

# More Summary Statistics

Oct 13, 2015

# The Big Picture

## Overall Goal

Build a Concordance of a text

- *Locations* of words
- *Frequency* of words

## Today: Summary Statistics

- For Loops and Booleans from last time
- 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 `For` Statements (For Loops)

“For each element in list `myList`, do something”

```
>>> myList = [1, 2, 3]
>>>
```

# Python `FOR` Statements (For Loops)

“For each element in list `myList`, do something”

```
>>> myList = [1,2,3]
>>> for element in myList:
...     print element


1
2
3
>>>
```

# Python For Statements (For Loops)

“For each element in list myList, do something”

```
>>> myList = [1, 2, 3]
>>> for element in myList:
...     print element

1
2
3
>>>
```



# Python For Statements (For Loops)

“For each element in list myList, do something”

```
>>> myList = [1, 2, 3]
>>> for element in myList:
...     print element
1
2
3
>>>
```

# Python For Statements (For Loops)

“For each element in list myList, do something”

```
>>> myList = [1, 2, 3]
>>> for num in myList:
...     print num
1
2
3
>>>
```

# Python For Statements (For Loops)

“For each element in list myList, do something”

```
>>> myList = [1, 2, 3]
>>> for num in myList:
...     print num
1
2
3
>>>
```

The diagram illustrates the execution of a Python for loop. It shows a code block with three lines of code. The first line is `myList = [1, 2, 3]`, where `myList` is highlighted in blue. A blue box labeled "List" has an arrow pointing to `myList`. The second line is `for num in myList:`, where `num` is highlighted in green. A green box labeled "Variable" has an arrow pointing to `num`. The third line is `print num`, which is indented. A red box labeled "Indentation Matters!!" has a bracket pointing to the indentation of this line. Below the code block, the numbers 1, 2, and 3 are printed on separate lines, representing the output of the loop. The prompt `>>>` is shown at the beginning and end of the code block.



# Activity 2-2

- Do Task 3

# Word Count for Shel's Poem

```
def countWordsInShel():  
    '''Returns the number of words in the poem.'''  
  
    return count
```

# Word Count for Shel's Poem

```
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
```

# Word Count for Shel's Poem

## Good Programming Practices: Documentation!

```
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
```

Program Description  
(triple quotes)

Comment (#)

Print Statement

# 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

# The Big Picture

## Overall Goal

Build a Concordance of a text

- *Locations* of words
- *Frequency* of words

## From Last Lecture

- Review material from last class
- Count the number of words in poem.txt (by Shel Silverstein)
- Count the number of words in Moby Dick
  - There's a shortcut...
- Compute the average word length of Moby Dick
- Find the longest word in Moby Dick

# A Shortcut to List Length

## Preloaded Functions

len

List

Integer

```
>>> len(myList)
```

# A Shortcut to List Length

## Preloaded Functions

len

List

Integer

```
>>> len(myList)
```

## From Last Lecture

- Review material from last class
- Count the number of words in poem.txt (by Shel Silverstein)
- Count the number of words in Moby Dick
  - There's a shortcut...
- Compute the average word length of Moby Dick
- Find the longest word in Moby Dick



# Python Functions

## Preloaded Functions

len

List OR String

Integer

# Python Functions

Preloaded Functions		
<code>len</code>	List OR String	Integer
<code>float</code>	Number (as an Integer, Float, or String)	Float
<code>int</code>	Number (as an Integer, Float, or String)	Integer
<code>str</code>	Integer, Float, String, or List	String

These functions *cast* a variable of one type to another type

- `3 / 4 -> 0.75`
- `3/float(4) -> 0.75`, `float(3)/4 -> 0.75`, `float(3)/float(4) -> 0.75`
- if an arithmetic expression involves a float, the result will be a float. `3 + 0.0 -> 3.0`, `3 + float(0) -> 3.0`
- New shorthand: “->” means “evaluates to”

# Python Functions

## Preloaded Functions

len	List OR String	Integer
float	Number (as an Integer, Float, or String)	Float
int	Number (as an Integer, Float, or String)	Integer
str	Integer, Float, String, or List	String
range	Two Integers 1. Start Index (Inclusive) 2. End Index (Exclusive)	List of Integers

These functions *cast* a variable of one type to another type



# Activity

- Do Task 4

# Compute the Average Word Length of Moby Dick

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

# Compute the Average Word Length of Moby Dick

```
def avgWordLengthInMobyDick():  
    '''Gets the average word length in MobyDick.txt'''  
    myList = readMobyDick()  
    s = 0  
    for word in myList:  
        s = s + len(word)  
    avg = s / len(myList)  
    return avg
```

# Is our Program Correct?

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

## From Last Lecture

- Review material from last class
- Count the number of words in poem.txt (by Shel Silverstein)
- Count the number of words in Moby Dick
  - There's a shortcut...
- Compute the average word length of Moby Dick
- Find the longest word in Moby Dick



