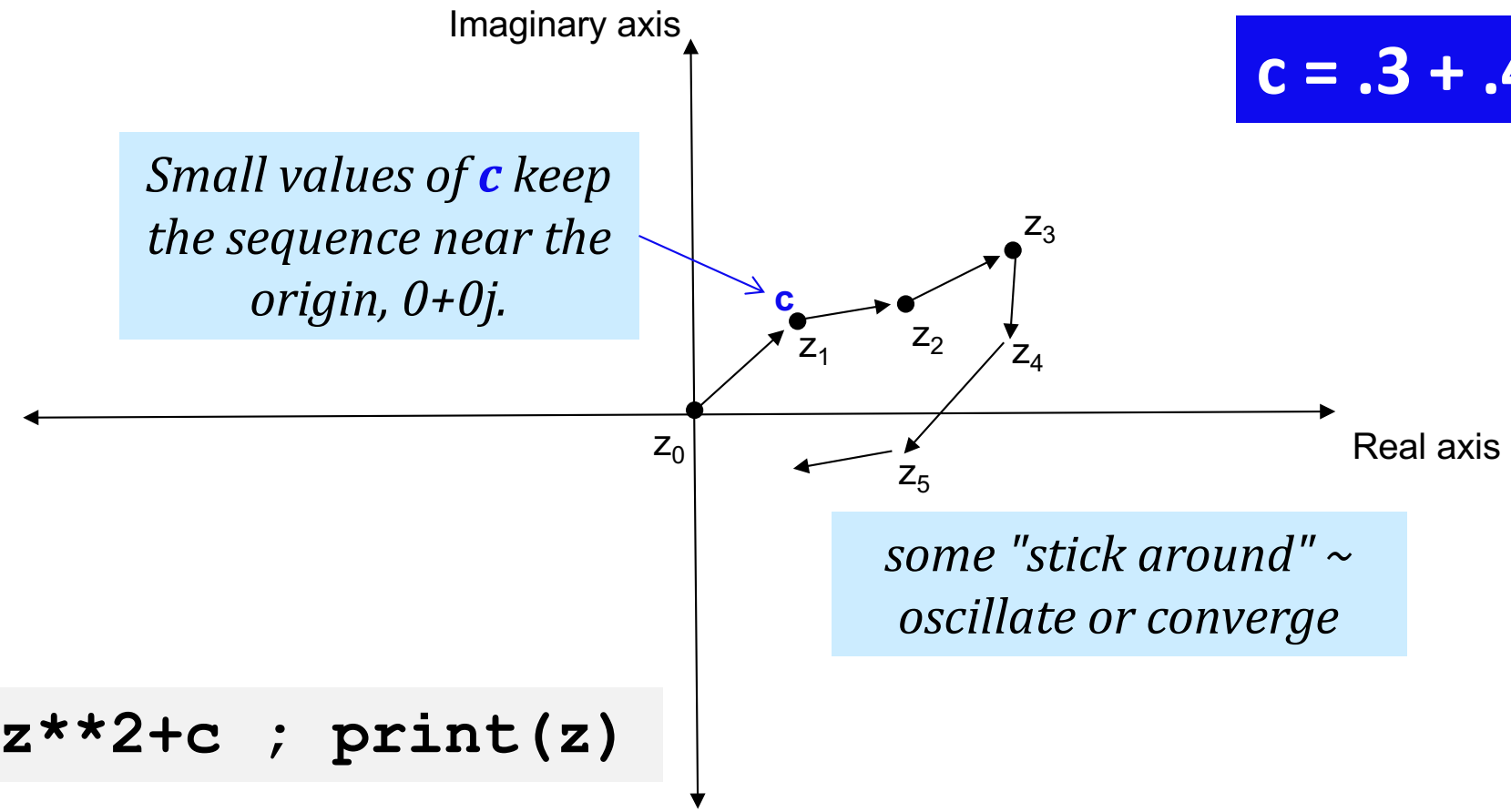
Mandelbrot Definition

Consider an ***update rule*** for all complex numbers ***c***

$$z_0 = 0$$

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

c = .3 + .4j



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

some "stick around" ~ oscillate or converge

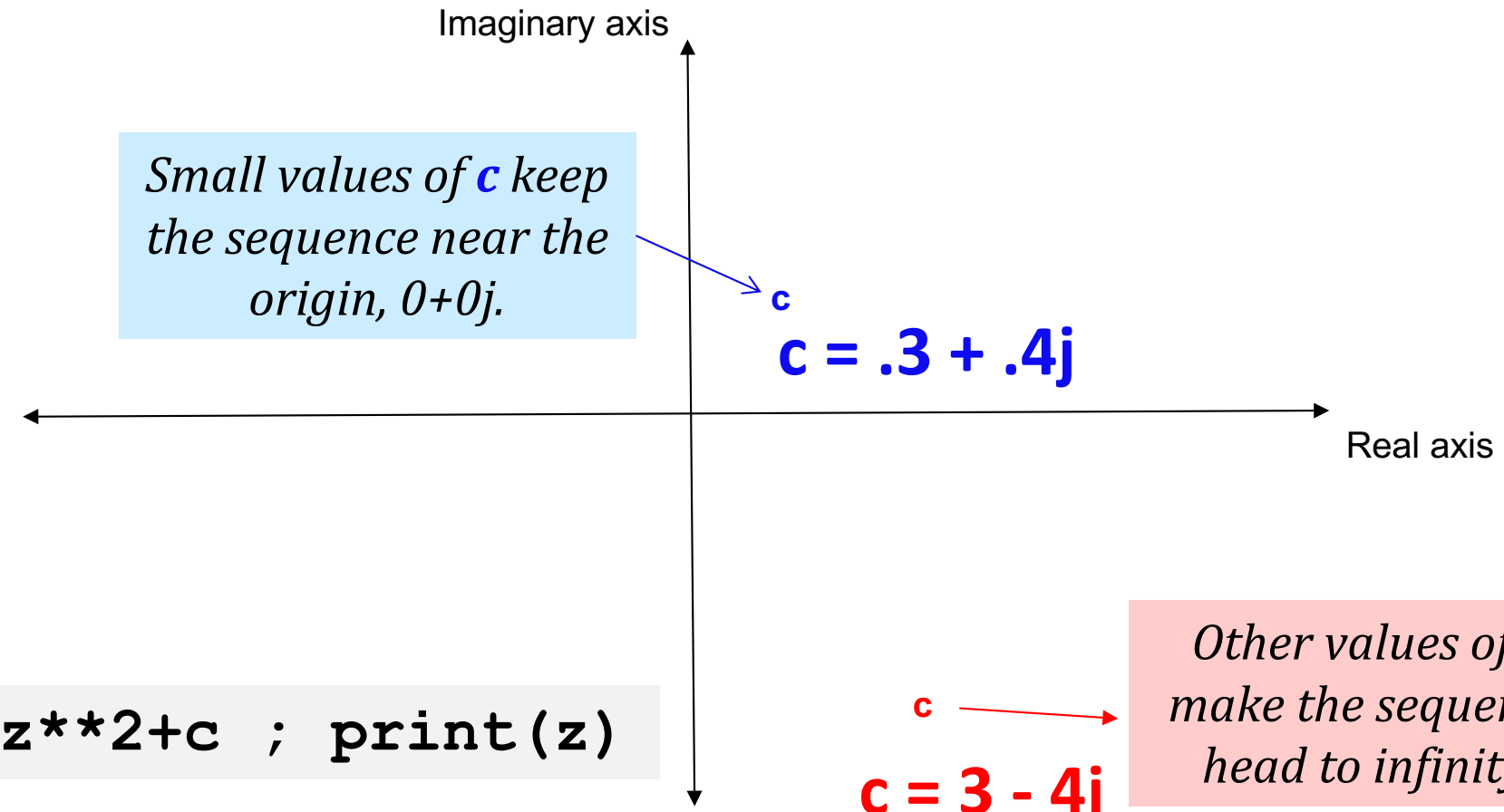
```
