More List Comprehensions
An Example
Write a program that reads a literary text (e.g., “War and Peace” or “The Illiad”) and does simple text analysis to figure out the principal characters of the novel.

For example, when I ran my program on “The Illiad” the most frequent characters were:

(563, 'Trojans'), (548, 'Achaeans'), (447, 'Jove'), (421, 'Hector'), (383, 'Achilles'), (183, 'Agamemnon'), (178, 'Priam'), (160, 'Patroclus'), (146, 'Minerva'), (137, 'Ajax')
Main Idea

• Since character names are proper nouns, starting with upper case letters, the idea is to look for words starting with upper case letters that do not appear at the beginning of sentences.

• So the program also attempts to partition the text into sentences, assuming that ".", "!", and "?" are all the possible sentence delimiters.

• Then we count the frequency of the proper nouns and report the most frequent of these. We only keep names that are at least 4 letters long.
Function parseSentences

# Takes a string as parameter and "splits" it into "sentences."
# We assume that ".", "!", and "?" are sentence delimiters

def parseSentences(bigString):
    return bigString.replace("!", ".").replace("?", ".").split(".")

• This returns a list – each element in the list is a string representing a sentence.
Next task: split sentences in word sequences

- We have solved this problem earlier and written a function called “parse” for it.

- That algorithm examined the string character-by-character and pulled out contiguous sequences of letters.

- Now we will use a different algorithm to solve this problem.
Algorithmic Idea

1. Replace every non-letter in each sentence by space.

2. Then split on spaces.

- **Question:** How do we specify all non-letter characters?
Two useful built-in functions

- `ord(ch)`
  If `ch` is a single character string, this function returns the ASCII code for `ch`

- `chr(i)`
  Returns a string of one character whose ASCII code is the integer `i`

What is ASCII?
It stands for the *American Standard Code for Information Interchange*. It assigns a number in the range 0..255 to every character that can be entered at the keyboard.
More on ASCII

