**Introductory Programming (Python)**

**Password Assignment**

**INTRODUCTION**

*Password management systems* are programs that allow users to store, generate, and manage their passwords securely. Believe it or not, you probably interact with these programs regularly. If you have an iPhone (or some other Apple product) you may have familiarity with the Keychain password management system that stores all of your passwords for all of your personal website accounts, and even suggests strong passwords to use for new accounts. Most web browsers including Edge, Chrome, and Firefox have password managers that save and autocompletes user’s password forms. Examples of third party password management systems include 1Password, Bitwarden, and Dashlane.

In this assignment you will build your own password management system as an introduction to topics in lists and strings by implementing the following features:

* A function that allows users to check how strong their current password is.
* A function that allows users to keep track of older passwords.
* A function that allows user to review all their older passwords.
* A function that allows users to update their current password to a new one.

As you work on implementing these features, reflect on the importance of including each of these features for sensitive programs such as password management systems. Read through the “Security and Ethics Objectives” section of this write-up before starting the assignment.

**PRIOR KNOWLEDGE**

This assignment recommends you have prior knowledge in the following topics:

* Declaring functions and variables
* Writing conditionals (e.g., if(...), elif(...), else, switch(...))
* How to utilize relational and logical operators
* How to implement both for and while loops

**COMPUTING OBJECTIVES**

By the end of this assignment, you will have:

* Utilized loops to access and traverse lists.
* Utilized loops to access the individual characters of a string.
* Learned to compare strings.
* Reinforced knowledge in previous topics (e.g., operators, loops, conditionals, etc.).

**SECURITY AND ETHICS OBJECTIVES**

Based on the lecture materials and through your own reflection on creating a password manager, by the end of this assignment, we hope you will have considered and reflected on the following issues and questions:

* Who are the different people who would be affected by your password management system? What do you think each group of people, or *stakeholders*, would value most about this system?
* What features do you think are necessary for a good password management system?
* When designing new password management systems, what proactive steps could be taken to address the implications of previous, poorly designed password management systems?
* Regardless of how you implemented password requirements in a technical sense, how might you encourage users of a system to have good password practices?

**GETTING STARTED**

You will be writing functions that represent several different features that would be used in a password management system. As you write out each specified function, we want you to reflect on several things:

* Why would this feature in a system be desirable?
* What would a *good* implementation for this feature look like?
* What would a *bad* implementation for this feature look like?
* What do you think could be further improved about this feature beyond the instructions we give to you?

**Function 1: Password Strength Checker**

def passwordStrength(password)

The passwordStrength() function should accept a string as its only input parameter which can represent any password. For this function, you will want to initialize a counter called satisfaction\_counter that will record how well a given password satisfies a few criteria. You will add 1 to the satisfaction\_counter if a given password satisfies each of the following conditions:

* The length of the password is greater than 12.
* The password contains a lowercase letter.
* The password contains an uppercase letter.
* The password contains a numerical value.
* The password contains a special ASCII symbol.

You will only want to count each condition *once*. This means that the greatest number satisfaction\_counter can be is 5. If when the function is called with a particular password the satisfaction\_counter is equal to 0 or 1, then your function should print “Your password is WEAK in strength.” If the satisfaction\_counter is equal to 2 or 3, then your function should print “Your password is MEDIUM in strength.” If the satisfaction\_counter is equal to 4 or 5, then your function should print “Your password is STRONG in strength.”

Take a look at these input/output examples below to see how passwordStrength()works.

**Input:**

passwordStrength(‘password’)

passwordStrength(‘P@$sW0rdz’)

**Output:**

Your satisfaction counter is: 1

Your password is WEAK in strength.

Your satisfaction counter is: 4

Your password is STRONG in strength.

**Function 2: Password List Updater**

def updatePasswords(previous\_passwords, current\_password):

The updatePasswords() function accepts a list of strings (representing a list of previous passwords) and a string (representing the user’s current password) as its two input parameters. When this function is called, it shifts all the previous passwords contained inside the previous\_passwords list one space to the left and then changes the last element of the previous\_passwords list to equal the current\_password.

Take a look at our input/output example below to see how updatePasswords()works.

**Input:**

previous\_passwords = [“”, “”, “”, “”, “password”]

current\_password = p@ssw0rd

updatePasswords(previous\_passwords, current\_password)

**Output:**

previous\_passwords = [“”, “”, “”, “password”, “p@ssw0rd”]

**Function 3: Password Review**

def reviewOldPasswords(previous\_passwords):

The reviewOldPasswords()function accepts a list of strings (representing a list of previous passwords) as its only input parameter. When this function is called, it prints all previous passwords from the previous\_passwords list from oldest to newest. You will recall from updatePasswords()that the last element of previous\_passwords represents the most recent previous password, while the first element represents the oldest previous password.

There is a particular format for previous passwords to be printed, so please refer to the input/output example below to see how reviewOldPasswords() works.

**Input:**

previous\_passwords = [“”, “”, “123459”, “helloW0rlDz”, “password”]

reviewOldPasswords(previous\_passwords)

**Output:**

............

...Oldest...

123459

helloW0rlDz

password

...Newest...

............

**Function 4: Password Changer**

def changePassword(current\_password):

The changePassword() function will accept a string representing the user’s current password as its only input parameter. When this function is called, it will first make a call to updatePasswords() and then pass previous\_passwords and current\_password to that function. After making a call to updatePasswords(), you will then want to make a call to reviewOldPasswords() and pass your newly updated previous\_passwords to that function call. Once you have made both of those function calls, you will then want to update current\_password to your new desired password. If your new password is equal to your previous current\_password, then you will want your user to input a different new password.

Take a look at our input/output example below to see how changePassword()works.

**Input:**

current\_password = password

changePassword(current\_password)

**Output:**

............

...Oldest...

password

...Newest...

............

Please input your new password:

> P@$w0rDz

Your new password is: P@$w0rDz

**Starter Code**

def get\_menu\_choice():

In addition to the main() function, we have provided you with the get\_menu\_choice() function. This function is meant to print a list of choices a user may choose to utilize in this password management system.

def main():

In addition to the get\_menu\_choice() function, we will provide you the main() function. This function will initialize variables such as previous\_passwords and current\_password, as well as create a while loop that continues to introduce menu choices.

You should note in the code that the default starting password or this system is ‘password’ – you will need to use this to “log in” at the start!

This assignment is licensed under a Creative Commons CC-BY license. For more information, see:

Casey Fiesler, Samantha Dalal, and Joshua Paup. Passwords and Python: Introducing Security Concepts in Lower-Division Programming. ACM EngageCSEdu. 2023. 10.1145/3631988

A grey and black sign with a person in a circle

Description automatically generated