



APTECH COMPUTER EDUCATION
METRO STAR GATE

ADSE-1

PROJECT REPORT

PyPasswordGenerator

DEVELOPERS

ZAINAB MUSHARAF

AKBAR AMIN

FACULTY: SIR FARAZ AHMED

DATE: 13th JULY 2023

Catalog

PROJECT CERTIFICATION	3
OBJECT OF THE PROJECT	4
PROBLEM STATEMENT	4
PROJECT WORKFLOW	5
User Interface Setup:	5
Event Loop:	5
User Input:	5
Password Generation:	5
Display Password:	6
Error Handling:	6
Copying Password:	6
Documentation and Testing:	6
SOURCE CODE	7
SCREENSHOTS	11
Main output	11
Validation for empty Inputs	11
Generated Random Passwords	12
Generated Password Copied	12
PROJECT ANALYSIS	13
Overall, the "py-password-generator" project offers a user-friendly and efficient solution for generating strong passwords. The utilization of PySimpleGUI simplifies the user interaction, while the password generation functions provide flexibility and randomness to create secure passwords. ...	14
TASK SHEET:	15
FINAL CHECK:	15

PROJECT CERTIFICATION

This is to certify that

Mr. Akbar Amin (1280756) and Ms. Zainab Musharaf (1260377)
has successfully designed & developed



PY_PASSWORD_GENERATOR



Authorized Sign By
Sir Faraz Ahmed

Date of submission: 13th July 2023

OBJECT OF THE PROJECT

The Objective of this program is to give a sample project to work on real life projects. These applications help you build a larger more robust application.

The objective is not to teach you the concepts but to provide you with a real life scenario and help you create applications using the tools.

You can revise them before you start with the project.

It is very essential that a student has a clear understanding of the subject.

Kindly get back to eProjects Team in case of any doubts regarding the application or its objectives.

PROBLEM STATEMENT

Write a Python / R Program to find Armstrong number.

Having a weak password is not good for a system that demands high confidentiality and security of user credentials. It turns out that people find it difficult to make up a strong password that is strong enough to prevent unauthorized users from memorizing it.

Creating a strong password and remembering it is a tedious task.

You need to build a program that intakes some words from the user and then generates a random password using those words.

The user can remember the password with the help of the words he gave as an input.

PROJECT WORKFLOW

User Interface Setup:

- Import the necessary modules and libraries, including PySimpleGUI.
- Define the layout of the graphical user interface (GUI) using PySimpleGUI elements such as input fields, buttons, and text areas.
- Create a window object using PySimpleGUI and set its layout to the defined GUI layout.
- Initialize any necessary variables.

Event Loop:

- Enter an event loop that listens for user interactions with the GUI.
- Handle different events such as button clicks or window closures.
- Implement event handlers to respond to user actions appropriately.

User Input:

- Retrieve user input from the GUI elements.
- Obtain the words or phrases provided by the user and the desired length of the password.

Password Generation:

- Implement two password generation functions: `generate_password1` and `generate_password2`.
- `generate_password1` takes the user's words, counts the length, and generates a password with shuffled words and additional random characters for complexity.

- `generate_password2` takes the user's words, shuffles them, counts the length, and generates a password with shuffled words and additional random characters for complexity.

Display Password:

- Update the GUI to display the generated passwords.
- Show the passwords to the user in separate text areas.
- Allow the user to copy each password to the clipboard if needed.

Error Handling:

- Implement error handling mechanisms to handle any potential issues during the password generation process.
- Display informative error messages to the user if input validation fails or other errors occur.

Copying Password:

- Implement the functionality to copy the generated passwords to the clipboard.
- Handle user interactions with the "Copy" buttons.
- Copy the respective passwords to the clipboard using the `pyperclip` module.
- Provide feedback to the user indicating successful copying of the password.

Documentation and Testing:

- Document the project, including the overall functionality, code structure, and usage instructions.
- Include any dependencies and installation steps in the documentation.
- Perform thorough testing to ensure the password generator works as expected in different scenarios.

