HMC Summer Research Celebration

Curious about research opportunities at HMC?

Want to learn more about your friend’s summer project?

Come to the poster session to learn about projects happening across campus!

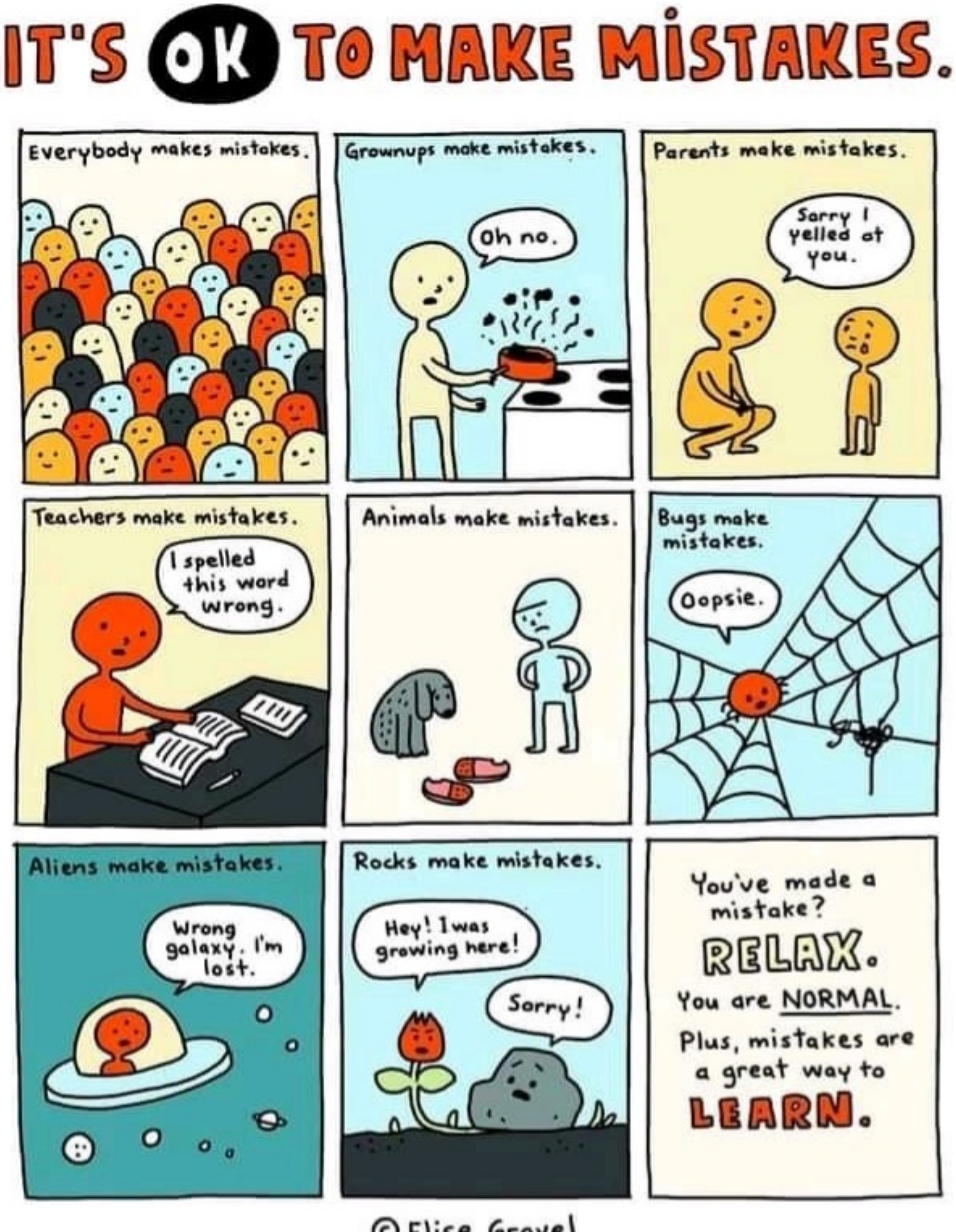
Thursday Sept 23
Drop by anytime between
4:30 - 6:30 pm

Attendees are eligible to win raffle prizes

Zoom Meeting 868 2909 2950, Passcode D5sDRH

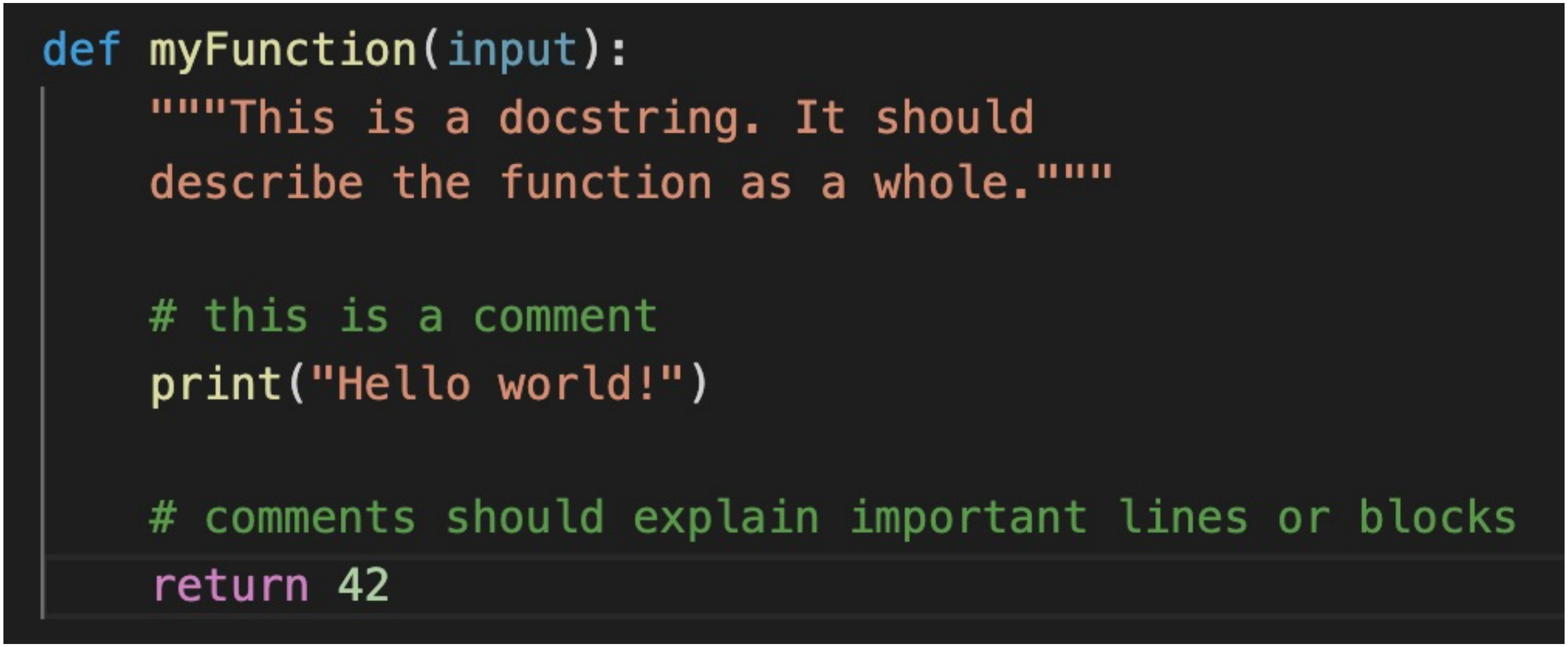
Make sure you have the latest version of Zoom

Common Mistakes



© Elise Gravel

Common Mistakes (HW1)



```
def myFunction(input):
    """This is a docstring. It should
    describe the function as a whole."""

    # this is a comment
    print("Hello world!")

    # comments should explain important lines or blocks
    return 42
```

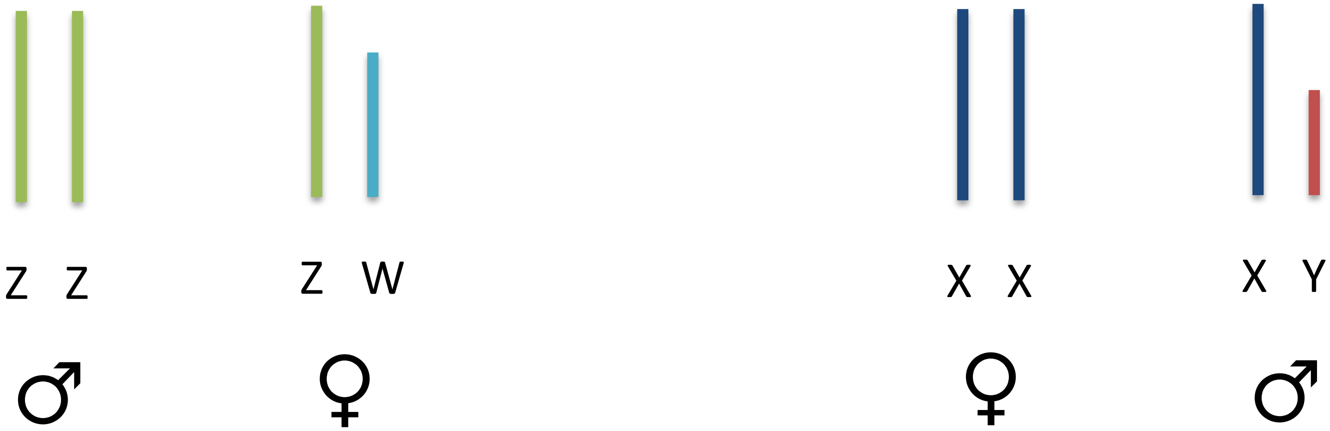
You ***will*** be graded on style and readability.

