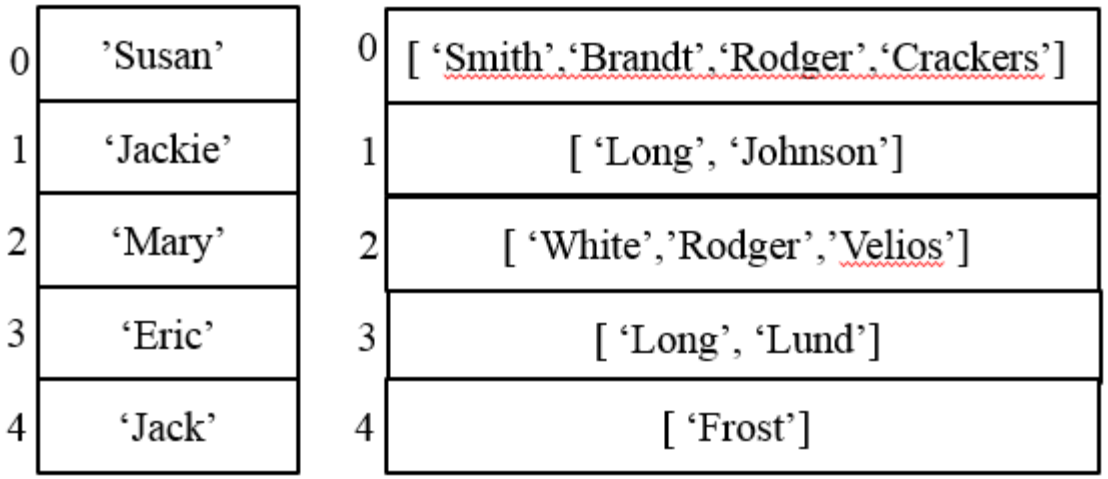
# CompSci 101
# Introduction to Computer Science




Nov 1, 2016

Prof. Rodger

# Announcements

- Reading and RQ due next time
- APT 6 due today, APT 7 out
  - Do more APTs to catch up….
- APT QUIZ 2 – Nov. 6-8
- Next Assignment out Thursday
- Lab this week!

- Today:
  - Processing data – how to organize it?
  - enumerate

# Lab this week! - Odds with poker!


# ACM Programming Contest

- Need volunteers to help on Saturday Nov 5 from 11:20am to 6pm. (includes food)
- Contest
  - Team of three, one computer, 6-8 problems, 5 hours
  - Problems “APT-like”
- See Piazza post for how to sign up

# Registration time…

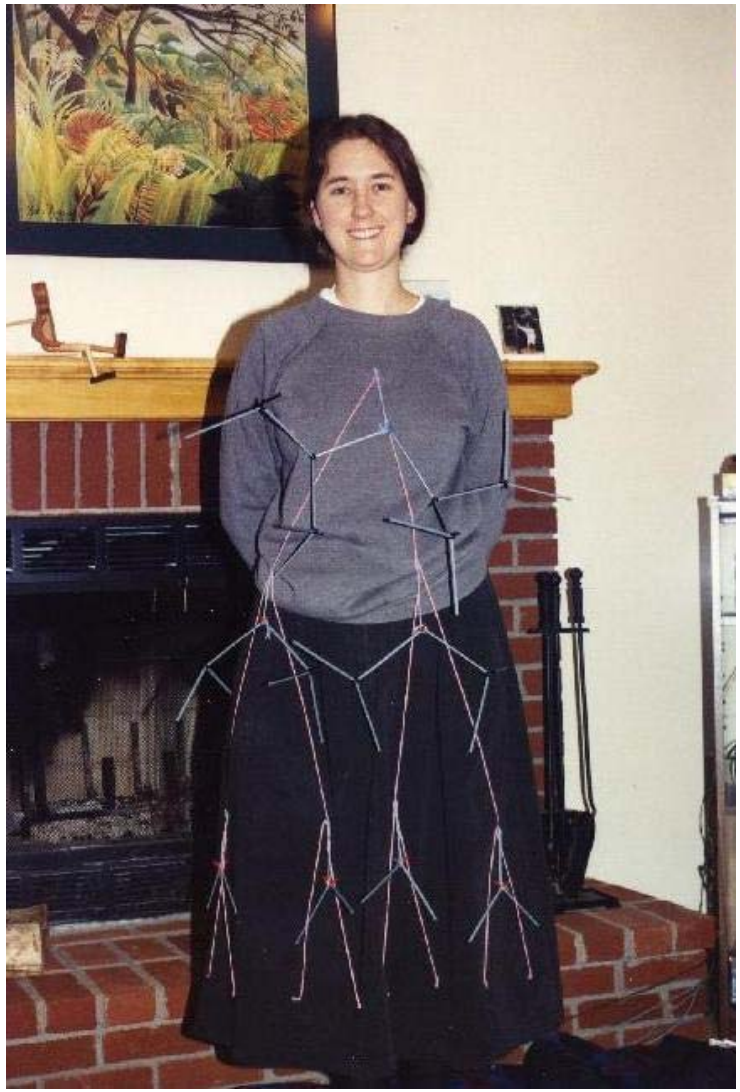
- What CS courses can you take next?
  - CompSci 201
  - CompSci 230