z = z**2+c ; print(z)
```

Mandelbrot Definition

Consider an ***update rule*** for all complex numbers ***c***

$$z_0 = 0$$

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



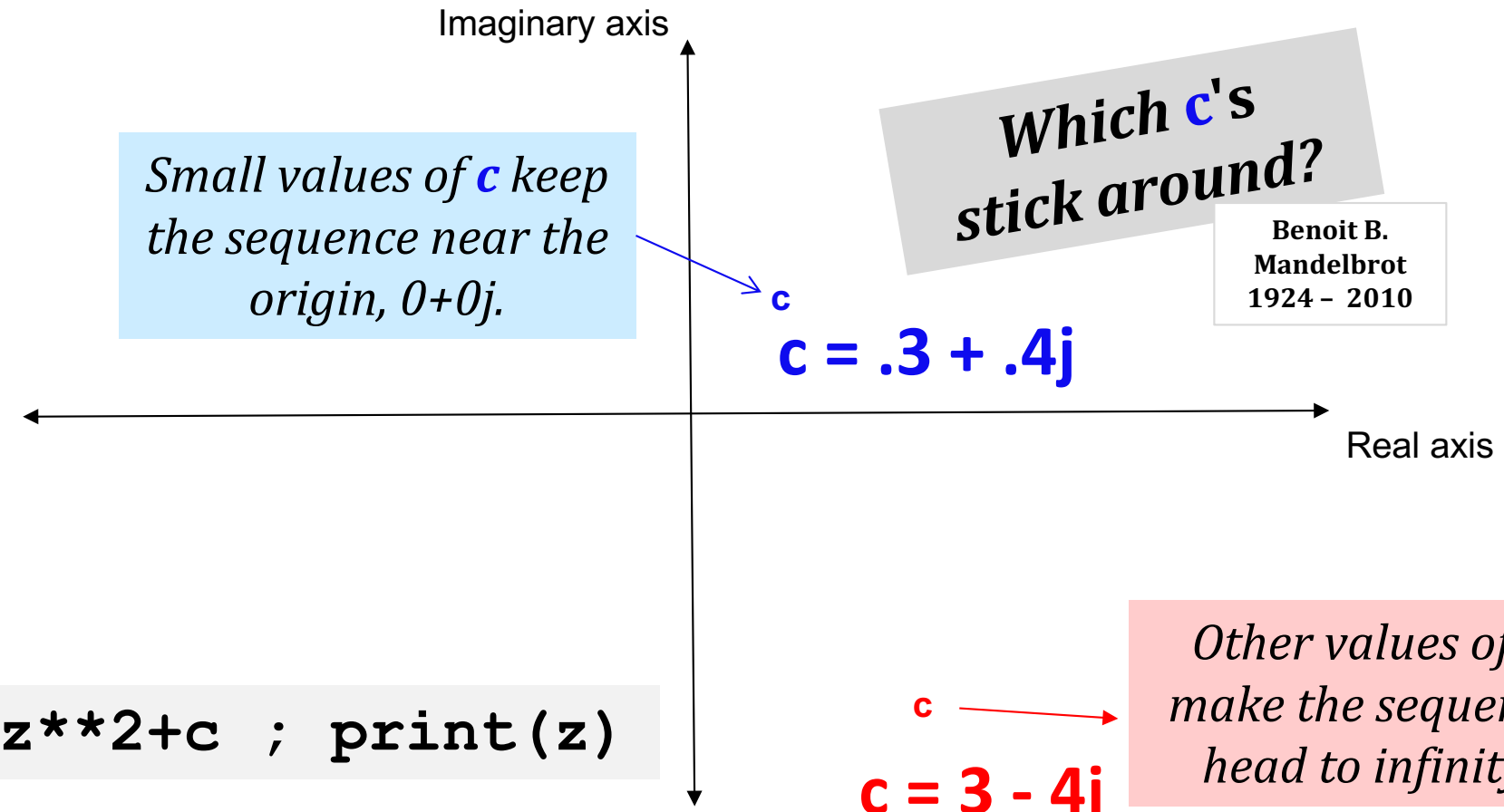

```
z = z**2+c ; print(z)
```

Mandelbrot Definition

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.*

Which c's stick around?

