

Project: 2

Project 2: Phishing Website Detection Tool

Statement:

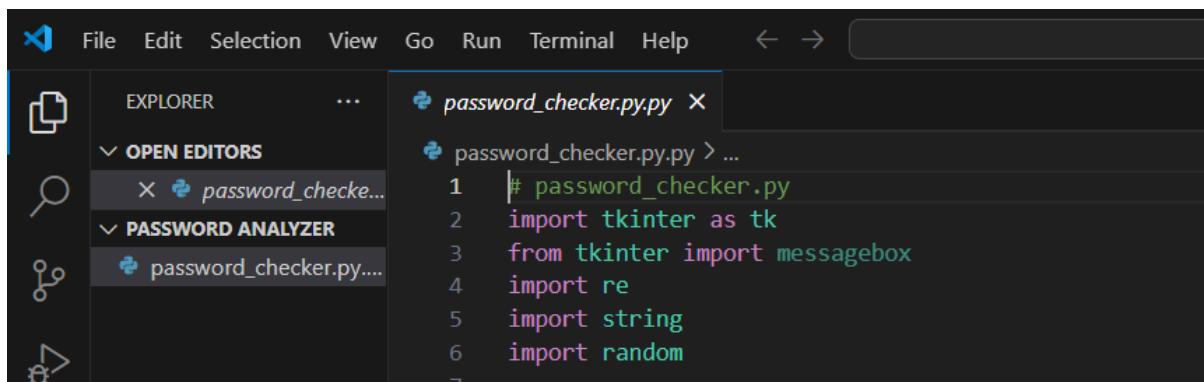
Weak passwords are a major reason for data breaches.

Objective:

Develop a GUI-based tool that analyzes passwords and provides strength feedback and improvement suggestions.

Explanation:

- ◆ Importing Libraries



```
# password_checker.py
import tkinter as tk
from tkinter import messagebox
import re
import string
import random
```

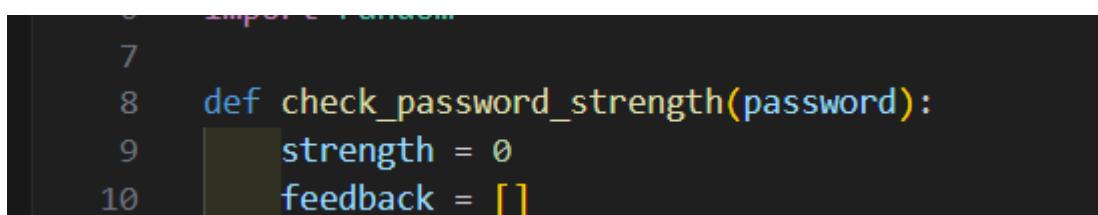
tkinter: The main GUI library used to create the window, buttons, text input, etc.

re: Python's regex module for checking password patterns.

string: Used to build a set of characters for strong password generation.

random: To randomly choose characters for generating suggestions.

- ◆ Password Strength Checker Function



```
import random
import re
import string
```

```
def check_password_strength(password):
    strength = 0
    feedback = []
```

strength: A score (0–5) based on how many rules the password meets.

feedback: A list of comments to help improve the password.

◆ Rules and Evaluation

```

11
12     if len(password) >= 8:
13         strength += 1
14     else:
15         feedback.append("Password should be at least 8 characters long.")
16

```

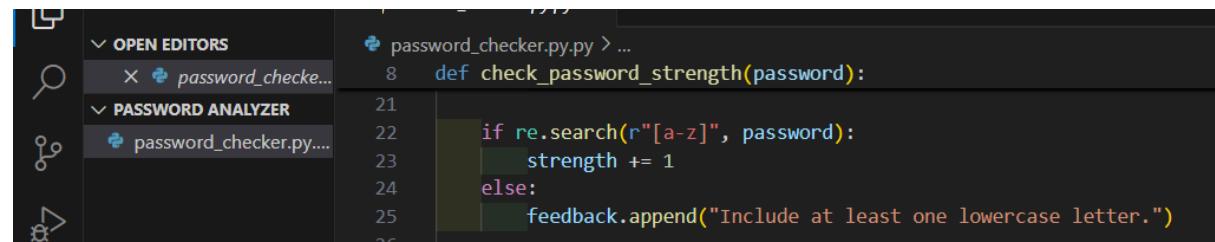
Rule 1: Password must be at least 8 characters.

```

16
17     if re.search(r"[A-Z]", password):
18         strength += 1
19     else:
20         feedback.append("Include at least one uppercase letter.")