- Docstrings
- Comments
- Purposeful variable names

Evolution of sex determination systems

http://en.wikipedia.org/wiki/American_alligator
http://en.wikipedia.org/wiki/Amphiprioninae
http://commons.wikimedia.org/wiki/File:Male_mallard_duck_3.jpg
http://en.wikipedia.org/wiki/Hippopotamus

Chromosomal sex determination in birds and mammals

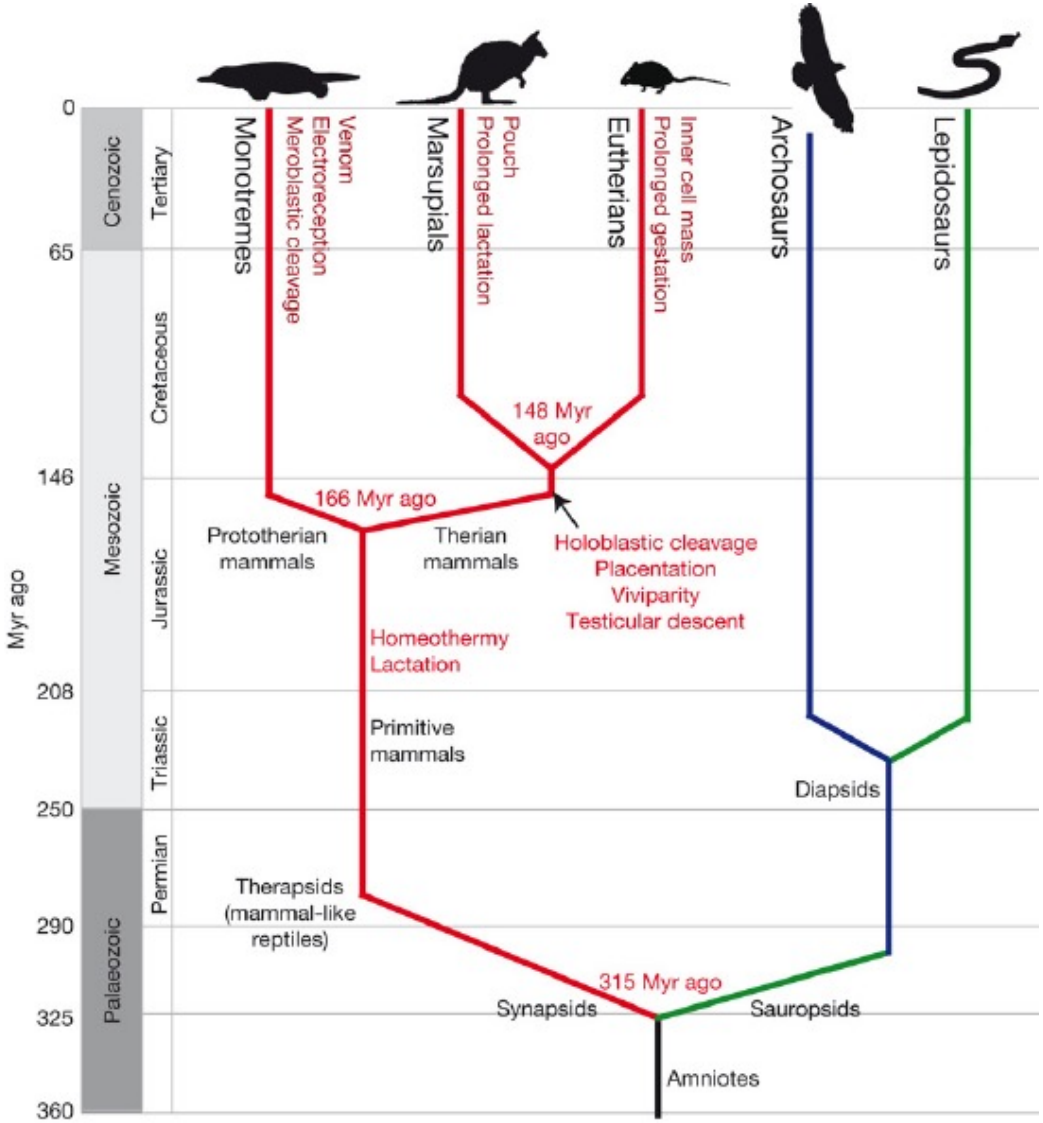


Do these sex-determination systems share a common ancestor or did they evolve independently???

Characteristics shared by descent are homologous

http://www.flickr.com/photos/sunstones/2664993674/
http://www.flickr.com/photos/nycgeo/1065447484/sizes/z/in/photostream/
http://www.flickr.com/photos/bbum/98144389/sizes/z/in/photostream/
http://commons.wikimedia.org/wiki/File:Mother_And_Baby_Elephant.jpg

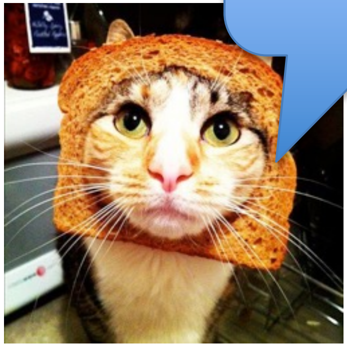
Is the mammalian X homologous to the avian Z?



Nature 453, 175-183 fig1

Some things that “require” more than `for` and `while` loops

I’d be “toast” if I had to do this right now!

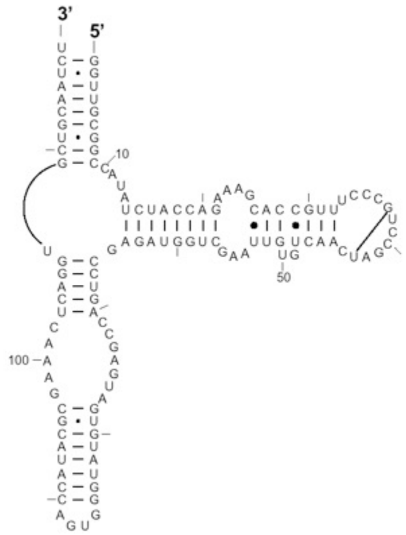


Measure similarity of two DNA sequences…

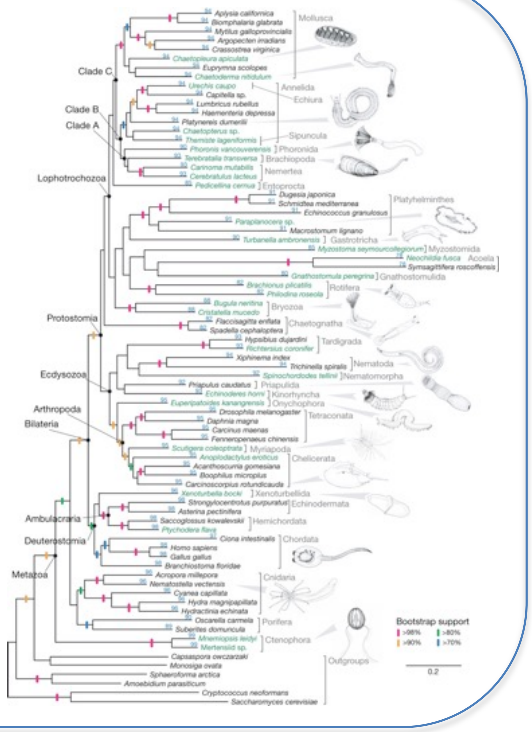
GGGACTCACTCATCAGTT

CACTCATTTGCAGTCATG

Predict how an RNA sequence will fold…



Compute and draw phylogenetic trees…

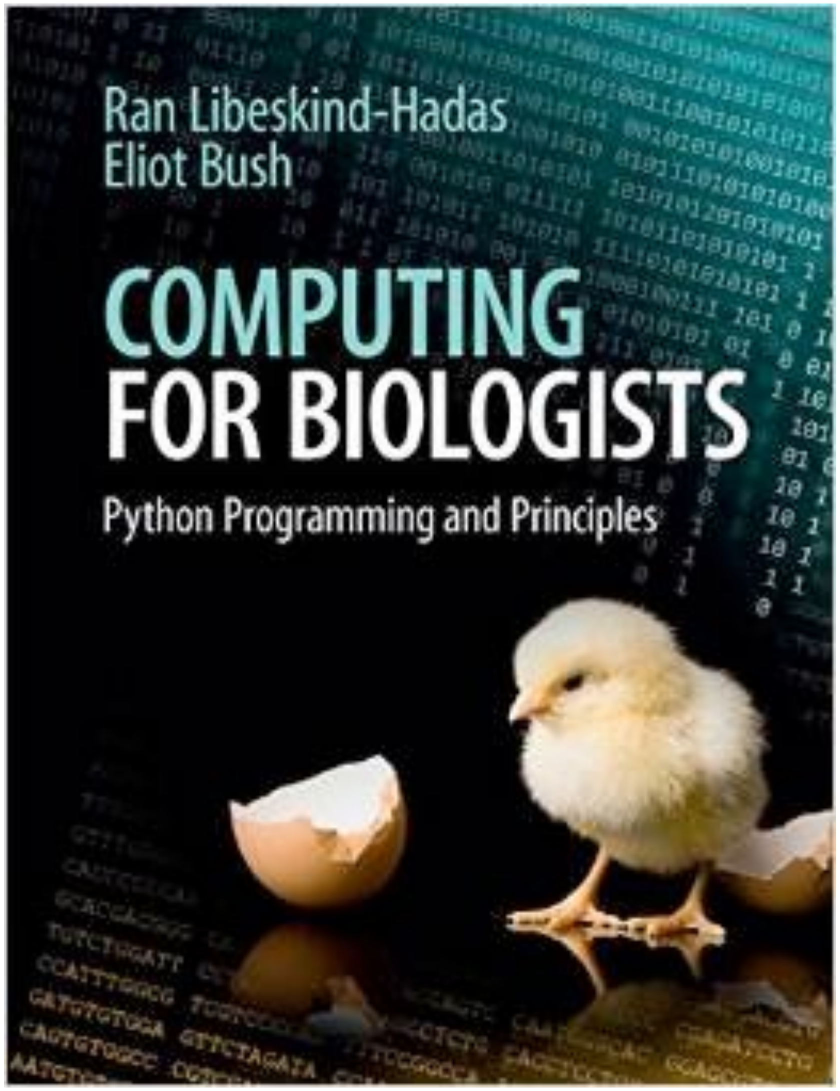


CS 5 Green

Learning Goals

- Describe the concept of recursion
- State the difference between the base case and recursive case
- Practice recursion
- State the tradeoffs between recursion and iteration