Benoit B. Mandelbrot
1924 – 2010

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

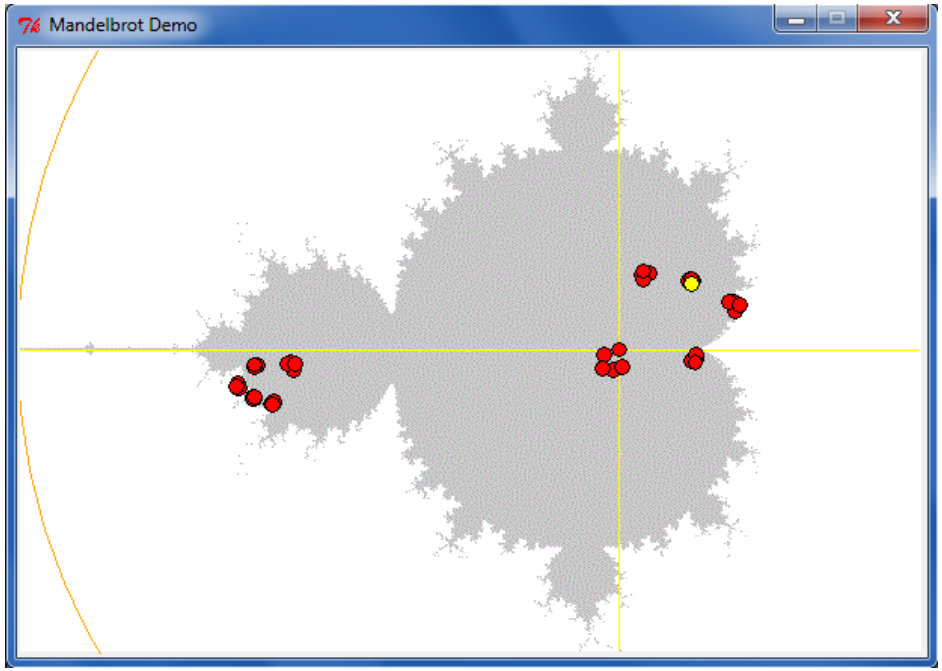
```
z = z**2+c ; print(z)
```

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 = 0.232813899367 + 0.232530407823 * 1j
iter # 3 : z = -1.21355756129 + -0.0546346462374 * 1j
iter # 4 : z = 0.256047527535 + -0.0303026920702 * 1j
iter # 5 : z = -1.14904739926 + -0.178425[illegible]6935 * 1j
iter # 6 : z = 0.0747849173552 [illegible]7964 * 1j
iter # 7 : z = -1.269[illegible]6977 * 1j
iter # 8 : [illegible]88 * 1j
iter # 9 [illegible]08 * 1j
iter # 10 [illegible]056431 * 1j
iter # 11 [illegible]602 * 1j
iter # 12 [illegible]14 * 1j
iter # 13 [illegible]5207 * 1j
iter # 14 : [illegible]9227677689 * 1j
