

Compsci 101

How Dictionaries work, Recursion

Live Lecture



Susan Rodger
Nicki Washington
April 20, 2021

W is for ...



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

x is for ...



- **XOR**
 - (a or b) and not (a and b), a.k.a. symmetric difference
- **XML**
 - eXtensible Markup Language
- **Xerox Parc**
 - From Mice to Windows

Rediet Abebe



- CS PhD '19 Cornell
- Harvard Fellow til '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-8 due Tuesday, April 20, Today!
- Assign 6 Recommender, due Thurs 4/22
 - One grace day, NO LATE DAYS!
 - MUST TURN in BY FRIDAY 4/23
- Assign 7 Create due, Friday, April 23!
 - Grace period is through Monday, April 26
 - No Late days!
- Lab 12 Friday, prelab available later today!
- Exam 3 back soon,...

Final Exam

- 3 hour exam – giving you 6 hours to take it
- 3 parts
 - PART A) on Sakai: (programming, like an APT Quiz)
 - 50 min – giving you 2 hours
 - Take any time April 27-29
 - PART B) – more programming, like Part B)
 - 50 min – giving you 2 hours
 - Take any time April 27-29
 - PART C) on GradeScope:
 - 80 min – giving you 2 hours
 - **MUST BE taken** on April 29.

Assignment 7: Create, Due Apr 23

Grace period til Apr 26, No late days!

Must be turned in by Apr 26

This assignment is required!

Pick one:

Video: Green dance, advertisement for 101, song, other

Poem or Multiple Haikus

Story

Comic

One-pager

Feedback

Let's see some examples

PFTD

- How do Dictionaries work so fast!
 - Access an element in constant time
- Recursion
 - Solving a problem by solving smaller problems

How do Dictionaries work so fast?

- How are they implemented?



Simple Example

Want a mapping of Soc Sec Num to Names

- Duke's CS Student Union wants to be able to quickly find out info about its members. Also add, delete and update members. Doesn't need members sorted.

267-89-5431 John Smith

703-25-6141 Ademola Olayinka

319-86-2115 Betty Harris

476-82-5120 Rose Black

- Dictionary d – SSN to names
 - $d['267-89-5431'] = \text{'John Smith'}$
 - How does it find 'John Smith' so fast?

Dictionary $d(SSN) = (SSN, name)$

- We actually would map the SSN to the tuple of (SSN, name).
- That is a lot to display on a slide, so we will just show SSN to name
- But remember name is really a tuple of (SSN,name)

Simple Example

Let's look under the hood.

How are dictionaries implemented?

- Dictionaries implemented with a list, in a clever way
- How do we put something into the list fast?
- How do we find it in the list quickly?
 - `d['267-89-5431'] = 'John Smith'`
- List size is 11 – from 0 to 10
- `d['267-89-5431']` calculates index location in list of where to put this tuple (SSN,name)
- Use a function to calculate where to store 'John Smith'
 - $H(ssn) = (\text{last 2 digits of ssn}) \bmod 11$
 - Called a Hash function

Have a list of size 11 from 0 to 10

- Insert these into the list
- Insert as (key, value) tuple
(267-89-5431, John Smith)
(in example, only showing name)

