More Summary Statistics

March 3, 2015
The Big Picture

Overall Goal
Build a Concordance of a text
• Locations of words
• Frequency of words

Today: Summary Statistics
• Python/IDLE stuff
• Nitpicky Python details
• A new kind of statement
• Count the number of words in *Moby Dick*
• Compute the average word length of *Moby Dick*
• Find the longest word in *Moby Dick*
Python vs. IDLE

• Python is a *language*: a set of rules for what’s “allowable”, much like English grammar (but more sensible).

• This language can be “interpreted”: turned into actions on a computer that do things like read and write files, print output to the screen, etc.

• IDLE is a program that takes input typed in Python and interprets it.
What IDLE does

• Prints “>>>” and waits for you to type Python
• When you type an expression, IDLE prints out the expression’s value to be helpful
• When you type an assignment, or list-assignment, or function-definition, IDLE just re-prints “>>>”
IDLE: working directory

- Python has a notion of “current folder” (called “current working directory”)

- If you type

  >>> f = open ("myfile.txt", "r")

and myfile.txt is in the current directory, the “open” will succeed. If not, it’ll fail.
IDLE: working directory

• If you open a new window in IDLE, it gives you a place to write a program

• Type your program there, and press F5
  – Python will insist on saving your program somewhere
  – Program is interpreted by Python in the Shell
  – Before that happens, the “working directory” is changed to the location where you saved the program!
IDLE: working directory

• **Current Directory**: place where your Python program is saved!

• **General rule**: Save in the same folder:
  – Your Python program
  – Your data

• All your “read” statements will work nicely!
IDLE: working directory

• You can use the full-path of the file
  >>> f = open ("C:\Users\Steve\...\myfile.txt", "r")
i.e., the “full path” to the file

• Or you can use the relative-path of the file
  >>> f = open ("myfile.txt", "r")
  if the file is in the current working directory
The Big Picture

Overall Goal
Build a Concordance of a text
- Locations of words
- Frequency of words

Today: Summary Statistics
- Python/IDLE stuff
- Nitpicky Python details
- A new kind of statement
- Count the number of words in Moby Dick
- Compute the average word length of Moby Dick
- Find the longest word in Moby Dick
- Get the vocabulary size of Moby Dick
Literals vs. Variables

"How does Python know what's a variable?"

• A literal is a piece of data that we give directly to Python
  - 'hello' is a string (str) literal
  - So are "hey there" and 'what\'s up'
  - 5 is an integer (int) literal
  - 32.8 is a floating-point (float) literal
"How does Python know what's a variable?"

• Variable names are made up of:
  – Letters (uppercase and lowercase)
  – Numbers (but only after the first letter)
  – Underscores

• Names for functions and types follow the same rules

• Anything else must be a literal or operator!
Using String Literals

def getFile(fnRelative):
    '''Opens the appropriate file in my folder'''
    fnAbsolute = "~/Users/hmendes/" + fnRelative
    return open(fnAbsolute, "r")

myFile = getFile("MobyDick.txt")
Using Functions

```python
def addOneBAD(t):
    t = t + 1
    return t

def addOneGOOD(t):
    x = t + 1
    return x
```

*t is a parameter, not a “scratchpad”

*Another variable (say x) may be your “scratchpad” variable

*Do not change argument values inside your functions;

*Use new variables instead
Order of evaluation

• When Python encounters an expression statement, it evaluates it (and IDLE prints it, in the interactive version)
• Evaluation (for number stuff) proceeds in “math order”: parenthesis, multiplication, division, addition, subtraction
• If expression contains a function-invocation, like `sum3(myList), we`
  – Evaluate the argument(s)
  – Apply the function
  – Replace the function-invocation with the returned value and continue evaluation
• What’s the difference between those?
  – `float(3/4) vs float(3)/4`
The Big Picture

Overall Goal
Build a Concordance of a text
• Locations of words
• Frequency of words

Today: Summary Statistics
• Administrative stuff
• Nitpicky Python details
• A new kind of statement
• Count the number of words in Moby Dick
• Compute the average word length of Moby Dick
• Find the longest word in Moby Dick
• Get the vocabulary size of Moby Dick
Review: Basic Types

- Integers
  - 3
  - -100
  - 1234

- Floats
  - 12.7
  - -99.99
  - 1234.0

- Strings
  - '12'
  - 'hi'
  - 'Moby'

- Booleans
  - True
  - False

New literals representing... truth and falseness
New Type: Booleans

- Either True or False
  - Note the capitalization

```
>>> x = True
>>> x
True
>>> y = False
>>> y
False
```
New Type: Booleans