iter # 15 : [illegible] + -0.180471770237 * 1j
iter # 16 : [illegible]360076381 + 0.255350697358 * 1j
iter # 17 : z = -1.26957426034 + -0.113606194429 * 1j
iter # 18 : z = 0.385222952638 + 0.125555732355 * 1j
iter # 19 : z = -1.08105700116 + -0.0661733682935 * 1j
iter # 20 : z = -0.0493841573866 + -0.0198329020025 * 1j
iter # 21 : z = -1.21164403147 + -0.160948405863 * 1j
iter # 22 : z = 0.228487387181 + 0.227117082506 * 1j
```

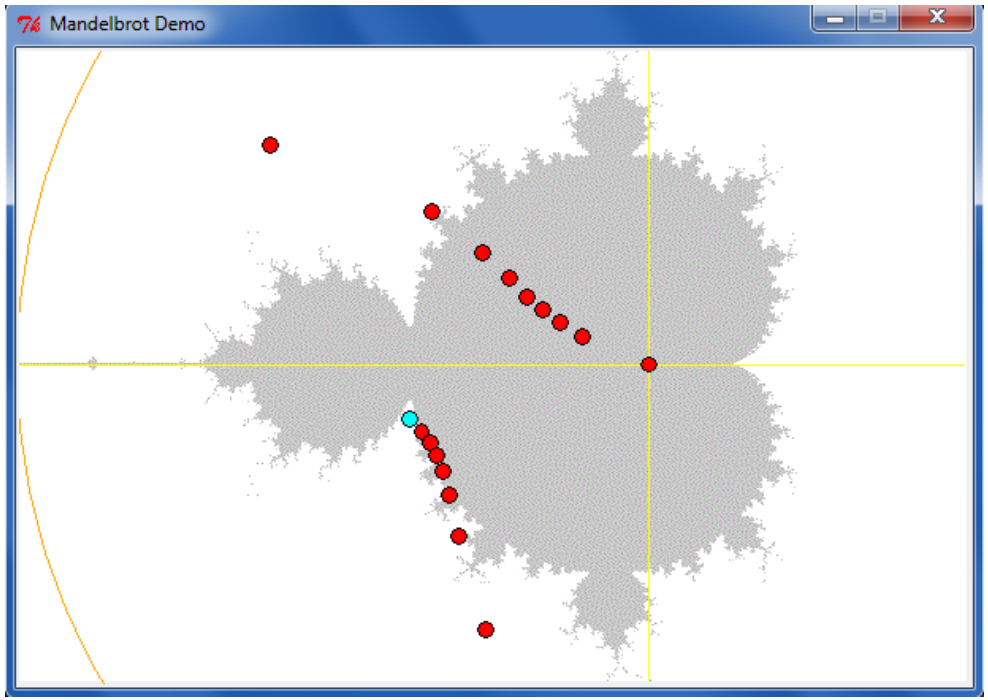
some c's stick around

Lab 8: *the Mandelbrot Set*

Consider an ***update rule*** for all complex numbers ***c***

$$z_0 = 0$$

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

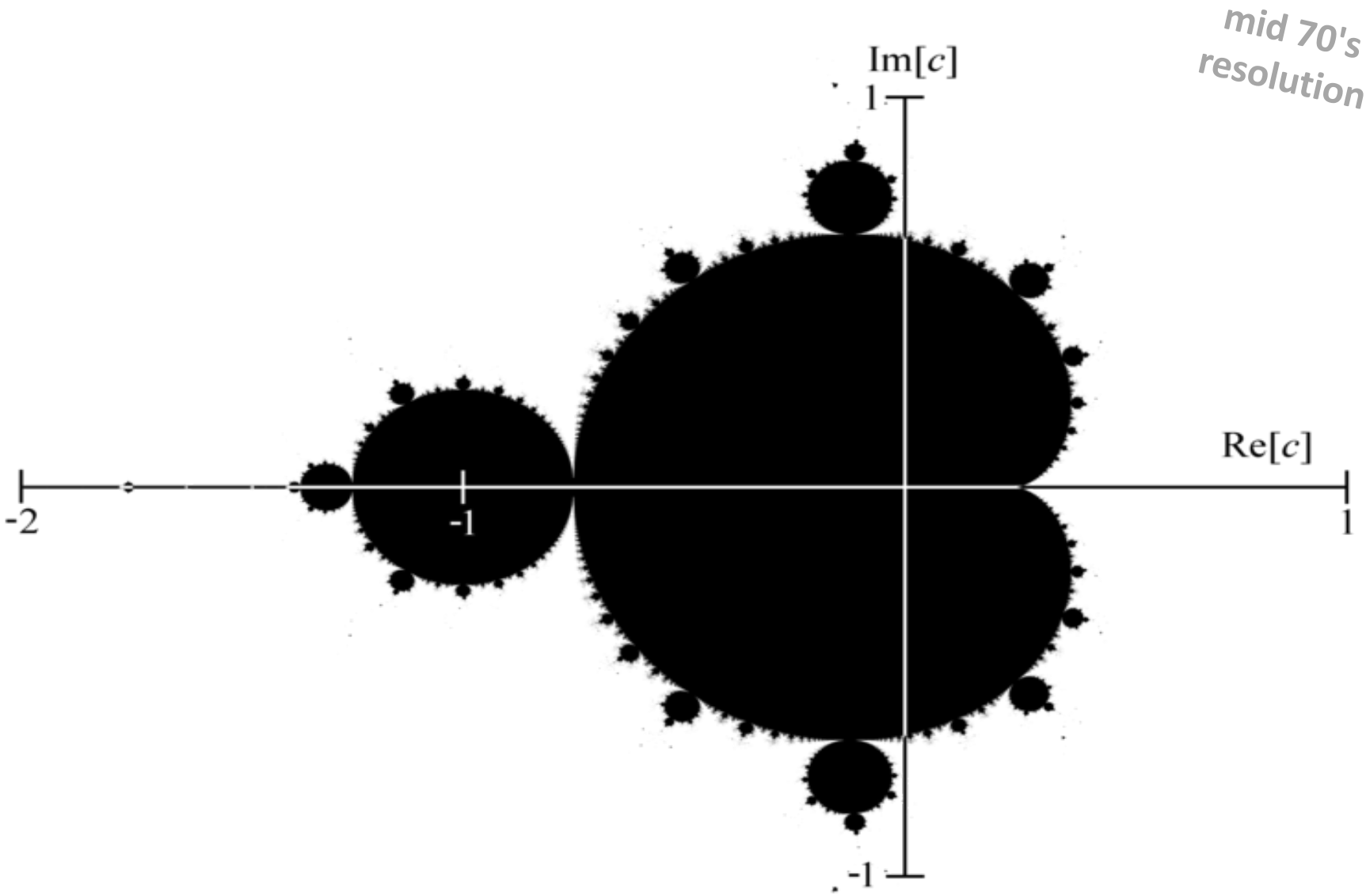