  - CompSci 230 is prereq for CompSci 330
  - CompSci 201 is prereq for many electives

# When Halloween and Computer Science COLLIDE!

# CS Concepts Coming Alive

- What data structure is this?



YARN,

in the

shap

binar

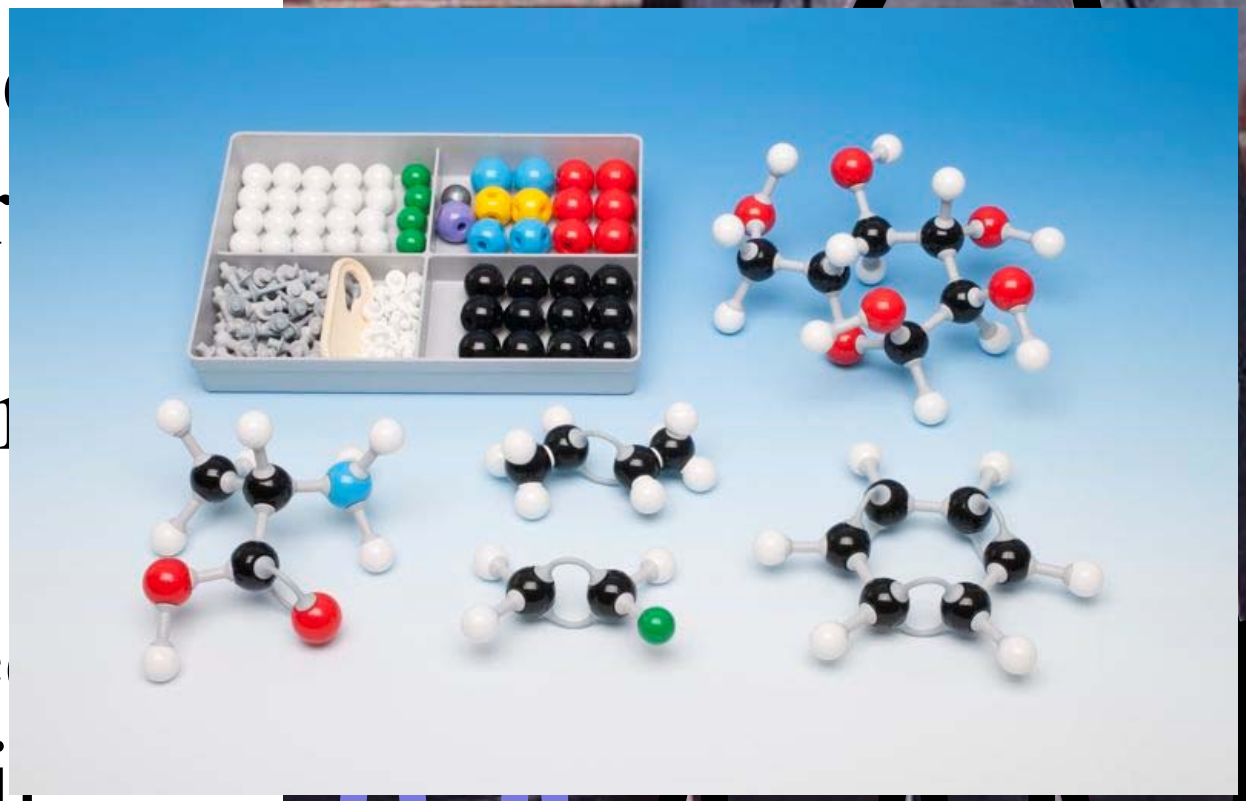
Subt

made

mole

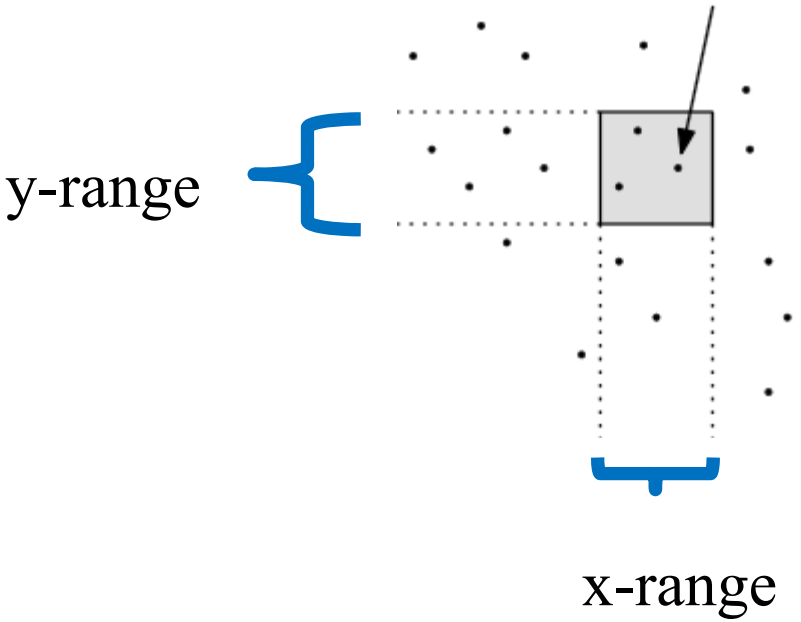
ki

What is it?

