

CompSci 101

Recursion



Susan Rodger
April 18, 2023



W is for ...

- **World Wide Web**
 - Where http meets tcp/ip?
- **WiFi**
 - We need and use this every day
- **Windows**
 - From OS to ...



Rediet Abebe



- **CS PhD '19 Cornell**
- **Harvard Fellow 19**
- **Now a Carnegie Fellow 22**
- **UC Berkeley Assist. Prof**
- **Research: AI, Inequality and Social Impact**
- **Co-founded Black in AI**
- **Co-founded Mechanism Design for Social Good**

“For the most part, algorithms didn’t create inequity and inequality, but the fact that we didn’t have people who were engaging with algorithms’ role was exacerbating this existing inequality. With any sort of social issue, an algorithm can make things a lot worse, or it can help you understand what’s going on better and try to move things in a positive direction.”

Announcements

- **APT-7 due in one week!**
- **Assign 6 Recommender, due Thursday**
 - Assign 6 Sakai quiz due tonight!
- **Assign 7 Create due, Wednesday, April 26!**
 - No penalty thru Sunday, April 30
- **Lab 11 Friday, do prelab!**

- **Final Exam – 9am, Thursday, May 4**
 - 3 hours, in person, covers topics through last day

Extra credit opportunity!

- **Fill out survey under Exam 3 Bonus in Sakai tests and quizzes**
 - If 65% fill out 1 extra credit point for Exam 3
 - If 75% fill out 1 additional extra credit point Exam 3
- **Right now at 44%**

More samples for Assignment 7

Compsci 101 Video



Video: APT Success



PFTD

- **Recursion**
 - Technique for solving a problem by solving smaller problems

Recursion

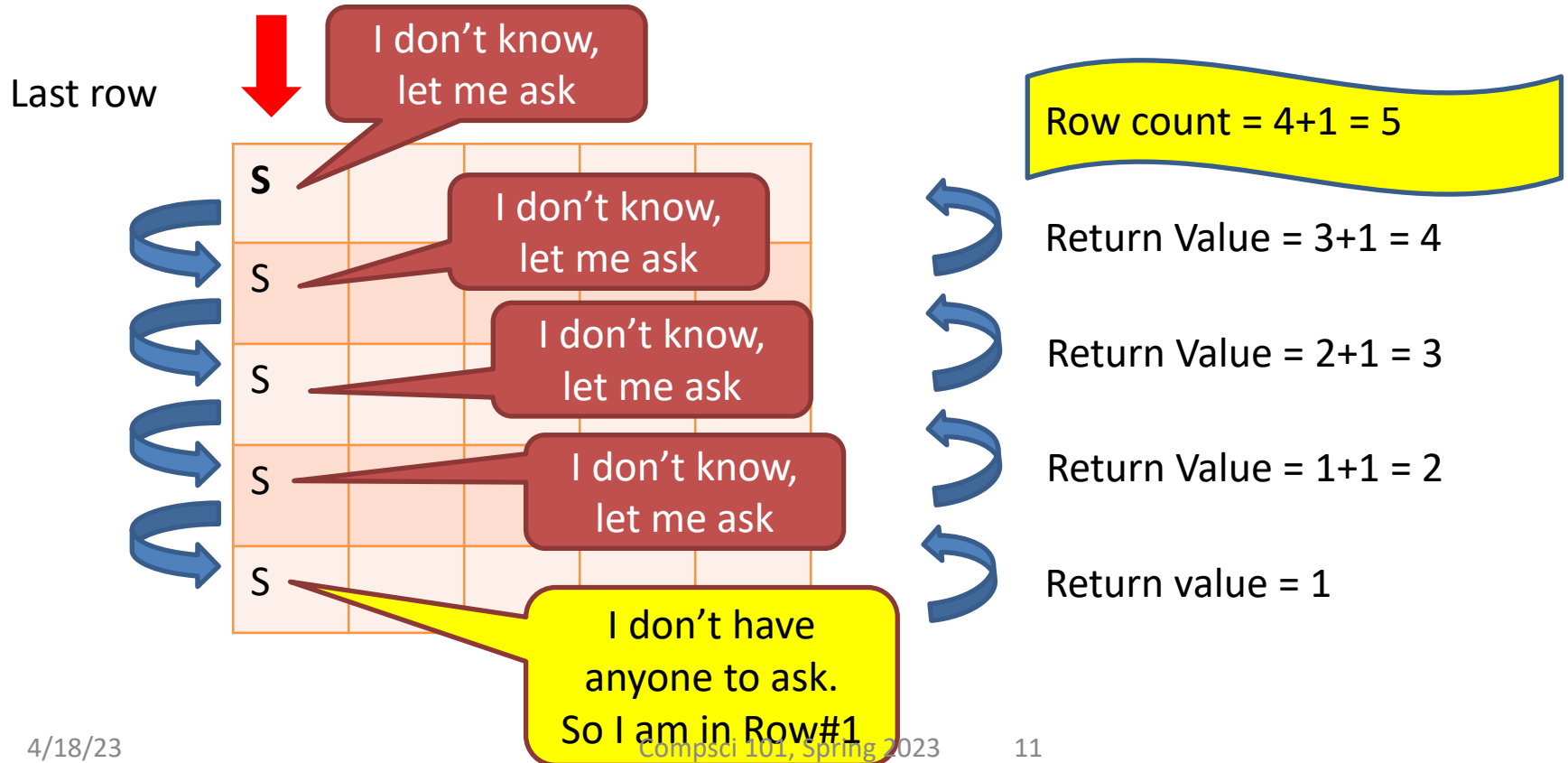
- **Solving a problem by solving similar but smaller problems**

