ASSIGNMENT 2

Frequency Counter

OCTOBER 1, 2020

Sukhman Atwal

Contents

Introduction

This assignment is split into two parts/tasks.

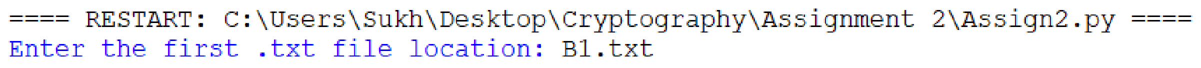
Task 1 consists of creating an application that will count the frequency of all letters in a text file. The user will specify a text file and it will create a file in CSV format with all the count data in the same directory as the script location.

Task 2 consists of using a substitution cipher such as Caesar with an objective to compute the probabilities distributions for the text file which we created above using P(M);P(K); and P(C). Afterwards, we'll be computing the conditional probability for the following letters: e, t, a, i, o, n.

The program was created with python3, importing the "panda" module to convert the data to csv format.

How to Use

You would run the script using the command line, which should then ask for a user input for the text file location. The script is located in the 'Script' folder.



```
==== RESTART: C:\Users\Sukh\Desktop\Cryptography\Assignment 2\Assign2.py ====
Enter the first .txt file location: B1.txt
```

Afterwards, all relevant information will be displayed and saved to a csv file located in the same directory as the script.

Diagram

a	b	c	d	e	f	g	h	i	j
c	d	e	f	g	h	i	J	k	l

Above we can see a diagram of how a Caesar cipher would work with an offset of 2. An offset of 2 means that you would skip 2 letters in order to encrypt it. For example, if you have the word ‘apple’, we would get ‘crrng’ if we applied the Caesar cipher with an offset of 2. With an offset of 3, we would get ‘dssoh’ and so on.

Conditional Probabilities

Here I’ll be using Alice in Wonderland, B1.txt, for the purposes of finding the conditional probabilities.

M represents the set of possible plaintexts.

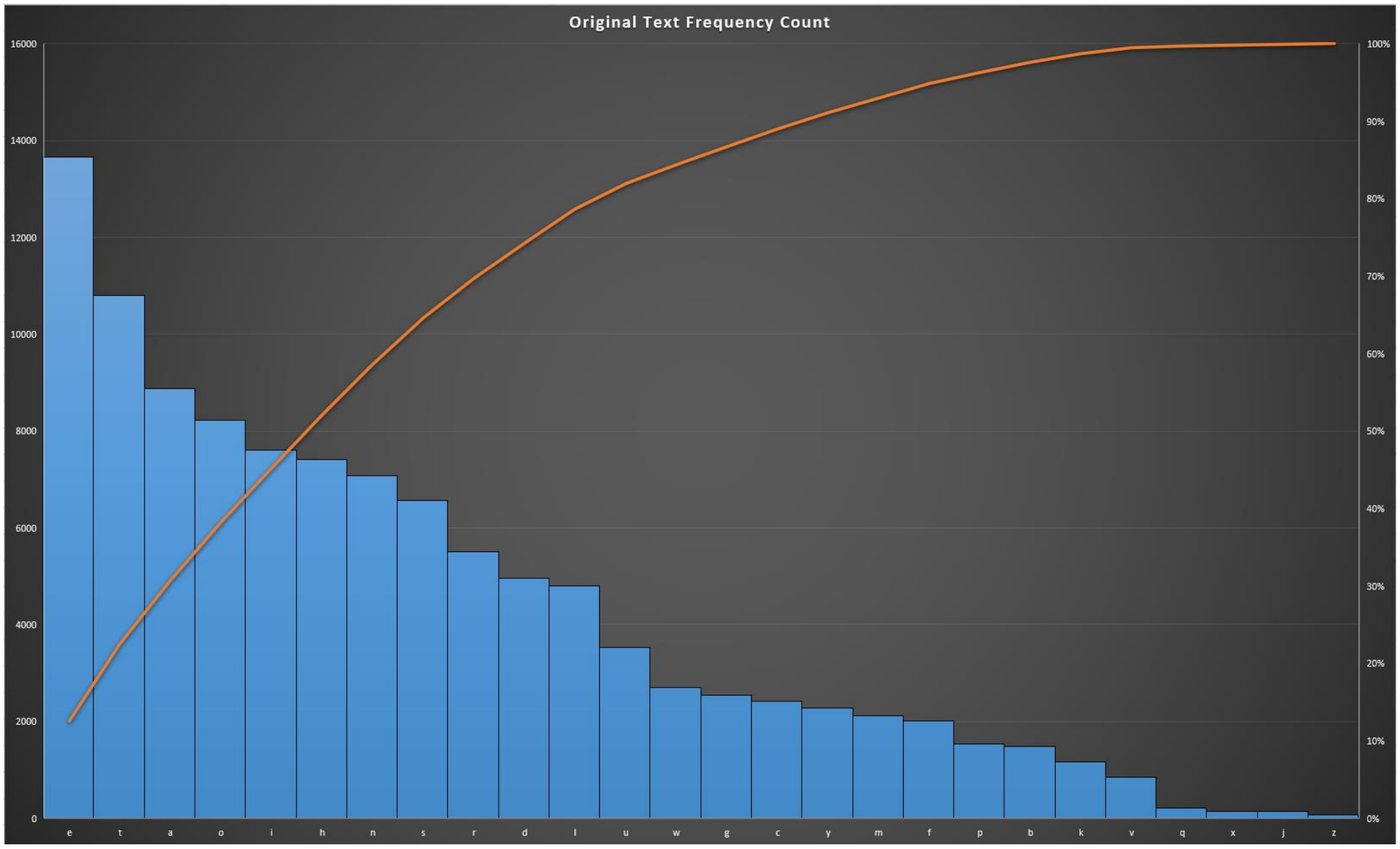
K represents the set of possible key values.

C represents the set of possible ciphertexts.