- The numbers 0..31 are reserved for unprintable characters, e.g., the tab character ("\t"), the end of line character ("\n"), etc.
- 32 is the ASCII value of the space character (" ")
- 33..47 is used for some punctuation characters
- 48..57 is used for digits “0” through “9”
- 65..90 is used for upper case letters
- 97..122 is used for lower case letters
<table>
<thead>
<tr>
<th>Dec</th>
<th>Hx Oct</th>
<th>Char</th>
<th>Dec</th>
<th>Hx Oct</th>
<th>Html</th>
<th>Char</th>
<th>Dec</th>
<th>Hx Oct</th>
<th>Html</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>NUL</td>
<td>32</td>
<td>20</td>
<td>040</td>
<td>$32</td>
<td>64</td>
<td>40</td>
<td>100</td>
<td>$64</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>SOH</td>
<td>33</td>
<td>21</td>
<td>041</td>
<td>$33</td>
<td>65</td>
<td>41</td>
<td>101</td>
<td>$65</td>
</tr>
<tr>
<td>2</td>
<td>002</td>
<td>STX</td>
<td>34</td>
<td>22</td>
<td>042</td>
<td>$34</td>
<td>66</td>
<td>42</td>
<td>102</td>
<td>$66</td>
</tr>
<tr>
<td>3</td>
<td>003</td>
<td>ETX</td>
<td>35</td>
<td>23</td>
<td>043</td>
<td>$35</td>
<td>67</td>
<td>43</td>
<td>103</td>
<td>$67</td>
</tr>
<tr>
<td>4</td>
<td>004</td>
<td>EOT</td>
<td>36</td>
<td>24</td>
<td>044</td>
<td>$36</td>
<td>68</td>
<td>44</td>
<td>104</td>
<td>$68</td>
</tr>
<tr>
<td>5</td>
<td>005</td>
<td>ENQ</td>
<td>37</td>
<td>25</td>
<td>045</td>
<td>$37</td>
<td>69</td>
<td>45</td>
<td>105</td>
<td>$69</td>
</tr>
<tr>
<td>6</td>
<td>006</td>
<td>ACK</td>
<td>38</td>
<td>26</td>
<td>046</td>
<td>$38</td>
<td>70</td>
<td>46</td>
<td>106</td>
<td>$70</td>
</tr>
<tr>
<td>7</td>
<td>007</td>
<td>BEL</td>
<td>39</td>
<td>27</td>
<td>047</td>
<td>$39</td>
<td>71</td>
<td>47</td>
<td>107</td>
<td>$71</td>
</tr>
<tr>
<td>8</td>
<td>010</td>
<td>BS</td>
<td>40</td>
<td>28</td>
<td>050</td>
<td>$40</td>
<td>72</td>
<td>48</td>
<td>108</td>
<td>$72</td>
</tr>
<tr>
<td>9</td>
<td>011</td>
<td>TAB</td>
<td>41</td>
<td>29</td>
<td>051</td>
<td>$41</td>
<td>73</td>
<td>49</td>
<td>109</td>
<td>$73</td>
</tr>
<tr>
<td>10</td>
<td>012</td>
<td>LF</td>
<td>42</td>
<td>30</td>
<td>052</td>
<td>$42</td>
<td>74</td>
<td>50</td>
<td>110</td>
<td>$74</td>
</tr>
<tr>
<td>11</td>
<td>013</td>
<td>VT</td>
<td>43</td>
<td>31</td>
<td>053</td>
<td>$43</td>
<td>75</td>
<td>51</td>
<td>111</td>
<td>$75</td>
</tr>
<tr>
<td>12</td>
<td>014</td>
<td>FF</td>
<td>44</td>
<td>32</td>
<td>054</td>
<td>$44</td>
<td>76</td>
<td>52</td>
<td>112</td>
<td>$76</td>
</tr>
<tr>
<td>13</td>
<td>015</td>
<td>CR</td>
<td>45</td>
<td>33</td>
<td>055</td>
<td>$45</td>
<td>77</td>
<td>53</td>
<td>113</td>
<td>$77</td>
</tr>
<tr>
<td>14</td>
<td>016</td>
<td>SO</td>
<td>46</td>
<td>34</td>
<td>056</td>
<td>$46</td>
<td>78</td>
<td>54</td>
<td>114</td>
<td>$78</td>
</tr>
<tr>
<td>15</td>
<td>017</td>
<td>SI</td>
<td>47</td>
<td>35</td>
<td>057</td>
<td>$47</td>
<td>79</td>
<td>55</td>
<td>115</td>
<td>$79</td>
</tr>
<tr>
<td>16</td>
<td>020</td>
<td>DLE</td>
<td>48</td>
<td>36</td>
<td>060</td>
<td>$48</td>
<td>80</td>
<td>56</td>
<td>116</td>
<td>$80</td>
</tr>
<tr>
<td>17</td>
<td>021</td>
<td>DC1</td>
<td>49</td>
<td>37</td>
<td>061</td>
<td>$49</td>
<td>81</td>
<td>57</td>
<td>117</td>
<td>$81</td>
</tr>
<tr>
<td>18</td>
<td>022</td>
<td>DC2</td>
<td>50</td>
<td>38</td>
<td>062</td>
<td>$50</td>
<td>82</td>
<td>58</td>
<td>118</td>
<td>$82</td>
</tr>
<tr>
<td>19</td>
<td>023</td>
<td>DC3</td>
<td>51</td>
<td>39</td>
<td>063</td>
<td>$51</td>
<td>83</td>
<td>59</td>
<td>119</td>
<td>$83</td>
</tr>
<tr>
<td>20</td>
<td>024</td>
<td>DC4</td>
<td>52</td>
<td>40</td>
<td>064</td>
<td>$52</td>
<td>84</td>
<td>60</td>
<td>120</td>
<td>$84</td>
</tr>
<tr>
<td>21</td>
<td>025</td>
<td>NAK</td>
<td>53</td>
<td>41</td>
<td>065</td>
<td>$53</td>
<td>85</td>
<td>61</td>
<td>121</td>
<td>$85</td>
</tr>
<tr>
<td>22</td>
<td>026</td>
<td>SYN</td>
<td>54</td>
<td>42</td>
<td>066</td>
<td>$54</td>
<td>86</td>
<td>62</td>
<td>122</td>
<td>$86</td>
</tr>
<tr>
<td>23</td>
<td>027</td>
<td>ETB</td>
<td>55</td>
<td>43</td>
<td>067</td>
<td>$55</td>
<td>87</td>
<td>63</td>
<td>123</td>
<td>$87</td>
</tr>
<tr>
<td>24</td>
<td>030</td>
<td>CAN</td>
<td>56</td>
<td>44</td>
<td>068</td>
<td>$56</td>
<td>88</td>
<td>64</td>
<td>124</td>
<td>$88</td>
</tr>
<tr>
<td>25</td>
<td>031</td>
<td>EM</td>
<td>57</td>
<td>45</td>
<td>069</td>
<td>$57</td>
<td>89</td>
<td>65</td>
<td>125</td>
<td>$89</td>
</tr>
<tr>
<td>26</td>
<td>032</td>
<td>SUB</td>
<td>58</td>
<td>46</td>
<td>070</td>
<td>$58</td>
<td>90</td>
<td>66</td>
<td>126</td>
<td>$90</td>
</tr>
<tr>
<td>27</td>
<td>033</td>
<td>ESC</td>
<td>59</td>
<td>47</td>
<td>071</td>
<td>$59</td>
<td>91</td>
<td>67</td>
<td>127</td>
<td>$91</td>
</tr>
<tr>
<td>28</td>
<td>034</td>
<td>FS</td>
<td>60</td>
<td>48</td>
<td>072</td>
<td>$60</td>
<td>92</td>
<td>68</td>
<td>128</td>
<td>$92</td>
</tr>
<tr>
<td>29</td>
<td>035</td>
<td>GS</td>
<td>61</td>
<td>49</td>
<td>073</td>
<td>$61</td>
<td>93</td>
<td>69</td>
<td>129</td>
<td>$93</td>
</tr>
<tr>
<td>30</td>
<td>036</td>
<td>RS</td>
<td>62</td>
<td>50</td>
<td>074</td>
<td>$62</td>
<td>94</td>
<td>70</td>
<td>130</td>
<td>$94</td>
</tr>
<tr>
<td>31</td>
<td>037</td>
<td>US</td>
<td>63</td>
<td>51</td>
<td>077</td>
<td>$63</td>
<td>95</td>
<td>71</td>
<td>131</td>
<td>$95</td>
</tr>
</tbody>
</table>
Some examples of `chr` and `ord` in action

```python
>>> ord("a")
97
>>> chr(97)
'a'
>>> ord(" ")
32
>>> ord("o")
48
>>> chr(48)
'o'
>>> chr(49)
'1'
>>> ord("A")
65
>>> ord("B")
66
```
Function replaceNonLetters

