

Textual Analysis: Summary Statistics

March 1, 2012

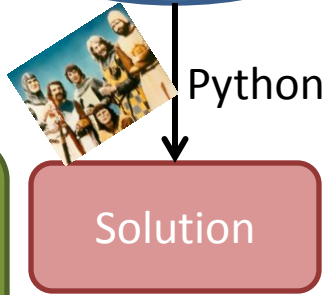
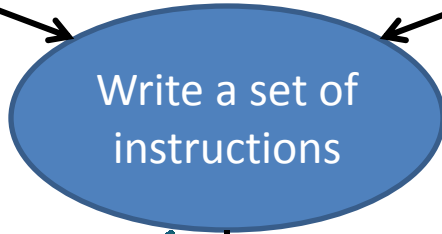
Textual Analysis



Build a Concordance of a text

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

- Word frequencies across time
 - Determine authorship
 - Count labels to determine liberal media bias



```
ACTACGTCGACTACGATCAC
GATCGCGCGATCACGTATTT
ACGATCAGCTACGATCGATC
TACGATCGTAGCTGTGATCG
```

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
- Compute the average word length of Moby Dick
- Find the longest word in Moby Dick

ACT2-2

- Do Task 1

ACT2-2

- Do Task 1

Question: What is the difference between *returning* a value and *printing* a value?

```
def addOne(t) :  
    y = t + 1  
    return y
```

vs.

```
def addOne(t) :  
    y = t + 1  
    print y  
    return
```

Working with Files

“Inputs” are also called *Arguments*.

Preloaded Functions

Name	Inputs	Outputs
open	Two Strings 1. File Name 2. “r” for read (for now)	File
read (On a File)	none	String
close (On a File)	none	none
split (On a String)	(optional) delimiter	List of Strings

Working with Files

```
>>> shellList = readShel()
>>> shellList
['Sarah', 'Cynthia', 'Sylvia', 'Stout', 'Would', 'not',
'take', 'the', 'garbage', 'out!', "She'd", 'scour', 'the',
'pots', 'and', 'scrape', 'the', 'pans,', 'Candy', 'the',
'yams', 'and', 'spice', 'the', 'hams,', 'And', 'though',
'her', 'daddy', 'would', 'scream', 'and', 'shout,', 'She',
'simply', 'would', 'not', 'take', 'the', 'garbage',
'out.', 'And', 'so', 'it', 'piled', 'up', 'to', 'the',
'ceilings:', 'Coffee', 'grounds,', 'potato', 'peelings,',
'Brown',
...
'an', 'awful', 'fate,', 'That', 'I', 'cannot', 'now',
'relate', 'Because', 'the', 'hour', 'is', 'much', 'too',
'late.', 'But', 'children,', 'remember', 'Sarah', 'Stout',
'And', 'always', 'take', 'the', 'garbage', 'out!']
```

Working with Files

```
>>> shellList = readShel()
>>> shellList
['Sarah', 'Cynthia', 'Sylvia', 'Stout', 'Would', 'not',
'take', 'the', 'garbage', 'out!', "She'd", 'scour', 'the',
'pots', 'and', 'scrape', 'the', 'pans,', 'Candy', 'the',
'yams', 'and', 'spice', 'the', 'hams,', 'And', 'though',
'her', 'daddy', 'would', 'scream', 'and', 'shout,', 'She',
'simply', 'would', 'not', 'take', 'the', 'garbage',
'out.', 'And', 'so', 'it', 'piled', 'up', 'to', 'the',
'ce', 's,',
'B
..
'a
'r
'l
'A
```

Escape Characters

`\\` means interpret the NEXT character differently.

- `\n`: “new line”
- `\'`: “apostrophe”
- `\t`: “tab”