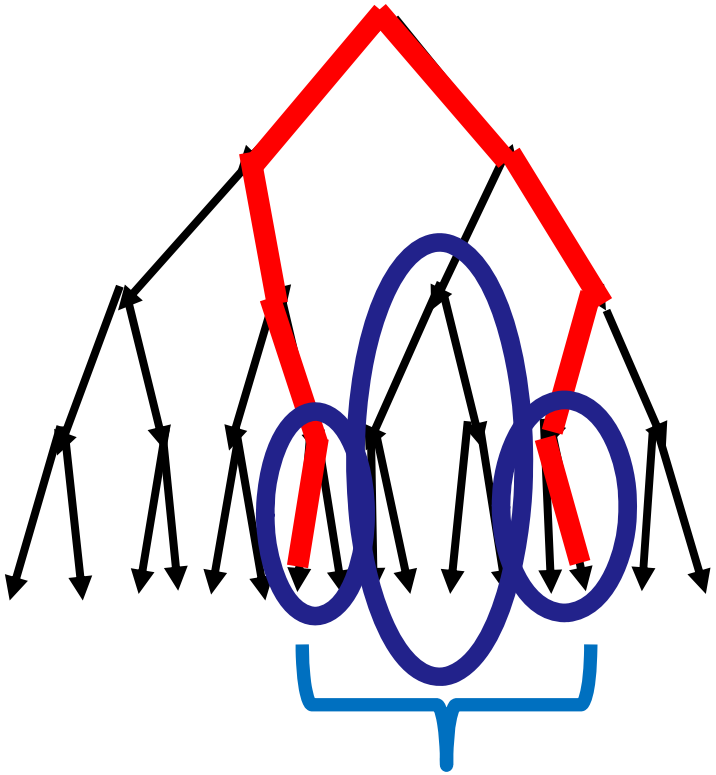


# 2D-range tree

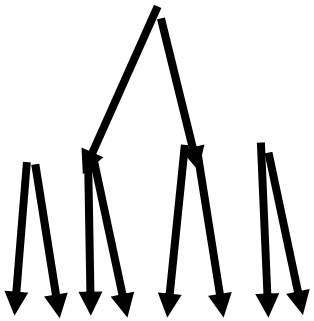
- Search in x-y plane
- Main tree organized by x-values
- Subtree organized by y values



# Binary Search tree of points in the plane – sorted by X-value



In the x-range



Each subtree organized by y-value

Search each subtree by y-value

# Problem: Longest Name

Given a list of names (one word only) and a letter (assume names start with capital letter, and letter is capital)

names = [‘Helen’, ‘Bob’, ‘Bart’, ‘Hugh’]

Find the longest name that starts with that letter

# Code for longest name

```
def longestName(alist, letter):
    longest = ''
    for name in alist:
        if letter == name[0] and
                len(name) > len(longest):
            longest = name
    return longest
```

How do you modify to find the location (position) of the longest name?

# Problem: Find the position of the longest name that starts with that letter

bitly/101f16-1101-1

# Enumerate

- An iterator, generates a sequence
- Generates tuples of (index, item)
- Used with for loop to get both index and item
- for (index,item) in enumerate(somelist):
  - You get both at the same time!

# Solve previous problem with enumerate

# Problem: Popular Name

- Given a list of names, determine the most popular first name and print that name with all of its last names.
- Input: Names are always two words, names are in a file. If multiple names are on the same line they are separated by a ":"
- Output: Most popular first name, followed by a ":", followed by corresponding last names separated by a blank

# Example Input File with 5 lines

```
Susan Smith:Jackie Long:Mary White
Susan Brandt
Jackie Johnson:Susan Rodger:Mary Rodger
Eric Long:Susan Crackers:Mary Velios
Jack Frost:Eric Lund
```