SOURCE CODE

```
import string
import random
import PySimpleGUI as sg
import pyperclip

def generate_password1(words, count):
    # Combine the words provided by the user

    combined_words = ''.join(words)
    word_length= len(str(combined_words))
    # Generate a random string of characters for additional complexity
    random_chars = ''.join(random.choices(
        string.ascii_letters + string.digits + string.punctuation,
        k=count-word_length))

    if word_length> count:
        # make list before shuffling
        List= list(combined_words)
        random.shuffle(List)
        mixed_combined_words = ''.join(List)
        password = mixed_combined_words + random_chars
        return password
    else:

        # Combine the words and random characters to create the password
        password = combined_words + random_chars
        return password

def generate_password2(words, count):
    # Combine the words provided by the user
```

```

combined_words = ''.join(words)

# make list before shuffling
List= list(combined_words)
random.shuffle(List)
mixed_combined_words = ''.join(List)

# find the length of the user input words
word_length= len(str(mixed_combined_words))

# Generate a random string of characters for additional complexity
random_chars = ''.join(random.choices(
    string.ascii_letters + string.digits + string.punctuation, k=count-
word_length))

# Combine the words and random characters to create the password
password = mixed_combined_words + random_chars

return password

# changing the theme of the window
sg.theme('DarkBlue6')

# changing the size and font
sg.set_options(font='verdana 15')

# GUI interface window design
layout = [
    [sg.Text('Enter some words:'), sg.Push(),
     sg.Input(size=15, key="-WORDS-")],
    [sg.Text('Length of Password :'), sg.Push(),
     sg.Input(size=15, key="-SIZE-")],
    [sg.Button('Generate', button_color="Green"), sg.Push(), sg.Button("Copy
1"), sg.Push(), sg.Button("Copy 2"), sg.Push(), sg.Button('Cancel')],
    [sg.Text('Generated Password 1:', visible=False, key="-PASSWORD_TEXT_1-"),
sg.Push(),
     sg.Multiline(size=15, key="-PASSWORD_1-", visible=False,
no_scrollbar=True)],

```



```

        [sg.Text('Generated Password 2:', visible=False, key="-PASSWORD_TEXT_2-"),
sg.Push(),

        sg.Multiline(size=15, key="-PASSWORD_2-", visible=False,
no_scrollbar=True)],

]

# Create the window
window = sg.Window("Password Generator", layout)

# Event loop
while True:
    event, values = window.read()

    if event == sg.WINDOW_CLOSED or event == "Cancel":
        break

    if event == "Generate":
        try:
            count = int(values['-SIZE-'])
            user_words = values["-WORDS-"].split()
            window["-PASSWORD_TEXT_1-"].update(visible=True)

            # For generated password 1
            password1 = generate_password1(user_words, count)
            window["-PASSWORD_TEXT_1-"].update(visible=True)
            window["-PASSWORD_1-"].update(password1, visible=True)

            # For generated password 2
            password2 = generate_password2(user_words, count)
            window["-PASSWORD_TEXT_2-"].update(visible=True)
            window["-PASSWORD_2-"].update(password2, visible=True)

        except ValueError:
            # Clearing Fields
            window["-PASSWORD_TEXT_1-"].update(visible=False)

```

```

        window["-PASSWORD_TEXT_2-"].update(visible=False)
        window["-PASSWORD_1-"].update("", visible=False)
        window["-PASSWORD_2-"].update("", visible=False)

        # error popup message
        sg.popup_no_buttons('Enter Valid Values', text_color='white',
                                auto_close=True, auto_close_duration=2,
background_color='Red', no_titlebar = True )

    # copying password
    if event == "Copy 1":
        password1 = values["-PASSWORD_1-"]
        if password1 != "":
            pyperclip.copy(password1)
            sg.popup_no_buttons('Password 1 copied successfully!',
text_color='white',
                                auto_close=True, auto_close_duration=2,
background_color='green', no_titlebar = True )

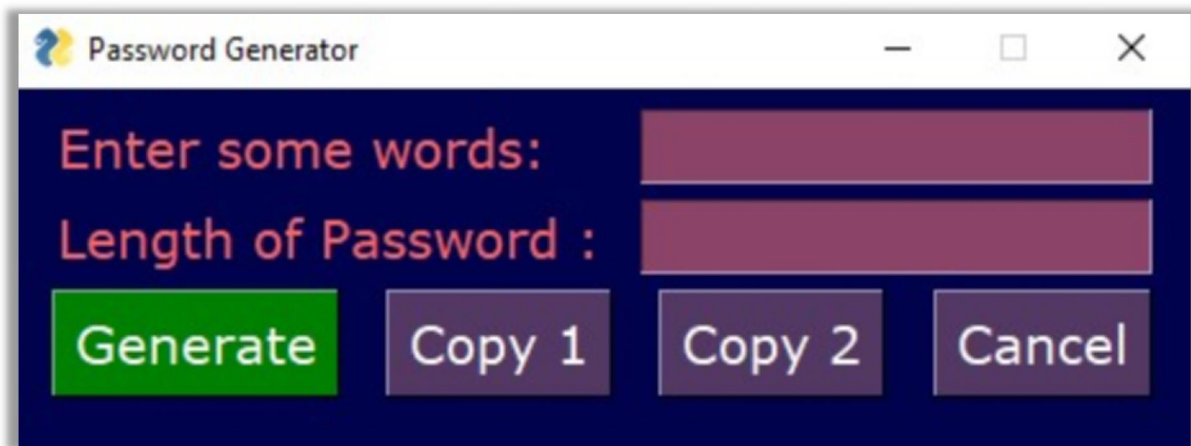
        if event == "Copy 2":
            password1 = values["-PASSWORD_2-"]
            if password1 != "":
                pyperclip.copy(password1)
                sg.popup_no_buttons('Password 2 copied successfully!',
text_color='white',
                                auto_close=True, auto_close_duration=2,
background_color='green', no_titlebar = True )

    # Close the window
window.close()

