

Criterion C - Development

Introduction:

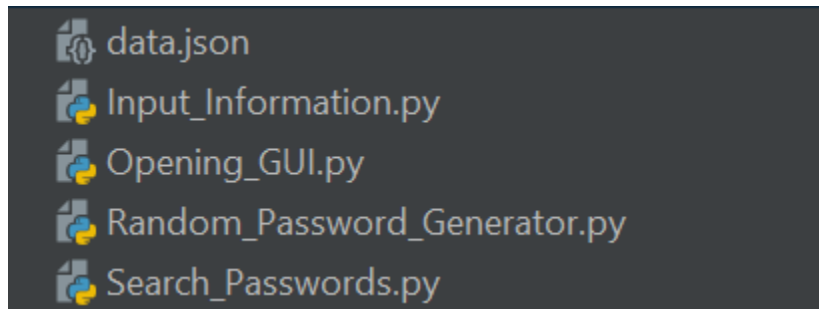
My code was developed with Python and I used Tkinter to make a Graphical User Interface. The program is a password manager that allows the user to input passwords, generate random passwords, and search for passwords that were previously inputted.

Summary List of All Techniques:

- Parameter passing
- If else loop
- For loop
- Use of a flag value
- Method returning a value
- Array of objects
- Graphical User Interface (GUI)
 - GUI tabs
 - GUI popup menus
- Import JSON
- Random string generation
- Creation of functions
- Saving to a file
 - JSON
- Message boxes
- Error handling

Structure of the Program:

The program contains 5 classes. Four out of five classes are GUI's to aid in the user's experience. These 4 classes execute and display information to the user. The fifth class holds all the data that is inputted from the user, which can be accessed from another class.



The Opening_GUI class allows the user to choose from the three main functions of the code: Input Information, Random Password Generator, and Search Passwords. The Input_Information class allows the user to input a website name accompanied with their username and password. This data is stored and can be accessed from the Data class. The Random_Password_Generator class allows the user to generate a random password. The Search_Passwords class returns a given password from the Data class.

Data Structures Used:

I used different types of lists in my program:

- *Tuples*- special characters on keyboard are stored in tuple as a single variable (special_characters)

```
special_characters = list("!@#$%^&*()")
```

- This allows for all the special characters to be called with one variable, so that they are all randomized. This makes the code more efficient as I did not have to individually call each character for the code. This also increases the randomness of the password as the code shuffles through all the characters in this single list.
- *Arrays*- the generated password was stored as an array so each character could be randomized

```
password = []
```

- The randomly generated password is stored as an array so that all elements of the password, which are inputted by the user, are indexed by the numbers chosen by the user. This will be further explained in Main Unique Algorithms.

File reading:

JSON- My program inputs data into a JSON file because it can store simple data. It also makes it easier to access data to and from a GUI. The information is inputted from the Input_Information GUI and can be accessed for the Search_Passwords GUI.

```
{
  "hello": {
    "email": "bye",
    "password": "ok"
  },
  "morning": {
    "email": "evening",
    "password": "night"
  }
}
```

Main Unique Algorithms:

Interactions between GUI's:

There are 3 GUIs where the user can interact with. From each one, you can move onto the other 2. There are 2 buttons on the bottom of the GUI's which allow the user to move between the GUI pages.

```
def nextPage():  
    mgr.destroy()  
    import Input_Information
```

The GUI that the user is currently on is closed and the other GUI is imported, when the button is clicked.

The information inputted from the “Input_Information” GUI can later be accessed from the “Search_Passwords” GUI.

The image displays two side-by-side screenshots of a Python-based GUI application titled "Password Manager".

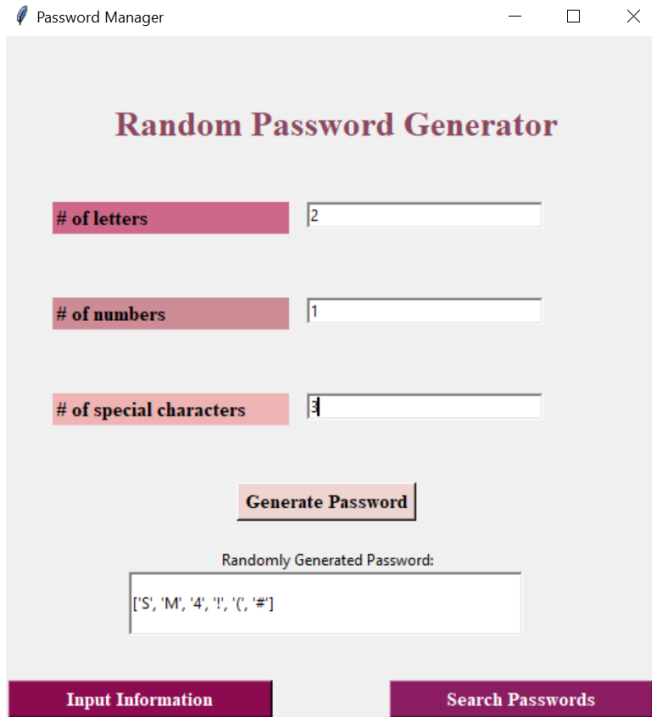
The left window, titled "New Account Information", features three input fields: "Website:" with the value "morning", "Username:" with the value "evening", and "Password:" with the value "night". Below these fields are two buttons: "Enter" and "Cancel". At the bottom of the window are two buttons: "Random Password Generator" and "Search Passwords".