- Either True or False
  - Note the capitalization

- New Operators

Remember

<table>
<thead>
<tr>
<th>Numerical Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operator</strong></td>
</tr>
<tr>
<td>Sum</td>
</tr>
<tr>
<td>Difference</td>
</tr>
</tbody>
</table>
New Type: Booleans

• Either True or False  
  – Note the capitalization

• New Operators

  Remember

<table>
<thead>
<tr>
<th>Numerical Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operator</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>Sum</td>
</tr>
<tr>
<td>Difference</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Boolean Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operator</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>Equality</td>
</tr>
<tr>
<td>Inequality</td>
</tr>
<tr>
<td>Less Than</td>
</tr>
<tr>
<td>Less Than or Equal To</td>
</tr>
<tr>
<td>Greater Than</td>
</tr>
<tr>
<td>Greater Than or Equal To</td>
</tr>
</tbody>
</table>
New Type: Booleans

- Either **True** or **False**
  - Note the capitalization

- New Operators

**Remember**

<table>
<thead>
<tr>
<th>Numerical Operators</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operator</strong></td>
<td><strong>Example</strong></td>
</tr>
<tr>
<td>Sum</td>
<td>1 + 2</td>
</tr>
<tr>
<td>Difference</td>
<td>1 - 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Boolean Operators</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operator</strong></td>
<td><strong>Example</strong></td>
</tr>
<tr>
<td>Equality</td>
<td>1 == 2</td>
</tr>
<tr>
<td>Inequality</td>
<td>1 != 2</td>
</tr>
<tr>
<td>Less Than</td>
<td>1 &lt; 2</td>
</tr>
<tr>
<td>Less Than or Equal To</td>
<td>1 &lt;= 2</td>
</tr>
<tr>
<td>Greater Than</td>
<td>1 &gt; 2</td>
</tr>
<tr>
<td>Greater Than or Equal To</td>
<td>1 &gt;= 2</td>
</tr>
</tbody>
</table>
New Type: Booleans

- Either True or False
  - Note the capitalization
- New Operators
- These are expressions
- Assignments have only one equals sign.

<table>
<thead>
<tr>
<th>Boolean Operators</th>
<th>Operator</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Equality</td>
<td>1 == 2</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Inequality</td>
<td>1 != 2</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>Less Than</td>
<td>1 &lt; 2</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>Less Than or Equal To</td>
<td>1 &lt;= 2</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>Greater Than</td>
<td>1 &gt; 2</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>Greater Than or Equal To</td>
<td>1 &gt;= 2</td>
<td>False</td>
</tr>
</tbody>
</table>
Boolean Types

Last Boolean Operators: **and, or and not**

<table>
<thead>
<tr>
<th>Boolean Operators</th>
<th>Examples</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>and</strong></td>
<td>(4&lt;5) and (6&lt;3)</td>
<td></td>
</tr>
<tr>
<td><strong>or</strong></td>
<td>(4&lt;5) or (6&lt;3)</td>
<td></td>
</tr>
<tr>
<td><strong>not</strong></td>
<td>not (4&lt;5)</td>
<td></td>
</tr>
</tbody>
</table>
## Boolean Types

**Last Boolean Operators:** and, or and not

<table>
<thead>
<tr>
<th>Boolean Operators</th>
<th>Operator</th>
<th>Examples</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>and</td>
<td>(4&lt;5) and (6&lt;3)</td>
<td>True and False</td>
</tr>
<tr>
<td></td>
<td>or</td>
<td>(4&lt;5) or (6&lt;3)</td>
<td>True or False</td>
</tr>
<tr>
<td></td>
<td>not</td>
<td>not(4&lt;5)</td>
<td>not(True)</td>
</tr>
</tbody>
</table>
Boolean Types

Last Boolean Operators: **and**, **or** and **not**

<table>
<thead>
<tr>
<th>Boolean Operators</th>
<th>Operator</th>
<th>Examples</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>and</td>
<td>(4&lt;5) and (6&lt;3)</td>
<td>True and False</td>
</tr>
<tr>
<td></td>
<td>or</td>
<td>(4&lt;5) or (6&lt;3)</td>
<td>True or False</td>
</tr>
<tr>
<td></td>
<td>not</td>
<td>not(4&lt;5)</td>
<td>not(True)</td>
</tr>
</tbody>
</table>
Boolean Types