This week's reading…



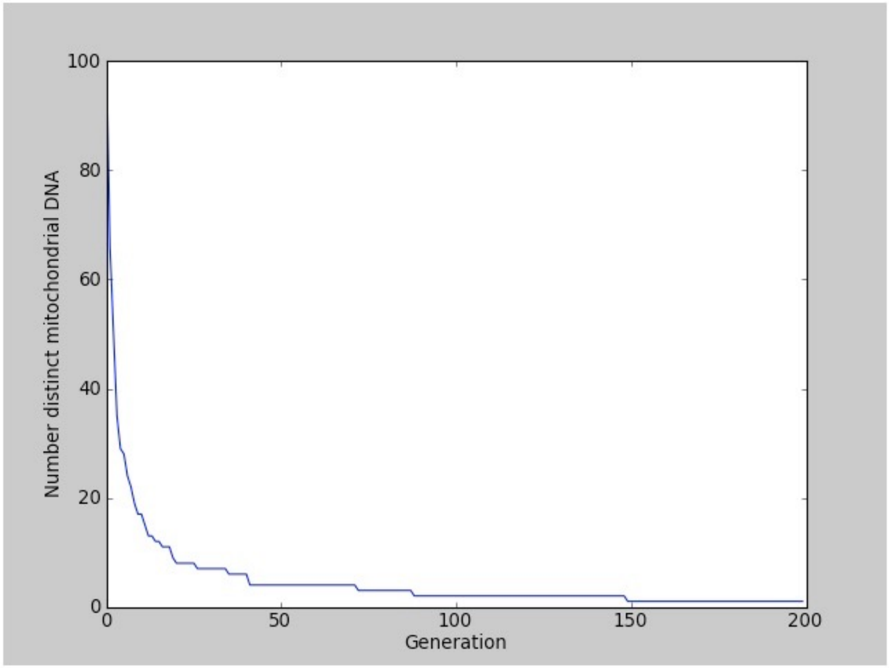
Chapter 5: Recursion

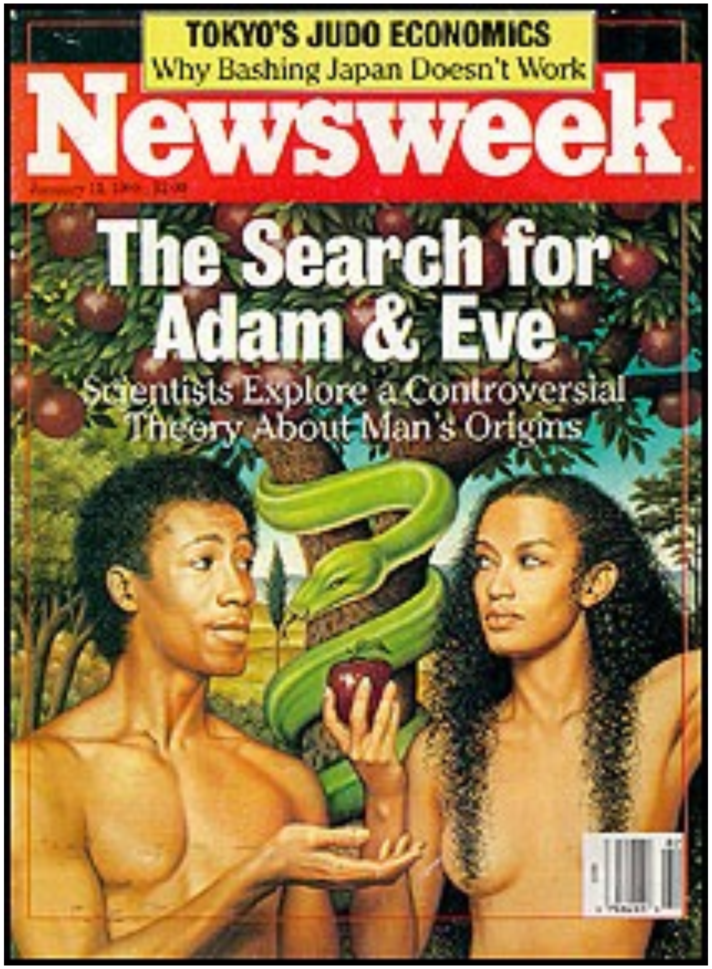
only 11 pages (dbl spaced!)

https://2.flexiple.com/free-illustrations/reading-newspaper

This week's homework…

Mitochondrial Eve!



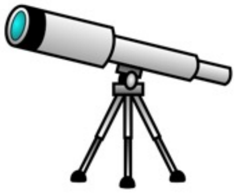

TOKYO'S JUDO ECONOMICS
Why Bashing Japan Doesn't Work
Newsweek
The Search for Adam & Eve
Scientists Explore a Controversial Theory About Man's Origins

This week's homework…

Got Milk?

Recursion! (Lab and Bonus)

A word about “scope”



```
def joe(x):
    y = bjorn(x)
    z = x + y
    return z

def bjorn(x):
    x = 42
    return 2
```

```
>>> joe(1)
```

What does `joe(1)` return?

What Happens Inside a Function?

```
def h(x):
    return f(x) + x

def f(x):
    x = x-1
    return g(x)+1

def g(x):
    return x*2
```

Remember, each function has its own private variables!

Two key points…

- Functions return to where they were called from
- Each function keeps its own values of its variables

Factorial (iterative)

```
n! = n × (n-1) × (n-2) × … × 1
```

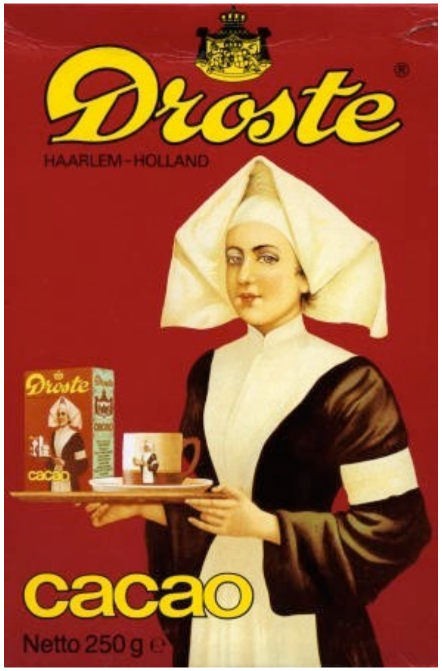
```
def factorial(n):
    # initialize result
    result = 1

    # multiply each number between 1 and n
    for curNum in range(1, n+1):
        result = result * curNum

    return result
```

Using loops to solve problems is called **iteration.**

Factorial (recursive)



iterative solution

$n! = n \times (n-1) \times (n-2) \times \ldots \times 1$

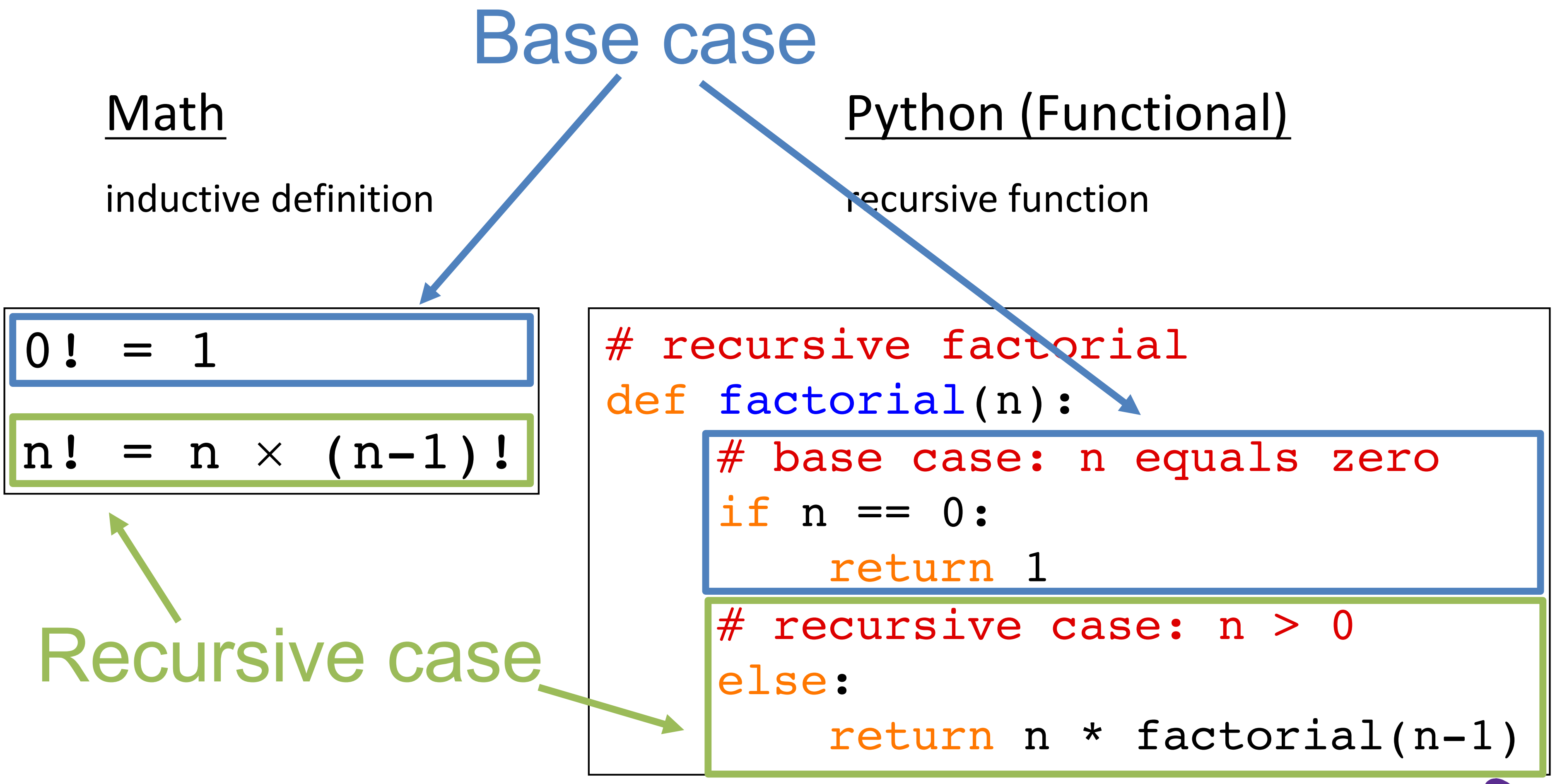
$n! = n$ x $(n-1)!$ “recursive case”

$0! = 1$ “base case”

recursive solution

Recursive function: a function which includes **itself** as part of its definition.

