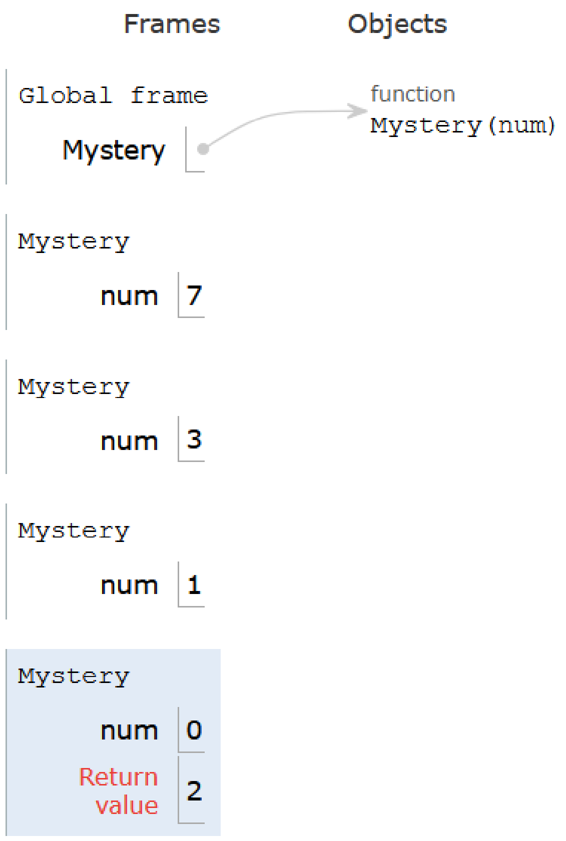
Compsci 101
More Recursion, Last Lecture
Live Lecture



Susan Rodger
Nicki Washington
April 22, 2020

`Y` is for …

- **YAML and YACC**
  - Yet Another ...
- **Y2K:** https://www.youtube.com/watch?v=rblt2EtFfC4
  - How many bits are enough bits?
- **YouTube**
  - Connected to computing ...

`z` is for …

- **Zero**
  - There are two, or 10 bits in the universe
- **Zip**
  - Compressed file archive format
- **Zork**
  - Text-based adventure game
  - https://www.youtube.com/watch?v=TNN4VPlRBJ8

The Tech Twins

- Troy and Travis Nunnally
- Between them: 2 master's and 1 doctorate from Georgia Tech
- Cofounders of Brain Rain Solutions
  - Augmented-reality
  - Internet-of-things
- Applied machine learning

https://www.wired.com/story/what-atlanta-can-teach-tech-about-cultivating-black-talent/

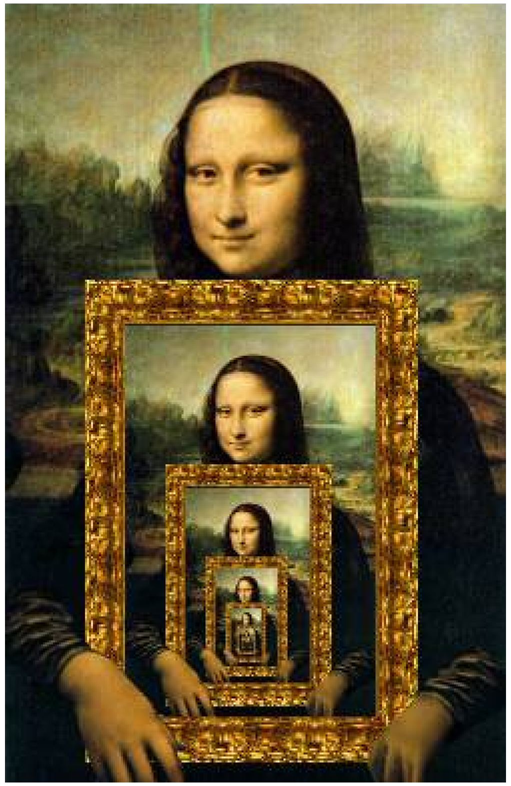
Travis advice: "You have to be passionate and find something that you simply love and enjoy. Not only find that thing – but actually be a life long learner around that."

Troy advice: "Start around a problem… from there, start using skills and learning skills that are related to building that."

Recursion in Pictures

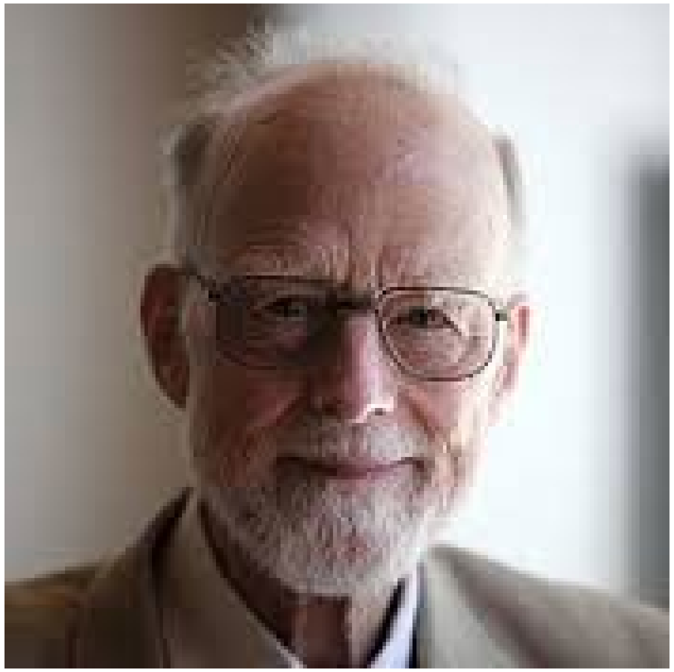
- http://xkcd.com/543/

HAPPY VALENTINE'S DAY.
-XKCD



Sir Anthony (Tony) Hoare

There are two ways of constructing a software design. One way is to make it so simple that there are obviously no deficiencies. And the other way is to make it so complicated that there are no obvious deficiencies.