Last Boolean Operators: **and**, **or**, **and not**

<table>
<thead>
<tr>
<th>Boolean Operators</th>
<th>Operator</th>
<th>Examples</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>and</td>
<td>(4&lt;5) and (6&lt;3)</td>
<td>True and False</td>
</tr>
<tr>
<td></td>
<td>or</td>
<td>(4&lt;5) or (6&lt;3)</td>
<td>True or False</td>
</tr>
<tr>
<td></td>
<td>not</td>
<td>not(4&lt;5)</td>
<td>not(True)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>More Examples</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4&lt;5) and ((6&lt;3) or (5==5))</td>
<td></td>
</tr>
<tr>
<td>(5==4) or (not(6&lt;3))</td>
<td></td>
</tr>
</tbody>
</table>
**Boolean Types**

Last Boolean Operators: **and, or and not**

<table>
<thead>
<tr>
<th>Boolean Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operator</strong></td>
</tr>
<tr>
<td><strong>and</strong></td>
</tr>
<tr>
<td><strong>or</strong></td>
</tr>
<tr>
<td><strong>not</strong></td>
</tr>
</tbody>
</table>

**More Examples**

<table>
<thead>
<tr>
<th><strong>Result</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>(4&lt;5) and ((6&lt;3) or (5==5)) True and (False or True)</td>
</tr>
<tr>
<td>(5==4) or (not(6&lt;3))</td>
</tr>
</tbody>
</table>

CSCI 0931 - Intro. to Comp. for the Humanities and Social Sciences
## Boolean Types

### Last Boolean Operators: and, or and not

<table>
<thead>
<tr>
<th>Boolean Operators</th>
<th>Examples</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operator</strong></td>
<td><strong>Examples</strong></td>
<td><strong>Result</strong></td>
</tr>
<tr>
<td>and</td>
<td>(4&lt;5) and (6&lt;3)</td>
<td>True and False</td>
</tr>
<tr>
<td>or</td>
<td>(4&lt;5) or (6&lt;3)</td>
<td>True or False</td>
</tr>
<tr>
<td>not</td>
<td>not(4&lt;5)</td>
<td>not(True)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>More Examples</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4&lt;5) and ((6&lt;3) or (5==5))</td>
<td>True</td>
</tr>
<tr>
<td>(5==4) or (not(6&lt;3))</td>
<td></td>
</tr>
</tbody>
</table>
### Boolean Types

Last Boolean Operators: `and`, `or`, and `not`

<table>
<thead>
<tr>
<th>Boolean Operators</th>
<th>Examples</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>and</strong></td>
<td><code>(4&lt;5)</code> and <code>(6&lt;3)</code></td>
<td>True and False</td>
</tr>
<tr>
<td><strong>or</strong></td>
<td><code>(4&lt;5)</code> or <code>(6&lt;3)</code></td>
<td>True or False</td>
</tr>
<tr>
<td><strong>not</strong></td>
<td><code>not(4&lt;5)</code></td>
<td><code>not(True)</code></td>
</tr>
</tbody>
</table>

**More Examples**

<table>
<thead>
<tr>
<th>Examples</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>(4&lt;5)</code> and <code>((6&lt;3) or (5==5))</code></td>
<td>True and (False or True)</td>
</tr>
<tr>
<td><code>(5==4)</code> or <code>(not(6&lt;3))</code></td>
<td>False or not(False)</td>
</tr>
</tbody>
</table>

CSCI 0931 - Intro. to Comp. for the Humanities and Social Sciences 28
# Boolean Types

**Last Boolean Operators:** and, or and not

<table>
<thead>
<tr>
<th>Boolean Operators</th>
<th>Examples</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operator</strong></td>
<td><strong>Examples</strong></td>
<td><strong>Result</strong></td>
</tr>
<tr>
<td>and</td>
<td>(4&lt;5) and (6&lt;3)</td>
<td>True and False</td>
</tr>
<tr>
<td>or</td>
<td>(4&lt;5) or (6&lt;3)</td>
<td>True or False</td>
</tr>
<tr>
<td>not</td>
<td>not(4&lt;5)</td>
<td>not(True)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>More Examples</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4&lt;5) and ((6&lt;3) or (5==5))</td>
<td>True and (False or True)</td>
</tr>
<tr>
<td>(5==4) or (not(6&lt;3))</td>
<td>False or not(False)</td>
</tr>
</tbody>
</table>
## Boolean Expressions on Strings

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equality</td>
<td>'a' == 'b'</td>
<td>False</td>
</tr>
<tr>
<td>Inequality</td>
<td>'a' != 'b'</td>
<td>True</td>
</tr>
<tr>
<td>Less Than</td>
<td>'a' &lt; 'b'</td>
<td>True</td>
</tr>
<tr>
<td>Less Than or Equal To</td>
<td>'a' &lt;= 'b'</td>
<td>True</td>
</tr>
<tr>
<td>Greater Than</td>
<td>'a' &gt; 'b'</td>
<td>False</td>
</tr>
<tr>
<td>Greater Than or Equal To</td>
<td>'a' &gt;= 'b'</td>
<td>False</td>
</tr>
</tbody>
</table>
Review: Statements