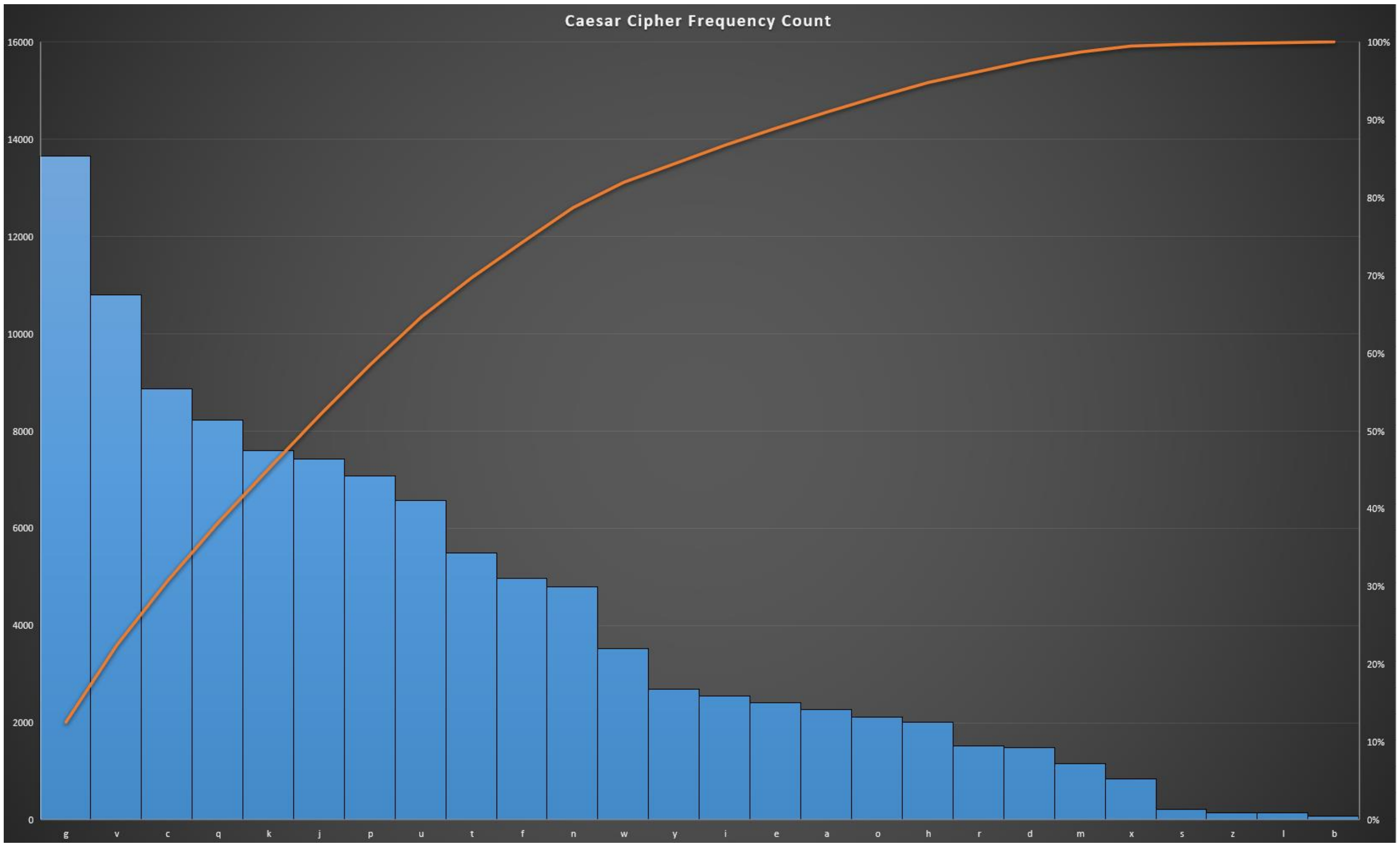
Conditional Probability			

In this scenario, I used an offset of 2. Additionally, in Caesar ciphers, the ‘key’ is limited by the number of characters in the alphabet which would be 26. Therefore, we can conclude the probability to be 1/26 or 0.0385. -----

