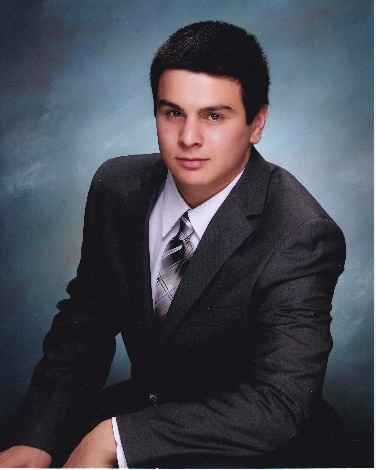
Peter Parianos



Grail Login ID: peparian

**Description of the code:**

The python IDE used in this project was IDLE, that is prepackaged with version 3.6. The program is structure with the entry point of the program at **main**(). This is initially called at the global level. **Main**() function includes: the description of the project; parsing the filename from the argumentlist *argv*, and processing the data to meet the condition set forth by the assignment.

**Find\_file**(*file\_name*) takes the string literally, and opens it as file. The information from the file is read row by row within a for loop, and then added to a list called *data*. If there is a new line character, it is striped from the word that contains it. This function returns the list *data*.

**Create\_dict**(*data*) takes the list and then creates a **Counter**() dictionary. This from the *collections* library. A **Counter**() takes an iterable as a parameter, counts the number of occurences of items in that iterable, sets the items as keys, and sets the value of the keys to the number of occurrences. It returns a ***dict***() Very helpful function. This function then returns *datadict*

The returned data structures *datadict* and *data* give us all we need to print the required information. Number of unique words is the length of *datadict*, total number of words is the length of *data*. This makes use of the built-in **len()** function.

The *sortedList* contains the returned value from **sorted**(). Parameters are the iterable to be sorted, key to sort by, and reverse. In order to sort by the length of each word in data, I needed to create a lambda expression (a one line function) in order to pass it into *key*. It is a simplified function to take each *word* in *data* and return the length of that word. This length will be used to sort the list by smallest length first, which is why I needed to reverse it.

To find the words that have more than 5 characters, I needed to view the keys of *datadict,* where each one is referred to as *key.* This was done using the dict built-in function, ***keys***(), which returns the keys as a list of strings. I iterated through that list and created yet another list called *temp\_list,* which includes only keys that were greater than four. This temporarily list was sorted by the value of the keys, in reverse order (descending). I preceded to print this list afterwards. The logic afterwards is pretty straightforward.

**Experiences In Debugging and Testing:**

All debugging was done using Windows PowerShell, as a simple command line interface. I did not use python’s built in debugger, only the errors that were returned from the running program.

Some notable errors:

I purposefully added a new line in the text file, and this created problems because the \n was attached to a word, which would have counter as letters in that word. Fixed with loop

I first tried to iterate through *data* manually, but this created a problem which includes **keyError** where you cannot have more than one key. After sifting through the python documentation, I found that there is a collections function called Counter(), which fits my needs perfectly.

To find words with more than 5 characters, I almost unnecessarily complicated the program by iterating through the list instead of the dictionary. After going through the documentation, I found a way to use the keys in the dictionaries as an iterable item. This saved a tremendous amount of time.