# Replaces all non-letters in a given string s by space
def replaceNonLetters(s):
    # Make a list of all non-letters.
    nonLetters = [chr(x) for x in range(0, 128) if not chr(x).isalpha()]

    # Replaces each nonletter character in s by space
    for char in nonLetters:
        s = s.replace(char, " ")

    return s
Function parseWords

# Takes a list of sentences and parses each sentence in this list into a list of words.
# So the result is a list of lists, e.g., [['This', 'is', 'ok'], ['This', 'is', 'not']].
# We use the same definition of a word as before. It is a contiguous sequence of
# letters.

def parseWords(sentenceList):

    # Once non-letters have been replaced by spaces then a simple split() using
    # blank as the delimiter will help us get all the words. Note that this
    # constructs a nested list of words for each sentence.

    return [replaceNonLetters(x).split() for x in sentenceList]
# main program
f = open("illiad.txt", "r")
bigString = f.read()
sentenceList = parseSentences(bigString)
nestedWordList = parseWords(sentenceList)

# This block of code walks through the list of words, ignores
# the first word in each sentence and of the remaining words, picks
# ones that start with an upper case and have length at least 4.

nestedWordList = [x[1:] for x in nestedWordList]
wordList = [y for x in nestedWordList for y in x]
characterNames = [x for x in wordList if x[0].isupper() and len(x) > 3]

[masterList, frequencies] = computeFrequencies(characterNames)