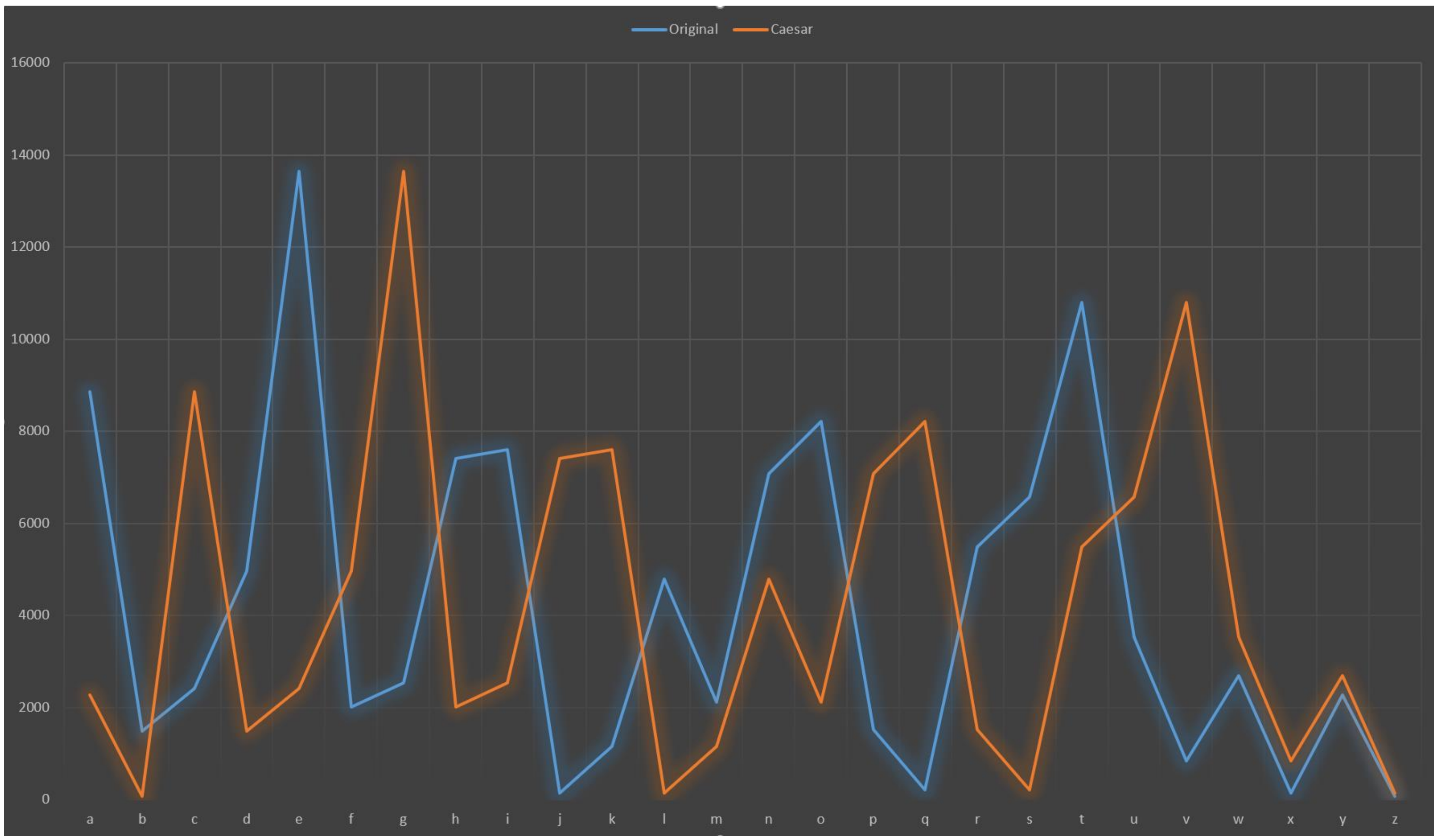
Alice in Wonderland



Above is a graph of the frequency counter for the original text of Alice in Wonderland. We can see that many of the vowels have the most occurrence such as e, a, o, and i.

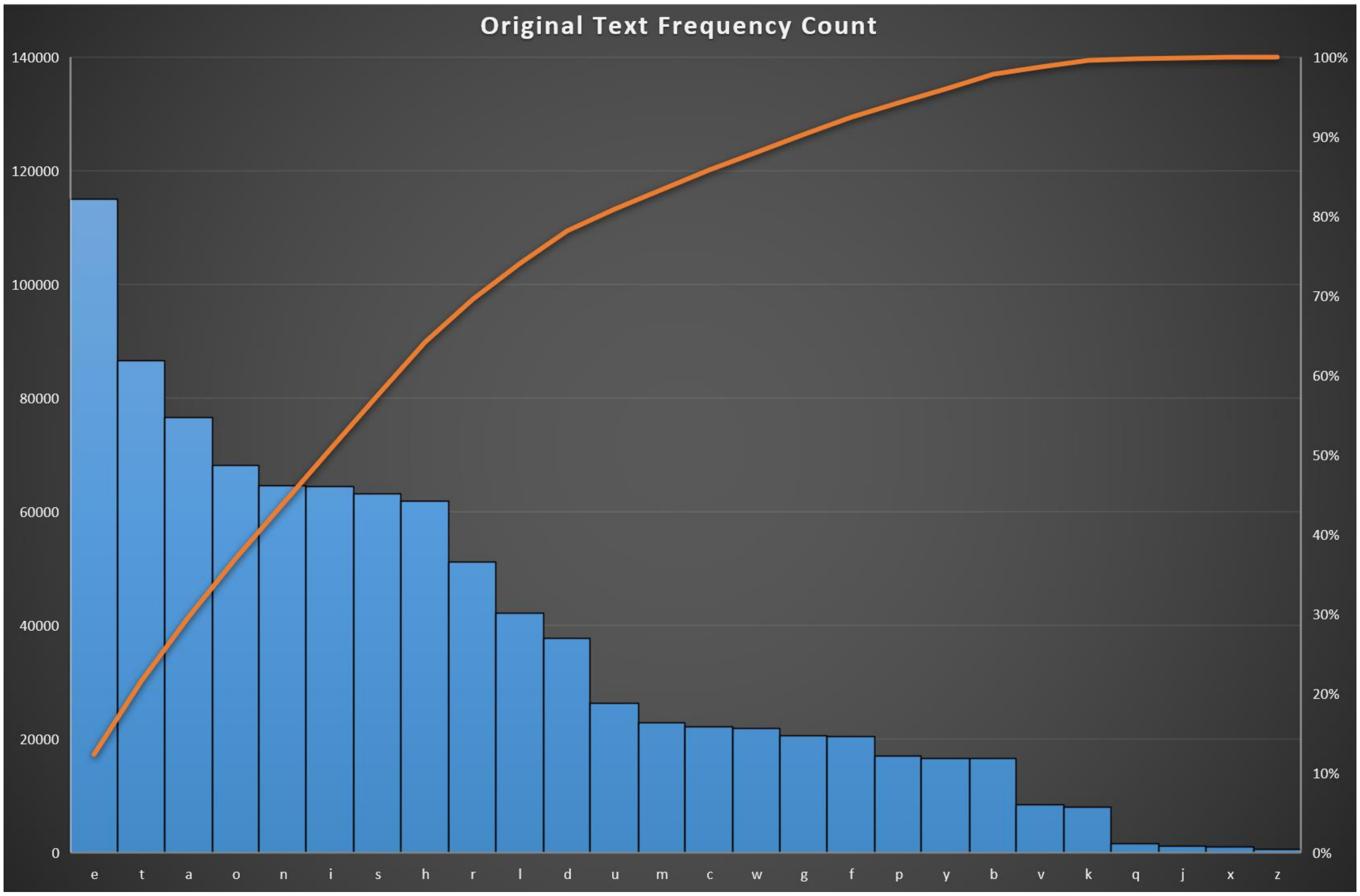


Next, we have a graph of the frequency counter for the text after running it through a Caesar cipher using an offset of 2. If we look at the most frequent letter in this graph “g” we can see that it’s a letter 2 offsets after “e” which we can see in the original text graph.