Factorial (recursive)



Base case

Math

inductive definition

```
0! = 1
n! = n × (n-1)!
```

Python (Functional)

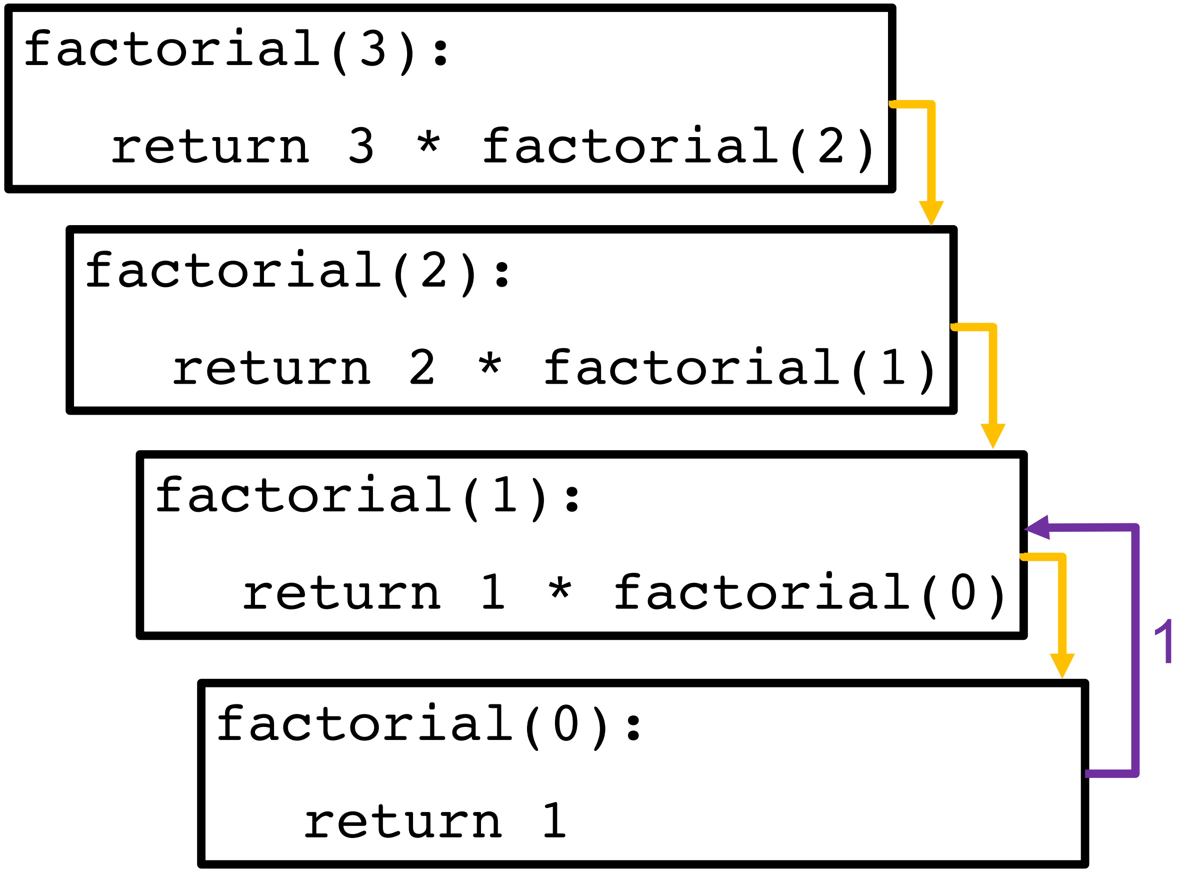
recursive function

```
# recursive factorial
def factorial(n):
    # base case: n equals zero
    if n == 0:
        return 1
    # recursive case: n > 0
    else:
        return n * factorial(n-1)
```

Recursive case

The input to the recursive call is **simpler** than the original input!!

Is Recursion Magic?



```
factorial(3):
   return 3 * factorial(2)
```

```
factorial(2):
   return 2 * factorial(1)
```

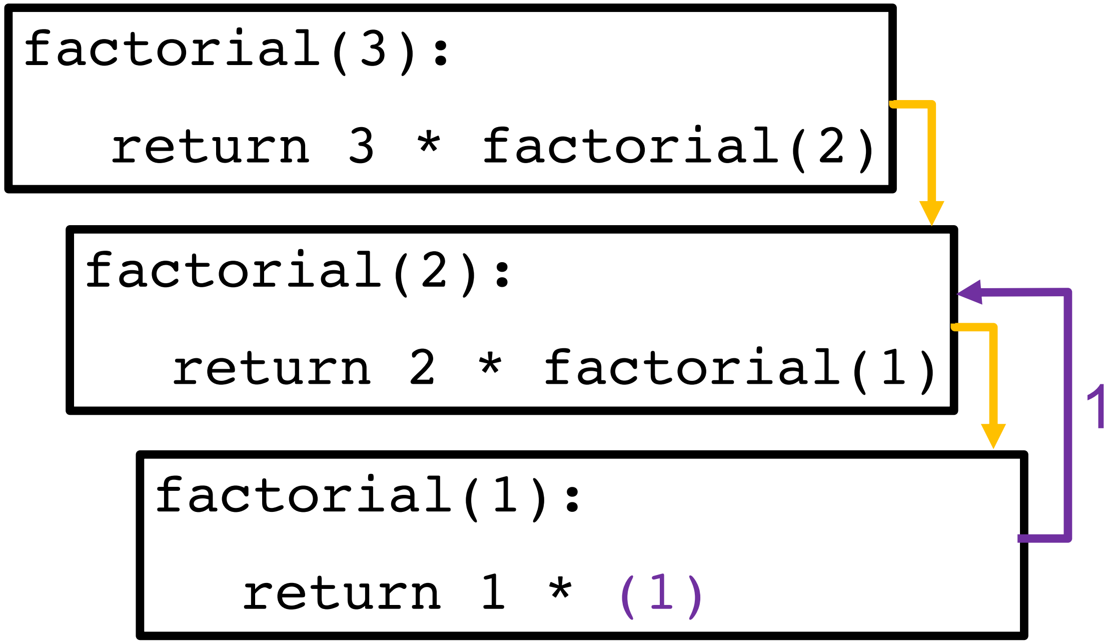
```
factorial(1):
   return 1 * factorial(0)
```

```
factorial(0):
   return 1
```

1

```
# recursive factorial
def factorial(n):
    if n == 0:
        return 1
    else:
        return n*factorial(n-1)
```

Is Recursion Magic?



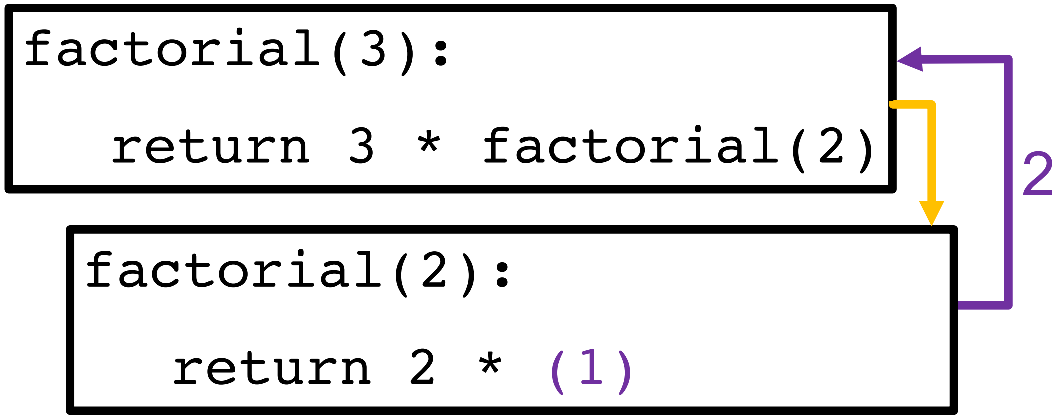
```
factorial(3):
   return 3 * factorial(2)
```

```
factorial(2):
   return 2 * factorial(1)
```

```
factorial(1):
   return 1 * (1)
```

```
# recursive factorial
def factorial(n):
    if n == 0:
        return 1
    else:
        return n*factorial(n-1)
```

Is Recursion Magic?

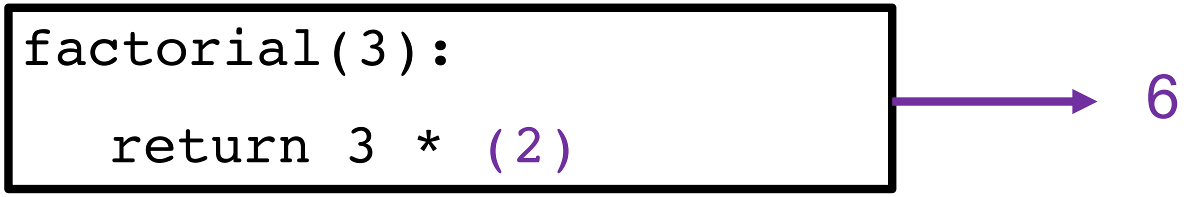


```
factorial(3):
   return 3 * factorial(2)
```

```
factorial(2):
   return 2 * (1)
```

```
# recursive factorial
def factorial(n):
    if n == 0:
        return 1
    else:
        return n*factorial(n-1)
```

Is Recursion Magic?