```

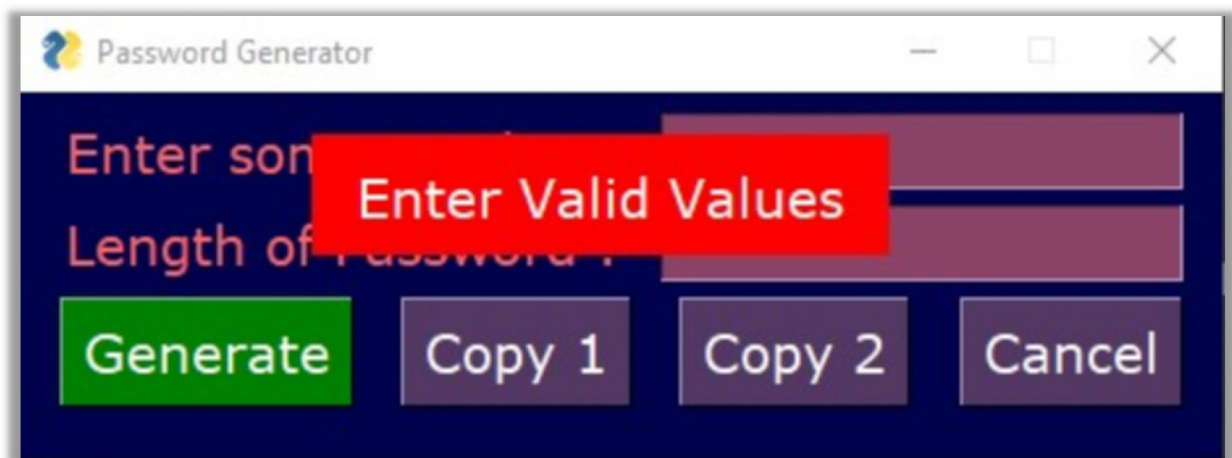
SCREENSHOTS

Main output



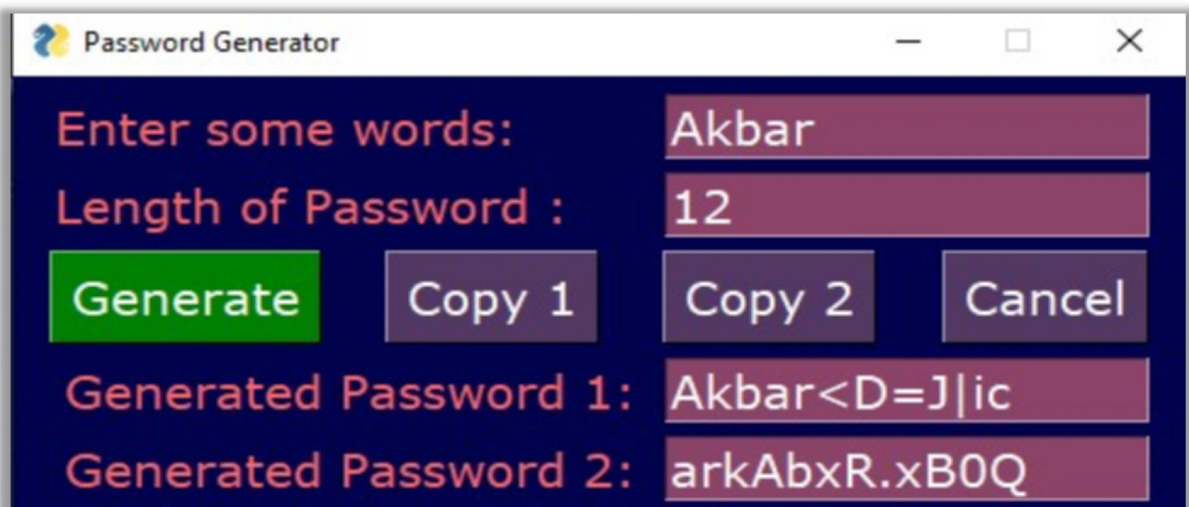
The screenshot shows a window titled "Password Generator" with a dark blue background. It contains two input fields with red text labels: "Enter some words:" and "Length of Password :". Below these fields are four buttons: "Generate" (green), "Copy 1", "Copy 2", and "Cancel" (all purple).