• Expression Statements
  Calculates something

• Assignment Statements
  Stores a value for a variable in memory table

• List-Assignment Stmts.
  Replaces
  An item or slices of an existing list with new value(s)

• For Statements
  "For each element in myList, do something"

• If Statements
  If A is true, then do something, otherwise do something else
Boolean Statements (If Stmts)

• “If something’s true, do A”

```python
def compare(x, y):
    if x > y:
        print(x, ' is greater than ', y)
```
Boolean Statements (\texttt{If Stmts})

• “If something’s true, do A, otherwise, do B”

```python
def compare(x, y):
    if x > y:
        print(x, ' is greater than ', y)
    else:
        print(x, ' is less than or equal to ', y)
```
Boolean Statements (If Stmts)

• “If something’s true, do A, otherwise, check something else; if that's true, do B, otherwise, do C”

```python
def compare(x, y):
    ' ' if x > y:
        print(x, ' is greater than ', y)
    else:
        if x < y:
            print(x, ' is less than ', y)
        else:
            print(x, ' is equal to ', y)
```
Boolean Statements (İf Stmts) shorthand!

- “If something’s true, do A, otherwise, check something else; if that's true, do B, otherwise, do C”

```python
def compare(x, y):
    ' 
    if x > y:
        print(x, ' is greater than ', y)
    elif x < y:
        print(x, ' is less than ', y)
    else:
        print(x, ' is equal to ', y)
```
Review: Other Things

• Lists (a type of **data structure**)

```python
[0, 1, 2]  ['hi', 'there']  ['hi', 0.0]
[1, 2, 3, 4, 5, True, False, 'true', 'one']
```
Review: Other Things

• Lists (a type of **data structure**)

  - [0,1,2]
  - ['hi','there']
  - ['hi',0.0]
  - [1,2,3,4,5,True,False,`true','one']

• Files (an **object** that we can open, read, close)

  ```python
  myFile = open(fileName,`r')
  ```
def countWordsInShel():
    '''Returns the number of words in the poem.'''
    myList = readShel()
    # the 'count' variable counts the number of words
    count = 0
    for word in myList:
        count = count + 1
    print "There are ",count," words in the poem."
    return count
Execution model for “for” loops

• If the loop variable isn’t in the memory table...add it

• Repeatedly assign to it sequential items in the list...

• ...and execute the statements within the loop

• Note: when done, the loop variable will be in the memory table, with its last value
def countWordsInShel():
    '''Returns the number of words in the poem.'''
    myList = readShel()
    # the 'count' variable counts the number of words
    count = 0
    for word in myList:
        count = count + 1
    print "There are ", count, " words in the poem."
    return count

Good Programming Practices:
Documentation!
The Big Picture

Overall Goal
Build a Concordance of a text
• Locations of words
• Frequency of words

Today: Summary Statistics
• Administrative stuff
• Nitpicky Python details
• A new kind of statement
• Count the number of words in Moby Dick
• Compute the average word length of Moby Dick
• Find the longest word in Moby Dick
Python Functions

<table>
<thead>
<tr>
<th>Preloaded Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>len</td>
</tr>
<tr>
<td>len</td>
</tr>
<tr>
<td>len</td>
</tr>
</tbody>
</table>

```python
>>> len([3, 47, 91, -6, 18])

>>> uselessList = ['contextless', 'words']
>>> len(uselessList)

>>> creature = 'woodchuck'
>>> len(creature)
```
# Python Functions

<table>
<thead>
<tr>
<th>Preloaded Functions</th>
<th>Len</th>
<th>List OR String</th>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>len</td>
<td></td>
<td>List OR String</td>
<td>Integer</td>
</tr>
</tbody>
</table>
## Python Functions

<table>
<thead>
<tr>
<th>Preloaded Functions</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>len</td>
<td>List OR String OR ...</td>
<td>Integer</td>
</tr>
<tr>
<td>float</td>
<td>Number (as an Integer, Float, or String)</td>
<td>Float</td>
</tr>
<tr>
<td>int</td>
<td>Number (as an Integer, Float, or String)</td>
<td>Integer</td>
</tr>
<tr>
<td>str</td>
<td>Integer, Float, String, or List</td>
<td>String</td>
</tr>
</tbody>
</table>
Python Functions