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

WOTO-1 How Dictionaries Work

<http://bit.ly/101s21-0420-1>



Recursion

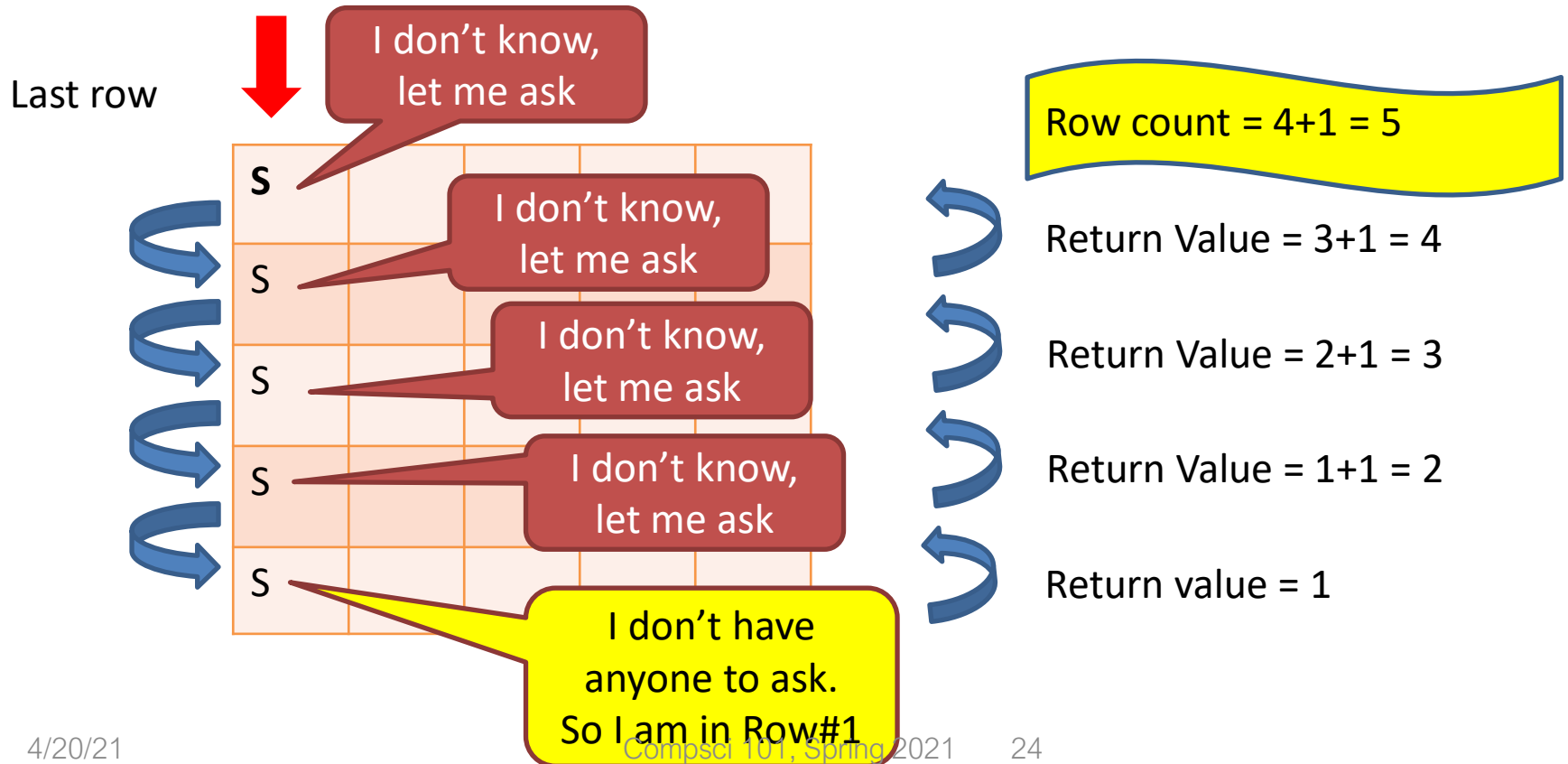
- Solving a problem by solving similar but smaller problems