Validation for empty Inputs



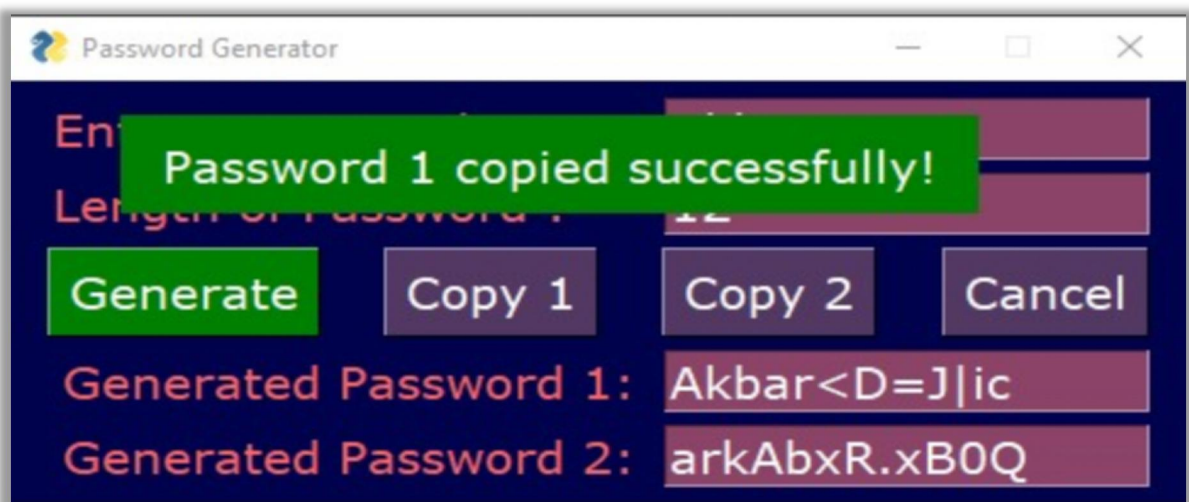
The screenshot shows the same "Password Generator" window, but with a red rectangular overlay in the center containing the text "Enter Valid Values" in white. This overlay is positioned over the input fields, indicating a validation error for empty inputs.