Recursion

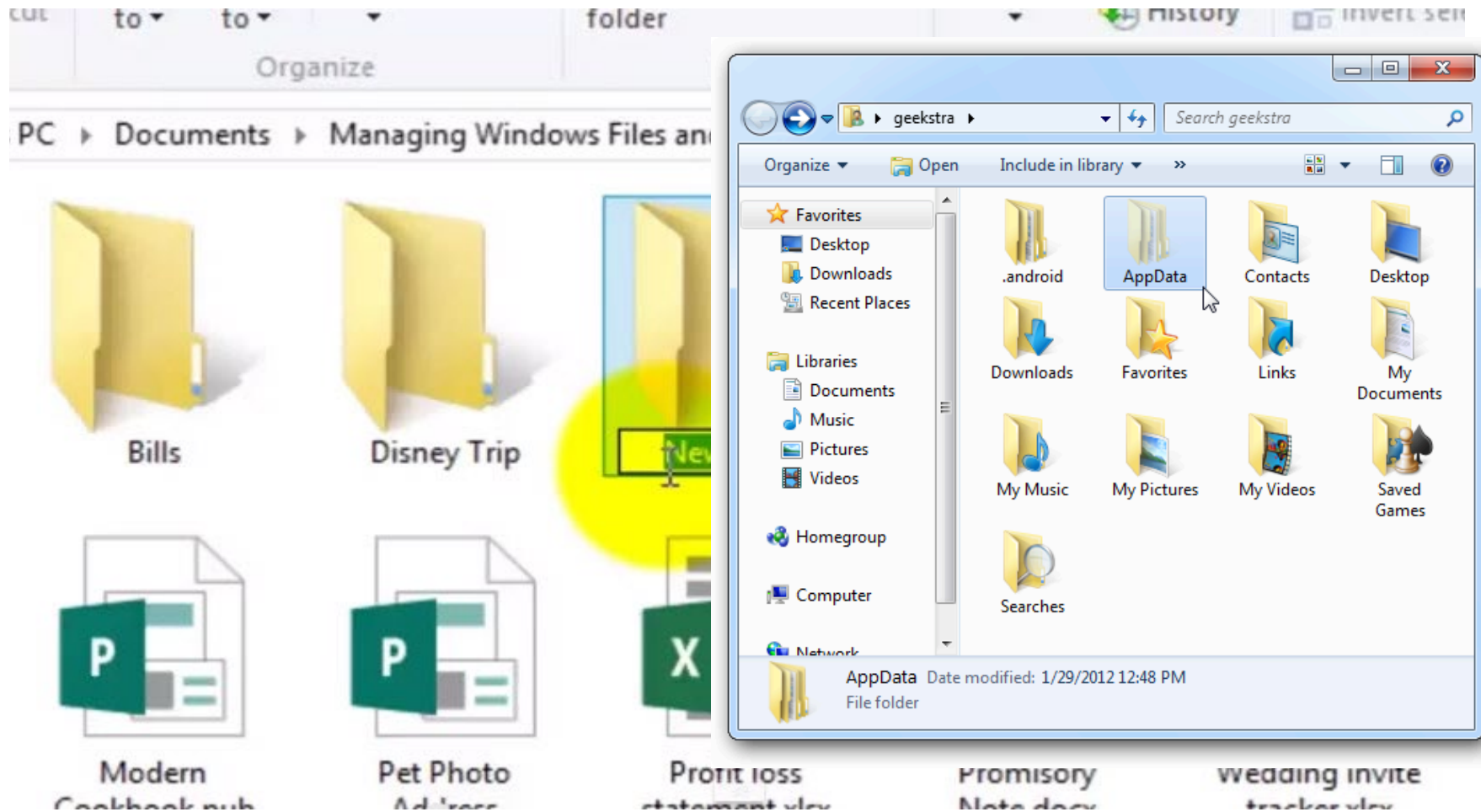
Solving a problem by solving similar but smaller problems