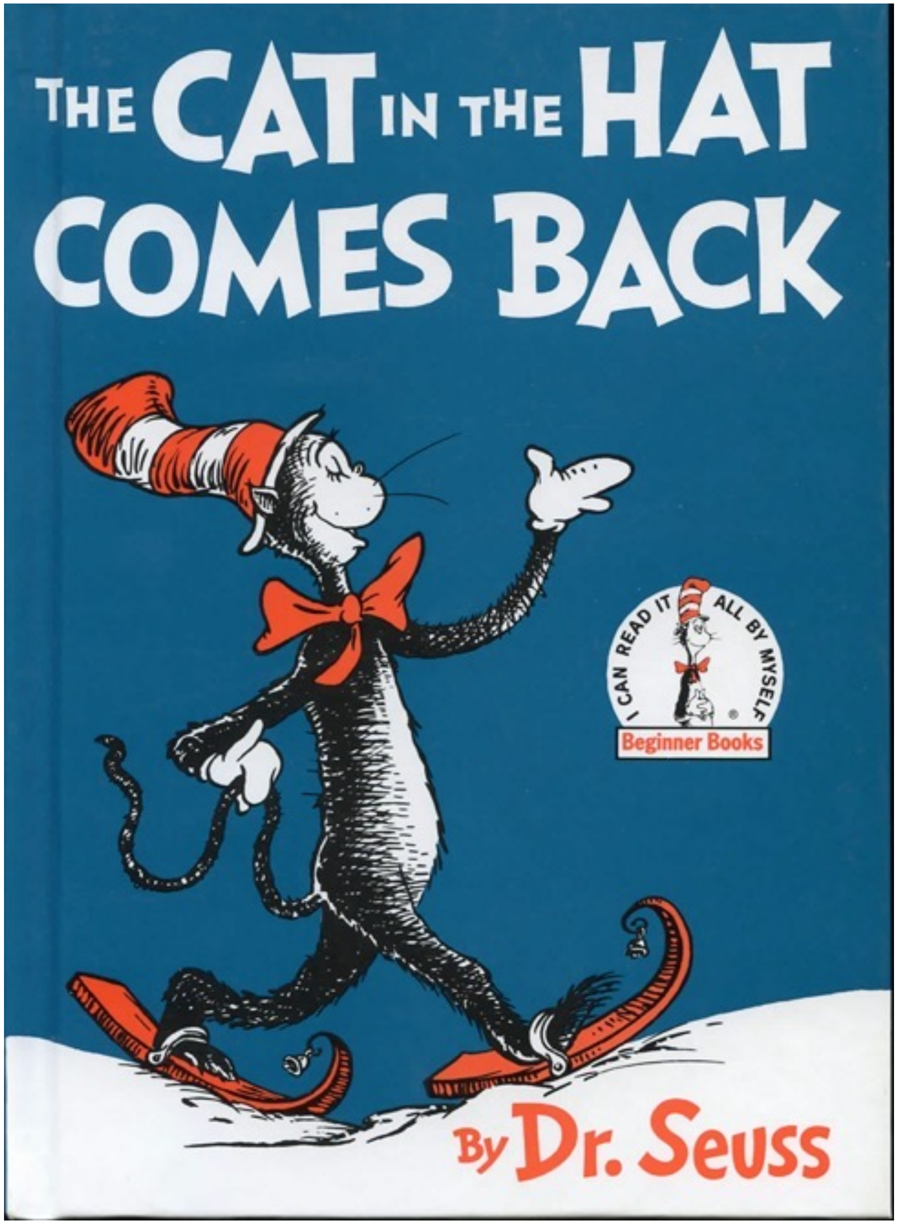
```
factorial(3):

   return 3 * (2)
```

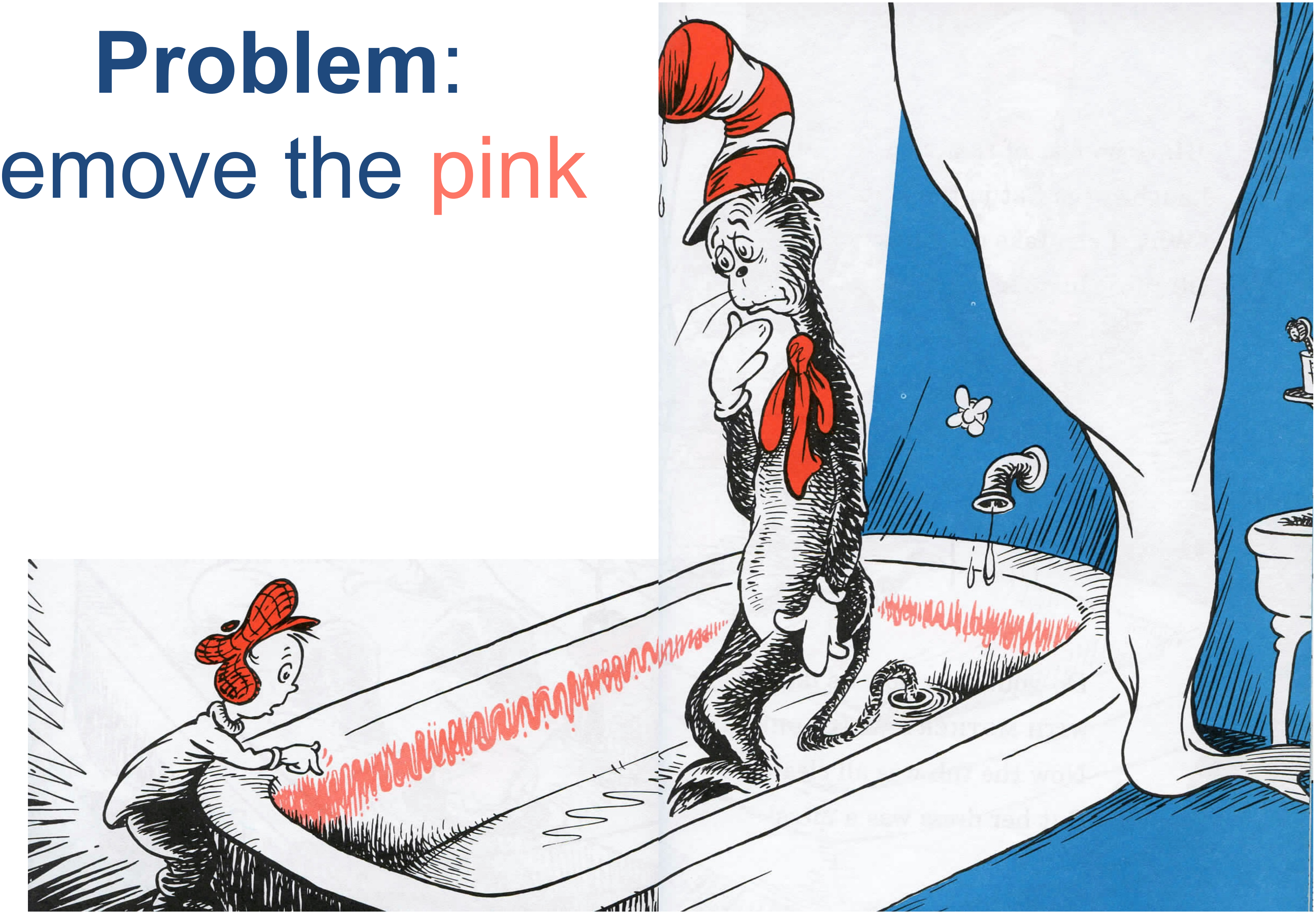
6

```
# recursive factorial
def factorial(n):
    if n == 0:
        return 1
    else:
        return n*factorial(n-1)
```

Recursion in a children's book



Problem:
remove the pink

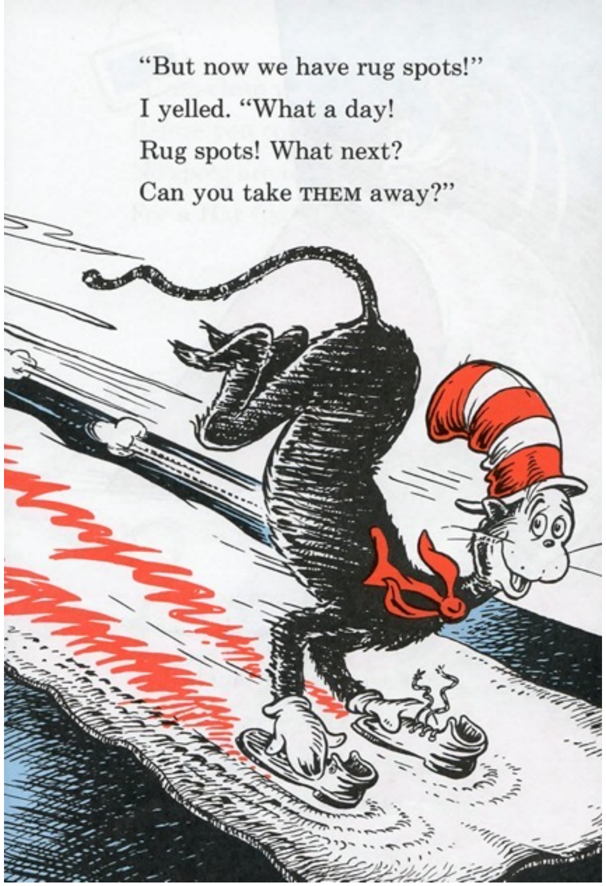


Non-recursive attempts fail

But the cat laughed, "Ho! Ho!
I can make the spot go.
The way I take spots off a dress
Is just so!"