Turing Award, didn’t get recursion…..

Inventor of quicksort

Assignment 7: Create, Due April 23

Grace period til April 26, No late days!

Must be turned in by April 26

This assignment is required!

Pick one:

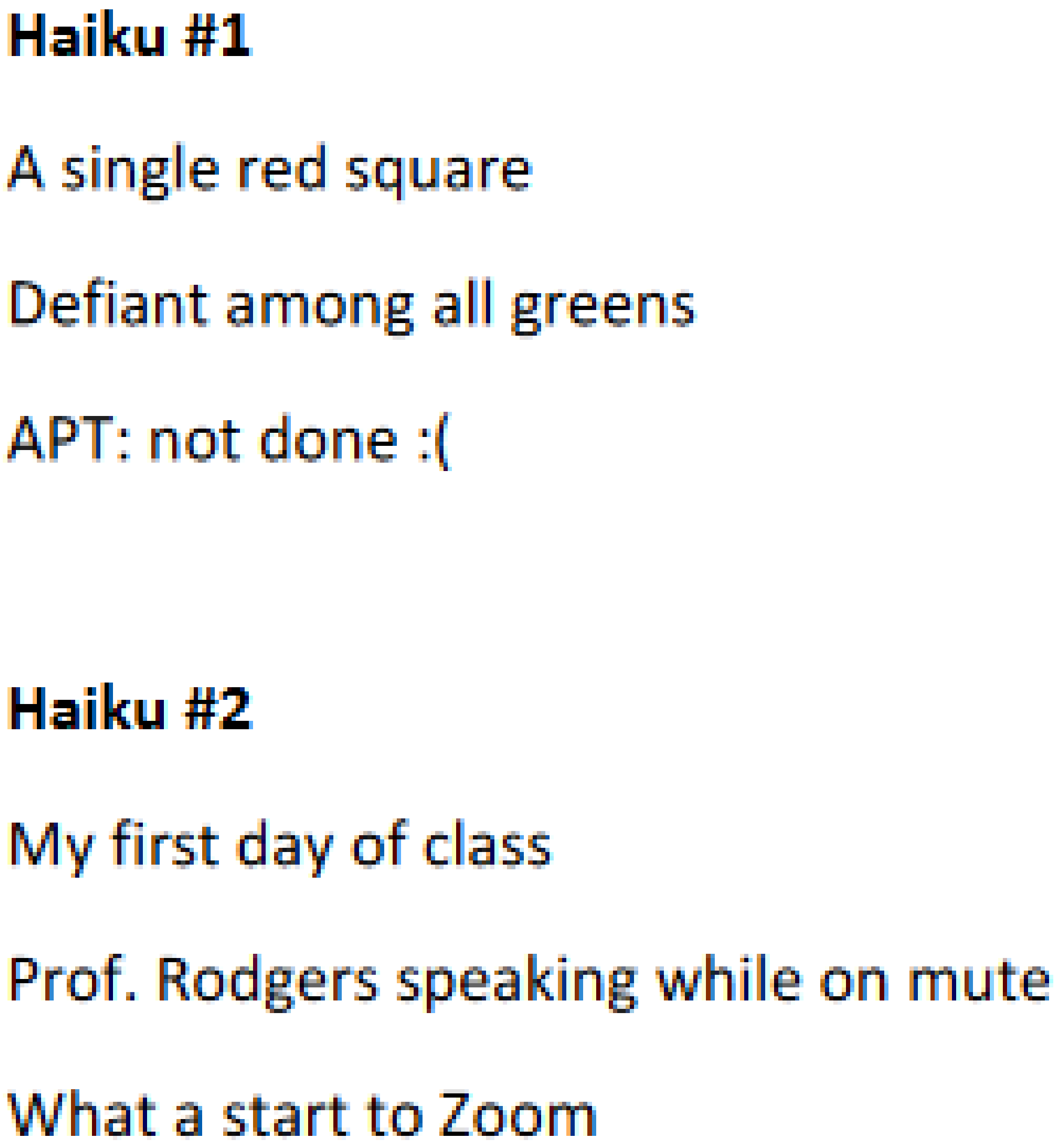
- Video: Green dance, advertisement for 101, song, other
- Poem or Multiple Haikus
- Story
- Comic
- One-pager
- Feedback

Let's see some examples

A Story – One Eternity Later

APT Due

Some Haikus



Haiku #1

A single red square

Defiant among all greens

APT: not done :(

Haiku #2

My first day of class

Prof. Rodgers speaking while on mute

What a start to Zoom

Announcements

- Assign 6 Recommender, due Thurs 4/22, today!
  - 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, Apr 26
  - No Late days!

- Lab 12 Friday, prelab available now!

- Exam 3 almost done grading…. Coming back soon

PFTD

- Finish Recursion
  - Solving a problem by solving smaller problems
- Finishing up
  - What more is there in CS?

Final Exam

- **Study like you studied for Exam 3**
  - Use Exam 3 handout
- **We only have a little material since then**
  - Recommender
    - – this is all about stuff we did before
  - Modules
    - Exceptions
  - Recursion – reading only, no writing
- **Not on the exam**
  - Images, turtles, exceptions