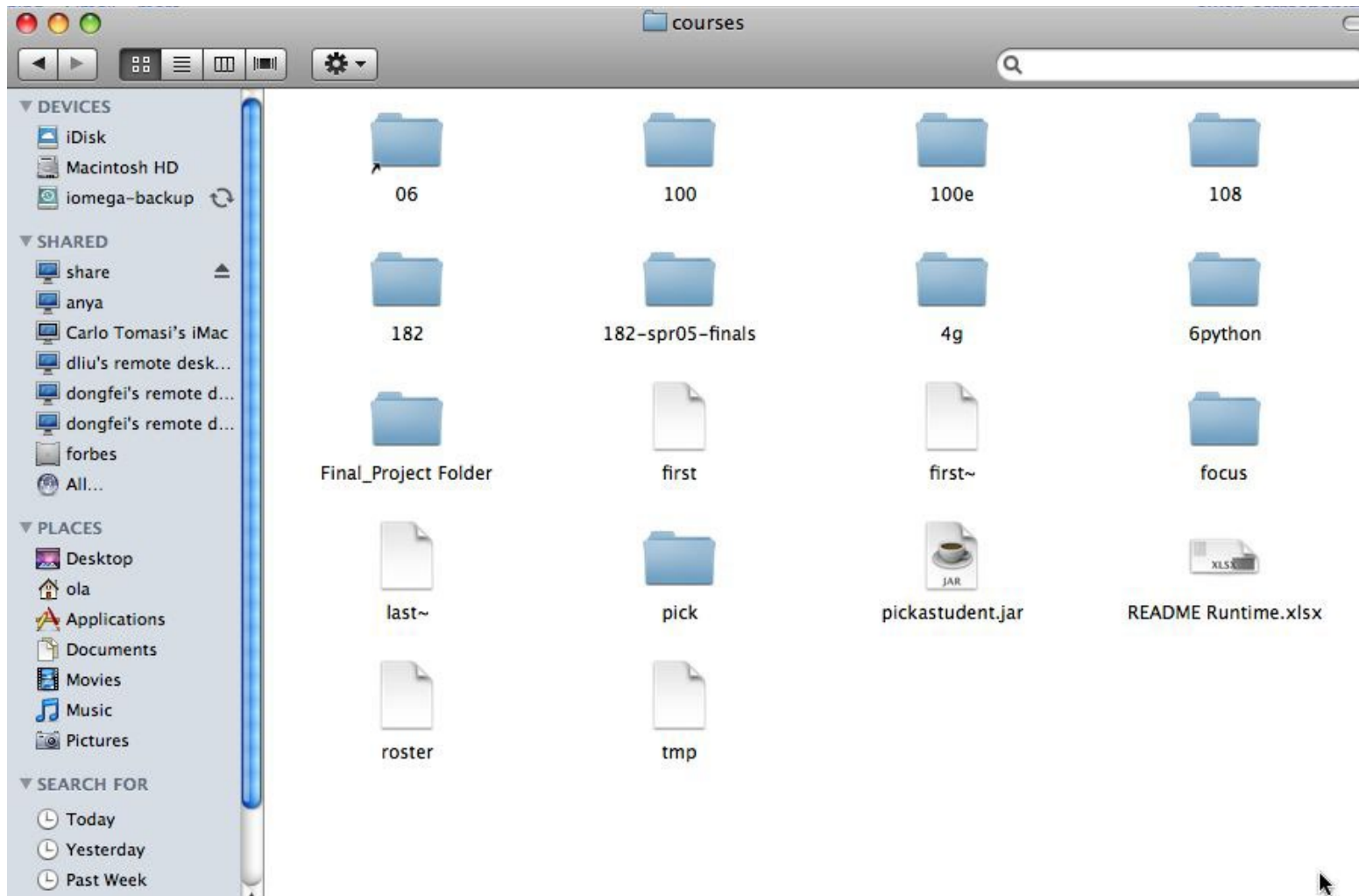
Question - How many **rows** are there in this classroom?

Similar but smaller question - How many **rows** are there until your row?



What's in a file-system Folder?