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
- 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
>>>
```

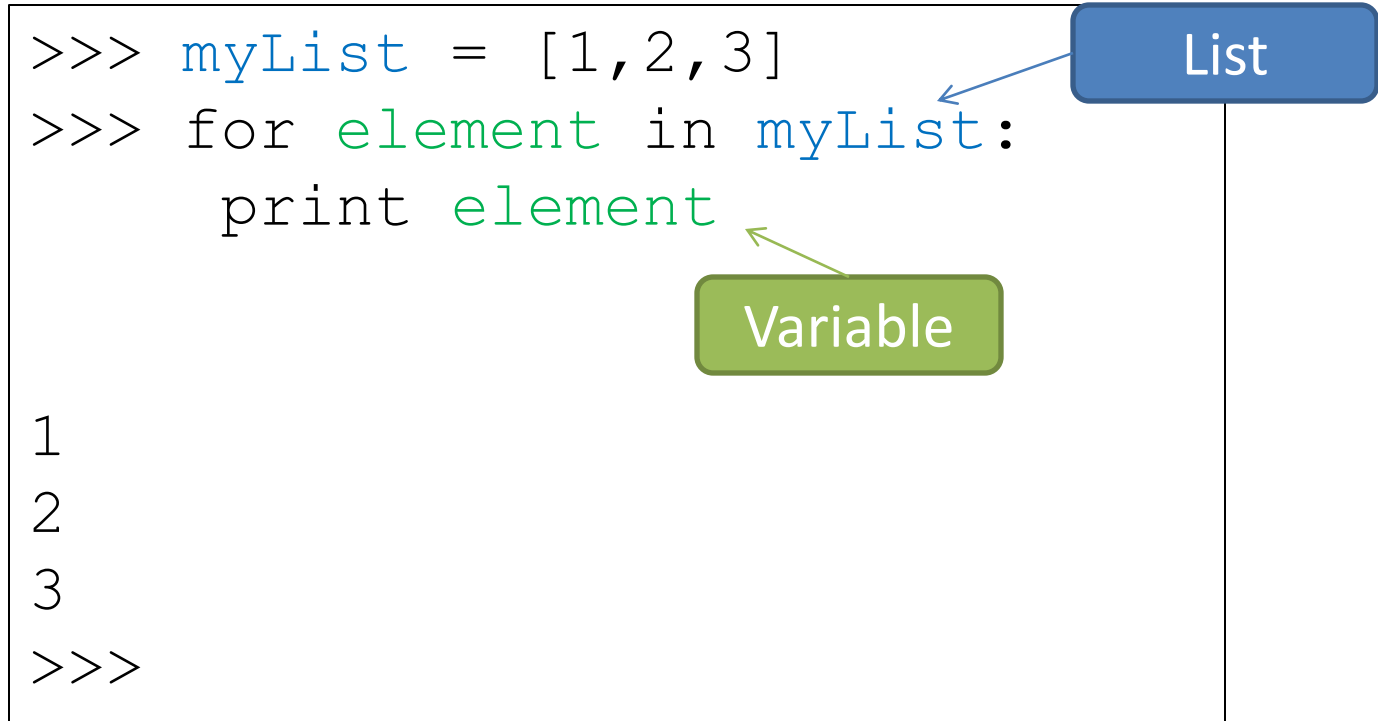


A blue rounded rectangular box containing the word "List" in white text. A blue arrow points from the box to the variable "myList" in the code snippet above.

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
>>>
```



The code block is annotated with two callout boxes. A blue box labeled "List" has an arrow pointing to the `myList` variable in the second line of code. A green box labeled "Variable" has an arrow pointing to the `element` variable in the `print` statement of the `for` loop.

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 shows a Python code block with two annotations. A blue rounded rectangle labeled "List" has a blue arrow pointing to the variable `myList` in the first line of code. A green rounded rectangle labeled "Variable" has a green arrow pointing to the variable `num` in the second line of code. The code block is enclosed in a black border.

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 the code being run in a shell, with the output of the loop. Annotations highlight key concepts: a blue box labeled "List" points to the variable `myList` in the first line; a green box labeled "Variable" points to the variable `num` in the second line; and a red box labeled "Indentation Matters!!" points to the indentation of the `print num` line. The output of the loop is shown as the numbers 1, 2, and 3, followed by the shell prompt `>>>`.

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

Activity

- Do Task 2

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

A Shortcut to List Length

Preloaded Functions

len

List

Integer

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

A Shortcut to List Length

Preloaded Functions

len

List

Integer

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

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 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

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
<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

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
<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

Do Task 3

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/float(len(myList))
    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/float(len(myList))
    return avg
```

Is our Program Accurate?

```
>>> 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']
```

Is our Program Accurate?

