A GUI Password Creator

For this PA, you will write a single program with four functions. **It isn't sufficient for your program to run correctly; 25% of your grade will be based on the following:**

- **use of a header with required information (5%),**
- **use of a docstring in each function (5%),**
- **use of comments,**
- **choice of lvalue and variable names, and**
- **readability, i.e., include whitespace (15% for last three).**

You will need to submit your program to Canvas as a zipped file so if you don't know how to zip a file, ask your TA during your lab or else google how to do it for your operating system. **Note that the grading rubric for this PA is available on Canvas, and you should look at it to see what's expected.**

Header information. Your program must include the following information in the header.

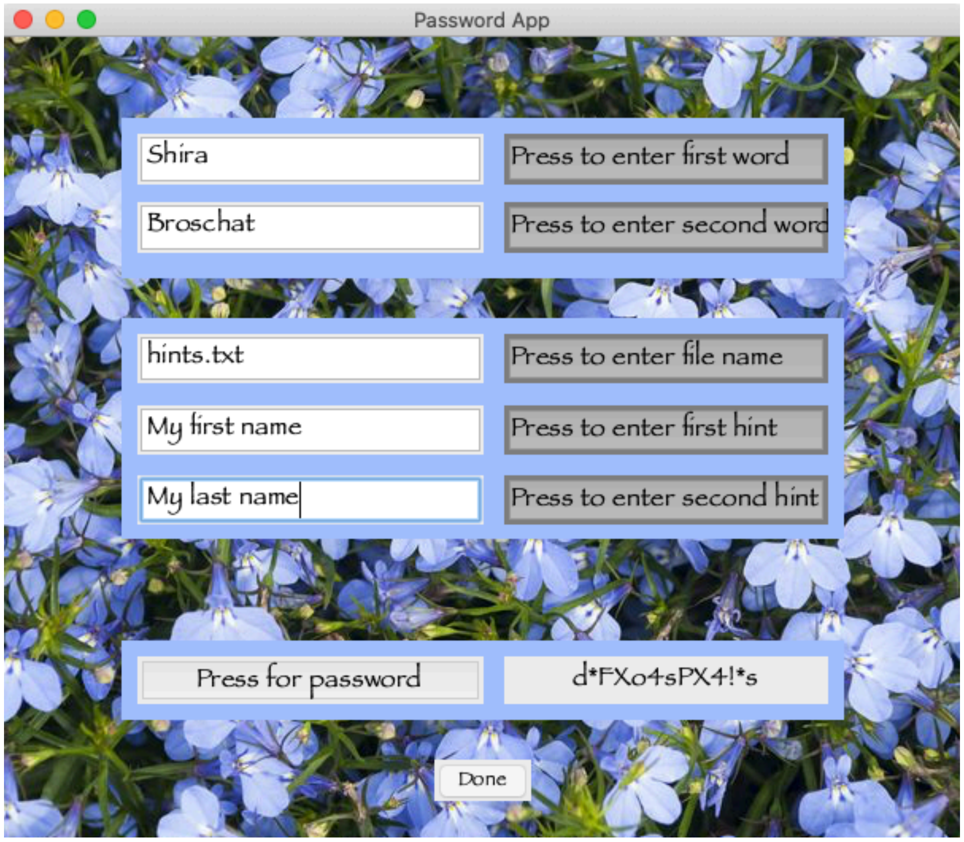
- Name
- Date
- CptS 111, Spring 2023
- Programming Assignment #4
- Name of program
- Brief description and/or purpose of the program; include sources if you used any, e.g., website URLs you used

Background Information

You know how some websites require you to have a strong password or recommend some weird combination of characters like `v$AcMQ?gz5!`? They can't really expect anyone to remember passwords like this, can they? Well, in this PA you're going to develop a password creator that will generate just such types of gobbledygook passwords, but the secret is that you won't have to remember them. Instead, you'll look in your hint file which will provide hints for the two ordinary words you used to create the password. This will allow you to regenerate it if necessary.