# Get the Longest Word in Moby Dick

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

# Get the Longest Word in Moby Dick

```
def getLongestWordInMobyDick():  
    '''Returns the longest word in MobyDick.txt'''  
    myList = readMobyDick()  
    longestLen = 0  
    longestWord = ""  
    for word in myList:  
        if len(word) > longestLen:  
            longestLen = len(word)  
            longestWord = word  
    return longestWord
```

# The Big Picture

## Overall Goal

Build a Concordance of a text

- *Locations* of words
- *Frequency* of words

## Today: Summary Statistics

- Review material from last class
- Count the number of words in poem.txt (by Shel Silverstein)
- Count the number of words in Moby Dick
  - There's a shortcut...
- 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.

# The Big Picture

## Overall Goal

Build a Concordance of a text

- *Locations* of words
- *Frequency* of words

## Today: Summary Statistics

- For Loops and Booleans from last time
- 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

# 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


# Literals vs. Variables

"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/alexandra/" + fnRelative  
    return open(fnAbsolute, "r")  
  
myFile = getFile("MobyDick.txt")
```



# Using Functions

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

t is a parameter, not a “scratchpad”

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

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



# 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

Numerical Operators		
Operator	Example	Result
Sum	$1 + 2$	3
Difference	$1 - 2$	-1

# New Type: Booleans

- Either `True` or `False`
  - Note the capitalization
- New Operators

Remember

Numerical Operators		
Operator	Example	Result
Sum	<code>1 + 2</code>	3
Difference	<code>1 - 2</code>	-1

Boolean Operators		
Operator	Example	Result
Equality	<code>1 == 2</code>	
Inequality	<code>1 != 2</code>	
Less Than	<code>1 &lt; 2</code>	
Less Than or Equal To	<code>1 &lt;= 2</code>	
Greater Than	<code>1 &gt; 2</code>	
Greater Than or Equal To	<code>1 &gt;= 2</code>	

# New Type: Booleans

- Either `True` or `False`
  - Note the capitalization
- New Operators

Remember

Numerical Operators		
Operator	Example	Result
Sum	<code>1 + 2</code>	3
Difference	<code>1 - 2</code>	-1

Boolean Operators		
Operator	Example	Result
Equality	<code>1 == 2</code>	False
Inequality	<code>1 != 2</code>	True
Less Than	<code>1 &lt; 2</code>	True
Less Than or Equal To	<code>1 &lt;= 2</code>	True
Greater Than	<code>1 &gt; 2</code>	False
Greater Than or Equal To	<code>1 &gt;= 2</code>	False

# New Type: Booleans

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

Boolean Operators		
Operator	Example	Result
Equality	<code>1 == 2</code>	<code>False</code>
Inequality	<code>1 != 2</code>	<code>True</code>
Less Than	<code>1 &lt; 2</code>	<code>True</code>
Less Than or Equal To	<code>1 &lt;= 2</code>	<code>True</code>
Greater Than	<code>1 &gt; 2</code>	<code>False</code>
Greater Than or Equal To	<code>1 &gt;= 2</code>	<code>False</code>

# Boolean Types

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

Boolean Operators			
Operator	Examples		Result
<code>and</code>	<code>(4&lt;5) and (6&lt;3)</code>		
<code>or</code>	<code>(4&lt;5) or (6&lt;3)</code>		
<code>not</code>	<code>not (4&lt;5)</code>		



# Boolean Types

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

Boolean Operators			
Operator	Examples		Result
<code>and</code>	<code>(4&lt;5) and (6&lt;3)</code>	<code>True and False</code>	
<code>or</code>	<code>(4&lt;5) or (6&lt;3)</code>	<code>True or False</code>	
<code>not</code>	<code>not (4&lt;5)</code>	<code>not (True)</code>	

# Boolean Types

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

Boolean Operators			
Operator	Examples		Result
<code>and</code>	<code>(4&lt;5) and (6&lt;3)</code>	<code>True and False</code>	<code>False</code>
<code>or</code>	<code>(4&lt;5) or (6&lt;3)</code>	<code>True or False</code>	<code>True</code>
<code>not</code>	<code>not (4&lt;5)</code>	<code>not (True)</code>	<code>False</code>

# Boolean Types

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

Boolean Operators			
Operator	Examples		Result
<code>and</code>	<code>(4&lt;5) and (6&lt;3)</code>	<code>True and False</code>	<code>False</code>
<code>or</code>	<code>(4&lt;5) or (6&lt;3)</code>	<code>True or False</code>	<code>True</code>
<code>not</code>	<code>not (4&lt;5)</code>	<code>not (True)</code>	<code>False</code>

More Examples		Result
<code>(4&lt;5) and ((6&lt;3) or (5==5))</code>		
<code>(5==4) or (not (6&lt;3))</code>		

# Boolean Types

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

Boolean Operators			
Operator	Examples		Result
<code>and</code>	<code>(4&lt;5) and (6&lt;3)</code>	<code>True and False</code>	<code>False</code>
<code>or</code>	<code>(4&lt;5) or (6&lt;3)</code>	<code>True or False</code>	<code>True</code>
<code>not</code>	<code>not (4&lt;5)</code>	<code>not (True)</code>	<code>False</code>

More Examples		Result
<code>(4&lt;5) and ((6&lt;3) or (5==5))</code>	<code>True and (False or True)</code>	
<code>(5==4) or (not (6&lt;3))</code>		

# Boolean Types

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

Boolean Operators			
Operator	Examples		Result
<code>and</code>	<code>(4&lt;5) and (6&lt;3)</code>	<code>True and False</code>	<code>False</code>
<code>or</code>	<code>(4&lt;5) or (6&lt;3)</code>	<code>True or False</code>	<code>True</code>
<code>not</code>	<code>not (4&lt;5)</code>	<code>not (True)</code>	<code>False</code>

More Examples		Result
<code>(4&lt;5) and ((6&lt;3) or (5==5))</code>	<code>True and (False or True)</code>	<code>True</code>
<code>(5==4) or (not (6&lt;3))</code>		

# Boolean Types

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

Boolean Operators			
Operator	Examples		Result
<code>and</code>	<code>(4&lt;5) and (6&lt;3)</code>	<code>True and False</code>	<code>False</code>
<code>or</code>	<code>(4&lt;5) or (6&lt;3)</code>	<code>True or False</code>	<code>True</code>
<code>not</code>	<code>not (4&lt;5)</code>	<code>not (True)</code>	<code>False</code>

More Examples		Result
<code>(4&lt;5) and ((6&lt;3) or (5==5))</code>	<code>True and (False or True)</code>	<code>True</code>
<code>(5==4) or (not (6&lt;3))</code>	<code>False or not (False)</code>	

# Boolean Types

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

Boolean Operators			
Operator	Examples		Result
<code>and</code>	<code>(4&lt;5) and (6&lt;3)</code>	<code>True and False</code>	<code>False</code>
<code>or</code>	<code>(4&lt;5) or (6&lt;3)</code>	<code>True or False</code>	<code>True</code>
<code>not</code>	<code>not (4&lt;5)</code>	<code>not (True)</code>	<code>False</code>

More Examples		Result
<code>(4&lt;5) and ((6&lt;3) or (5==5))</code>	<code>True and (False or True)</code>	<code>True</code>
<code>(5==4) or (not (6&lt;3))</code>	<code>False or not (False)</code>	<code>True</code>

# Boolean Expressions on Strings

Boolean Operators on Strings		
Operator	Example	Result
Equality	'a' == 'b'	False
Inequality	'a' != 'b'	True
Less Than	'a' < 'b'	True
Less Than or Equal To	'a' <= 'b'	True
Greater Than	'a' > 'b'	False
Greater Than or Equal To	'a' >= 'b'	False



# Review: Statements

- Expression Statements
- Assignment Statements
- List-Assignment Sttmts.
  
- For Statements
  
- If Statements

Calculates something

*Stores* a value for a variable  
in memory table

*Replaces*  
An item or slices of an  
existing list with new  
value(s)

“For each element in  
`myList`, do something”

If `A` is true, then do  
something, otherwise do  
something else

# Boolean Statements (If Stmts)

- “If something’s true, do A”