The right window, titled "Search Account Passwords", features an input field labeled "Enter website" with the value "morning". Below this is a "Search" button. Further down is a "Password:" label above an input field containing the value "night". At the bottom of the window are two buttons: "Input Information" and "Random Password Generator".

The website name (“morning”) which is inputted in the Input_Information GUI can be entered into the Search_Passwords GUI to return the password (“night”) that was also inputted in the Input_Information GUI. This is because the information inputted in the first GUI is entered into a data.json file, and is called from the second GUI to be outputted to the user.

Random Password Generator:

The user inputs how many characters they want for each type of character (letters, numbers, special characters).



The screenshot shows a window titled "Password Manager" with a "Random Password Generator" section. It features three input fields: "# of letters" with value 2, "# of numbers" with value 1, and "# of special characters" with value 3. A "Generate Password" button is below these fields. The output area, labeled "Randomly Generated Password:", displays the string "[S, 'M', '4', '!', '(', '#']". At the bottom of the window are two buttons: "Input Information" and "Search Passwords".

The code creates an array and individual for loops for each type of character. Each for loop will be indexed by the number chosen by the user, for that specific type of character. This will randomly choose the given amount of characters from the pre-created lists for those characters. All the randomized characters are appended together and shuffled once more to make the password even more random. Finally the randomly generated password is displayed to the user.

```
password = []

for i in range(letter_count):
    password.append(random.choice(letters))

for i in range(number_count):
    password.append(random.choice(numbers))

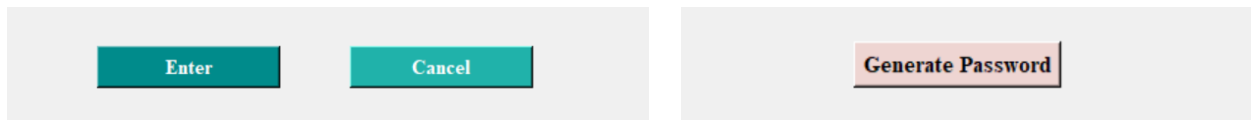
for i in range(special_character_count):
    password.append(random.choice(special_characters))
```

This is complex and unique because it doesn't just randomize for a set password length. The user is given the freedom of choosing the length of the password and the elements that go into it.

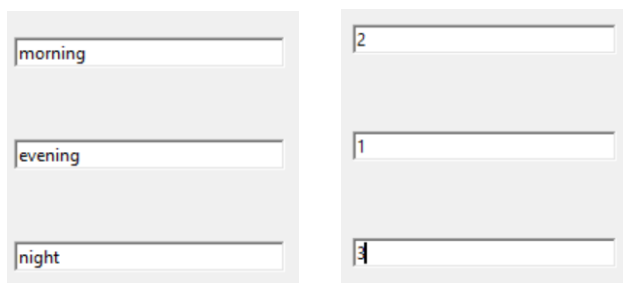
User Interface/GUI Work:

I used multiple elements in my GUI to help the user navigate through the program easily:

- *Buttons*- to submit information into a data file or perform a task/ carry out function: search a file, generate a password, destroy GUI, etc...



- *Entry boxes*- to input information, or commands

The image displays two columns of text entry boxes. The left column contains three boxes with the labels 'morning', 'evening', and 'night' respectively. The right column contains three boxes with the numerical values '2', '1', and '3' respectively.

- *Result boxes*- to display/return information to user

The image shows two result boxes. The first box, labeled 'Password:', contains the text 'night'. The second box, labeled 'Randomly Generated Password:', contains a list of characters: ['S', 'M', '4', '!', '(', '#'].

- *Message boxes*- to confirm information or show error messages



The message box on the left retrieves the values/information from entry boxes in Input_Information to display to the user before adding to the data.json file.

```
user_website = website_entry.get()  
user_email = email_entry.get()  
user_password = password_entry.get()
```

Software Tools Used:

The main development tool I used was Python because it is easy to navigate and I have more experience with it. Python would also allow me to complete this task with more complexity because it has many add ons, giving me a wider range of functions to use.