Recursion

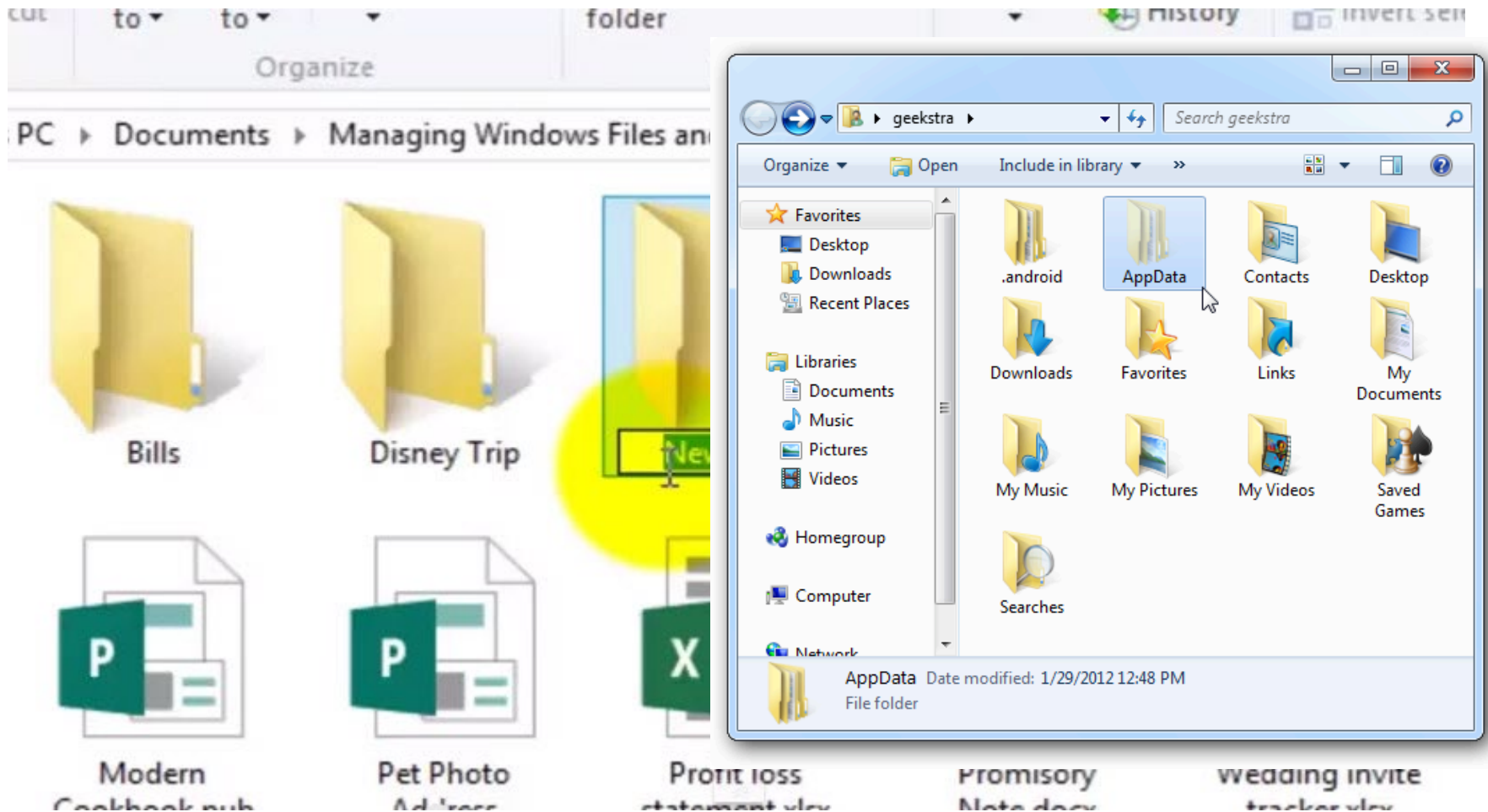
Solving a problem by solving similar but smaller problems