```
Click to choose c.
c is -0.757929883139 + -0.172932330827 * 1j

iter # 0 : z = 0.0 + 0.0 * 1j
iter # 1 : z = -0.757929883139 + -0.172932330827 * 1j
iter # 2 : z = -0.213377766429 + 0.0892088317622 * 1j
iter # 3 : z = -0.720358027597 + -0.211002693361 * 1j
iter # 4 : z = -0.283536331822 + 0.131066637188 * 1j
iter # 5 : z = -0.694714446542                369601 * 1j
iter # 6 : z = -0.3364                        34238 * 1j
iter # 7 :                                    197352 * 1j
iter # 8 :                                    19922 * 1j
iter # 9 :                                    50516 * 1j
iter # 10 :                                   34007 * 1j
iter # 11 :                                   0519 * 1j
iter # 12 :                                   438 * 1j
iter # 13 :                                   543792492805 * 1j
iter # 14 : z                 8018 + 0.48336747636 * 1j
iter # 15 : z              516174815348 + -0.839488323663 * 1j
iter # 16 : z = -1.19623408871 + 0.693713130081 * 1j
iter # 17 : z = 0.191808204996 + -1.8326189188 * 1j
iter # 18 : z = -4.07963159717 + -0.875955021339 * 1j
iter # 19 : z = 15.1181668861 + 6.97421523468 * 1j
iter # 20 : z = 179.161361972 + 210.701767303 * 1j
```

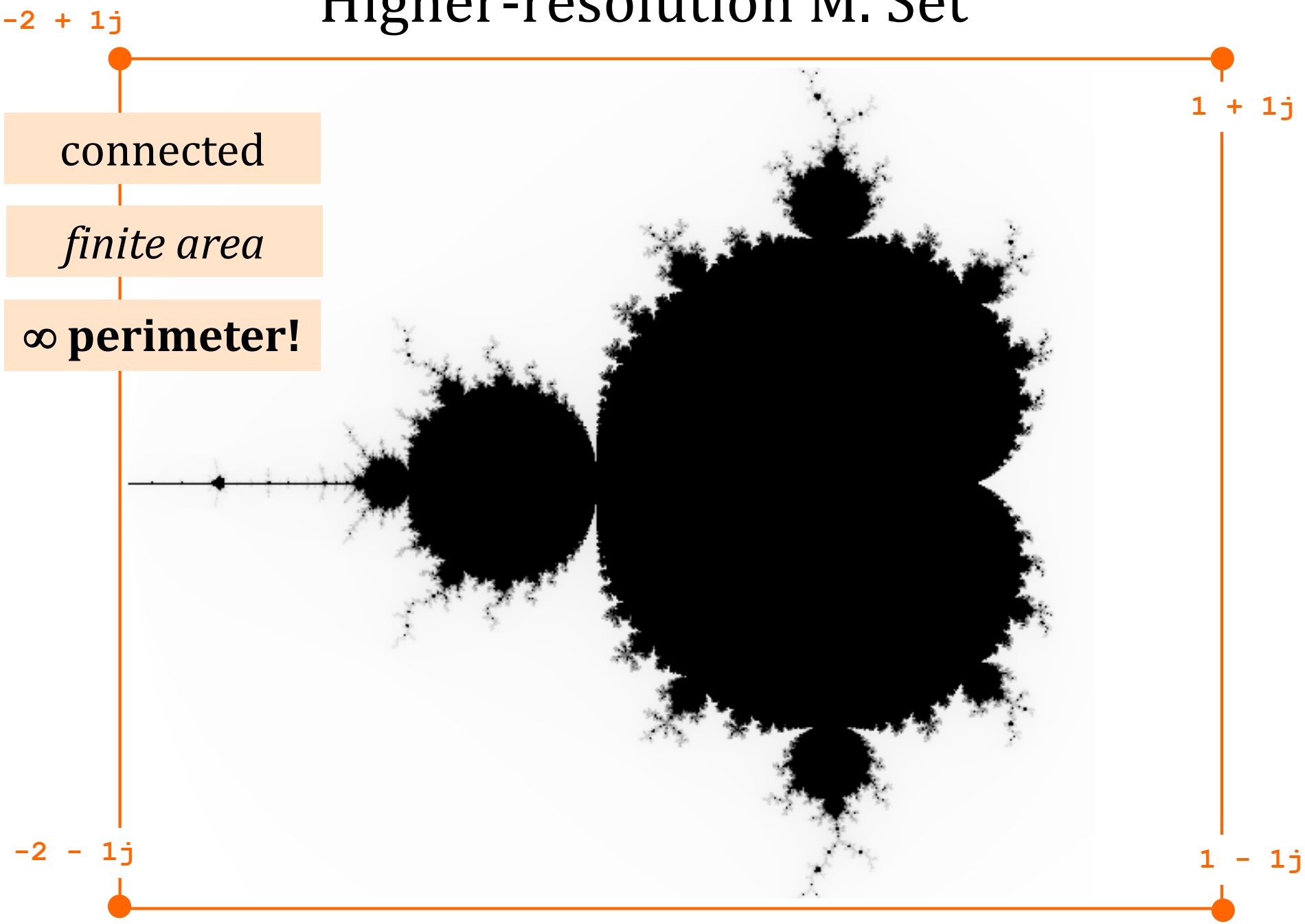
other c's diverge