# Corresponding Output

```
Susan: Smith Brandt Rodger Crackers
```

# What do you need to solve this problem?
# bit.ly/101f16-1101-2

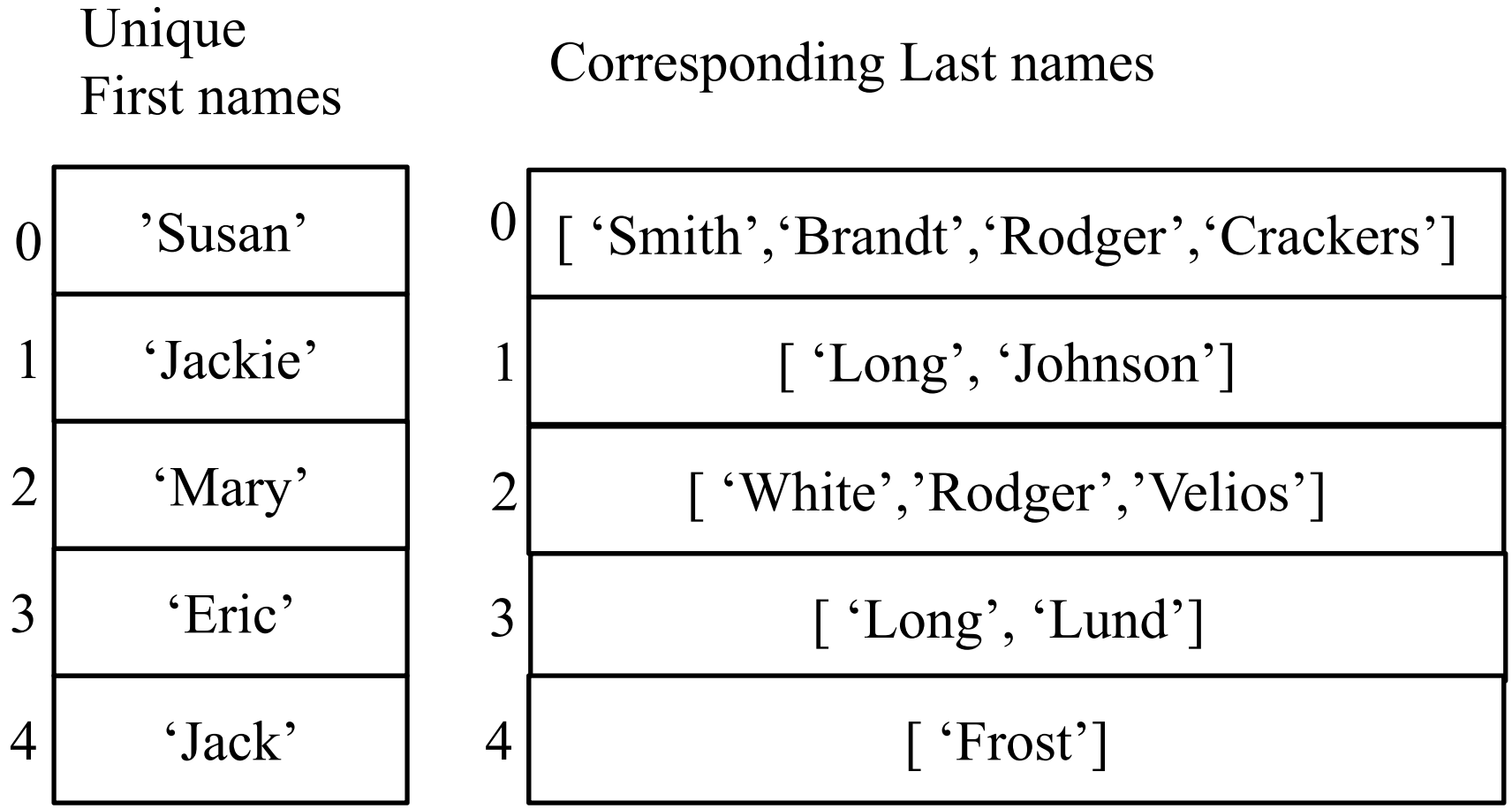
How might one organize the data to solve this problem?

How many different ways to solve this problem?