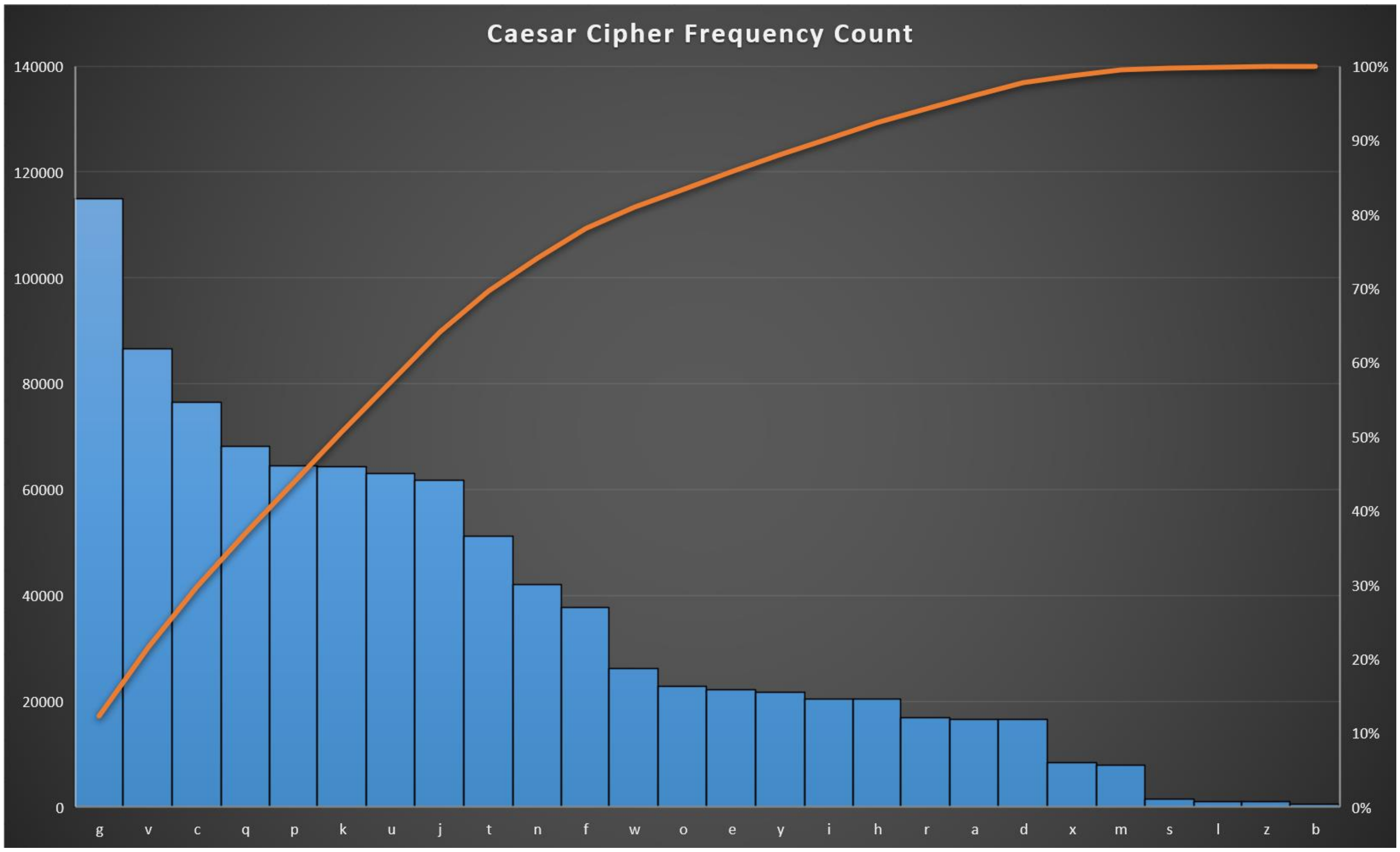


Here is a line graph, with the blue line representing the original text and the orange line representing the Caesar encrypted text. We can see the orange line is doing what the blue line has just done in terms of direction and shape.