20

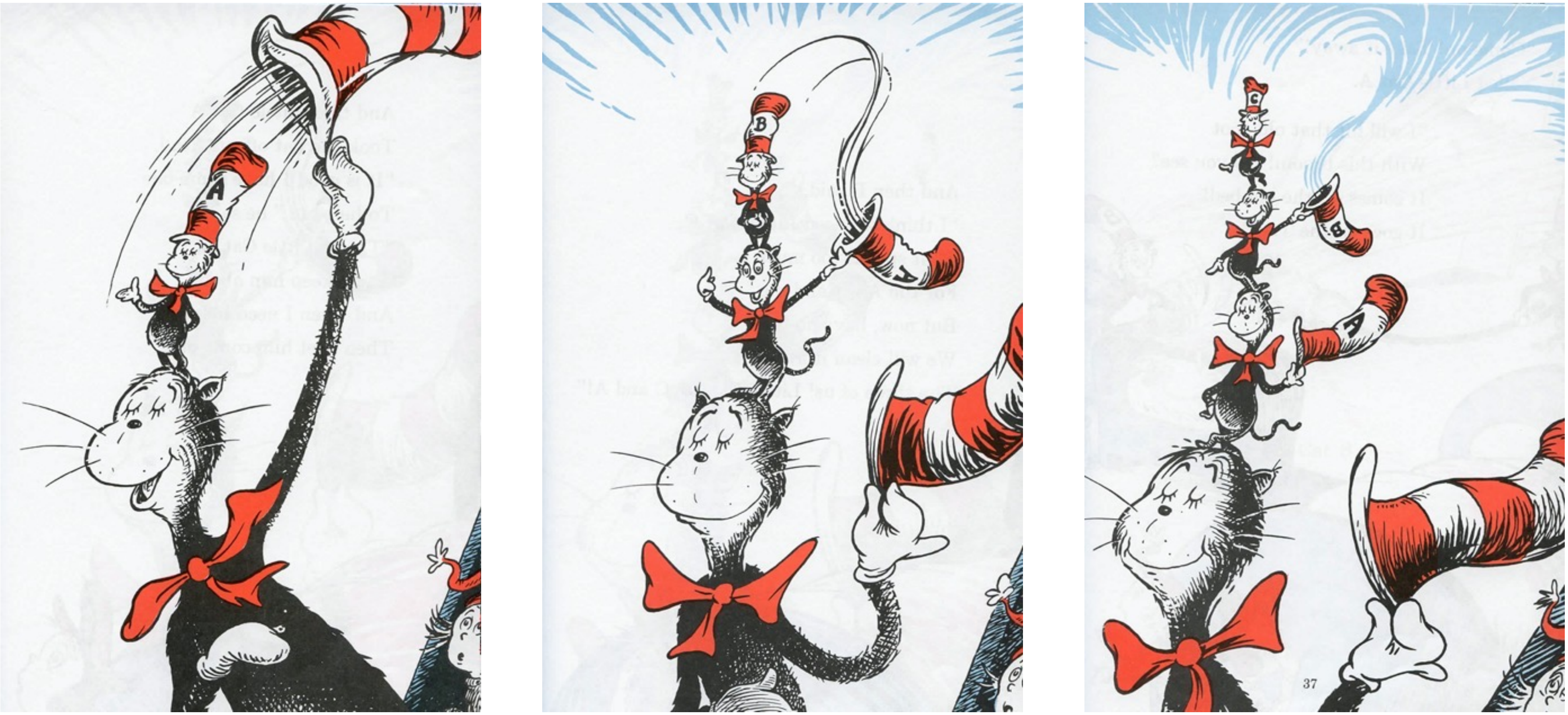
Non-recursive attempts fail



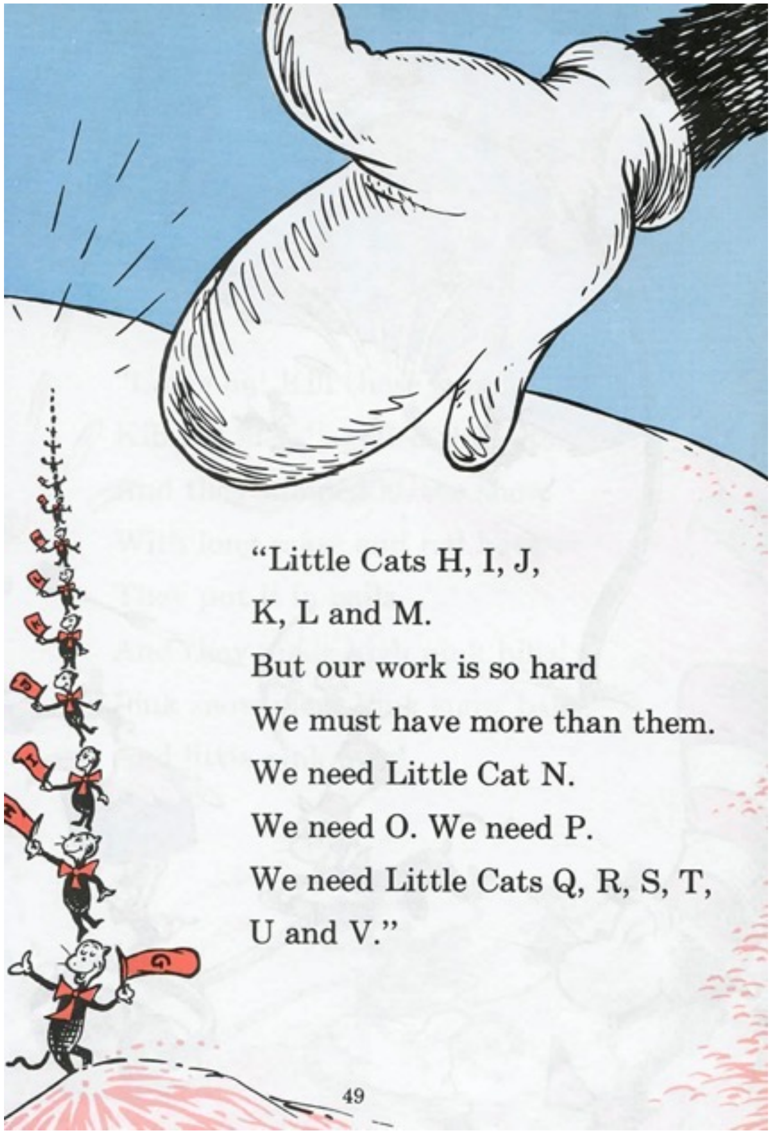
"But now we have rug spots!"
I yelled. "What a day!
Rug spots! What next?
Can you take THEM away?"

He ran into Dad's bedroom
And then the cat said,
"It is good that your dad
Has the right kind of bed."