<table>
<thead>
<tr>
<th>Preloaded Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>len</strong></td>
</tr>
<tr>
<td>List OR String OR ...</td>
</tr>
<tr>
<td>Integer</td>
</tr>
<tr>
<td><strong>float</strong></td>
</tr>
<tr>
<td>Number (as an Integer, Float, or String)</td>
</tr>
<tr>
<td>Float</td>
</tr>
<tr>
<td><strong>int</strong></td>
</tr>
<tr>
<td>Number (as an Integer, Float, or String)</td>
</tr>
<tr>
<td>Integer</td>
</tr>
<tr>
<td><strong>str</strong></td>
</tr>
<tr>
<td>Integer, Float, String, or List</td>
</tr>
<tr>
<td>String</td>
</tr>
<tr>
<td><strong>range</strong></td>
</tr>
<tr>
<td>Two Integers</td>
</tr>
<tr>
<td>1. Start Index (Inclusive)</td>
</tr>
<tr>
<td>2. End Index (Exclusive)</td>
</tr>
<tr>
<td>List of Integers</td>
</tr>
</tbody>
</table>

These functions cast a variable of one type to another type.
The Big Picture

Overall Goal
Build a Concordance of a text
  • Locations of words
  • Frequency of words

Today: Summary Statistics
• Administrative stuff
• Nitpicky Python details
• A new kind of statement
• Count the number of words in Moby Dick
• Compute the average word length of Moby Dick
• Find the longest word in Moby Dick
ACT2-3

• Do Task 1
ACT2-3

• Do Task 2
ACT2-3

• Do Task 3
Compute the Average Word Length of Moby Dick

def avgWordLengthInMobyDick():
    '''Gets the average word length in MobyDick.txt'''

    return avg
def avgWordLengthInMobyDick():
    '''Gets the average word length in MobyDick.txt'''
    myList = readMobyDick()
    s = 0
    for word in myList:
        s = s + len(word)
    avg = s/float(len(myList))
    return avg
Is our Program Correct?

```python
>>> MDList = readMobyDick()
```

```python
>>> MDList[0:99]
['CHAPTER', '1', 'Loomings', 'Call', 'me', 'Ishmael.', 'Some', 'years', 'ago--never', 'mind', 'how', 'long', 'precisely--', 'having', 'little', 'or', 'no', 'money', 'in', 'my', 'purse', 'and', 'nothing', 'particular', 'to', 'interest', 'me', 'on', 'shore', 'I', 'thought', 'I', 'would', 'sail', 'about', 'a', 'little', 'and', 'see', 'the', 'watery', 'part', 'of', 'the', 'world', 'It', 'is', 'a', 'way', 'I', 'have', 'of', 'driving', 'off', 'the', 'spleen', 'and', 'regulating', 'the', 'circulation', Whenever', 'I', 'find', 'myself', 'growing', 'grim', 'about', 'the', 'mouth', whenever', 'it', 'is', 'a', 'damp', 'drizzly', 'November', 'in', 'my', 'soul', whenever', 'I', 'find', 'myself', 'involuntarily', 'pausing', 'before', 'coffin', 'warehouses', 'and', 'bringing', 'up', 'the', 'rear', 'of', 'every', 'funeral', 'I', 'meet', 'and']
```
The Big Picture

Overall Goal
Build a Concordance of a text
• Locations of words
• Frequency of words

Today: Summary Statistics
• Administrative stuff
• Nitpicky Python details
• A new kind of statement
• Count the number of words in *Moby Dick*
• Compute the average word length of *Moby Dick*
• Find the longest word in *Moby Dick*
Get the Longest Word in Moby Dick

```python
def getLongestWordInMobyDick():
    '''Returns the longest word in MobyDick.txt'''

    return longestword
```
def getLongestWordInMobyDick():
    '''Returns the longest word in MobyDick.txt'''
    myList = readMobyDick()
    longestword = ""
    for word in myList:
        if len(word) > len(longestword):
            longestword = word
    return longestword
Get the Longest Word in Moby Dick

def getLongestWordInMobyDick():
    '''Returns the longest word in MobyDick.txt'''
    myList = readMobyDick()
    longestword = ""
    for word in myList:
        if len(word) > len(longestword):
            longestword = word
    return longestword

Is our program correct?
The Big Picture

Overall Goal
Build a Concordance of a text
• Locations of words
• Frequency of words

Today: Summary Statistics
• Administrative stuff
• Nitpicky Python details
• A new kind of statement
• Count the number of words in Moby Dick
• Compute the average word length of Moby Dick
• Find the longest word in Moby Dick
Next Class

Next time, we’ll look at counting the *vocabulary size*, not just the total number of words.