```
def compare(x, y):  
    if x > y:  
        print(x, ' is greater than ', y)
```

# Boolean Statements (If Stmts)

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

```
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”

```
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 (If Stmts) shorthand!

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

```
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**)

```
[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)

```
myFile = open(fileName,`r`)
```

# 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

## Preloaded Functions

len	List	Integer
len	String	Integer
len	...	Integer

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

```
>>> uselessList = ['contextless', 'words']
```

```
>>> len(uselessList)
```

```
>>> creature = 'woodchuck'
```

```
>>> len(creature)
```

# Python Functions

## Preloaded Functions

len

List **OR** String

Integer

# Python Functions

Preloaded Functions		
<code>len</code>	List OR String OR ...	Integer
<code>float</code>	Number (as an Integer, Float, or String)	Float
<code>int</code>	Number (as an Integer, Float, or String)	Integer
<code>str</code>	Integer, Float, String, or List	String

These functions *cast* a variable of one type to another type

# Python Functions

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

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
```

# Compute the Average Word Length of Moby Dick

```
def avgWordLengthInMobyDick():
    '''Gets the average word length in MobyDick.txt'''
    myList = readMobyDick()
    s = 0
    for word in myList:
        s = s + len(word)
    avg = s/len(myList)
    return avg
```

# Is our Program Correct?

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

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

# 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