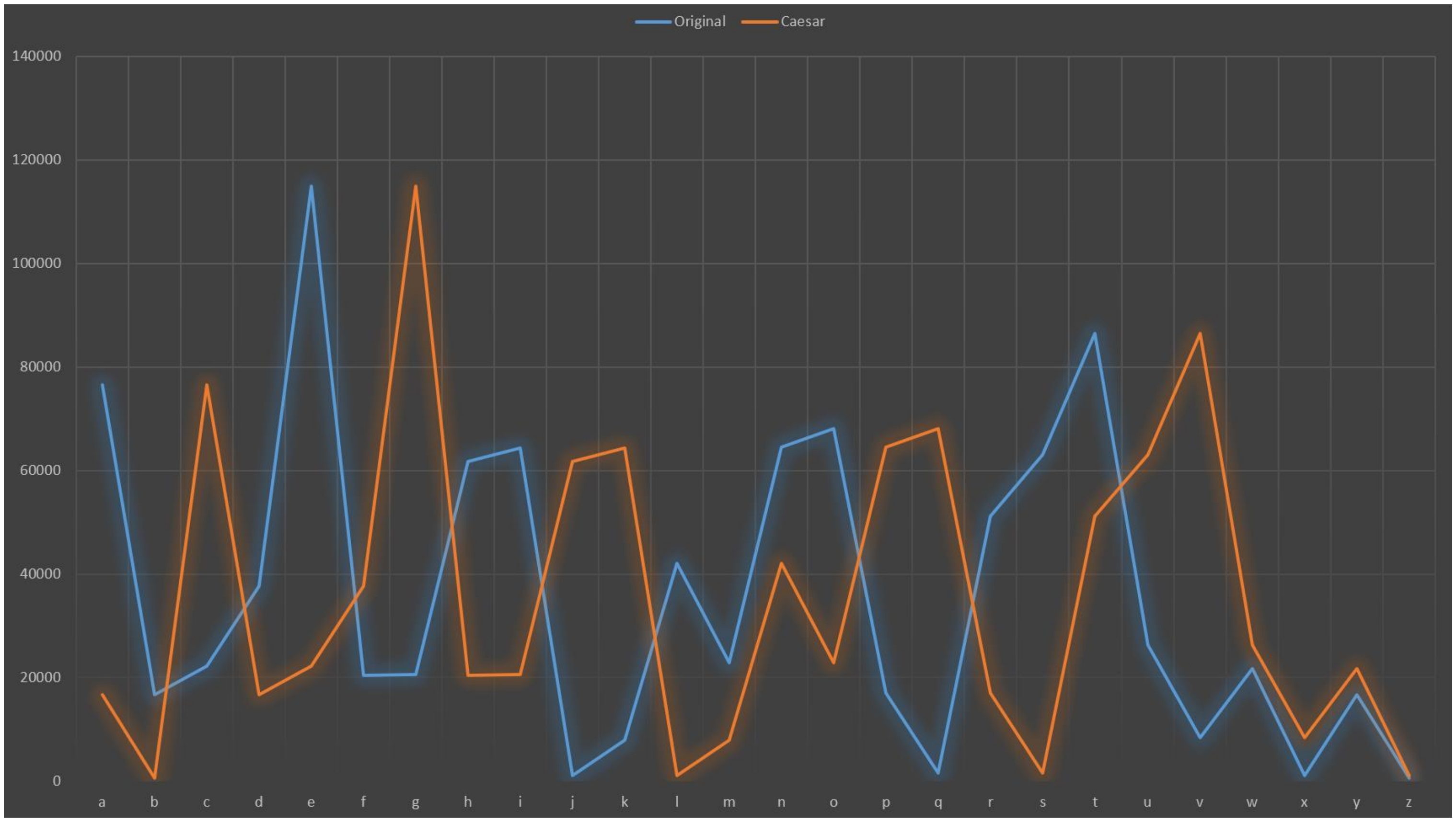
Moby Dick



Above is the frequency counter of each letter for Moby Dick. Similar to Alice in Wonder, we can see the most frequent characters are vowels like e, a, o and i. For both graphs of the original text, the most frequent letter by a pretty large margin is ‘e’.



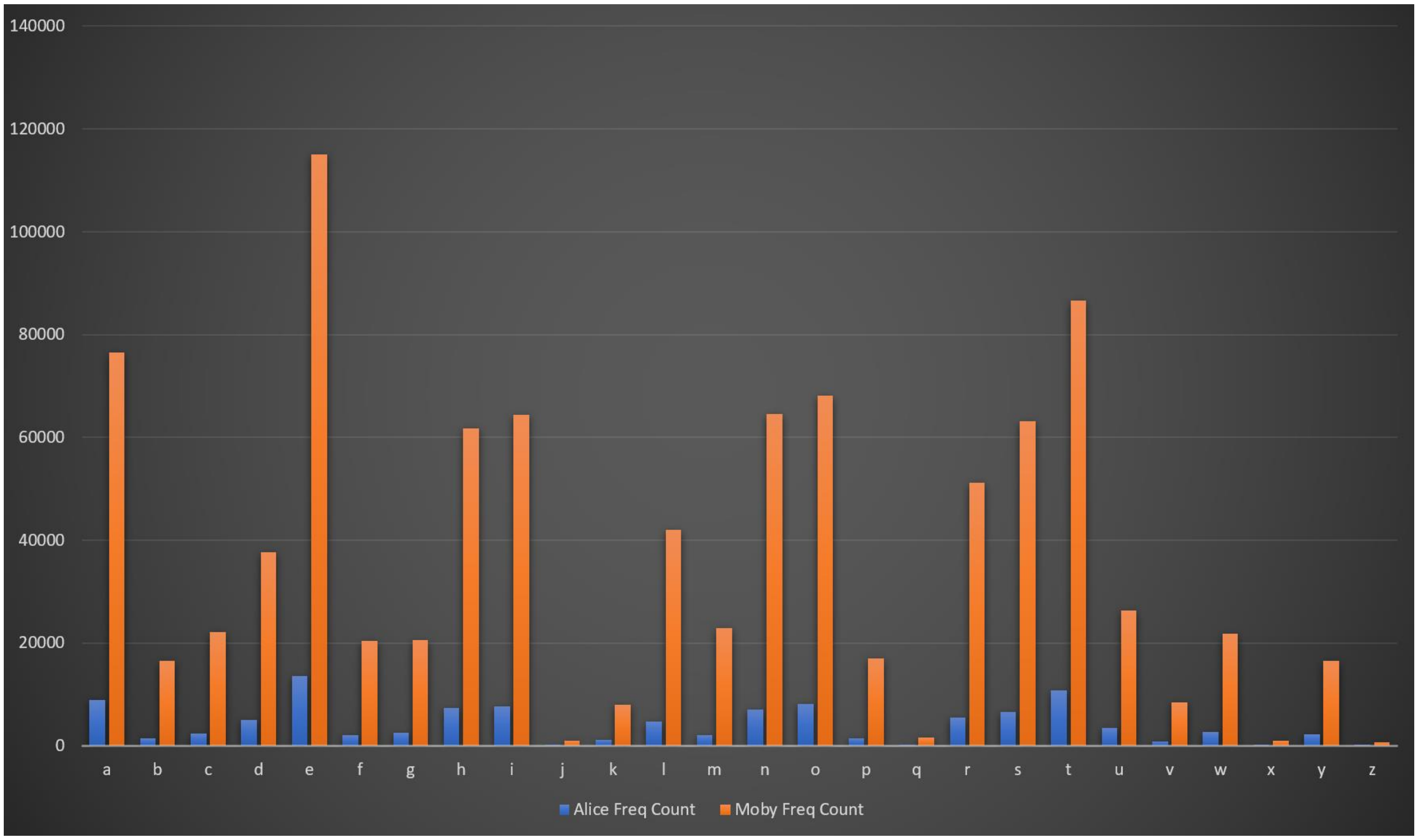
We again ran it through a Caesar cipher encryption with an offset of 2. We can use the same logic as Alice in Wonderland and see that the most frequent character ‘g’ is 2 offsets from the most frequent character in the original text ‘e’.