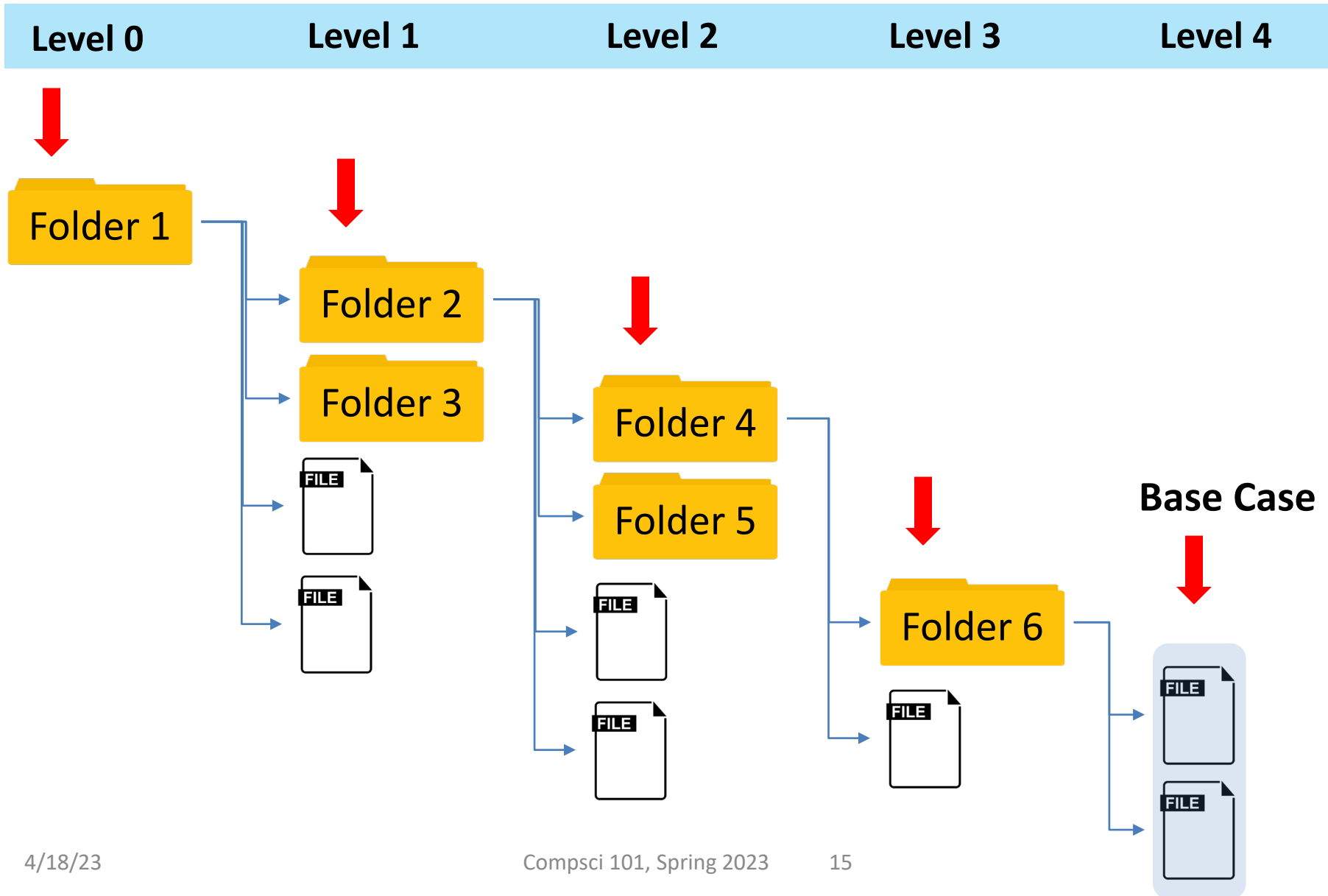




What's in a folder on your computer?



- Where are the **large** files?
- How do you **find them**?
- They take up space!
 - What's the plan –
 1. Erase?
 2. Backup?

Hierarchy in Folder Structure



Recursion (idea) to print ALL files in a folder

- A folder can have sub folders and files
- A file cannot have sub files

```
def visit(dirname):  
    for inner in dirname:  
        if isdir(inner):  Is that a directory?  
            visit(inner)  
        else:  If not a directory, it will be a file  
            print(name(inner), size(inner))
```

Recursion (idea) to print ALL files in a folder

- A folder can have sub folders and files
- A file cannot have sub files

Name of function is
visit

```
def visit(dirname):  
    for inner in dirname:  
        if isdir(inner):  
            visit(inner)  
        else:  
            print(name(inner), size(inner))
```

Is that a directory?