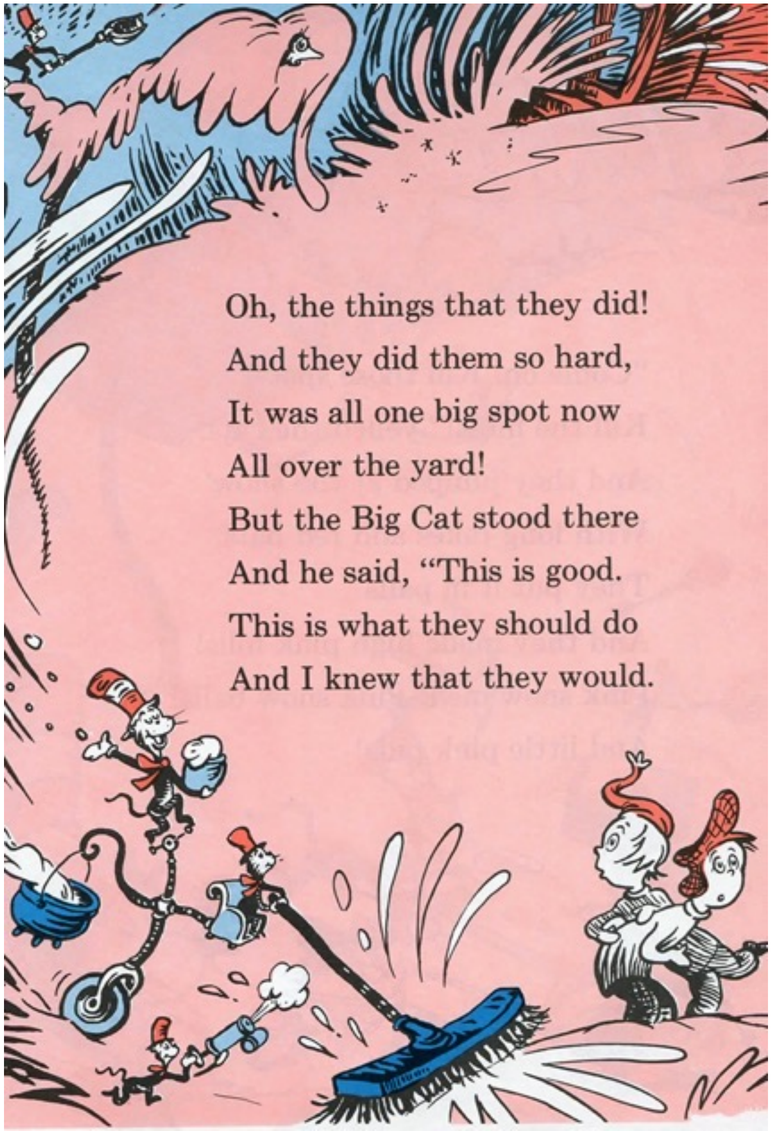
The Cat implements recursion



The Cat implements recursion

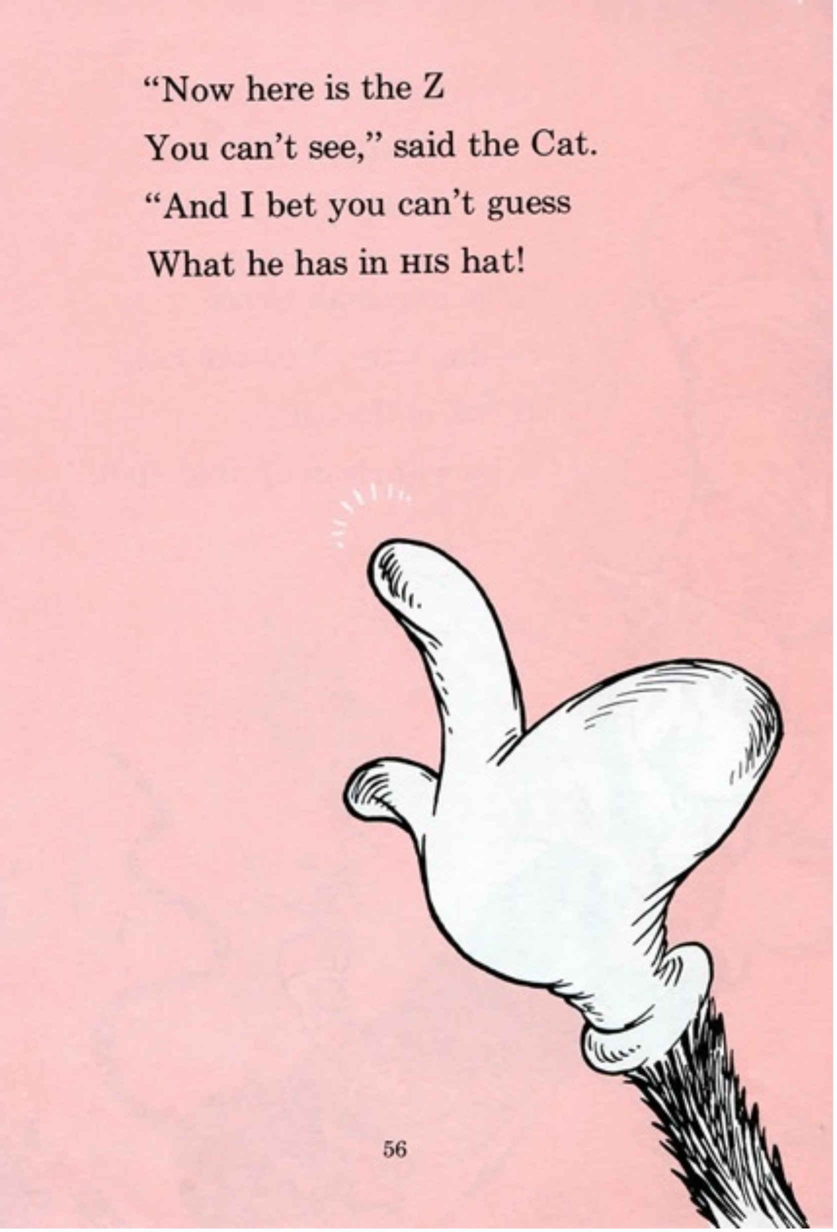


"Little Cats H, I, J,
K, L and M.
But our work is so hard
We must have more than them.
We need Little Cat N.
We need O. We need P.
We need Little Cats Q, R, S, T,
U and V."

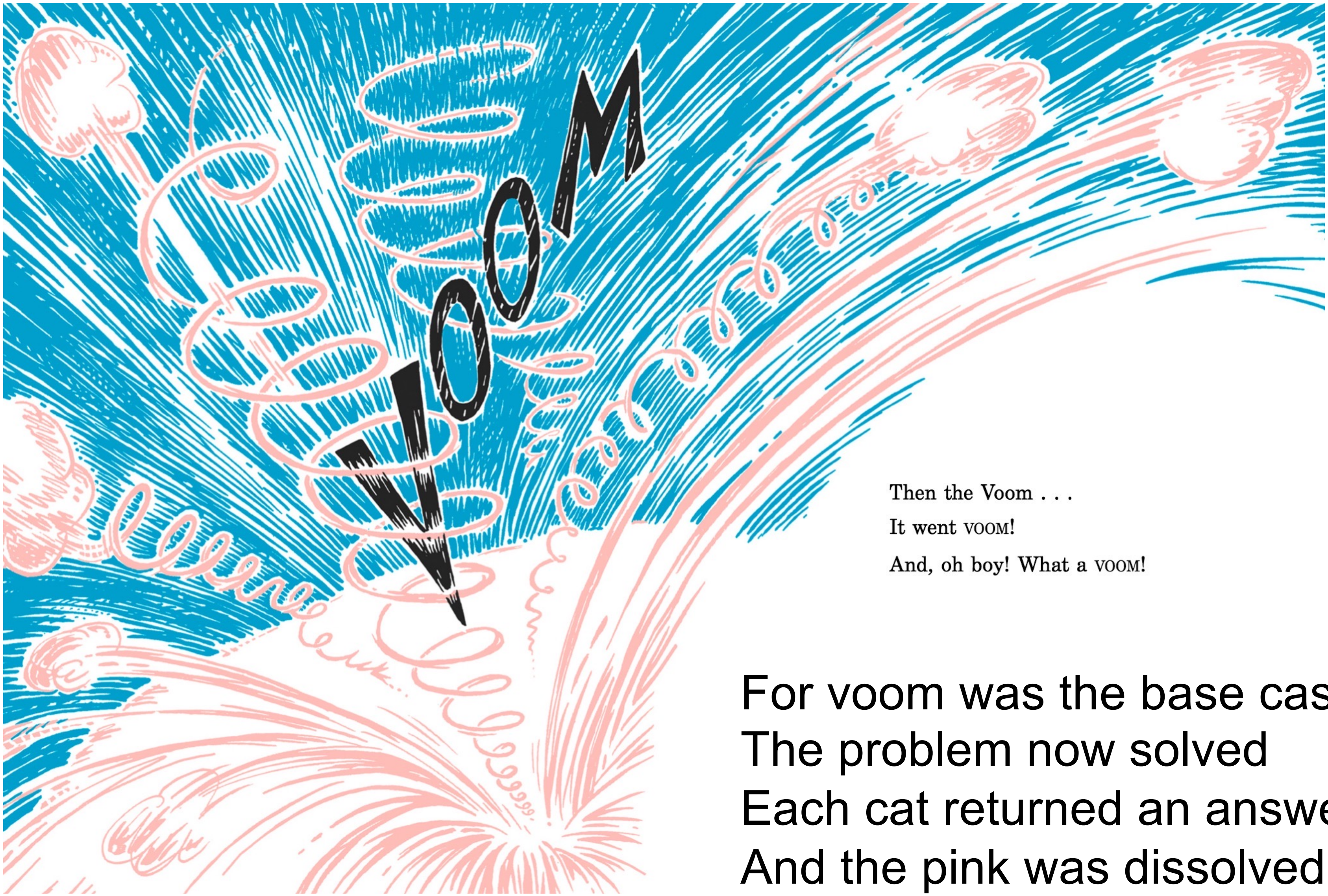


Oh, the things that they did!
And they did them so hard,
It was all one big spot now
All over the yard!
But the Big Cat stood there
And he said, "This is good.
This is what they should do
And I knew that they would.

Cat Z reaches the base case



“Now here is the Z
You can’t see,” said the Cat.
“And I bet you can’t guess
What he has in HIS hat!



Then the Voom . . .
It went VOOM!
And, oh boy! What a VOOM!

For voom was the base case
The problem now solved
Each cat returned an answer
And the pink was dissolved

Worksheet

Q

Computing the length of a list

```
>>> len([1, 42, "spam"])
3
>>> len([1, [2, [3, 4]]])
2
```

Python has this built-in!

```
def len(inputL):
    '''Returns the length of a list’’’
```

Summing up the numbers in a list

```
>>> sum([1, 42, 7])
50
>>> sum([42])
42
>>> sum([])
0
```

Python has this built-in too!

```
def sum(inputL):
    '''Returns the sum of numbers in a list’’’
```

No new variables required!

```
def len(inputL):
    '''RECURSIVE VERSION'''
    if inputL == []:
        return 0
    else:
        return 1 + len(inputL[1:])

def lenV2(inputL):
    ’’ITERATIVE VERSION'''
    counter = 0  # a new variable!
    for x in inputL: # another new variable
        counter += 1
    return counter
```

Intermediate values stored in “stack frames” instead!

[🐢, 🐢, 🐢] Reversing a list

Q

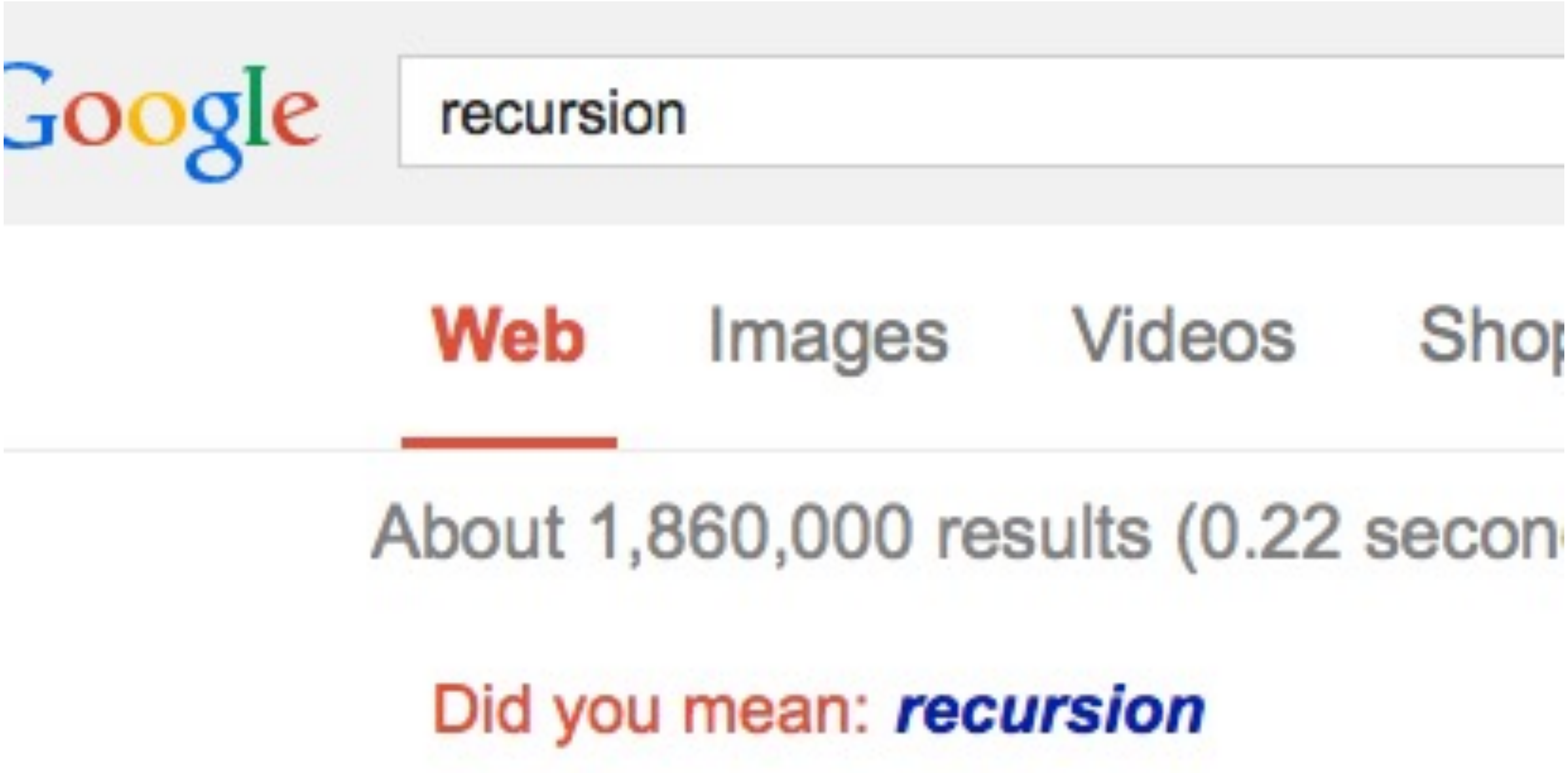
```
>>> reverse([1, 2, 3, 4])
[4, 3, 2, 1]

def reverse(inputL):
    '''reverses the order of a list'''
```