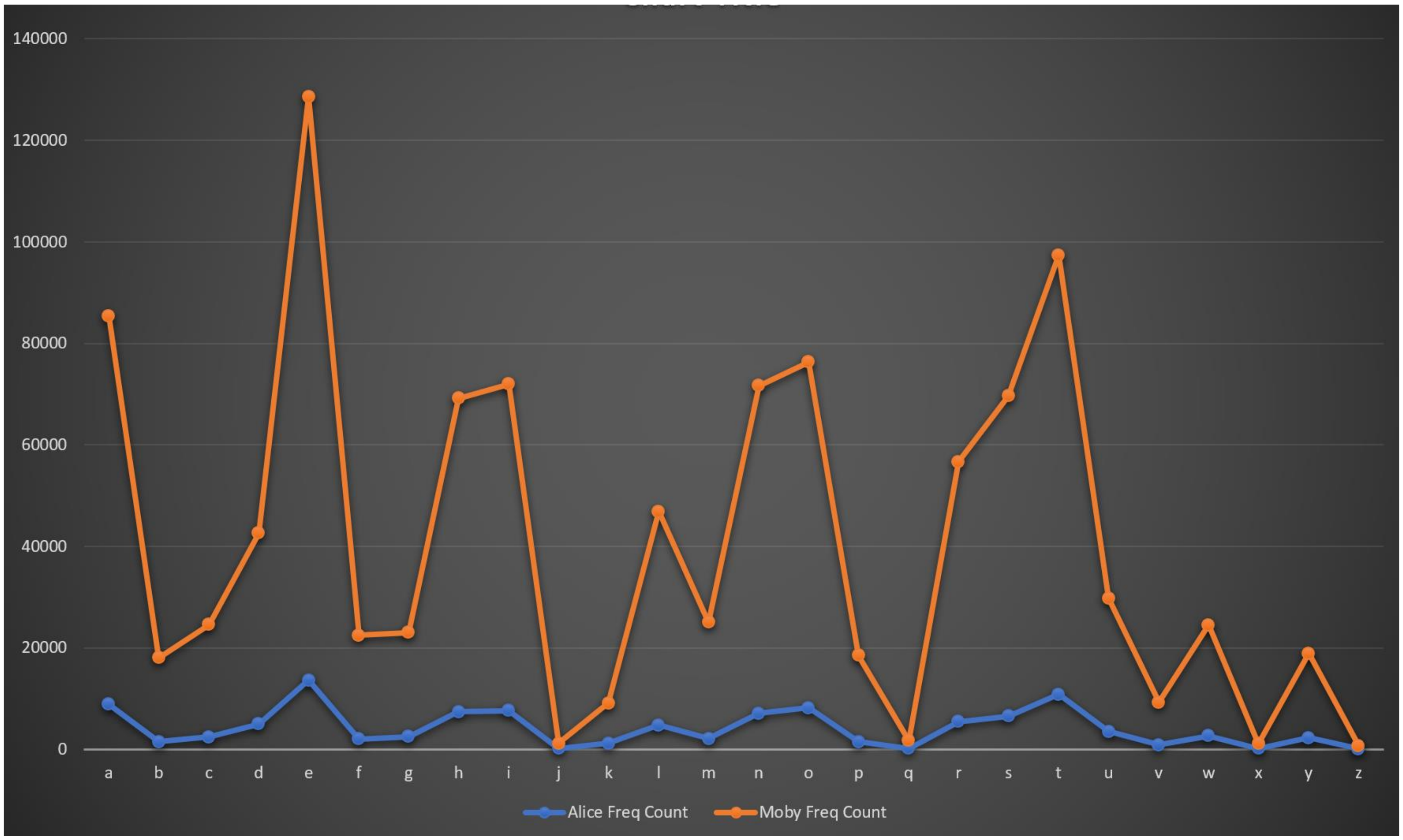
And finally, here's the line graph comparing the original and Caesar cipher text files, seeing the similarity in movements between the orange and blue lines.

Comparing Alice in Wonder and Moby Dick

Here I'll be comparing the original text between the two books to look for any similarities.



Due to Moby Dick being a longer book, we can see it's larger in terms of sample size. To get a better idea, I'll covert it to a line graph.



Here we can see that the spikes are quite similar between the two. We can see big and small spikes in ‘e’, ‘n’, ‘o’, ‘s’, ‘t’, ‘w’ and ‘y’ in both books.

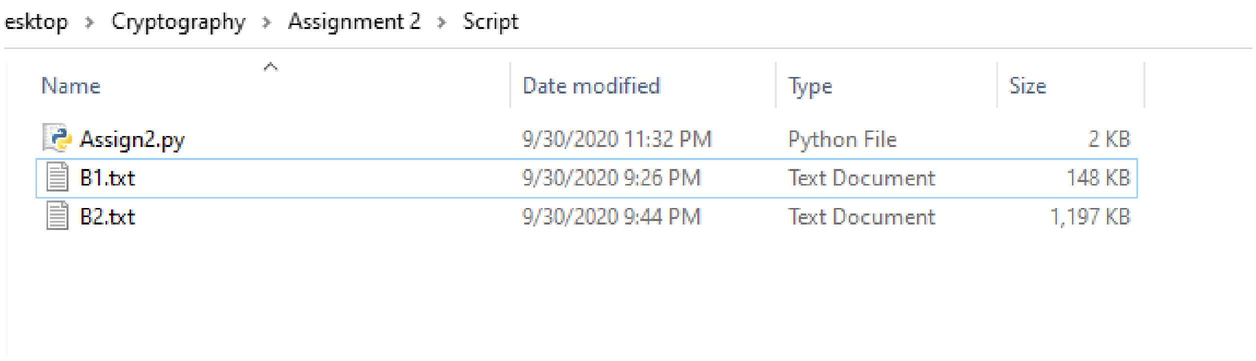
Testing

Here we are checking to ensure we are getting the correct outputs from the user inputs and the script is working as intended.

Test #	Tools Used	Expectations	Actuality	Pass/Fail
1	IDLE	Only currently existing txt files should go through, any other file types will terminate the script.	Only existing txt files created went through	Pass
2	IDLE, Excel	Converts txt file to csv format and the csv file is correctly formatted to easily display graphs	Correctly converted to csv in an easy to read format	Pass
3				
4				