If not a directory, it will be a file

Call visit
on a
smaller
problem

Recursion (idea) to print ALL files in a folder

- A folder can have sub folders and files
- A file cannot have sub files

```
def visit(dirname):  
    for inner in dirname:  
        if isdir(inner):  
            visit(inner)  
        else:  
            print(name(inner), size(inner))
```

Calling the function we are defining, but on a SMALLER problem

Is that a directory?

If not a directory, it will be a file

Finding large files: FileVisit.py

```
def bigfiles(dirname,min_size):
    large = []
    for sub in os.listdir(dirname):
        path = os.path.join(dirname,sub)
        if os.path.isdir(path):
            subs = bigfiles(path,min_size)
            large.extend(subs)
        else:
            size = os.path.getsize(path)
            if size > min_size:
                large.append((path,size))
    return large

# on Mac like this:
#big = bigfiles("/Users/Susan/Documents",10000)
# on Windows like this:
big = bigfiles("C:\\Users\\Susan\\Documents",10000)
```

Finding large files: FileVisit.py

```
def bigfiles(dirname, min_size):  
    large = []  
    for sub in os.listdir(dirname):  
        path = os.path.join(dirname, sub)  
        if os.path.isdir(path):  
            subs = bigfiles(path, min_size)  
            large.extend(subs)  
        else:  
            size = os.path.getsize(path)  
            if size > min_size:  
                large.append((path, size))  
    return large
```

Name of function is
bigFiles

Call
bigFiles
With a
smaller
problem

```
# on Mac like this:  
#big = bigfiles("/Users/Susan/Documents", 10000)  
# on Windows like this:  
big = bigfiles("C:\\Users\\Susan\\Documents", 10000)
```


Example Run

- ('C:\\Users\\Susan\\files\\courses\\cps101\\workspace\\spring2015\\assign4_transform\\data\\romeo.txt', 153088L)
- ('C:\\Users\\Susan\\files\\courses\\cps101\\workspace\\spring2015\\assign4_transform\\data\\twain.txt', 13421L)
- ('C:\\Users\\Susan\\files\\courses\\cps101\\workspace\\spring2015\\assign5_hangman\\src\\lowerwords.txt', 408679L)
- ...

Finding Large Files questions

bit.ly/101s23-0418-1

The os and os.path libraries

- **Libraries use an API to isolate system dependencies**
 - C:\\x\\y **# windows**
 - /Users/Susan/Desktop **# mac**
- **FAT-32, ReFS, WinFS, HFS, HSF+, fs**
 - Underneath, these systems are different
 - Python API insulates and protects programmer
- **Why do we have `os.path.join(x, y)`?**
 - x = /Users/Susan/Documents
 - y = file1.txt
 - Output = /Users/Susan/Documents/file1.txt

Dissecting FileVisit.py

- **How do we find the contents of a folder?**
 - Another name for folder: directory
- **How do we identify folder? (by name)**
 - `os.listdir(dirname)` returns a list of files and folder
- **Path is `c:\user\rodger\foo` or `/Users/rodger/bar`**
 - `os.path.join(dir,sub)` returns full path
 - Platform independent paths
- **What's the difference between file and folder?**
 - `os.path.isdir()` and `os.path.getsize()`

Does the function call itself? No!

```
def visit(dirname) :  
    for inner in dirname:  
        if isdir(inner) :  
            visit(inner)  
        else:  
            print(name(inner), size(inner))
```

- **Is a file inside itself? No!**
- **Does pseudo code make sense?**
 - Details make this a little harder in Python, but close!

Does the function call itself? No!

```
def visit(dirname):  
    for inner in dirname:  
        if isdir(inner):  
            visit(inner)  
        else:  
            print(name(inner), size(inner))
```

Function calls a
smaller instance
of itself, on
smaller data

- **Is a file inside itself? No!**
- **Does pseudo code make sense?**
 - Details make this a little harder in Python, but close!

Structure matches code

Find large files

If you see a folder,

1. Find the large files and subfolders
2. For the subfolders, repeat the process of finding large files and any other folders within that subfolder
3. Repeat the process until you reach the last folder

Compress or Zip a folder

If you see a folder,

1. Find the files and subfolders
2. For the subfolders, repeat the process of finding files and any other folders within that subfolder
3. At the last stage, start compressing files and move up the folder hierarchy

Structure matches code

- **Structure of list of lists**

- Can also lead to processing a list which requires processing a list which ...