Final Exam – 3hr, you get 6 hr

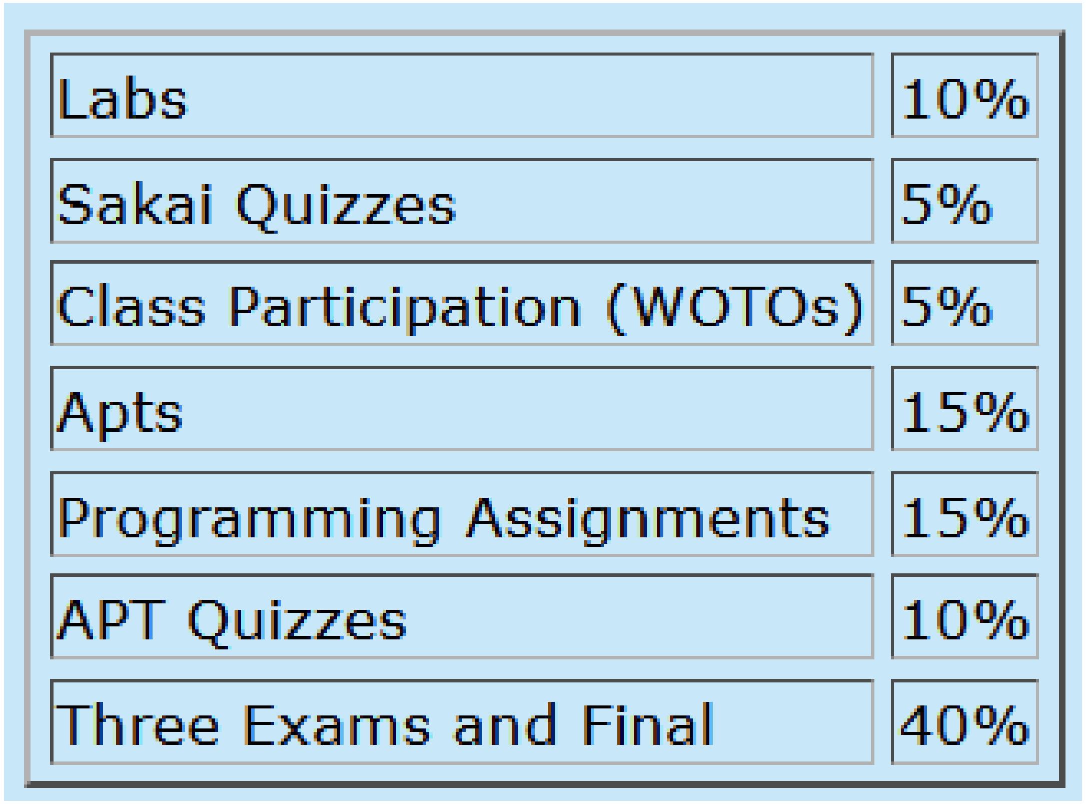
- **3 parts**
  - PART A) on Sakai: (programming, like APT Quiz)
    - 50 min – giving you 2 hours
    - **Take any time April 27, 7am to Apr 29, 11PM**
    - TURNS OFF AT 11PM on Apr. 29
  - PART B) –like Part A, start on Sakai
    - 50 min – giving you 2 hours
    - **Take any time April 27, 7am to Apr 29, 11PM**
  - PART C) on GradeScope:
    - 80 min – giving you 2 hours
    - **MUST BE taken** on Apr 29, 7AM til 11PM
    - **MUST START by** 9pm Apr 29

Advice on Final Exam APTs

- Work an example, or 2 or 3 by hand
  - How are you solving it?
- What would the algorithm be?
- What tools do you need to implement?
  - Dictionary? Set? What do you need to loop over?
- What helper function could you write? Or two?
  - Debug it
- As you write code, print a lot!
- See debugging tips on APT main page

Calculate Your Grade

- From "About" tab on course web page



Labs	10%
Sakai Quizzes	5%
Class Participation (WOTOs)	5%
Apts	15%
Programming Assignments	15%
APT Quizzes	10%
Three Exams and Final	40%

- Each exam is 10%, the final is 10%

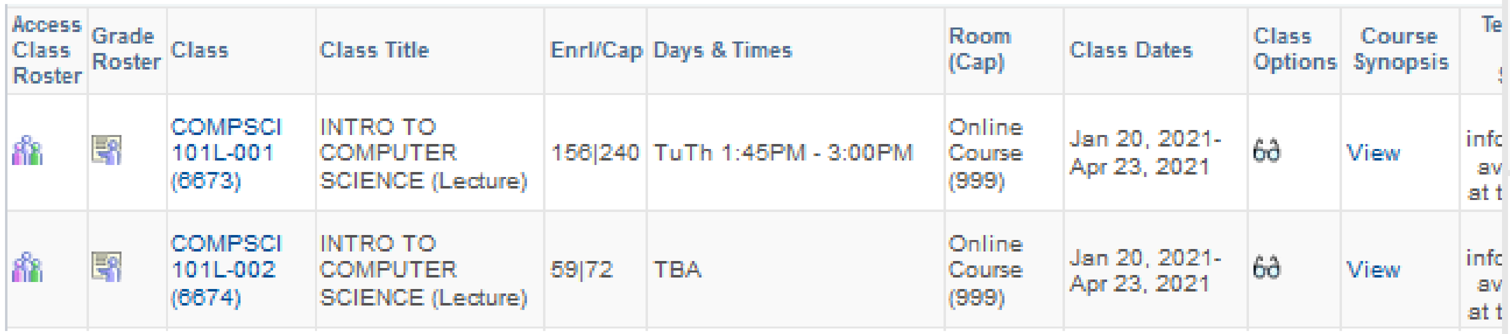
More on Grades

- Class Participation-WOTOs – **ignore** the first two weeks (drop/add period), plus drop 10 **points**
- Sakai Quizzes **275** points– will drop **40** points
  - Your points/235, can’t get more than 100
- Lab – drop **15** points (each lab is 5 pts)
  - Lab 12 covers recursion and debugging!