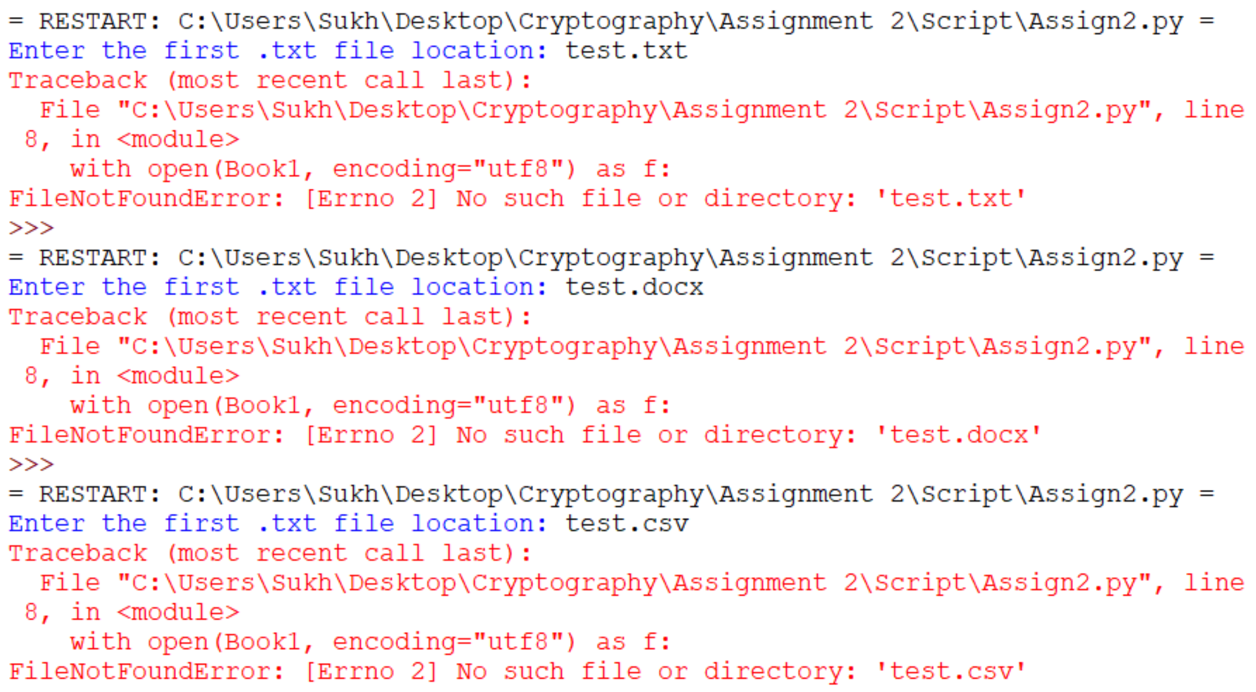
Test 1

For reference, the only files I currently have in the folder is the script, B1.txt and B2.txt.



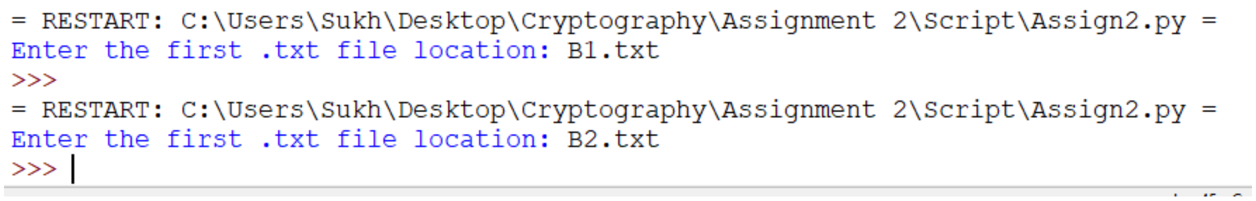
esktop › Cryptography › Assignment 2 › Script

Name	Date modified	Type	Size
Assign2.py	9/30/2020 11:32 PM	Python File	2 KB
B1.txt	9/30/2020 9:26 PM	Text Document	148 KB
B2.txt	9/30/2020 9:44 PM	Text Document	1,197 KB



```
= RESTART: C:\Users\Sukh\Desktop\Cryptography\Assignment 2\Script\Assign2.py =
Enter the first .txt file location: test.txt
Traceback (most recent call last):
  File "C:\Users\Sukh\Desktop\Cryptography\Assignment 2\Script\Assign2.py", line
 8, in <module>
    with open(Book1, encoding="utf8") as f:
FileNotFoundError: [Errno 2] No such file or directory: 'test.txt'
>>>
= RESTART: C:\Users\Sukh\Desktop\Cryptography\Assignment 2\Script\Assign2.py =
Enter the first .txt file location: test.docx
Traceback (most recent call last):
  File "C:\Users\Sukh\Desktop\Cryptography\Assignment 2\Script\Assign2.py", line
 8, in <module>
    with open(Book1, encoding="utf8") as f:
FileNotFoundError: [Errno 2] No such file or directory: 'test.docx'
>>>
= RESTART: C:\Users\Sukh\Desktop\Cryptography\Assignment 2\Script\Assign2.py =
Enter the first .txt file location: test.csv
Traceback (most recent call last):
  File "C:\Users\Sukh\Desktop\Cryptography\Assignment 2\Script\Assign2.py", line
 8, in <module>
    with open(Book1, encoding="utf8") as f:
FileNotFoundError: [Errno 2] No such file or directory: 'test.csv'
```

Here I am entering various file formats and file names that do not exist in the current directory, producing errors as the file doesn't exist.



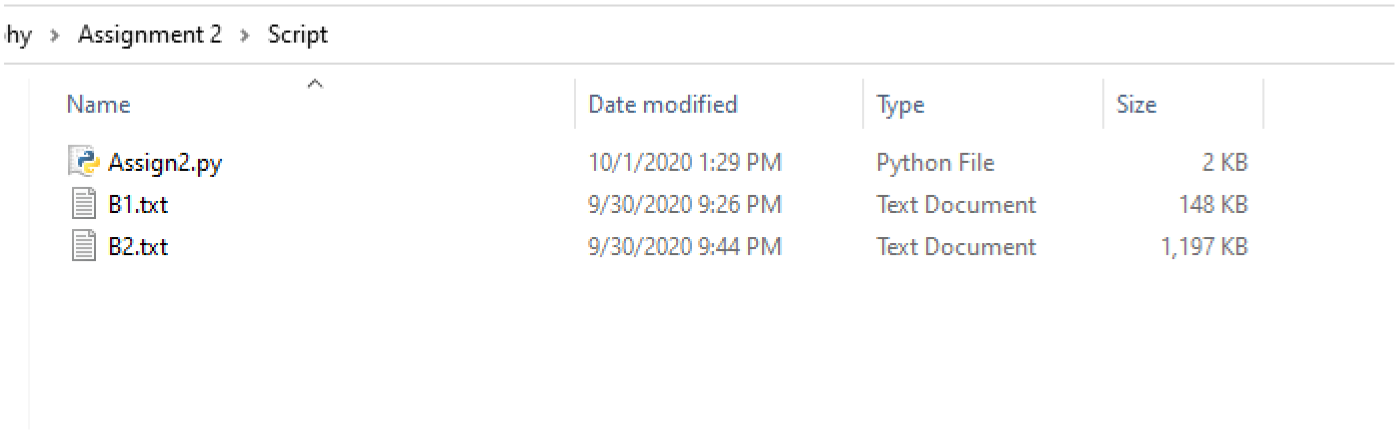
```
= RESTART: C:\Users\Sukh\Desktop\Cryptography\Assignment 2\Script\Assign2.py =
Enter the first .txt file location: B1.txt
>>>
= RESTART: C:\Users\Sukh\Desktop\Cryptography\Assignment 2\Script\Assign2.py =
Enter the first .txt file location: B2.txt
>>> |
```

I ran the two files shown at the start, B1.txt and B2.txt which produced no errors and was successfully converted to csv format.