- **Is e in this list?**

- How many lists do you have to look in?

[[[a,b], [c,d]], [a, [b,c],d]]

Structure matches Code

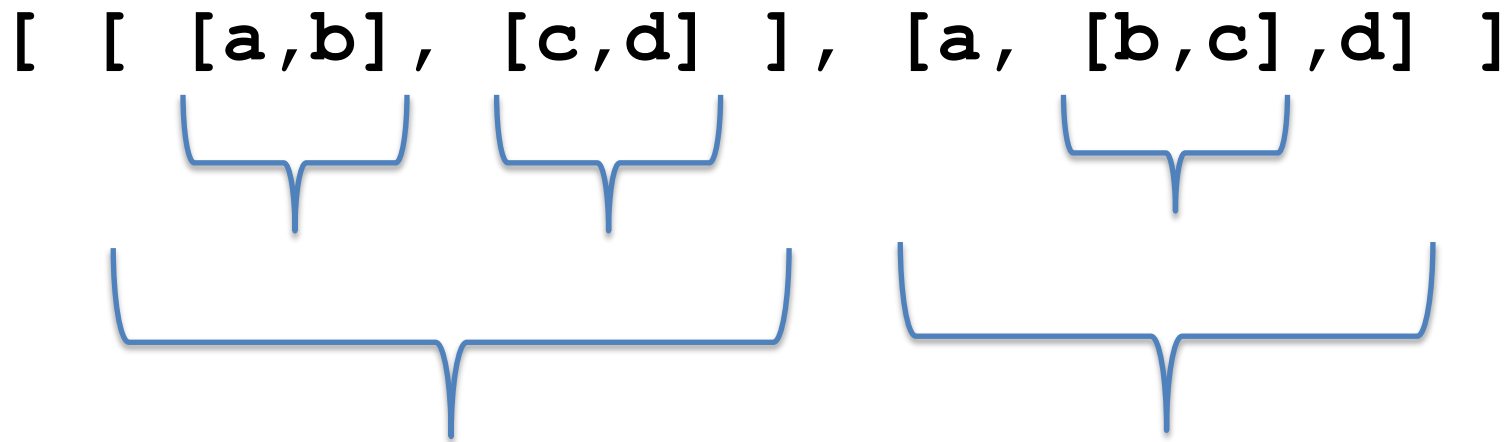
- **Structure of list of lists**

- Can also lead to processing a list which requires processing a list which ...

This structure fits with recursion

- **Is e in this list?**

- How many lists do you have to look in?



Structure matches Code

- **Structure of expressions**
 - Can also lead to processing an expressions which requires processing an expression...
- **How do you evaluate expression?**

$(a * (b + c + (d + e * f)) + (a * (b + d)))$

Structure matches Code

- **Structure of list of lists**
 - Can also lead to processing an expressions which requires processing an expression...
- **How do you evaluate expression?**

$$(a * (b + c + (d + e * f)) + (a * (b + d)))$$

The diagram illustrates the structure of the expression $(a * (b + c + (d + e * f)) + (a * (b + d)))$ using blue brackets. A large bracket underneath the first part, $(a * (b + c + (d + e * f)))$, indicates that this entire sub-expression is processed first. A smaller bracket underneath the inner expression $(d + e * f)$ shows that its evaluation is required before the multiplication with a . Another bracket underneath the second part, $(a * (b + d))$, shows that this sub-expression is processed as a whole.

Recursion Summary

- **Make Simpler or smaller calls**
 - Call a clone of itself with different input
- **Must have a base case when no recursive call can be made**
 - Example - The last folder in the folder hierarchy will not have any subfolders. It can only have files. That forms the base case

Mystery Recursion

bit.ly/101-s23-0418-2

Mystery Recursion

```
def Mystery(num):  
    if num > 0:  
        return 1 + Mystery(num//2)  
    else:  
        return 2 + num
```