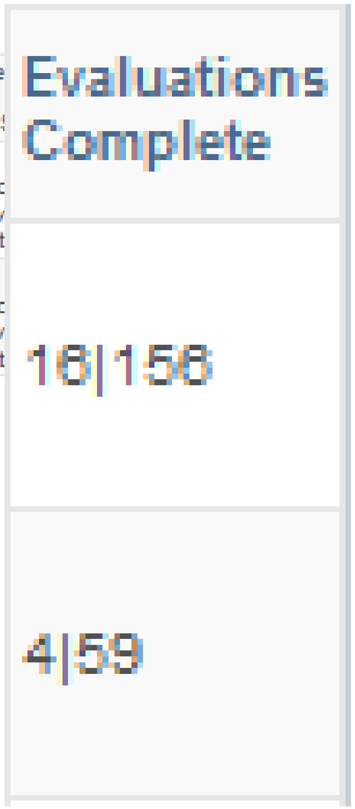
Time for Duke Course Eval and Seven Steps

1. Please fill out Duke Course Eval on DukeHub now

Only 9% have filled it in as of last night



Access Class Roster	Grade Roster	Class	Class Title	Enrl/Cap	Days & Times	Room (Cap)	Class Dates	Class Options	Course Synopsis	Te
👥	📋	COMPSCI 101L-001 (6673)	INTRO TO COMPUTER SCIENCE (Lecture)	156|240	TuTh 1:45PM - 3:00PM	Online Course (999)	Jan 20, 2021- Apr 23, 2021	👓	View	inf av at t
👥	📋	COMPSCI 101L-002 (6674)	INTRO TO COMPUTER SCIENCE (Lecture)	59|72	TBA	Online Course (999)	Jan 20, 2021- Apr 23, 2021	👓	View	inf av at t



Evaluations Complete
16|156
4|59

1. If 60% fill it out, everyone 1 extra point on final exam
2. If 75%, everyone gets 1 additional extra credit point on the Final exam

Recursion

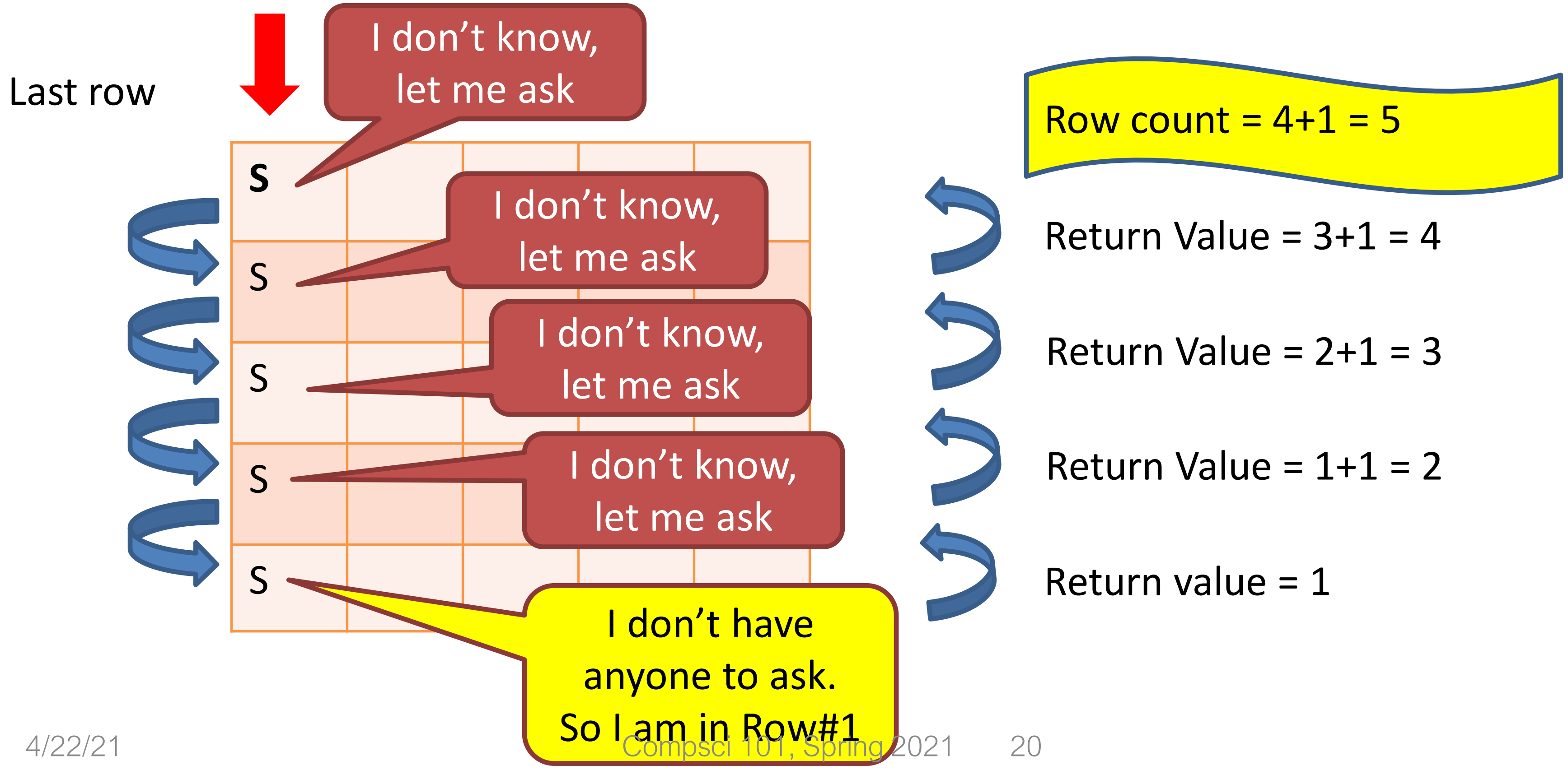
- Solving a problem by solving similar but smaller problems