Test 2

Here I'll be using Moby Dick, B2.txt as a reference.

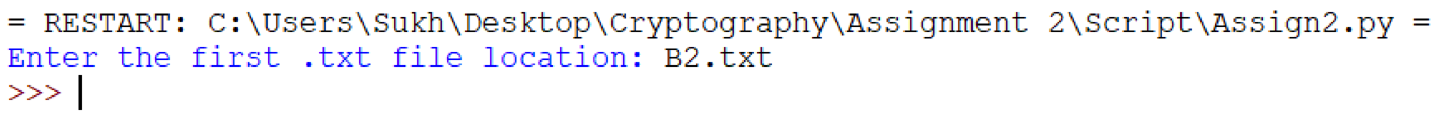
First I'll delete all existing csv files in the folder.



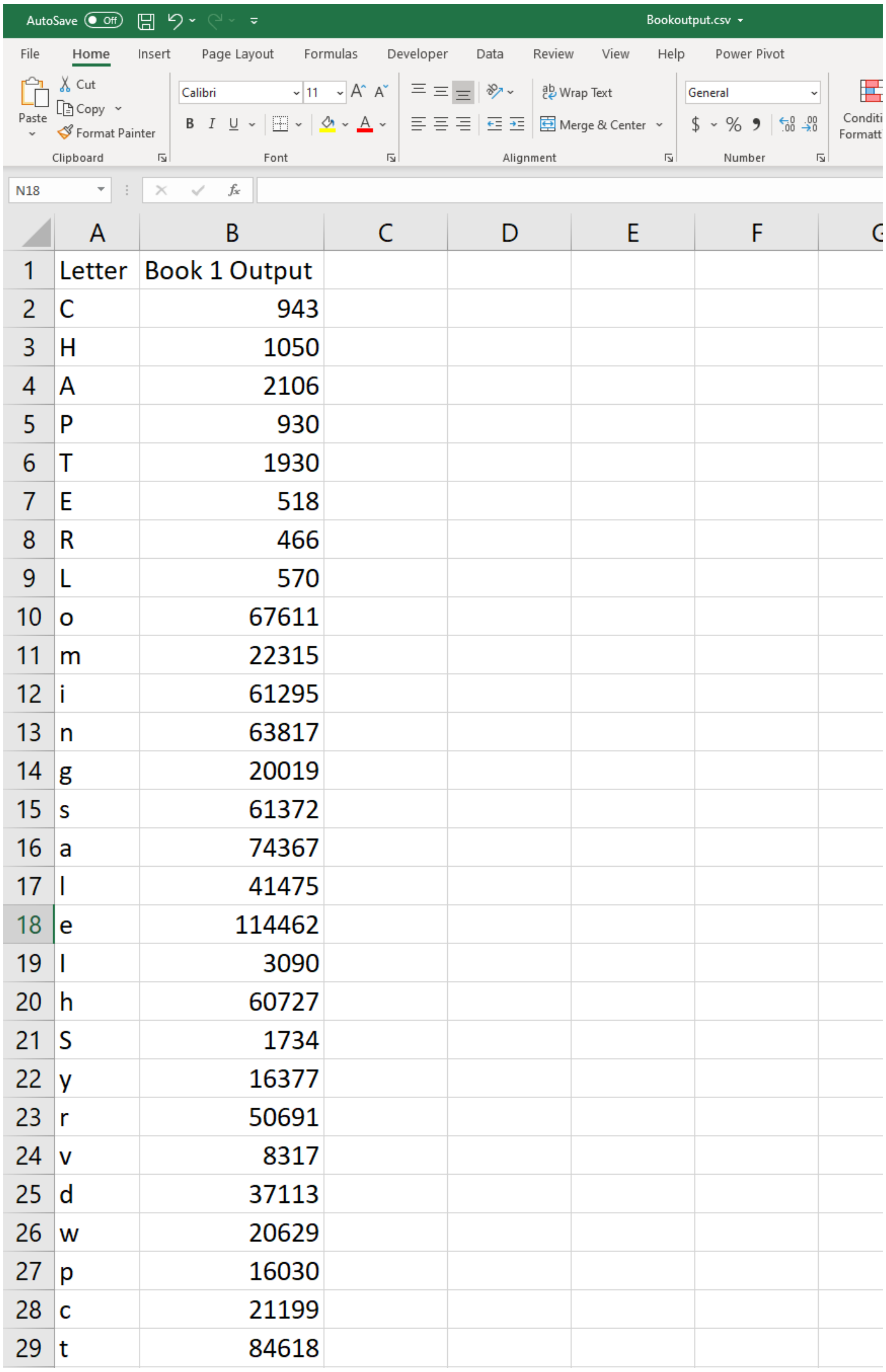
hy › Assignment 2 › Script

Name	Date modified	Type	Size
Assign2.py	10/1/2020 1:29 PM	Python File	2 KB
B1.txt	9/30/2020 9:26 PM	Text Document	148 KB
B2.txt	9/30/2020 9:44 PM	Text Document	1,197 KB

Next I'll run B2.txt through the script.



```
= RESTART: C:\Users\Sukh\Desktop\Cryptography\Assignment 2\Script\Assign2.py =
Enter the first .txt file location: B2.txt
>>> 
```



Letter	Book 1 Output
C	943
H	1050
A	2106
P	930
T	1930
E	518
R	466
L	570
o	67611
m	22315
i	61295
n	63817
g	20019
s	61372
a	74367
l	41475
e	114462
I	3090
h	60727
S	1734
y	16377
r	50691
v	8317
d	37113
w	20629
p	16030
c	21199
t	84618

Lastly, here is the output in csv format, with the letters in 1 column and the frequency count in the other. We can see it correctly formatted, easy to display format so we can easily use graphs or various other formats to play with the data.