Example

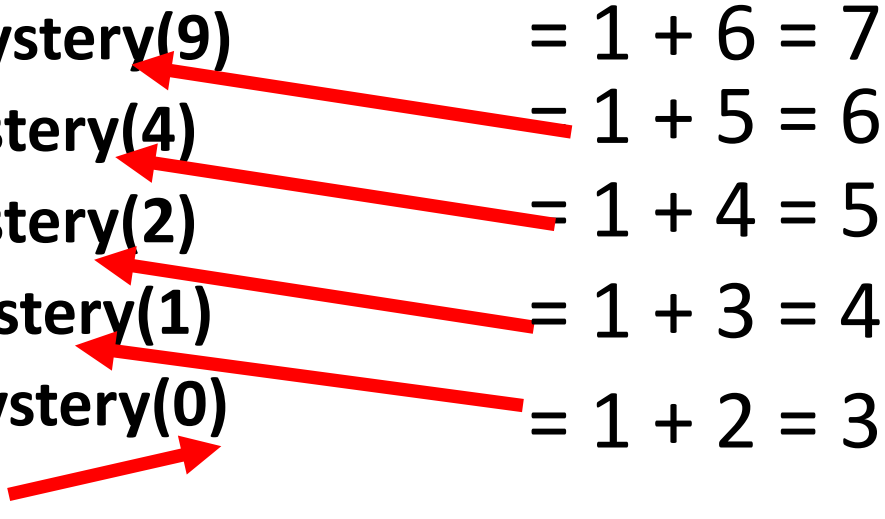
```
def Mystery(num):  
    if num > 0:  
        return 1 + Mystery(num//2)  
    else:  
        return 2 + num
```

- **Mystery(4) is 1 + Mystery(2)**
- **Mystery(2) is 1 + Mystery(1)**
- **Mystery(1) is 1 + Mystery(0)**
- **Mystery(0) is 2**

$$\begin{aligned} &= 1 + 4 = 5 \\ &= 1 + 3 = 4 \\ &= 1 + 2 = 3 \end{aligned}$$

Example

```
def Mystery(num):  
    if num > 0:  
        return 1 + Mystery(num//2)  
    else:  
        return 2 + num
```

- **Mystery(18) is** $1 + \text{Mystery}(9)$ $= 1 + 6 = 7$
 - **Mystery(9) is** $1 + \text{Mystery}(4)$ $= 1 + 5 = 6$
 - **Mystery(4) is** $1 + \text{Mystery}(2)$ $= 1 + 4 = 5$
 - **Mystery(2) is** $1 + \text{Mystery}(1)$ $= 1 + 3 = 4$
 - **Mystery(1) is** $1 + \text{Mystery}(0)$ $= 1 + 2 = 3$
 - **Mystery(0) is** $2 + 0$
- 

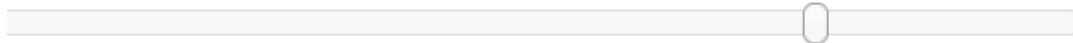
Mystery in Python Tutor

Python 3.6
([known limitations](#))

```
1 def Mystery(num):  
2     if num > 0:  
3         return 1 + Mystery(num//2)  
4     else:  
5         return 2 + num  
6  
7 if __name__ == '__main__':  
8     print("Mystery(7) is", Mystery(7))
```

[Edit this code](#)

: just executed
: to execute



<< First < Prev **Next >** Last >>

Step 16 of 19

[Visualization](#) (NEW!)

Print output (drag lower right corner to resize)

Frames

Objects

Global frame

Mystery

function
Mystery(num)

Mystery

num 7

Mystery

num 3

Mystery

num 1

Mystery

num 0

Return
value 2

Something Recursion

bit.ly/101s23-0418-3

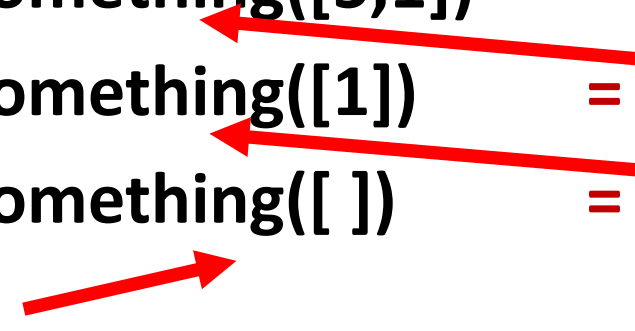
Something Recursion

What is Something([3,5,1]) ?

```
def Something(data):  
    # data is a list of integers  
    if len(data) == 0:  
        return 0  
    if data[0]%2 == 0: # it is even  
        return data[0] + Something(data[1:])  
    else:  
        return Something(data[1:])
```

Something Recursion

```
def Something(data):  
    # data is a list of integers  
    if len(data) == 0:  
        return 0  
    if data[0]%2 == 0: # it is even  
        return data[0] + Something(data[1:])  
    else:  
        return Something(data[1:])
```

- **Something([3,5,1]) is** **Something([5,1]) = 0**
 - **Something([5,1] is** **Something([1]) = 0**
 - **Something([1]) is** **Something([]) = 0**
 - **Something([]) is** **0**
- 