Recursion

Solving a problem by solving similar but smaller problems

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

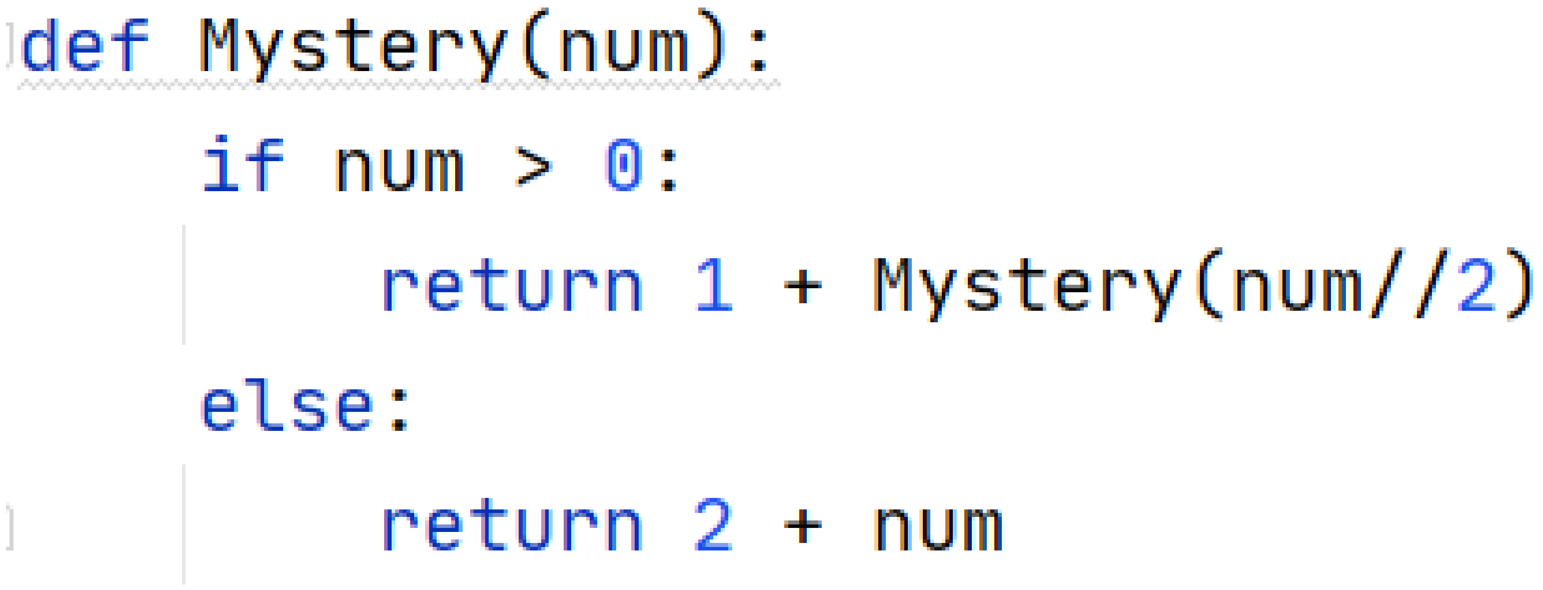
Similar but smaller question - How many **rows** are there until your row?



Review: Recursion Summary

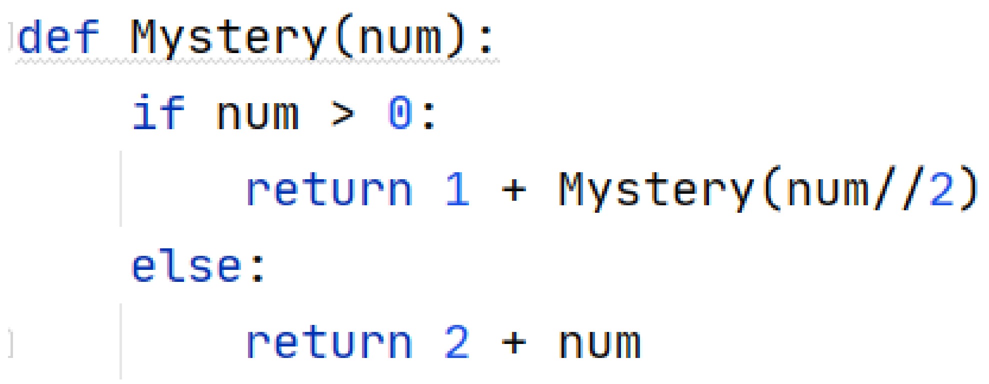
- **Make Simpler or smaller calls**
  - Call a clone of itself with different input
- **Must have a base case when no recursive call can be made**
    - Example - The last folder in the folder hierarchy will not have any subfolders. It can only have files. That forms the base case
    - This is the way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

