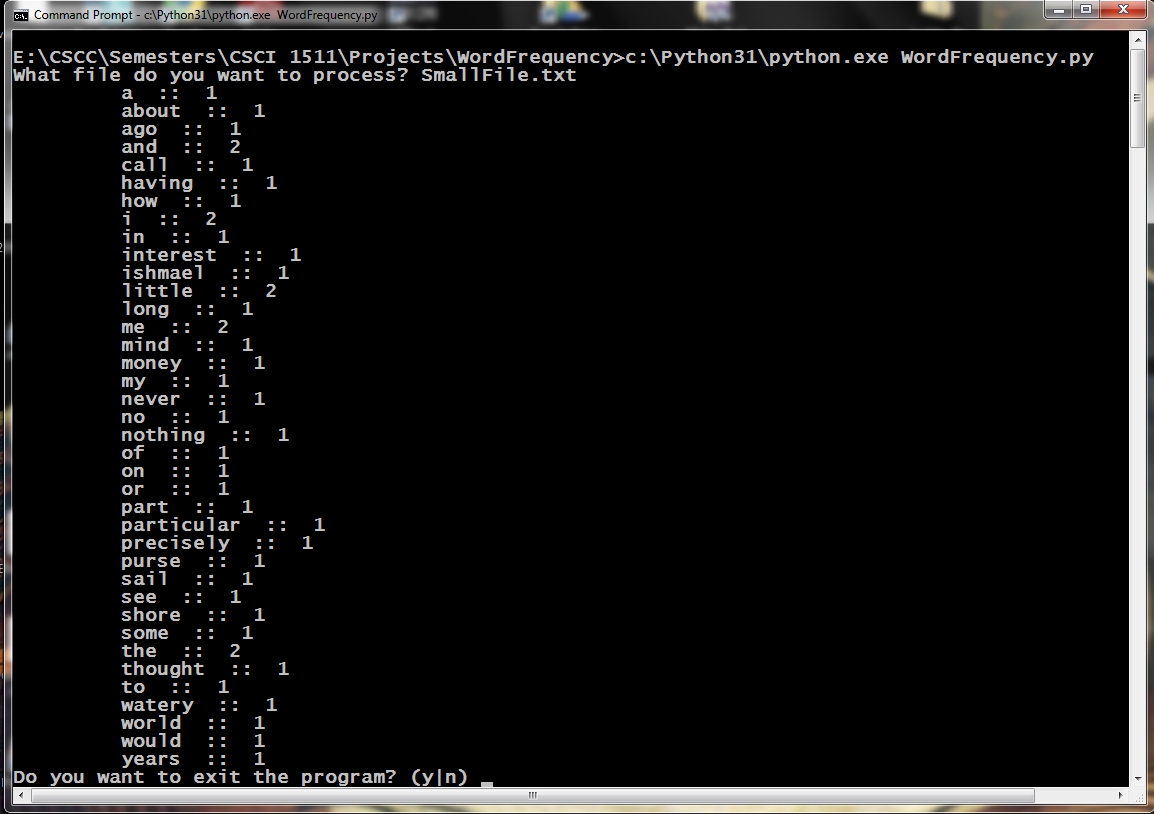
Pick one from the list below.

Each file should be a python program, root file name of “*Lab10\_pcarswel-#*”, extended by ".py". Insert your username for *pcarswel* and the number of the question you chose for ‘*#*’. The header of each file should be comments which tell the name of the program, your name [the author], relevant comments, and the date.

1. Word Count: Count the frequency of each word in a text file. Let the user choose a filename to read.
   1. The program will count the frequency with which each word appears in the text.
   2. Words which are the spelled the same but differ by case will be combined.
   3. Punctuation should be removed
   4. If the file does not exist, use a ‘try-execption’ block to handle the error
   5. Output will list the words alphabetically, with the word in the first column and the count in the second column … i.e. as : 1287



Word Frequency Example 1

The IPO chart is listed as:

|  |  |  |
| --- | --- | --- |
| IPO Chart for the **main()** function | | |
| Input | Processing | Output |
| NONE | * Gets the filename from the user and creates the file pointer * catches the exception if a file does not exist | NONE |
| IPO Chart for the **wordFreq()** function | | |
| File pointer | * creates the data type to store the word and the frequency * eliminate punctuation * combine the same words that differ by uppercase characters | Word frequency data?? |
| IPO Chart for the **printOut()** function | | |
| Word count data | * Prints out the word and it's frequency | NONE |

IPO for WordFrequency.py

Example UPC code

Pseudocode for Word Count:

Psuedo Code

printWds(data)

## print each word in our data and the respective count

## but we need them sorted, dictionaries have a nice module, ***sorted***

## and lists have ***sort***.

for x in sorted(data.keys()):

print() ## look at the formatting above

return

wordFreq(fptr)

## create empty data type, dictionary or list?

freq = {} or freq = []

## read in the first line

line = fptr.readline()

punctChars = (….create a tuple of punctuation characters to eliminate… )

while line:

for c in punctChars:

## use the string replace module to substitute an empty char

line = line.replace(c, “”)

## create a list of separate words that make up the string

words = line.split()

for each word in our split:

#create a temporary word converted to lower using the lower()

tmp = word.lower()

## add this to your database, this is where dictionaries have the

## advantage over dictionaries … the get() method

freq[tmp] = freq.get(tmp, 0) + 1

read another line

return freq