Mandelbrot Set ~ *points that stick around*



The shaded area are points that do ***not*** diverge for `z = z**2 + c`

Higher-resolution M. Set



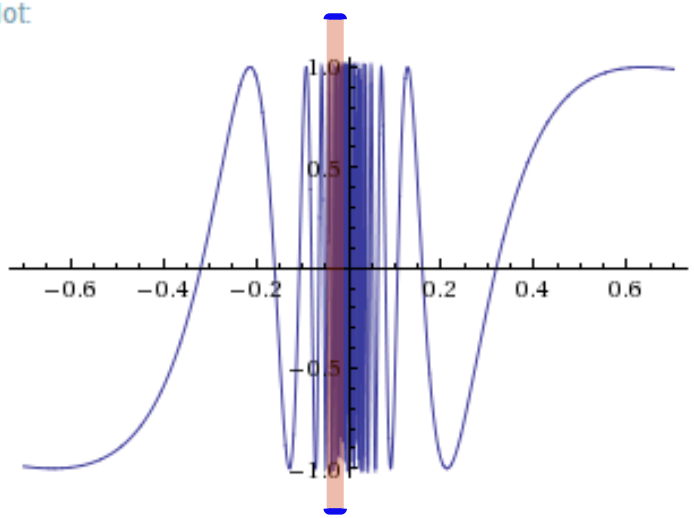
The black pixels are points that do ***not*** diverge for `z = z**2 + c`

Chaos?

Input interpretation:

plot	$y = \sin\left(\frac{1}{x}\right)$	$x = -0.7$ to 0.7

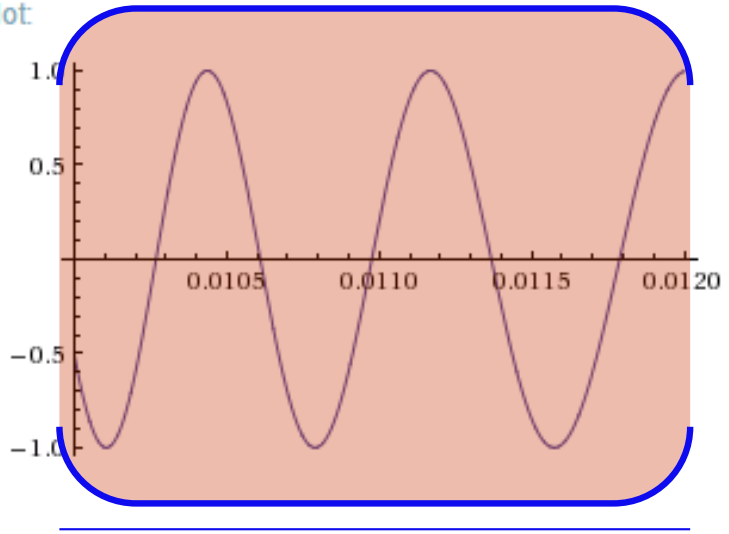
Plot:



Input interpretation:

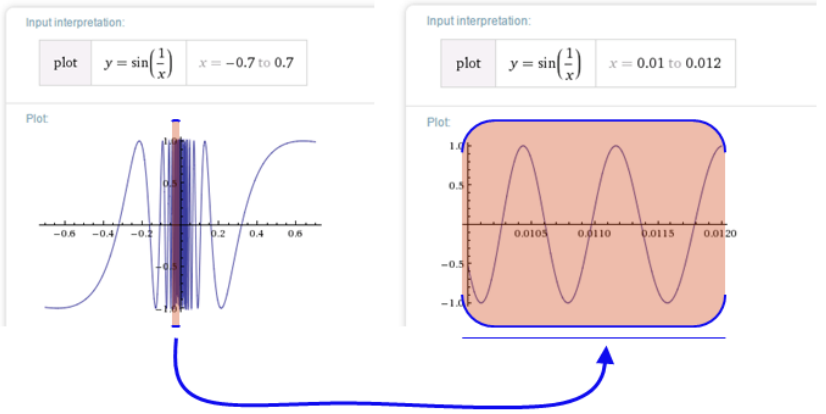
plot	$y = \sin\left(\frac{1}{x}\right)$	$x = 0.01$ to 0.012

Plot:



Complex things always consisted of simple parts…

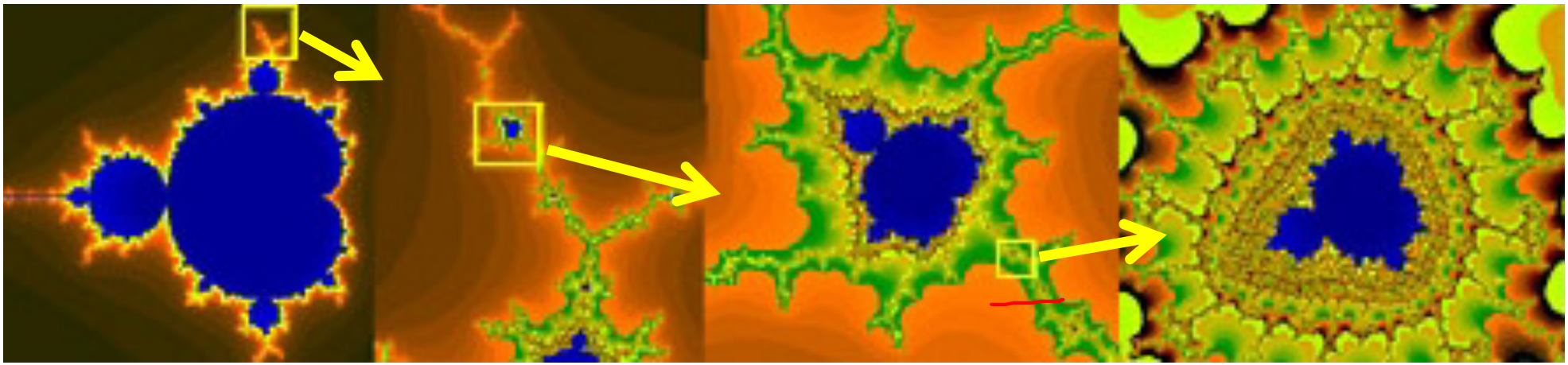
Before the M. Set, complex things were made of simple parts:



Chaos!

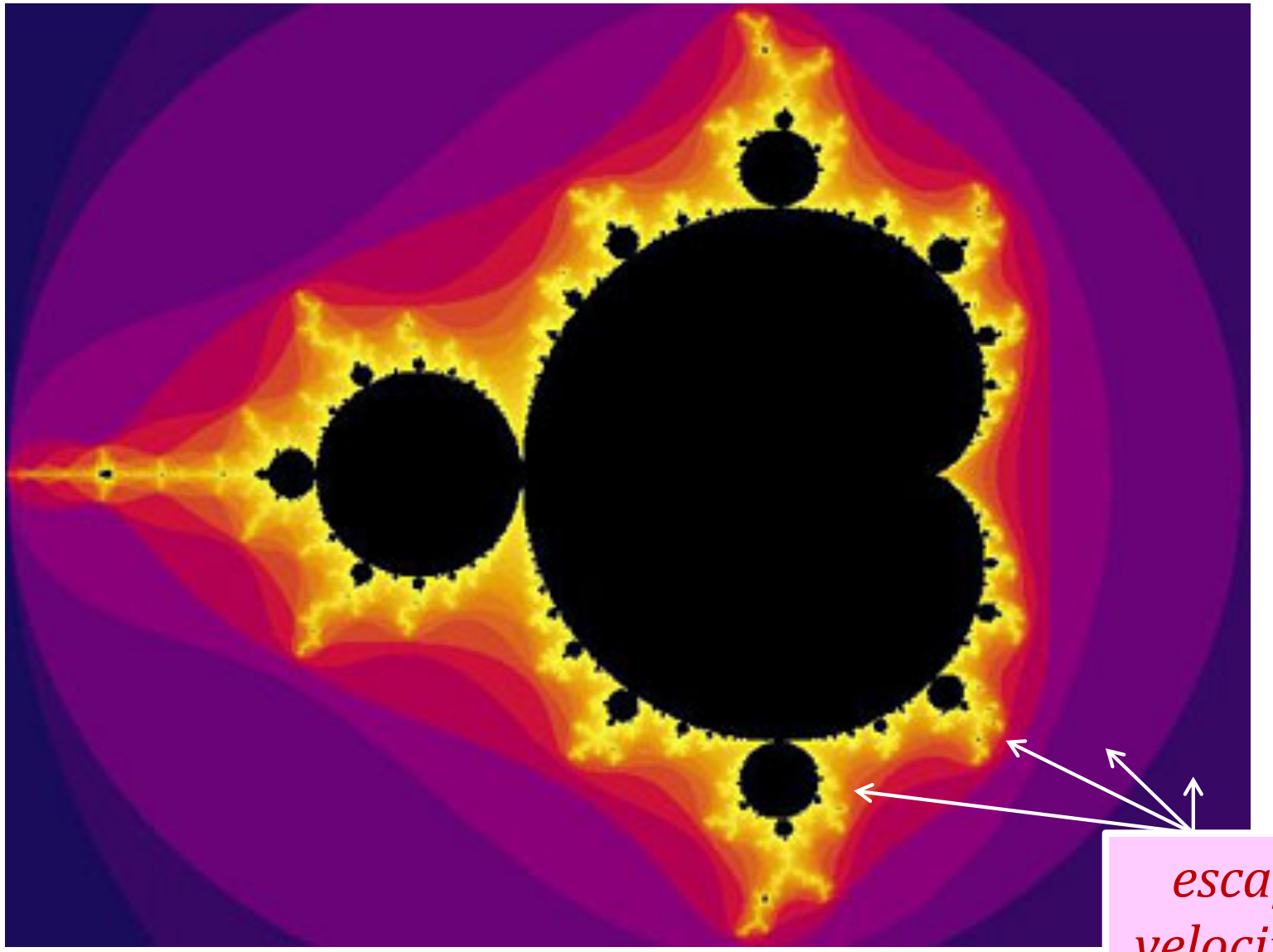
This was a *"naturally occurring"* object where zooming uncovers ***more*** detail, not less:

not self-similar but ***quasi***-self-similar



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

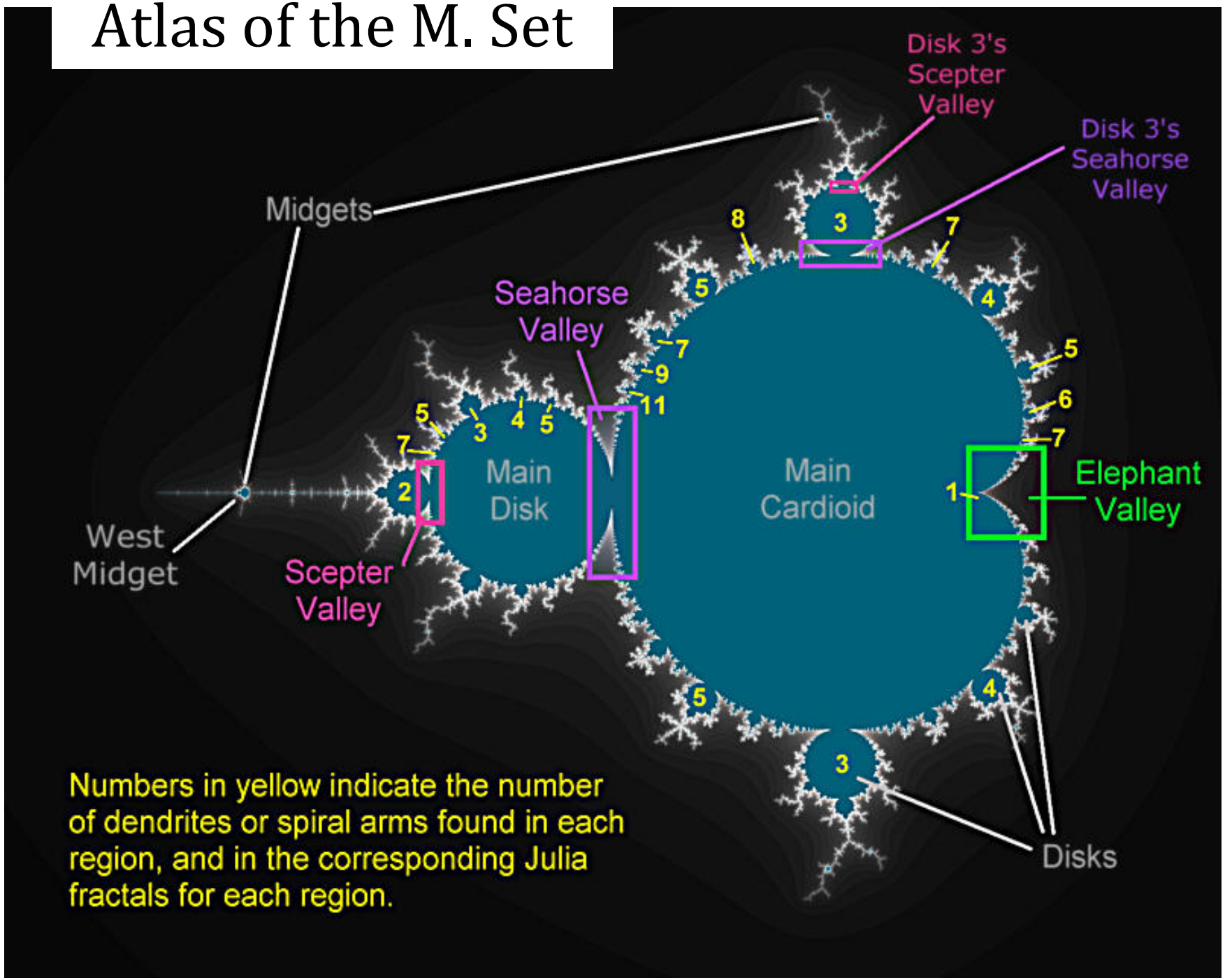
The black pixels are points that do ***not*** diverge for `z = z**2 + c`



What are these colors?

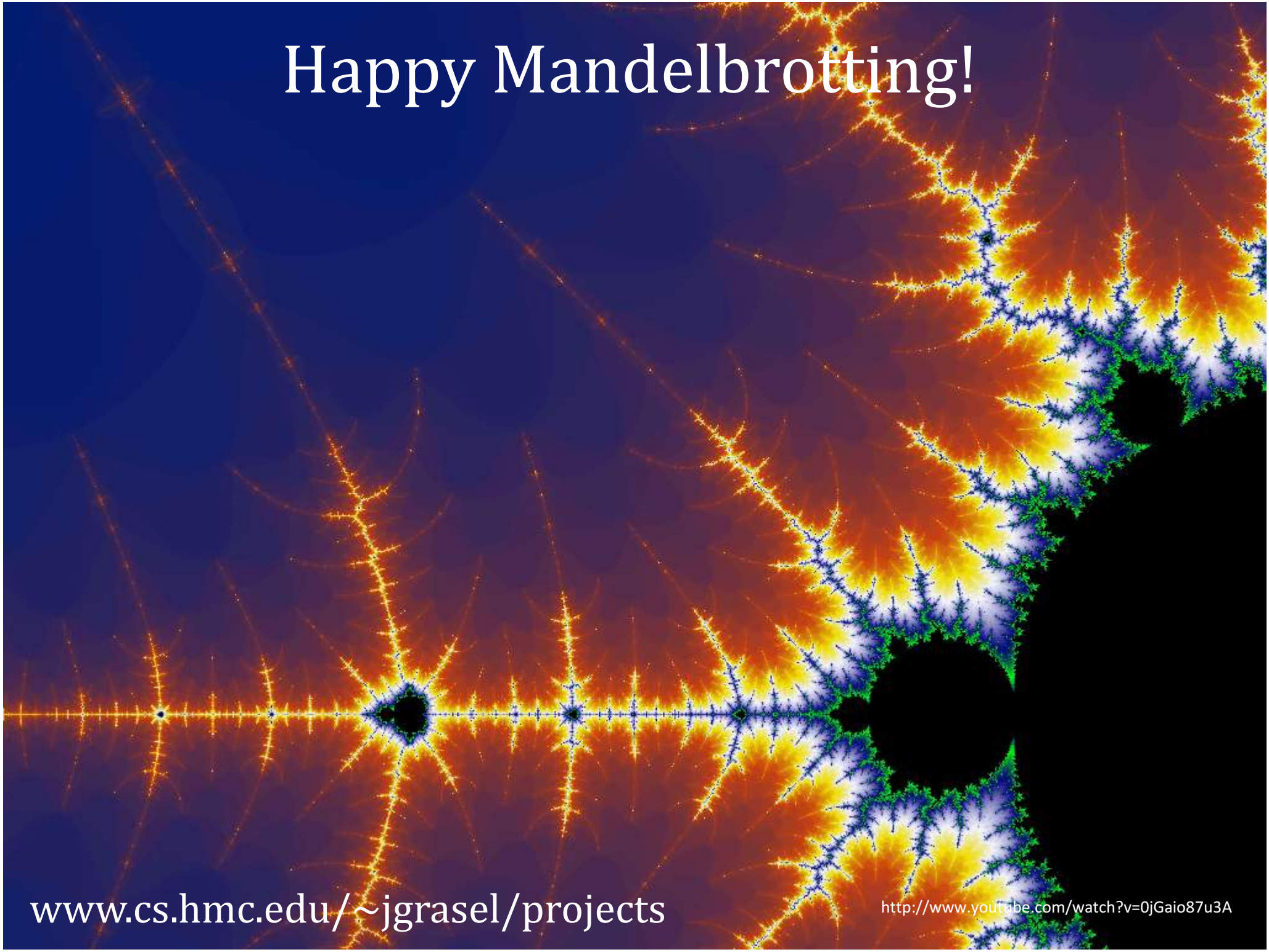
??

Atlas of the M. Set




In the *Seahorse Valley*....

Happy Mandelbrotting!



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

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