Mystery Recursion



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

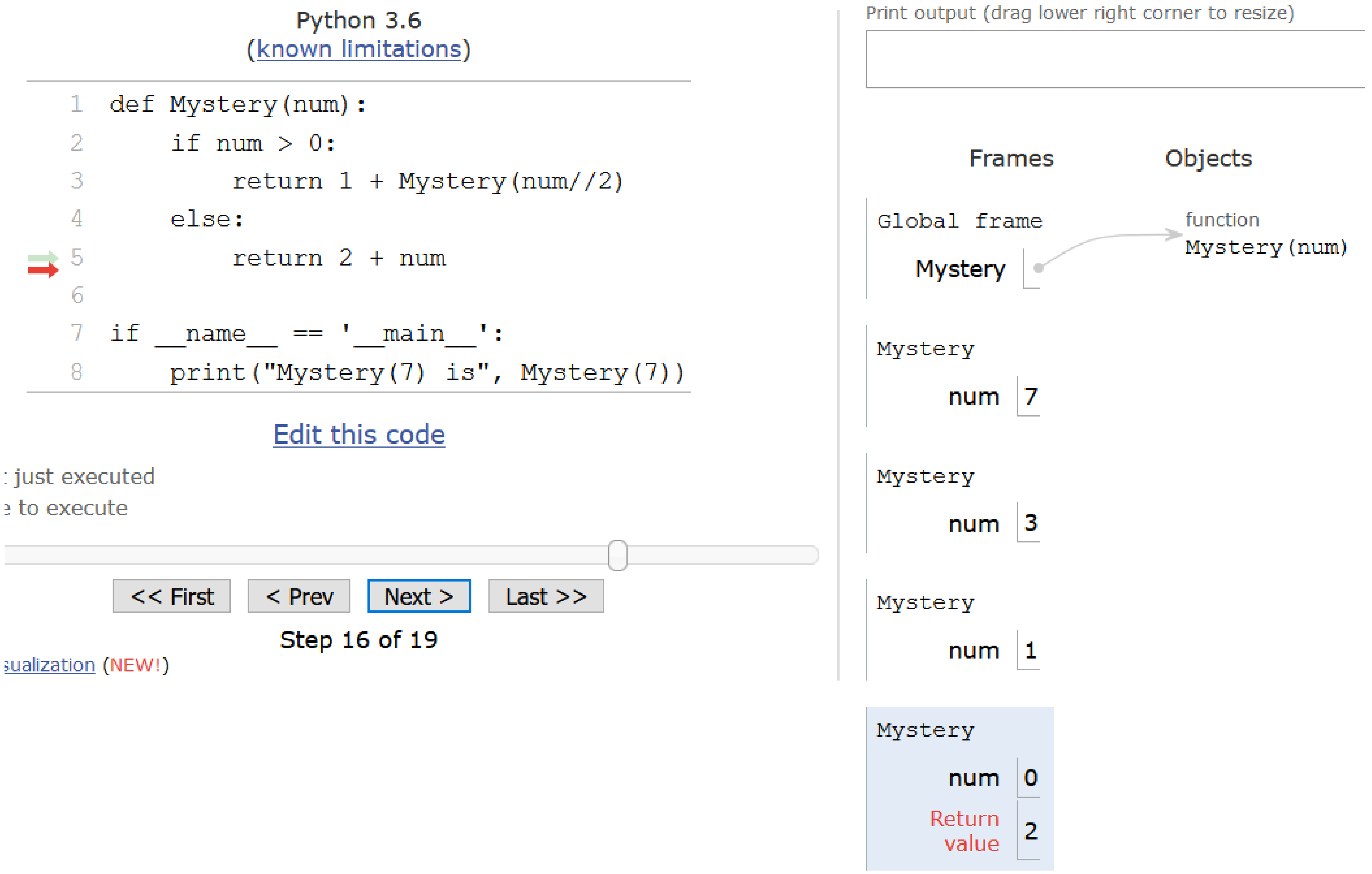
Example



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

- Mystery(7) is 1 + Mystery(3) = 1 + 4 = 5
- Mystery(3) is 1 + Mystery(1) = 1 + 3 = 4
- Mystery(1) is 1 + Mystery(0) = 1 + 2 = 3
- Mystery(0) is 2 + 0 = 2

Mystery in Python Tutor



Python 3.6
(known limitations)

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

if __name__ == '__main__':
    print("Mystery(7) is", Mystery(7))
```

Edit this code

just executed
to execute

<< First < Prev Next > Last >>

Step 16 of 19

sualization (NEW!)

Print output (drag lower right corner to resize)

Frames Objects

Global frame
Mystery → function Mystery(num)

Mystery
num 7

Mystery
num 3

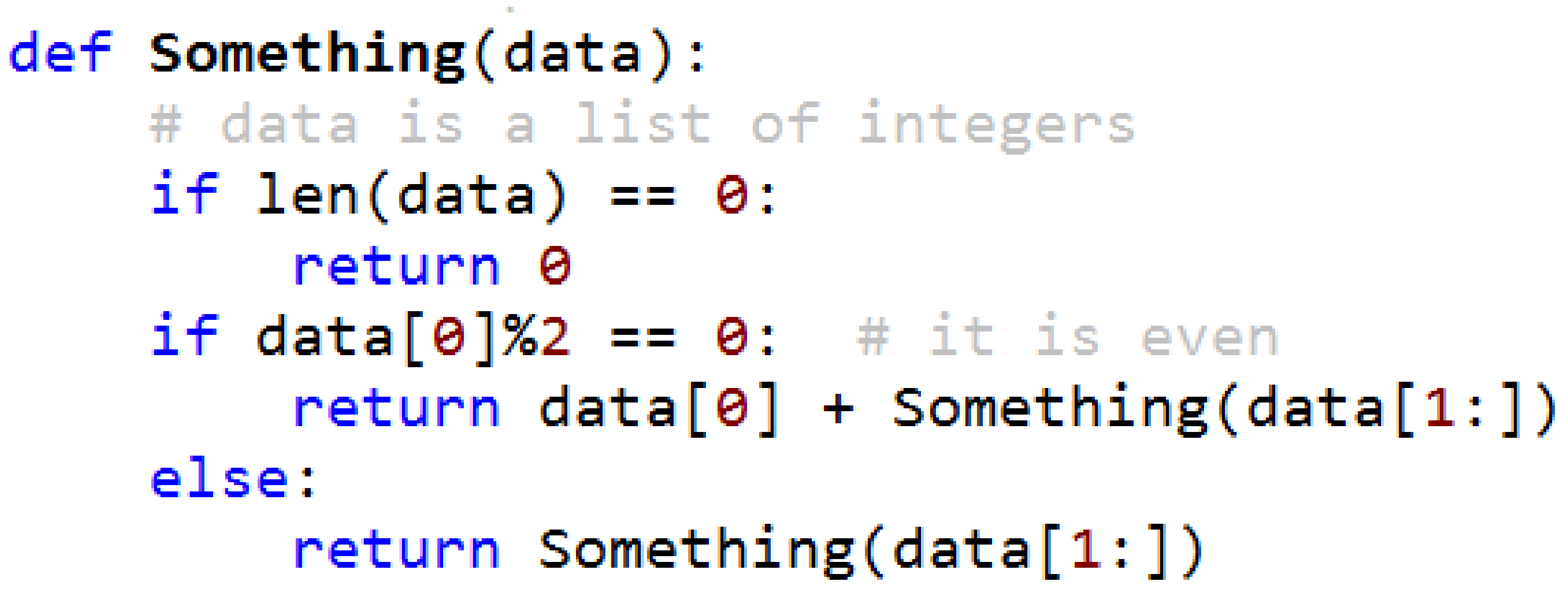
Mystery
num 1

Mystery
num 0
Return value 2

Something Recursion
bitly/101s21-0422-1

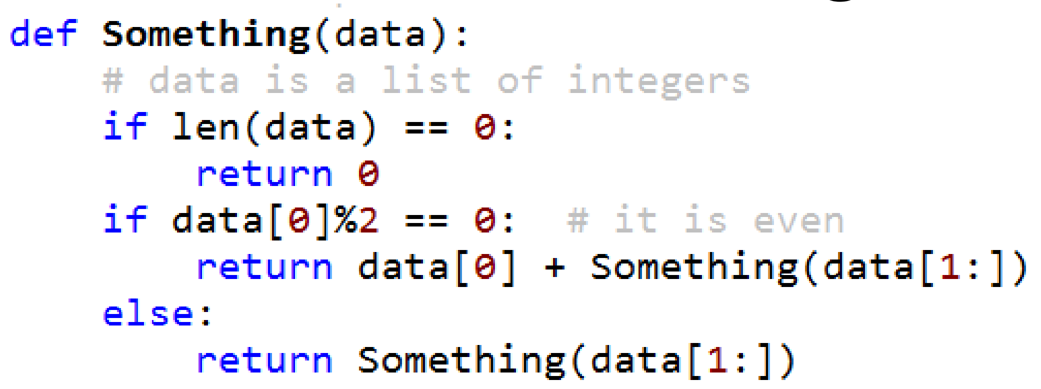
Something Recursion

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