```

Rule 2: Must have at least one uppercase letter.



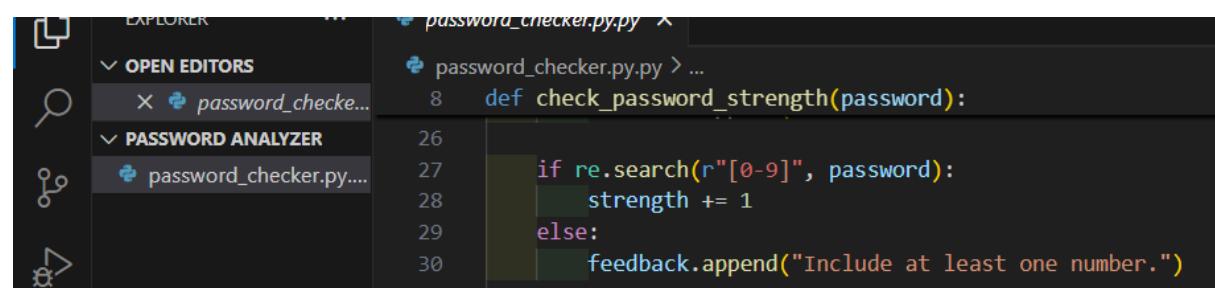
The screenshot shows the VS Code interface with the Explorer sidebar open. Under 'OPEN EDITORS', there is a file named 'password_checker.py'. In the main editor area, the following code is shown:

```

password_checker.py > ...
8 def check_password_strength(password):
21     if re.search(r"[a-z]", password):
22         strength += 1
23     else:
24         feedback.append("Include at least one lowercase letter.")
25
26

```

Rule 3: Must have at least one lowercase letter.



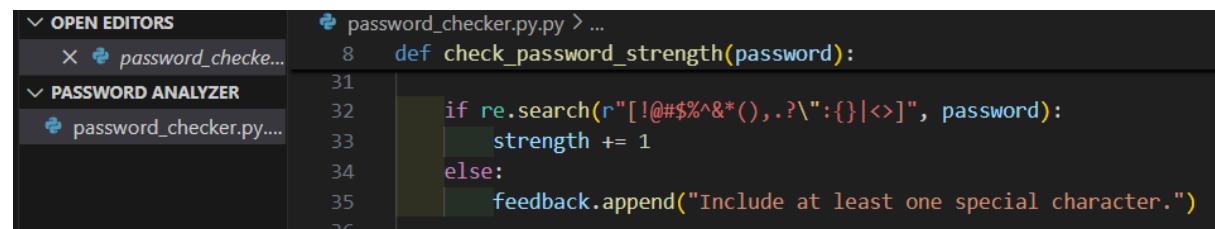
The screenshot shows the VS Code interface with the Explorer sidebar open. Under 'OPEN EDITORS', there is a file named 'password_checker.py'. In the main editor area, the following code is shown:

```

password_checker.py > ...
8 def check_password_strength(password):
26
27     if re.search(r"[0-9]", password):
28         strength += 1
29     else:
30         feedback.append("Include at least one number.")
31
32

```

Rule 4: Must include at least one digit.



The screenshot shows the VS Code interface with the Explorer sidebar open. Under 'OPEN EDITORS', there is a file named 'password_checker.py'. In the main editor area, the following code is shown:

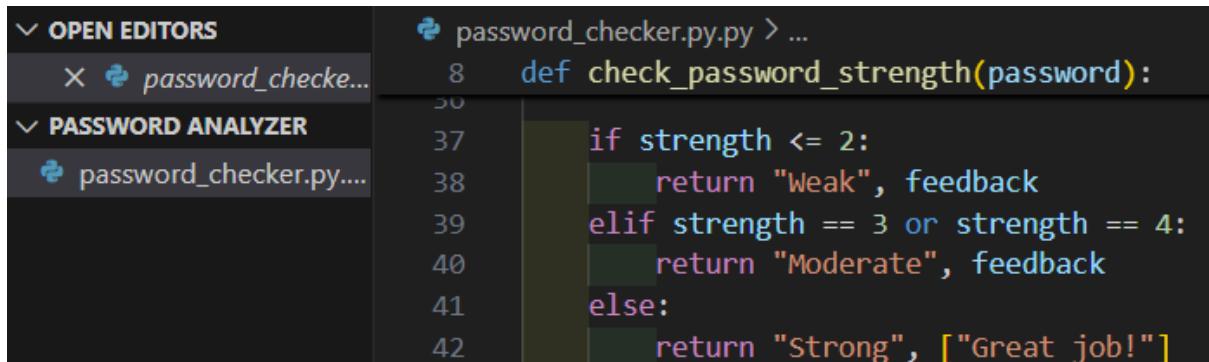
```

password_checker.py > ...
8 def check_password_strength(password):
31
32     if re.search(r"[@#$%^&*(),.?\"':{}|<>]", password):
33         strength += 1
34     else:
35         feedback.append("Include at least one special character.")
36

```

Rule 5: Must include **special characters**.