```
>>> shellList
['Sarah', 'Cynthia', 'Sylvia', 'Stout', 'Would', 'not',
'take', 'the', 'garbage', 'out!', "She'd", 'scour', 'the',
'pots', 'and', 'scrape', 'the', 'pans,', 'Candy', 'the',
'yams', 'and', 'spice', 'the', 'hams,', 'And', 'though',
'her', 'daddy', 'would', 'scream', 'and', 'shout,', 'She',
'simply', 'would', 'not', 'take', 'the', 'garbage',
'out.', 'And', 'so', 'it', 'piled', 'up', 'to', 'the',
'ceilings:', 'Coffee', 'grounds,', 'potato', 'peelings,',
'Brown',
...
'an', 'awful', 'fate,', 'That', 'I', 'cannot', 'now',
'relate', 'Because', 'the', 'hour', 'is', 'much', 'too',
'late.', 'But', 'children,', 'remember', 'Sarah', 'Stout',
'And', 'always', 'take', 'the', 'garbage', 'out!']
```

Is our Program Accurate?

```
>>> shellList
['Sarah', 'Cynthia', 'Sylvia', 'Stout', 'Would', 'not',
'take', 'the', 'garbage', 'out!', "She'd", 'scour', 'the',
'pots', 'and', 'scrape', 'the', 'pans,', 'Candy', 'the',
'yams', 'and', 'spice', 'the', 'hams,', 'And', 'though',
'her', 'daddy', 'would', 'scream', 'and', 'shout,', 'She',
'simply', 'would', 'not', 'take', 'the', 'garbage',
'out.', 'And', 'so', 'it', 'piled', 'up', 'to', 'the',
'ceilings:', 'Coffee', 'grounds,', 'potato', 'peelings,',
'Brown',
...
'an', 'awful', 'fate,', 'That', 'I', 'cannot', 'now',
'relate', 'Because', 'the', 'hour', 'is', 'much', 'too',
'late.', 'But', 'children,', 'remember', 'Sarah', 'Stout',
'And', 'always', 'take', 'the', 'garbage', 'out!']
```

We'll come back to this...

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

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 < 2</code>	
Less Than or Equal To	<code>1 <= 2</code>	
Greater Than	<code>1 > 2</code>	
Greater Than or Equal To	<code>1 >= 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>	<code>3</code>
Difference	<code>1 - 2</code>	<code>-1</code>

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 < 2</code>	<code>True</code>
Less Than or Equal To	<code>1 <= 2</code>	<code>True</code>
Greater Than	<code>1 > 2</code>	<code>False</code>
Greater Than or Equal To	<code>1 >= 2</code>	<code>False</code>

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 < 2</code>	<code>True</code>
Less Than or Equal To	<code>1 <= 2</code>	<code>True</code>
Greater Than	<code>1 > 2</code>	<code>False</code>
Greater Than or Equal To	<code>1 >= 2</code>	<code>False</code>

Boolean Types

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

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

Boolean Types

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

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

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

Boolean Types

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

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

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

Boolean Types

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

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

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

Boolean Types

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

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

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

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

Boolean Statements (If Stmts)

- “If something’s true, do A”

```
if 6 > 5:  
    print '6 is greater than 5'  
  
6 is greater than 5
```

Boolean Statements (If Stmts)

- “If something’s true, do A”

```
if 6 > 5:  
    print '6 is greater than 5'  
  
6 is greater than 5
```

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

```
if 5 == 6:  
    print '5 equals 6'  
else:  
    print '5 does not equal 6'  
  
5 does not equal 6
```

Boolean Statements (If Stmts)

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

```
if 5 == 6:
    print '5 equals 6'
else:
    if 5 > 6:
        print '5 is greater than 6'
    else:
        print '5 is less than 6'
```

5 is less than 6

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

Is our Program Accurate?

```
>>> shellList
['Sarah', 'Cynthia', 'Sylvia', 'Stout', 'Would', 'not',
'take', 'the', 'garbage', 'out!', "She'd", 'scour', 'the',
'pots', 'and', 'scrape', 'the', 'pans,', 'Candy', 'the',
'yams', 'and', 'spice', 'the', 'hams,', 'And', 'though',
'her', 'daddy', 'would', 'scream', 'and', 'shout,', 'She',
'simply', 'would', 'not', 'take', 'the', 'garbage',
'out.', 'And', 'so', 'it', 'piled', 'up', 'to', 'the',
'ceilings:', 'Coffee', 'grounds,', 'potato', 'peelings,',
'Brown',
...
'an', 'awful', 'fate,', 'That', 'I', 'cannot', 'now',
'relate', 'Because', 'the', 'hour', 'is', 'much', 'too',
'late.', 'But', 'children,', 'remember', 'Sarah', 'Stout',
'And', 'always', 'take', 'the', 'garbage', 'out!']
```

We'll come back to this...next class