```python
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



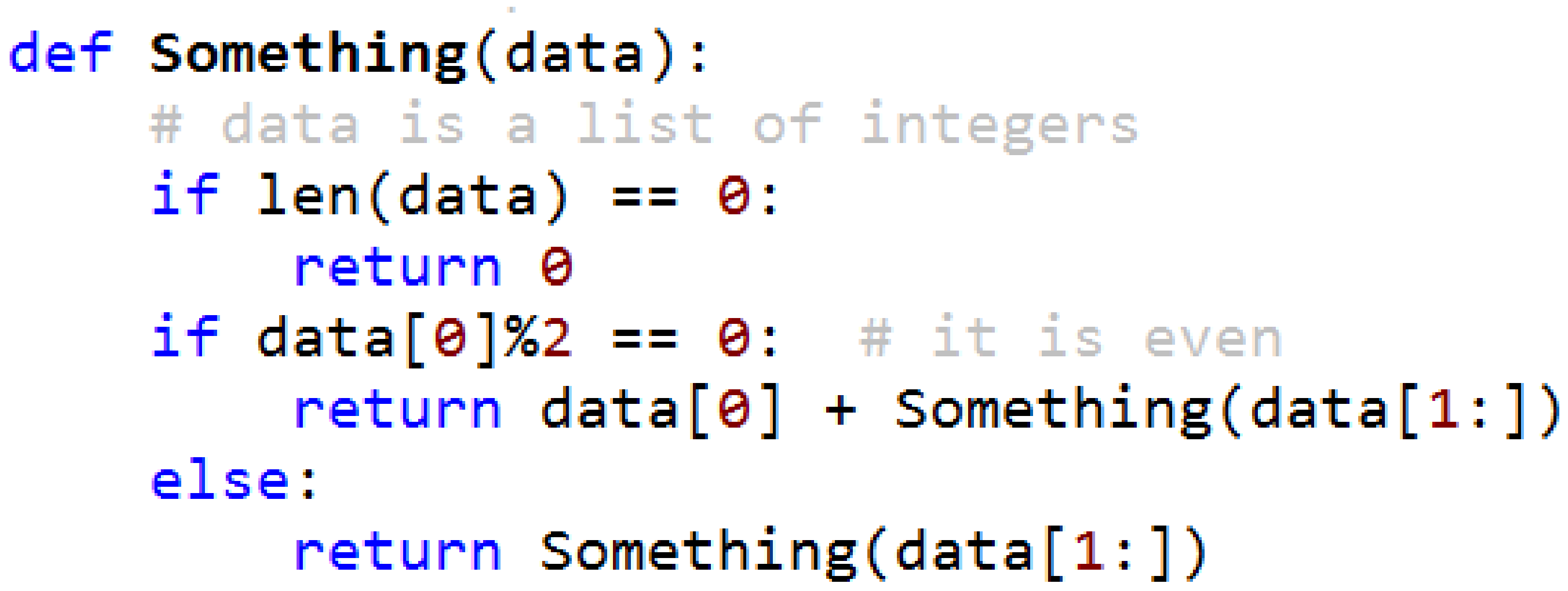
```python
def Something(data):
    # data is a list of integers
    if len(data) == 0:
        return 0
    if data[0]%2 == 0:  # it is even
        return data[0] + Something(data[1:])
    else:
        return Something(data[1:])
```

- 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

```python
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

Revisit the APT Bagels Recursively

```
filename:  Bagels.py

def bagelCount(orders) :
    """
    return number of bagels needed to fulfill
    the orders in integer list parameter orders
    """
```

1.
```
orders = [1,3,5,7]

Returns:  16
```

No order is for more than a dozen, return the total of all orders.

2.
```
orders = [11,22,33,44,55]