Something([3,5,1]) is 0

Something Recursion

What is Something([5,4,2,3]) ?

```
def Something(data):  
    # data is a list of integers  
    if len(data) == 0:  
        return 0  
    if data[0]%2 == 0: # it is even  
        return data[0] + Something(data[1:])  
    else:  
        return Something(data[1:])
```

Something Recursion

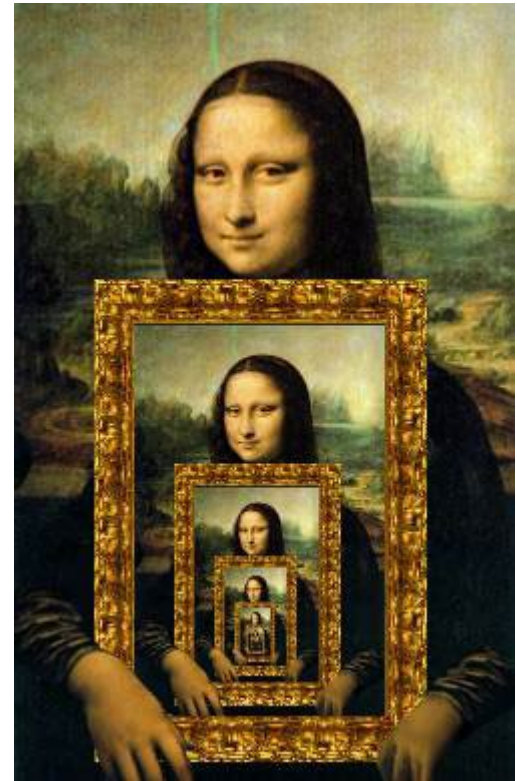
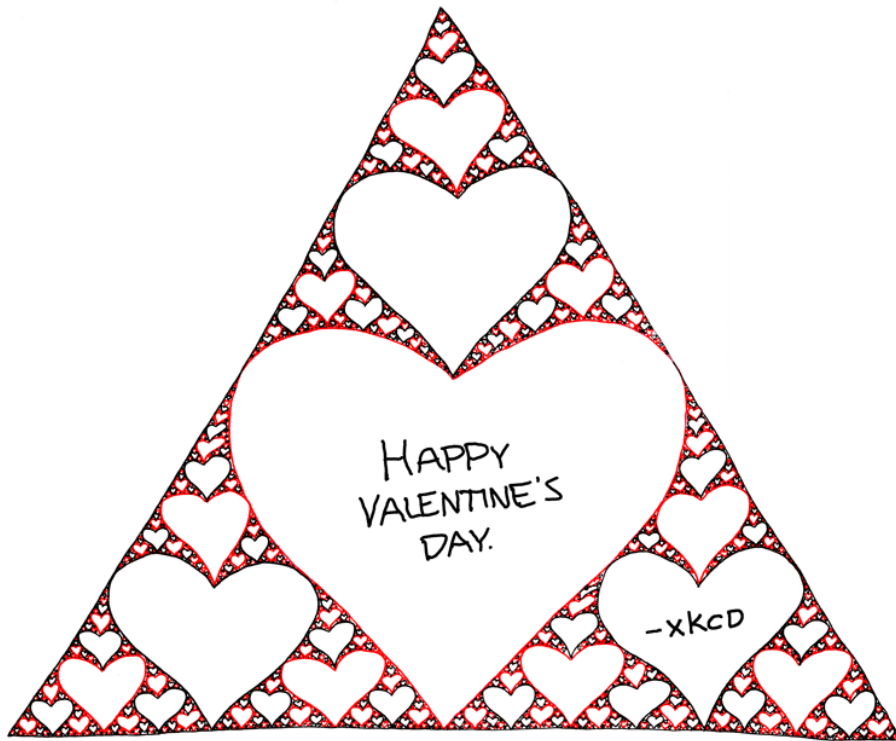
```
def Something(data):  
    # data is a list of integers  
    if len(data) == 0:  
        return 0  
    if data[0]%2 == 0: # it is even  
        return data[0] + Something(data[1:])  
    else:  
        return Something(data[1:])
```

Something([5,4,2,3]) is 6

- **Something([5,4,2,3]) is** **Something([4,2,3]) = 6**
 - **Something([4,2,3]) is** **4 + Something([2,3]) = 6**
 - **Something([2,3]) is** **2 + Something([3]) = 2**
 - **Something([3]) is** **Something([]) = 0**
 - **Something([]) is** **0**
-

Recursion in Pictures

- <http://xkcd.com/543/>



Enjoy a cookie!
You get one half of python logo
blue or yellow python

