# CompSci 101

## Introduction to Computer Science

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>'Susan'</td>
<td></td>
<td>['Smith', 'Brandt', 'Rodger', 'Crackers']</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>'Jackie'</td>
<td></td>
<td>['Long', 'Johnson']</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>'Mary'</td>
<td></td>
<td>['White', 'Rodger', 'Velios']</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>'Eric'</td>
<td></td>
<td>['Long', 'Lund']</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>'Jack'</td>
<td></td>
<td>['Frost']</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

March 21, 2017

Prof. Rodger
Announcements

• Reading and RQ due next time
• APT 6 due Thursday
• Next Assignment out Thursday
• Lab this week!

• FTASB:
  – Review Tuples/generators
  – Enumerate
  – Processing data – how to organize it?
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
Lab this week! - Odds with poker!
def makeGray(pixel):
    (r,g,b) = pixel
    gray = (r+g+b)/3
    return (gray,gray,gray)

def grayit2(picname):
    im = Image.open(picname)
    im.show()
    pixels = [makeGray(pix) for pix in im.getdata()]
    nim = Image.new("RGB",im.size)
    nim.putdata(pixels)
    nim.show()
    nim.save("gray"+picname)
Review: generator

```python
im.getdata(), accessing pixels
```

- Returns something *like* a list
  - Use: `for pix in im.getdata():`
  - Generates pixels on-the-fly, can't slice or index unless you use `list(im.getdata())`
  - Structure is called a Python generator!
  - Saves on storing all pixels in memory if only accessed one-at-a-time
Making Tuples and Generators

• Overuse and abuse of parentheses
  – To create a tuple, use parentheses

    for pix in im.getdata():
    (r,g,b) = pix
    npx = (255-r,255-g,255-b)

  – To create a generator use parentheses as though creating a list comprehension!

    [2*n for n in range(10000)]
    (2*n for n in range(10000))

• See this in PyDev console
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
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 in a list of the longest name that starts with that letter.
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/101s17-0321-2
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

<table>
<thead>
<tr>
<th>Unique First names</th>
<th>Corresponding Last names</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Susan’</td>
<td>0: [‘Smith’, ‘Brandt’, ‘Rodger’, ‘Crackers’]</td>
</tr>
<tr>
<td>1: Jackie’</td>
<td>1: [‘Long’, ‘Johnson’]</td>
</tr>
<tr>
<td>2: Mary’</td>
<td>2: [‘White’, ‘Rodger’, ‘Velios’]</td>
</tr>
<tr>
<td>4: Jack’</td>
<td>4: [‘Frost’]</td>
</tr>
</tbody>
</table>
**Example – two lists**

<table>
<thead>
<tr>
<th>Unique First names</th>
<th>Corresponding Last names</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: ‘Jackie’</td>
<td>[‘Long’, ‘Johnson’]</td>
</tr>
<tr>
<td>4: ‘Jack’</td>
<td>[‘Frost’]</td>
</tr>
</tbody>
</table>

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

- Which datafile is read in?
- What format is namelist in?
- Write the code for uniqueFirstNames
Write the code:
www.bit.ly/101s17-0321-4

- allLastNames
- correspondingLastNames
- printFirstWithLasts
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.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>['Susan', 'Smith', 'Brandt', 'Rodger', 'Crackers']</td>
</tr>
<tr>
<td>1</td>
<td>['Jackie', 'Long', 'Johnson']</td>
</tr>
<tr>
<td>2</td>
<td>['Mary', 'White', 'Rodger', 'Velios']</td>
</tr>
<tr>
<td>3</td>
<td>['Eric', 'Long', 'Lund']</td>
</tr>
<tr>
<td>4</td>
<td>['Jack', 'Frost']</td>
</tr>
</tbody>
</table>
Expanding the Problem

- Suppose we want to read from multiple data files
  names1.txt, names2.txt, names3.txt

See processFiles in popular.py