Recursion <(˚ε˚<)

"To understand recursion, you must first understand recursion"
- anonymous Mudd alum

Recursion <(˚ε˚<)

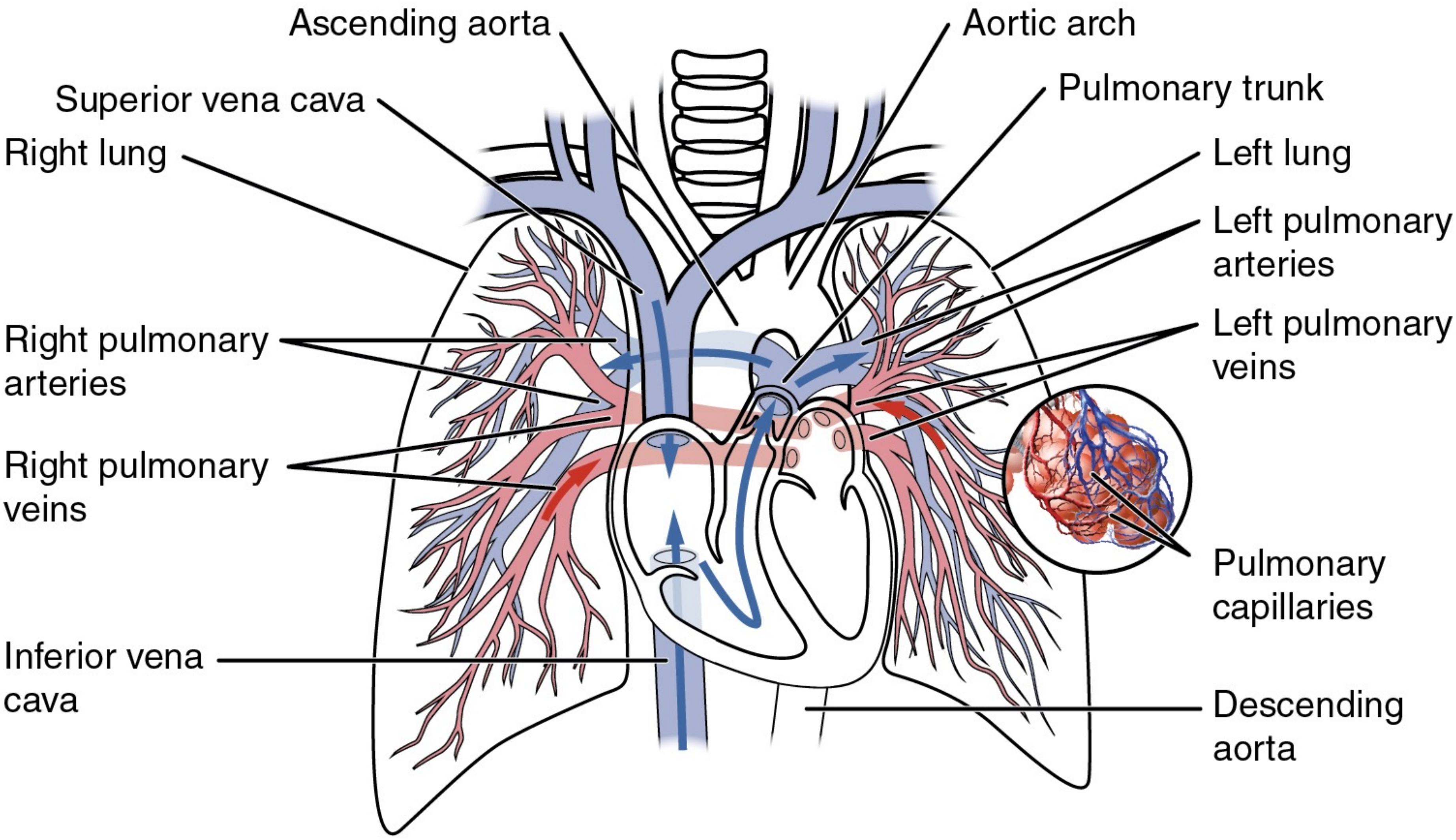


recursion

About 1,860,000 results (0.22 secon

Did you mean: ***recursion***

Recursion in nature

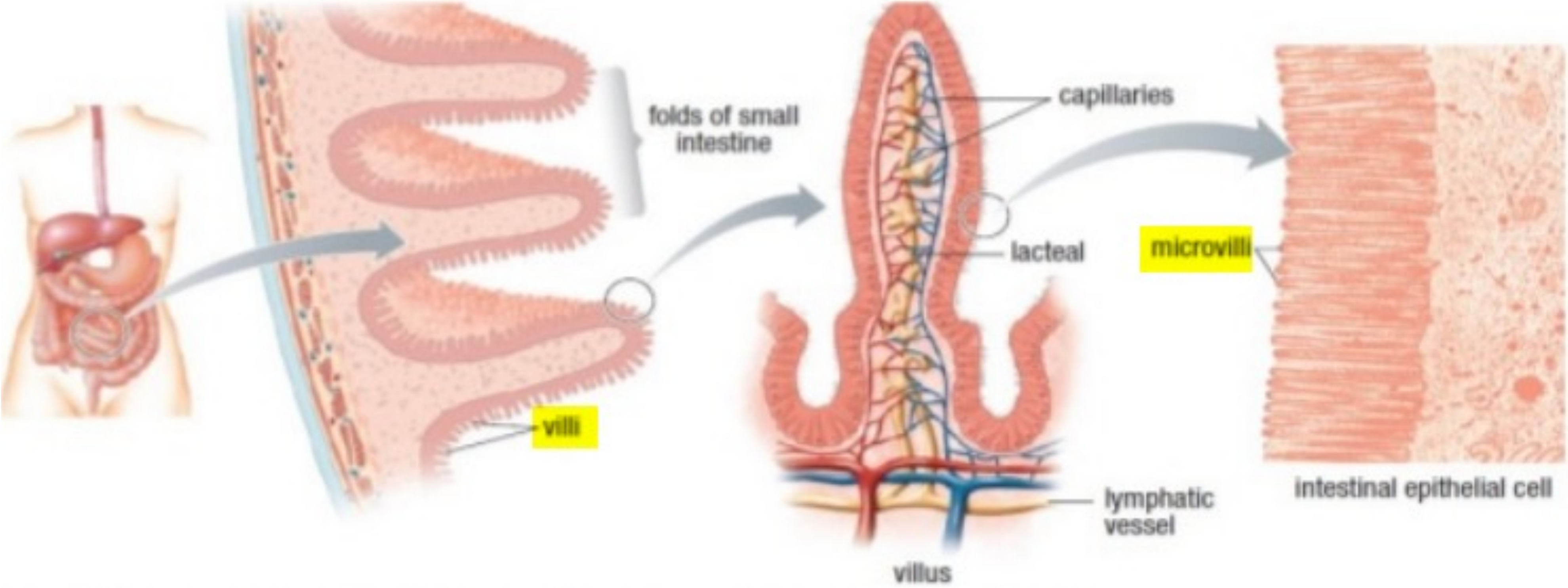


https://commons.wikimedia.org/w/index.php?curid=30148269

Recursion in nature

https://commons.wikimedia.org/w/index.php?curid=6777039

Recursion in nature



https://laboratoryinfo.com/microvilli/

Recursion in nature

The following pages have a number of exercises for you to do (in your notes). You’re welcome to work at your own pace.

```
min
member
pal

*insert/sort*
```

Minimum!

```
>>> min([372, 112, 42, 451])
42
>>> min([16])
16
```

Assume that the input list will never be empty! Use len as a helper function!

```
def min(inputL):
    '''Returns smallest value in a list'''
```

member

Q

```
>>> member(42, [1, 3, 5, 42, 7])
True
>>> member(42, ['spam', 'is', 'yummy'])
False
```

This is sort of like the "in" thing in Python, but don't use "in" here. Just list indexing, slicing, and recursion!

```
def member(thing, inputL):
    '''Return True if thing in inputL
    and False otherwise.'''
```

Palindrome?

```
>>> pal('radar')
True
>>> pal('amanaplanacanalpanama')
True
>>> pal('spam')
False
```

```
def pal(s):
    '''Returns True if s is a palindrome
    and False otherwise'''
```

Insertion Sorting

Challenge Question

```
>>> sort([42, 57, 1, 3])
[1, 3, 42, 57]
```

The idea… Given a list like L = `[42, 57, 1, 3]`

- Slice off the first element. Now we have a shorter list… `[57, 1, 3]`
- Use recursion to sort that list. Now we have… `[1, 3, 57]`
- Now, insert L[0](Which is 42)into the right place in `[1, 3, 57]`…
  `[1, 3, 42, 57]`

```
def insert(x, sorted_list):
    '''Takes a number and sorted list as input and returns a new list
    that has x inserted into the right place in the sorted list'''

def sort(my_list):
   '''Sorts a list using insert as a helper function'''
```