Program Requirements

You will create a GUI application that has entry boxes for entering text and buttons to press to perform actions. It should look as follows:


Password App
Press to enter first word
Press to enter second word
Press to enter file name
Press to enter first hint
Press to enter second hint
Press for password
Done

Shira

Broschat

hints.txt

My first name

My last name

d*FXo4sPX4!*s

Don't panic! Most of the code you need for this PA is in the template provided with this prompt. In fact, you don't need to do any of the GUI programming. However, if you feel up to it, you're free to change a few items in the portion of the template with the GUI code: the color of the three frames, the font type, and the background image. Instructions are given within the template if you want to do this, but it's perfectly fine if you don't and leave everything as it is. I've posted a zipped file of the image I used with my version `lobelias.png` together with a dozen other images you might want to try. You're also free to find an image on your own, but note that it has to be a `.png` file. **It will also be helpful if you go back and forth between the comments in the template and this prompt before you start coding.** The template comments should help you understand the structure of the program. **Importantly, the image file you use must be in the same folder (directory) as your `pa4.py` file. Ask me, a TA, or a VCEA tutor if you can't figure this out.**

The four functions in this program are described below in the same order they appear in the template. Note that you only need to add code in the template when you see **`<add>`** or **`<add_code>`**. However, before we get to the functions, note the following four points:

1. You need to add the header info as instructed at the top of the template.
2. This program uses the `tkinter` module. You'll see at the top of the program that it has been imported with the alias `tk` which is the typical alias used for this module.
3. You need to initialize some of the global values: `i` and `j` should be initialized to zero, and `words` should be initialized to an empty list. **The statements for initializing these variables should be placed where instructed!**

4. Last, save the template to the name of your program, i.e., **`pa4.py`**.

Now on to the functions!

- **`add_wd()`**: A void function with one parameter, a string representing one of the two words entered by the user to create the password. `add_wd()` is called twice, the first time when the user presses the button to enter the first word and again when the user presses the button to enter the second word. The `global` command allows the variables defined as global to be found outside the function. The `if` conditional allows you to create more than one password in one session. What you need to do to complete this function (but don't just blindly follow the instructions; think about what you're doing because that's how you learn!):
  - In the function header, add a parameter name for the word passed to the function.
  - Write two lines of code i) to increment `i` by 1 and ii) to append the word passed to the function to the `words` list, i.e., append the name you used in the function header.
  - Comment out (or delete) the two lines of code you don't need (depending on your operating system).
  - Add a docstring. Note that **`add_wd()`** creates a list of the two words entered by the user, and it also turns the buttons gray to show they have been pressed.
- **`create_file()`**: A void function with one parameter, a string representing the file name entered by the user. `filnam` is defined as global so that it can be found outside this function. What you need to do to complete this function:
  - In the function header, add a parameter name for the hint file passed to the function.
  - Write one line of code to assign the parameter passed to the function to the lvalue `filnam`, i.e., the name you used in the function header.
  - Comment out (or delete) the one line of code you don't need.
  - Add a docstring.
- **`write_hint()`**: A void function with one parameter, a string representing the hint entered by the user. This function is called twice, once for the first hint and again for the second hint, and is similar to the `add_wd()` function. What you need to do to complete this function:
  - In the function header, add a parameter name for the hint passed to the function.
  - Write four lines of code to i) increment `j` by 1, ii) open the file `filnam` using the `'a'` option (append to file), assigning it to an appropriate lvalue, iii) print or write the hint to the file, and iv) close the file using its lvalue. Here's how to do ii)-iv) (Important: iv) is done after the `if-else` construct):

    ```
    lvalue_name_chosen = open(filnam, 'a')
    print(variable_name_chosen, file=lvalue_name_chosen)

    lvalue_name_chosen.close()
    ```
  - Comment out (or delete) the two lines of code you don't need.

First Word	Second Word	Password
My Little	Pony	rFuRWR$l!PPl3
Joyride	721865	&Ba-D2LFR*!E3
Chicken	Nuggets	u_993PXo4!ok3u
Mocha	Latte	lsPP3WFo4s

  - Add a docstring.

- **`gen_pw()`**: A void function with no parameters that converts the two words entered by the user to create a gobbledygook password. The way this is accomplished is by first reversing the order of the two words and then substituting characters as shown in the template. You must make all the substitutions exactly as given for the PA. `text` is defined as global so it can be found by the GUI code. The statement for combining the two words in reverse and also changing the case of the words to lowercase is already given in the template. What you need to do to complete this function:
  - Write as many lines of code as you need to make the appropriate substitutions. There are a number of ways to do this, but be careful with the order in which you make the substitutions. Note that the statement for replacing a character is, for example, `passwd = passwd.replace('e', '3')`.

The examples given in the table above are provided to help ensure your program is correct. Note that you must try all four examples to verify that you made all the correct substitutions.

Submission information. Use Canvas to submit your program. Submit it as a zipped file called **`<first_initial+lastname>_pa4.zip`**. If you don't know how to zip a file, ask your TA or else google how to do it for your operating system (Windows, macOS, or Linux).