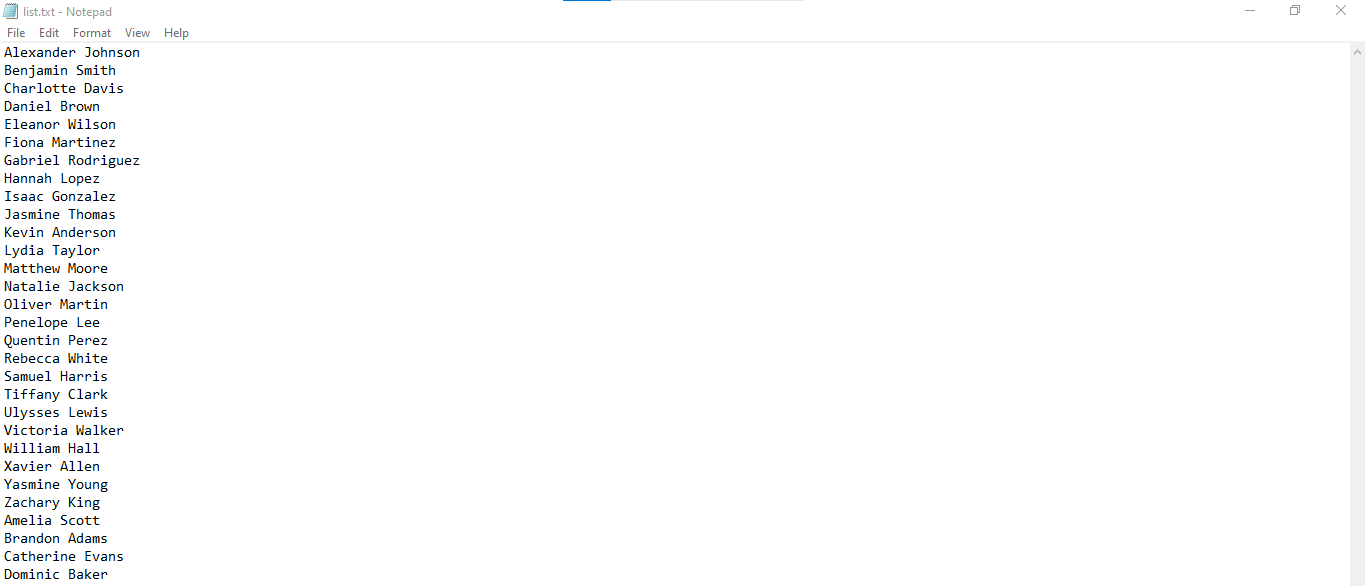
**Lab 03 Report**

**Part 1: Implementation**

**1. List.txt File**

First, I created text file that has 30 lists of names (first name and last name) then saved is as “list.text” file as shown below screenshot.



**2. Source Code (bloom\_filter.py)**

**A screen shot of a computer

AI-generated content may be incorrect.**

After creating text file of names lists, I then created python file for bloom filter function as shown above screenshot. This python file will read the names from the file I previously created which is list.txt file and allowing us to perform testing if given name is in the lists or not. This program I made will continually accepting names we entered and in order to exit, we can simply type” END END”.

**3. How to Run the Program**

* Open cmd command but make sure there is Python installed already in computer environment.
* Next, install the library of bloom filter using below command:

“pip install pybloom-live”

* Now we can run the program by simply navigating into its directory folder where the python program is saved, example, my program saved at: C:\Users\rose\Desktop\lab\_3,

Therefore, in cmd command, we will type as: C:\Users\rose\Desktop\lab\_3

* Execute the script: python bloom\_filter.py
* Enter Queries: We then now manually enter the **First and Last Name** to test if they are in the customer list. Example:

A screenshot of a computer

AI-generated content may be incorrect.

**Part 2: Boss’s Questions**

**a) Why can’t you just tell me if they are in the list or not?**

No, we cannot achieve 100% certainty due to the operational principles of Bloom Filters. In Greek mythology Odysseus wanted to listen to the Sirens' song although he knew it involved danger. He tied himself to the mast instead of drifting toward them. Bloom Filters operate in a similar way to Odysseus because they never miss identifying an affirmative response as positive. Under such circumstances a Bloom Filter guarantees that it will identify names that exist in the list. The behavior of a Bloom Filter functions like Odysseus' experience because it occasionally makes incorrect affirmative responses to queries that should return negative results. The filter operates through special patterns which check for name matches rather than keeping actual names inside its storage system. We say "probably a customer" since our system cannot guarantee "definitely a customer" when interacting with data."

**b) Why is the probability always the same for each name? Can it change?**

The probability remains fixed because the setup process of the Bloom Filter determines it rather than the name being verified. Bloom Filters achieve different false positive rates by assigning larger memory space or additional hash functions to minimize erroneous predictions of "customer" status and using minimal memory or hash functions for maximum misidentification. All names undergo the same level of false positive detection since we established fixed settings for the Bloom Filter beforehand. The program's performance becomes slower because adding more hash functions requires higher memory usage along with increased processing time although we could enhance it by enlarging both the filter size and number of hash functions.