Generated Random Passwords

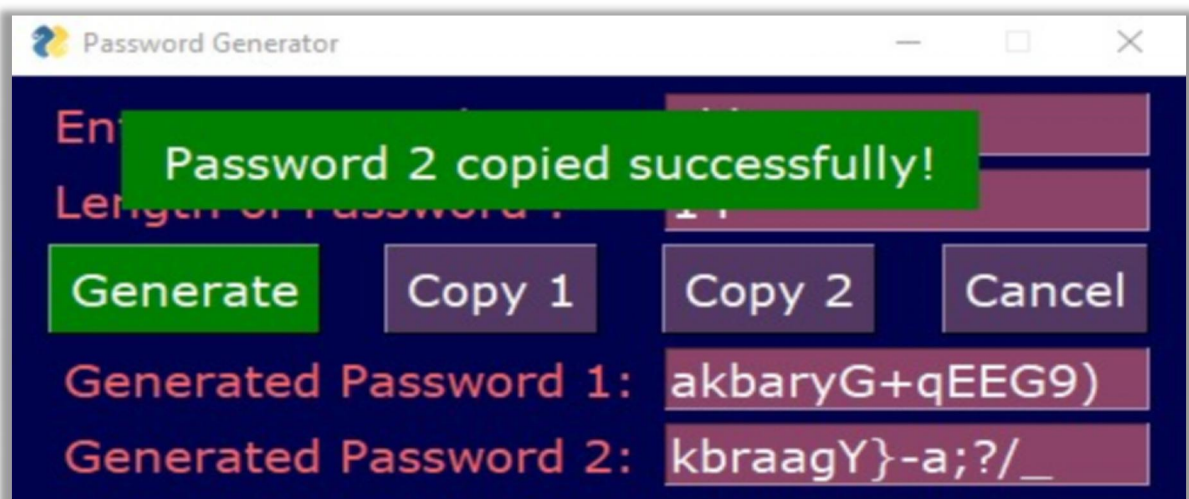


The screenshot shows a window titled "Password Generator". It has a dark blue background with red text for labels. The "Enter some words:" field contains "Akbar". The "Length of Password :" field contains "12". Below these are four buttons: "Generate" (green), "Copy 1" (purple), "Copy 2" (purple), and "Cancel" (purple). At the bottom, two generated passwords are shown: "Generated Password 1: Akbar<D=J|ic" and "Generated Password 2: arkAbxR.xB0Q".

Generated Password Copied



This screenshot is identical to the previous one, but with a green banner overlaying the top part of the form that reads "Password 1 copied successfully!". The "Generated Password 1" field still shows "Akbar<D=J|ic".



This screenshot is identical to the previous ones, but with a green banner overlaying the top part of the form that reads "Password 2 copied successfully!". The "Generated Password 1" field now shows "akbaryG+qEEG9)" and the "Generated Password 2" field shows "kbraagY}-a;?/_".

PROJECT ANALYSIS

The **py-password-generator** project is a Python-based password generator that allows users to generate strong and secure passwords based on their input and preferences. The project utilizes the **PySimpleGUI** library to create a graphical user interface (GUI) that simplifies the password generation process.

The code begins with importing necessary modules, including `string`, `random`, **PySimpleGUI**, and `pyperclip`. These modules provide functionalities for generating random characters, creating the GUI, and copying passwords to the clipboard.

The password generation logic is implemented through two functions: **generate_password1** and **generate_password2**.

generate_password1 takes user-provided words and counts the length of the combined words. It then generates additional random characters for complexity. If the word length exceeds the desired password length, the function shuffles the characters before appending the random characters. Finally, it returns the generated password.

generate_password2 follows a similar approach but shuffles the words provided by the user before counting the length and generating the password. This ensures that the resulting password has a randomized arrangement of the input words.

The **PySimpleGUI** library is used to create an intuitive GUI interface. The GUI includes input fields for users to enter words and the desired password length. It also provides buttons for generating passwords and copying them to the clipboard. The generated passwords are displayed in separate text areas within the GUI.

The code includes event handling within an event loop. It listens for user interactions, such as button clicks, and responds accordingly. When the **"Generate"** button is clicked, the code retrieves user input and calls the password generation functions. It then updates the GUI to display the generated passwords.

Error handling is also implemented. If the user enters invalid input or encounters an error, appropriate error messages are displayed through pop-up windows. The GUI ensures a smooth user experience by handling potential errors gracefully.

The code concludes by closing the GUI window when the user closes the application or clicks the "**Cancel**" button.

In terms of potential improvements, the project could benefit from incorporating input validation to ensure that users enter valid input for words and password length. Additionally, the code could be further modularized to enhance code readability and maintainability.

Overall, the "**py-password-generator**" project offers a user-friendly and efficient solution for generating strong passwords. The utilization of **PySimpleGUI** simplifies the user interaction, while the password generation functions provide flexibility and randomness to create secure passwords.

TASK SHEET:

1. Zainab Musharaf.
2. Akbar Amin.

Member Name	Task Done
Zainab Musharaf	Documentation
Akbar Amin	Design and Coding

FINAL CHECK:

Particular's	Checked
Design	Done
Validation	Done
Documentation	Done