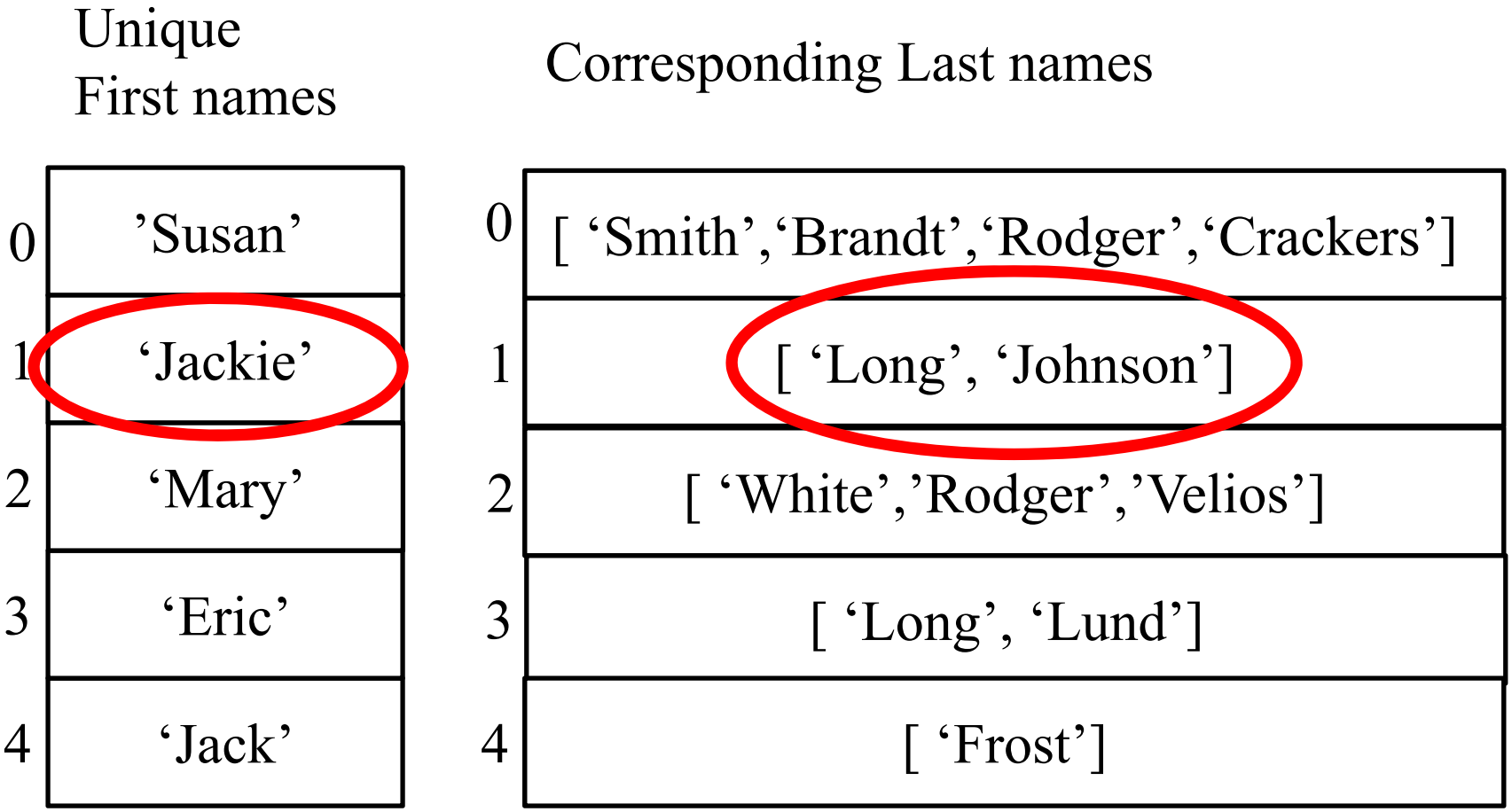
# One way to solve

- Create a list of unique first names
- Create a list of lists of last names that are associated with each first name

# Example – two lists



# Example – two lists




Jackie in position 1

Jackie's last names in position 1

# Now can we solve the problem?

- Compute those two lists that are associated with each other
  - List of unique first names
  - List of corresponding last names
- Compute the max list of last names
- Now easy to print the answer.
- See popular.py

# Look at the code for popular.py
# www.bit.ly/101f16-1101-3

- Which datafile is read in?
- What format is namelist in?
- Write the code for uniqueFirstNames

# Write the code: www.bit.ly/101f16-1101-4

- allLastNames
- correspondingLastNames
- printFirstWithLasts

# Finish

```
maxnum = max([len(item) for item in lastNames])
print maxnum
lastIndex = [index for (index, v) in
enumerate(lastNames) if len(v) == maxnum]
print "first name with most last names is:"
```

# Another way – list of lists

First word in each list is a first name
The rest are last names.

| | |
|---|---|
| 0 | [ ‘Susan’, ‘Smith’,‘Brandt’,‘Rodger’,‘Crackers’] |
| 1 | [‘Jackie’, ‘Long’, ‘Johnson’] |
| 2 | [‘Mary’, ‘White’,’Rodger’,’Velios’] |
| 3 | [ ‘Eric’, ‘Long’, ‘Lund’] |
| 4 | [ ‘Jack’, ‘Frost’] |

# Expanding the Problem

- Suppose we want to read from multiple data files

  names1.txt, names2.txt, names3.txt

See processFiles in popular.py