Returns: 175 since 11 + (22+1) +(33+2) + (44+3) + (55+4) = 175
```

APT Bagels Recursively bit.ly/101s21-0422-2

APT Bagels Recursively

A)
```
def bagelCount(orders):
    if len(orders) > 0:
        return orders[0]//12 + orders[0] + bagelCount(orders[1:])
    else:
        return 0
```

B)
```
def bagelCount(orders):
    if len(orders) > 0:
        return orders[-1]//12 + orders[-1] + bagelCount(orders[:-1])
    else:
        return 0
```

C)
```
def bagelCount(orders):
    return orders[0] + orders[0]//12 + bagelCount(orders[1:])
```

D)
```
def bagelCount(orders):
    if len(orders)>1:
        return orders[1] + orders[1]//12 + bagelCount(orders[2:])
    else:
        return bagelCount(orders[0])
```

What is Computing? Informatics?

- **What is computer science, what is its potential?**
  - What can we do with computers in our lives?
  - What can we do with computing for society?
  - Will networks transform thinking/knowing/doing?
  - Society affecting and affected by computing?
  - Changes in science: biology, physics, chemistry, …
  - Changes in humanity: access, revolution (?), …

- **Privileges and opportunities available if you know code**
  - Writing and reading code, understanding algorithms
  - Majestic, magical, mathematical, mysterious, …

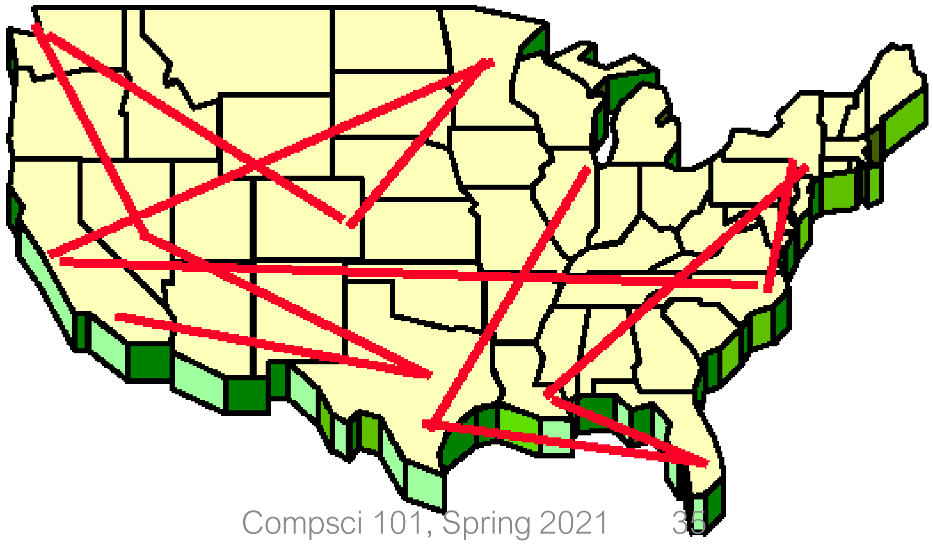
Computing - solve all problems?

- **Some problems can be solved 'efficiently'**
  - Run large versions fast on modern computers
  - What is 'efficient'? It depends
- **Some cannot be solved by computer.**
  - Provable! We can't wait for smarter algorithms

- **Some problems have no efficient solution**
  - Provably exponential 2^n so for "small" n …
- **Some have no known efficient solution, but**
  - If one does they all do!

Problem: Traveling Band

- Band wants you to schedule their concerts.
- They don't like to travel. Minimize the time they are on the bus!
- Given N cities, what is the best schedule (shortest distance) to visit all N cities once?



How do you calculate the best path?

- Try all paths
  - Atlanta, Raleigh, Dallas, Reno, Chicago
    - Add up the distance in this order
  - Dallas, Atlanta, Raleigh, Reno, Chicago
    - Add up the distance in this order
  - Etc.
- Would you agree to code this up?

Traveling Band questions bit.ly/101s21-0422-3

How long?

Number of Cities	All paths – N!	Time to solve - 10^9 Instructions per second
10	3 million	
15	10^{12}	
18	10^{15}	
20	10^{18}	
25	10^{25}	

How long?

Number of Cities	All paths – N!	Time to solve - 10^9 Instructions per second
10	3 million	< sec
15	10^{12}	
18	10^{15}	
20	10^{18}	
25	10^{25}	

How long?

Number of Cities	All paths – N!	Time to solve - 10^9 Instructions per second
10	3 million	< sec
15	10^{12}	16 min
18	10^{15}	
20	10^{18}	
25	10^{25}	

How long?

Number of Cities	All paths – N!	Time to solve - 10^9 Instructions per second
10	3 million	< sec
15	10^{12}	16 min
18	10^{15}	11 days
20	10^{18}	
25	10^{25}	

How long?

Number of Cities	All paths – N!	Time to solve - 10^9 Instructions per second
10	3 million	< sec
15	10^{12}	16 min
18	10^{15}	11 days
20	10^{18}	31 years
25	10^{25}	

How long?

Number of Cities	All paths – N!	Time to solve - 10^9 Instructions per second
10	3 million	< sec
15	10^{12}	16 min
18	10^{15}	11 days
20	10^{18}	31 years
25	10^{25}	10^8 years

How is Python like all other programming languages, how is it different?

Find all unique/different words in a file, in sorted order

Unique Words in Python

```
def main():
    f = open('/data/melville.txt', 'r')
    words = f.read().strip().split()
    allWords = set(words)

    for word in sorted(allWords):
        print(word)

if __name__ == "__main__":
    main()
```

Unique words in Java

```
import java.util.*;
import java.io.*;
public class Unique {
  public static void main(String[] args)
                              throws IOException{
    Scanner scan =
            new Scanner(new
   File("/data/melville.txt"));
    TreeSet<String> set = new TreeSet<String>();
    while (scan.hasNext()){
        String str = scan.next();
        set.add(str);
    }
    for(String s : set){
        System.out.println(s);
    }
  }
}
```

Unique words in C++

```
#include <iostream>
#include <fstream>
#include <set>
using namespace std;

int main(){
  ifstream input("/data/melville.txt");
  set<string> unique;
  string word;
  while (input >> word){
    unique.insert(word);
  }
  set<string>::iterator it = unique.begin();
  for(; it != unique.end(); it++){
    cout << *it << endl;
  }
  return 0;
}
```

Unique words in PHP

```
<?php

$wholething = file_get_contents("file:///data/melville.txt");
$wholething = trim($wholething);

$array = preg_split("/\s+/",$wholething);
$uni = array_unique($array);
sort($uni);
foreach ($uni as $word){
   echo $word."<br>";
}

?>
```

What is next?

- CompSci 201
  - Java, efficiency, other ways to organize data
- CompSci 230 – can take concurrently with 201
  - Discrete Mathematics

- CompSci 260 Computational Biology
- CompSci 216 Everything Data
- CompSci 240 Race, Gender, Class and Computing

End with A CS Story
bit.ly/101s21-0422-4