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.

The argument is saved in *file\_name,* and error handling is done to print a message if there is no argument because *argv* will be out-of-bounds. The error caught is an *IndexError,* because the array will not be size 2 if no arguments were entered.

**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 uses a built-in function called ***enumerate***() to return the index and the data. This is useful for rewriting the *data* list since we can use the indexing operator to reference the current word in the list, then replace it with *word.strip(‘\n’)* because we want to remove the new line character. This function returns the list *data*.

**Create\_dict**(*data*) takes the list and then creates a dictionary. A loop is created in order to iterate through the list, and add the *word* as the keys of the dictionary. If the word is not in *datadict*, then a new key is added. This function returns the dictionary *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.

We use a variable to store the longest word. We used the built-in function called len() which returns the integer value of the length of the word. We use a loop to iterate through all the words, and replace *maxword* with the larger length word. Then we print.

To find the words that have more than 4 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. The print function is very similar to printf in Java or C.

**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 wer used as help.

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.

To find words with more than 4 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.