

We Are Going to Hack Your Computer!

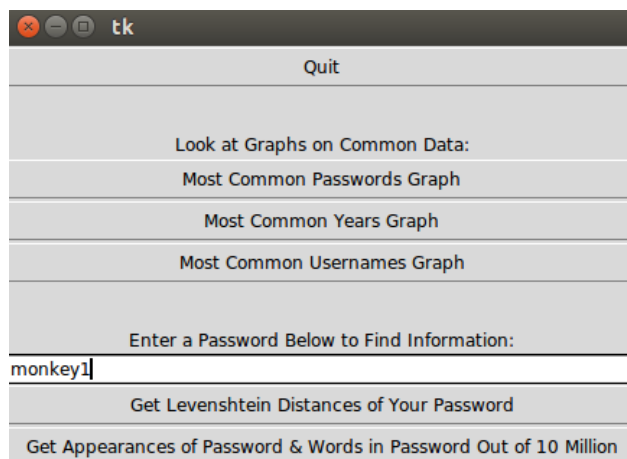
Lucy Wilcox and Nora Mohamed

This project's goal is to create an interactive visualization of our dataset of over 10 million username and password combinations, focusing on the password portion of the dataset. The program creates a GUI using TkInter, where the user can look at graphs of commonly occurring data on the passwords or look at graphs with information comparing user input to the data set. The graphs are generated in the user's web browser using Bokeh, which allows them to be interactive as well.

We made a user interface with TkInter, which has label which note if the user is getting common data, or data specific to the password they enter. There are buttons that they can click to select which data they want see. Lastly there is an entry box where the person is able to type in their password. The user can choose to look at common passwords, common years contained within passwords, or common usernames.

It also allows the user to enter a password of their choice and look at data about that password. They can see the Levenshtein distances between their password and other passwords. The Levenshtein distance is the number of changes that need to be made on one word to get to another word. This is relevant because if the Levenshtein distances are primarily small, their password is not particularly unique. They can also look at number of times different English words contained within their password and their password appear as passwords within the set of passwords.

We used Bokeh to display the data. For most of the data we used dot plots, which is basically a bar graph to show data. The user can zoom to view the data more clearly if desired. For the Levenshtein distance plot, the data is plotted where the radius is the distance. This creates circles, which we thought looked more interesting. In this graph, the user can hover their mouse over the data points to see more.



tk

Quit

Look at Graphs on Common Data:

Most Common Passwords Graph

Most Common Years Graph

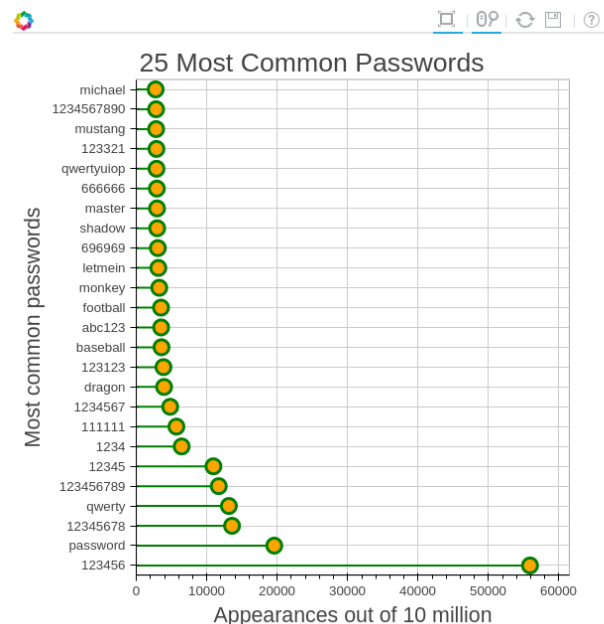
Most Common Usernames Graph

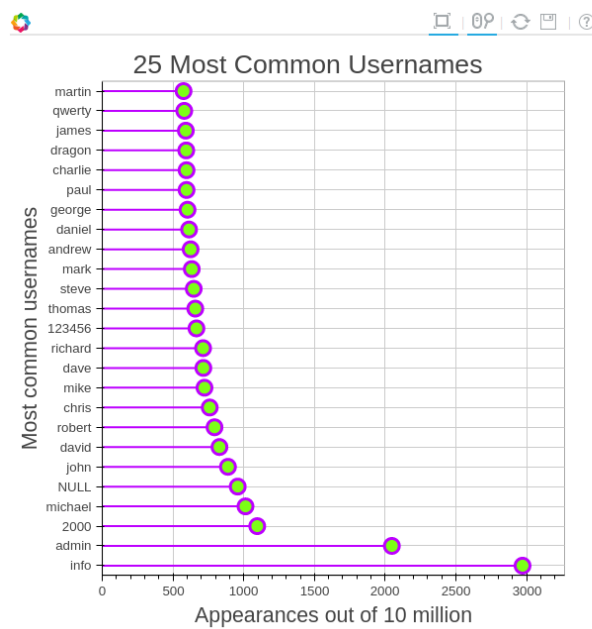
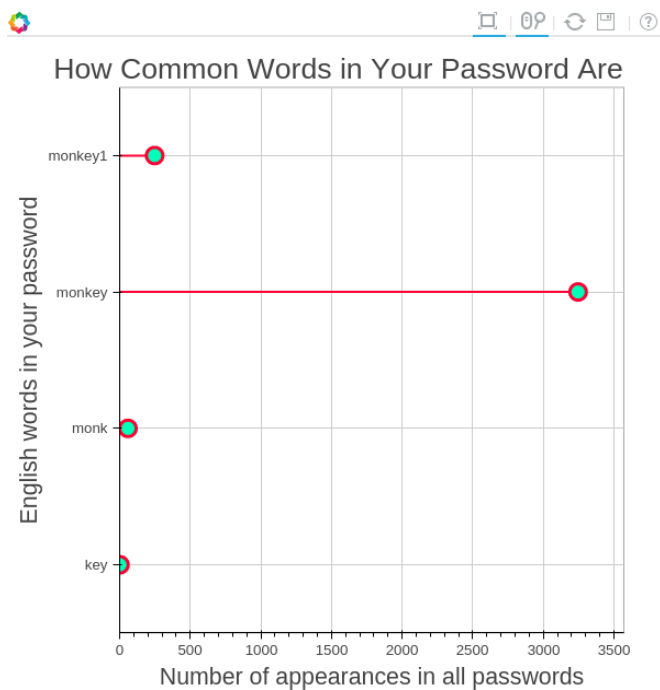
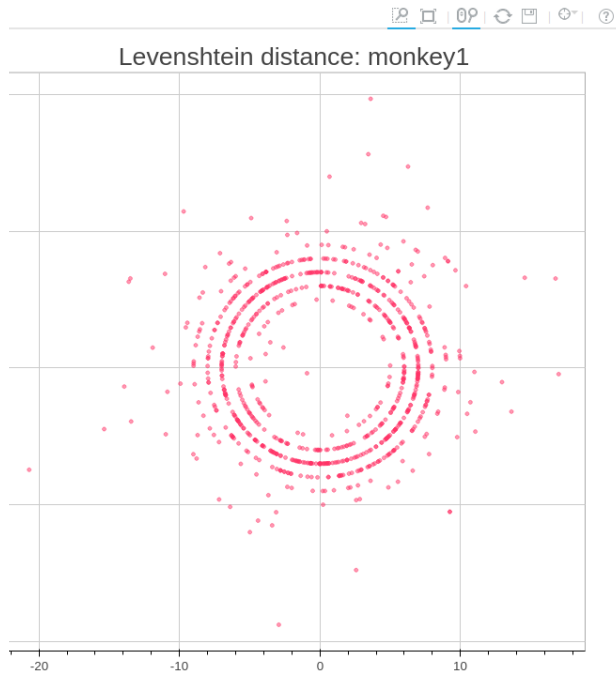
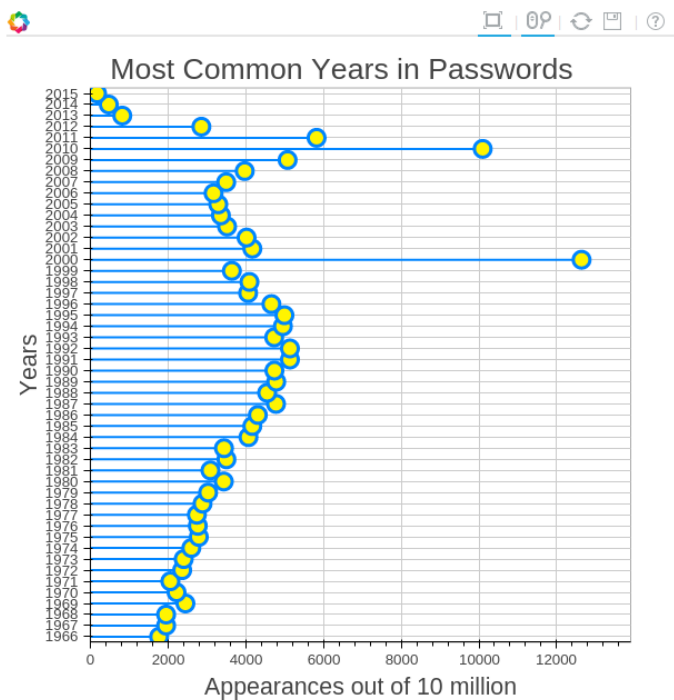
Enter a Password Below to Find Information:

monkey1

Get Levenshtein Distances of Your Password

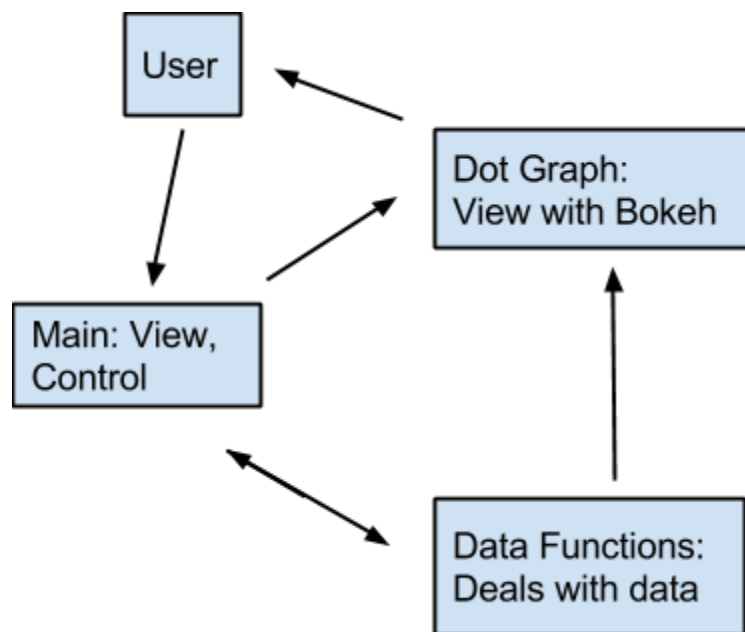
Get Appearances of Password & Words in Password Out of 10 Million





Our project has three different files, with four classes in total. In `main.py`, we have a class that manages the GUI. This class, *MakeWindow*, uses TkInter to create a GUI, whereby clicking buttons will run functions that create classes to generate graphs. These graphs are generated in a *DotGraph* in `dot_graphs.py`, which creates the two different types of graphs we have. The next two classes deal with two different data sets: *File* refers to the 10 million passwords data set, and *English* refers to the english dictionary (which we use for comparison between your password's english words and the data set's english words). These two classes were used to parse through the passwords and english dictionary in order to generate dictionaries or lists comparing the data to your password (or just analyzing the data in order to create the common data graphs). The data structures we used within the classes are mainly lists or dictionaries, where functions would return lists (made from dictionaries) to graph.

One design decision was on what library we should use for graphing and interactivity. We decided to use TkInter because *Think Python* included a chapter on it, and it seemed to be the simplest library for us to implement a GUI with. We also had to choose between what library we should use for graphing. Bokeh was confusing, and Matplotlib was more familiar to us. However, we decided to go with Bokeh because we hadn't used it before and it had the ability to hover over data points (along with other built in interactivity features).



We should have thought about classes earlier on instead of just trying to get the code working quickly, because this led to us taking more time in the long run and having to clean up a lot of dirty code. We were able to unit test by keeping our code in different files and test what was happening without trying to call the code from different places. This also made it easier for both of us to work on the code at the same time. Our original project was beyond what we were capable of. It would have taken far too long to learn a GUI

library that was powerful enough accomplish some of our original goals. We wish we knew more about using classes and realized that data visualization would be hard because all of the examples would be about games.

Sometimes we pair programmed, but mostly we each worked on different parts of the project while we were sitting together. Sometimes we worked on bits of the project on our own. Mostly Nora worked on Tkinter (main.py), Lucy dealt more with data (datafunctions.py), and we worked about equally on Bokeh (dotplot.py). Interestingly at the beginning, Lucy used Tkinter and Nora did the data functions, but we switched at some point through the process. We also started with Matplotlib because it was easier to use than Bokeh and gave us some results, but then improved our visualization by learning Bokeh. We did not have any big issues, but at first Lucy was really stressed out about picking a project and meeting the mid-project check-in. Once we choose and got started that stopped being an issue. Next time we might have chosen to do games because there was more guidance, though we not that interested in games.