Nick Edwards
Nucamp
Python Fundamentals
16 Nov 2024

WHIPPED PASS PY

Passwords as easy as 🥧

1. Introduction

Whipped Pass Py is a password aid powered by Python! This program helps the user create a dynamic and memorable password system for themselves that will create a unique password for each login. Whipped Pass Py also teaches users strong password creation techniques enhancing their own private online security.

Note: this application is still a work in progress, so I will discuss the design and planning, current functionality, and plans for for further development/

2. Design and Implementation

The early seeds of this project can be found in the z_dev_notes directory. In the subs_test file, you can see the notes from my research on which characters are commonly allowed or prohibited by most logins. I created a list of allowed symbols and prohibited characters, to check against. I created a few test functions at the bottom of this file to clean up each of the dictionaries by identifying issues programmatically. This was a critical step I had not anticipated which will result in a better program overall as users are less likely to encounter errors when registering credentials. It also means the lists and dictionaries can be modified and updated as standards for authentication change.

I then began work on the handle_alpha and handle_numeric files, to solicit an alphabetical and numeric input form the user. After both submissions are validated, the alphabetical component is run through a series of random replacements using data from the sub_dicts file. This enhances security by ensuring the words will not be spelled according to the dictionary and both numbers and symbols will be in the final password.

A major hurdle in this project was randomizing the number of replacements and which replacement characters were chosen so that the function does not return the same result each time. This means the user can re-run the program over and over again, to create many different password systems.

3. Conclusions

This project was a great exercise in working with many data types, especially manipulation of strings and lists. The randomization of the number of replacements and which replacement characters are used was much more difficult to test in hindsight. I'm not sure how another approach may have worked for this program and I definitely learned about how the random module works. I'm proud of how this feature turned out and believe the code is straightforward, legible and fairly DRY. As far as shortcomings, this largely came down to time–though I plan to continue to work on this as something to highlight and discuss during my job search.

For further development, I want to combine the results of the alphabetical and numeric strings in different orders to allow the user to choose a password that works best for them and is still secure. I also want to incorporate login "variables" chosen by the user that pull from the domain

to create unique credentials for every login (e.g. first letter, last vowel, etc). Further expansion of this project might include creating user “profiles” so a user can save, archive, modify, or delete their passwords.