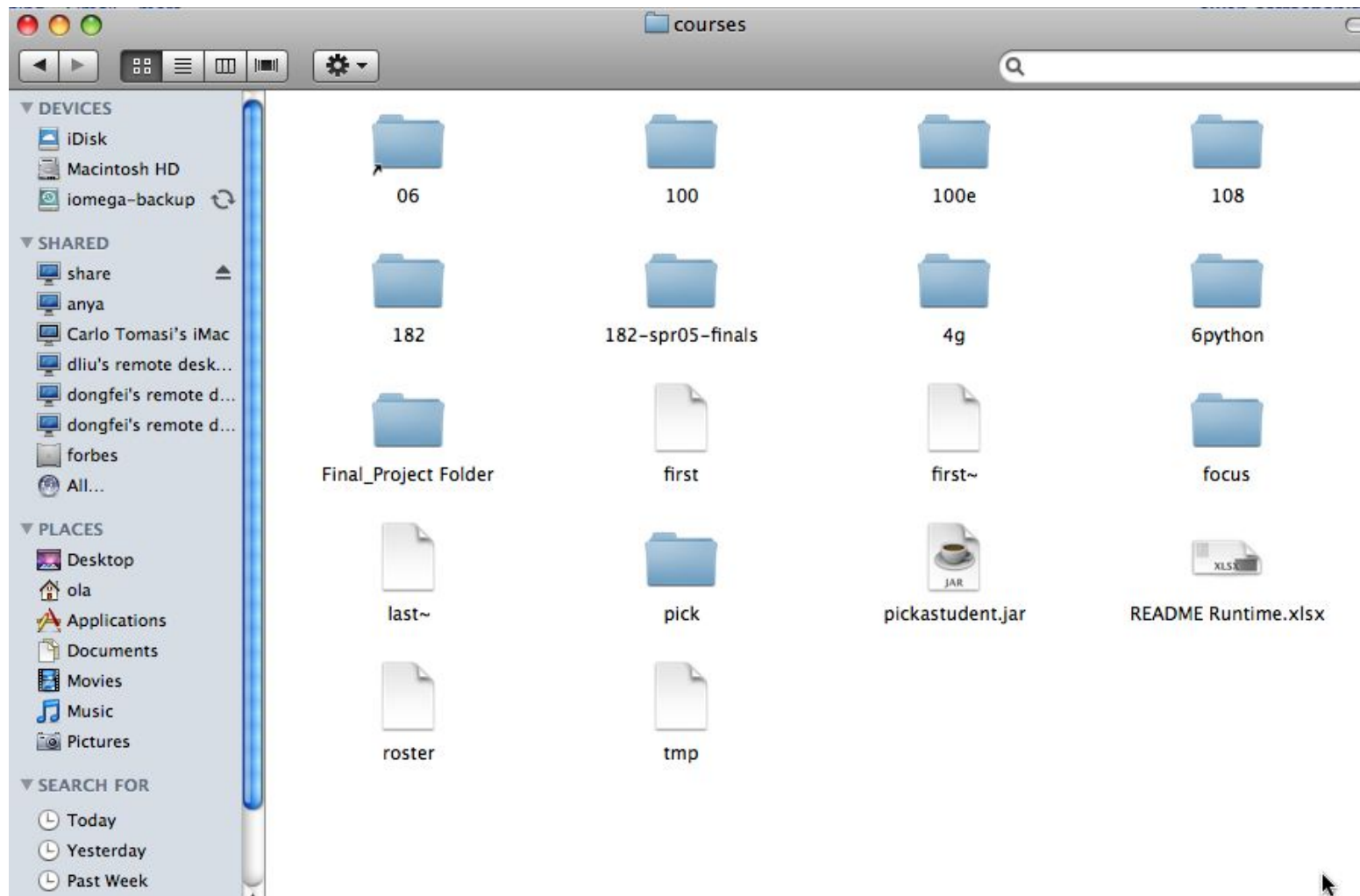
Question - How many **rows** are there in this classroom?

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



What's in a file-system Folder?

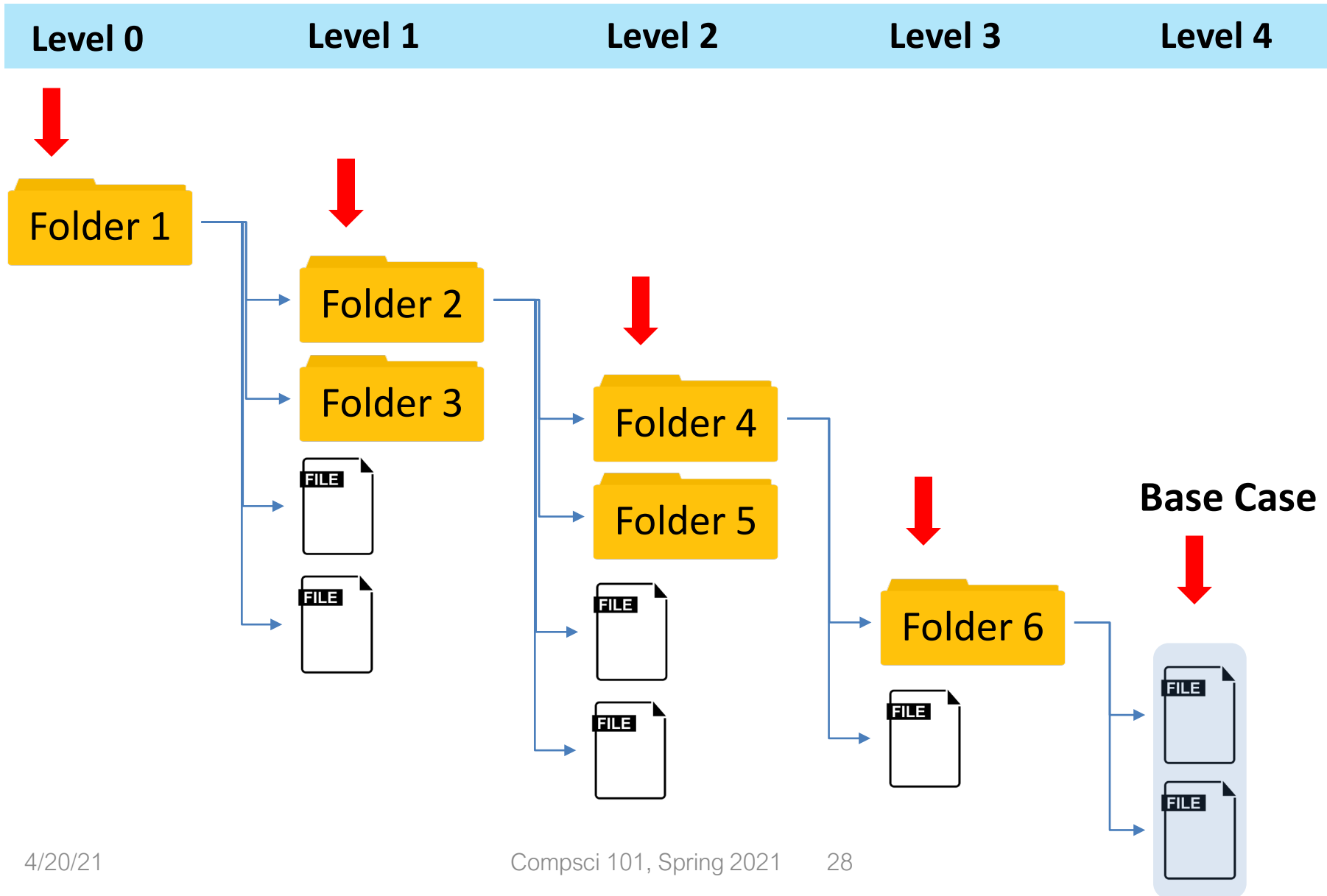




What's in a folder on your computer?



- 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 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))
```

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)
```

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/101s21-0420-2

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!

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 ...

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

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

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/101s21-0420-3

Mystery Recursion

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