◆ **Return Strength Label**



```

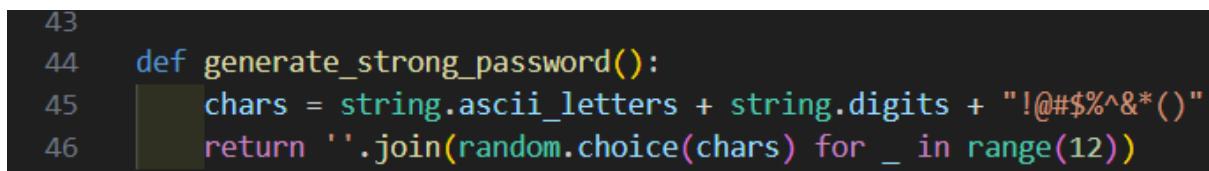
    password_checker.py.py > ...
8     def check_password_strength(password):
30
37         if strength <= 2:
38             return "Weak", feedback
39         elif strength == 3 or strength == 4:
40             return "Moderate", feedback
41         else:
42             return "Strong", ["Great job!"]

```

Depending on the score, it returns:

- "Weak" (0–2 points)
- "Moderate" (3–4 points)
- "Strong" (5 points)

◆ **Password Suggestion Function**



```

43
44     def generate_strong_password():
45         chars = string.ascii_letters + string.digits + "!@#$%^&*()"
46         return ''.join(random.choice(chars) for _ in range(12))

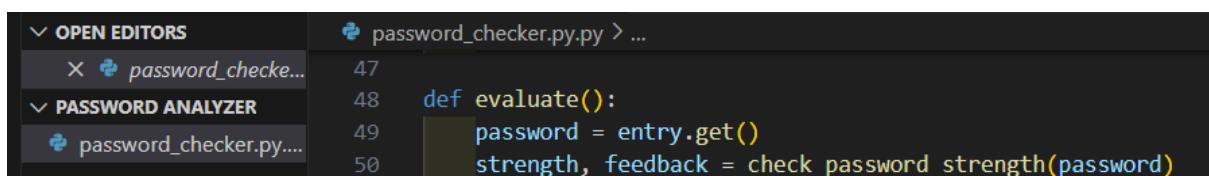
```

This function creates a 12-character strong password using:

- Uppercase and lowercase letters
- Numbers
- Special symbols

`random.choice()` picks one character at a time

◆ **Main Evaluation Logic (Called on Button Click)**



```

    password_checker.py.py > ...
47
48     def evaluate():
49         password = entry.get()
50         strength, feedback = check_password_strength(password)

```

Grabs the password from the input field.

Calls the checker to get the strength and feedback.

```
51 |     result_label.config(text=f"Strength: {strength}")
```

Displays the strength rating (Weak, Moderate, Strong).

```
52     feedback_text.delete(1.0, tk.END)
53     for item in feedback:
54         feedback_text.insert(tk.END, f"- {item}\n")
```

Clears the text box, then shows feedback messages line by line.



The screenshot shows a code editor with a sidebar containing a navigation bar and a search bar. The main area displays a Python script named `password_checker.py`. The code defines a class `PASSWORD ANALYZER` with a method `evaluate()`. Inside `evaluate()`, there is a conditional block that checks the password strength and generates a suggestion if it's not strong.

```
● 🔍 password_check... 48 def evaluate():
>Password ANALYZER 55     if strength != "Strong":
  🔍 password_checker.py... 56         suggestion = generate_strong_password()
  🔍 password_checker.py... 57         feedback_text.insert(tk.END, f"\nSuggested Password: {suggestion}")
```

If the password isn't strong, generate and display a suggested one.

◆ 7. GUI Layout



The screenshot shows a code editor interface with a dark theme. The left sidebar lists files under 'PASSWORD ANALYZER': 'password_checker.py...' (marked with a blue file icon) and 'password_checker.py....' (marked with a blue folder icon). The main area displays the following Python code:

```
58
59 # GUI Setup
60 root = tk.Tk()
61 root.title("Password Strength Checker")
62
```

Creates the main application window and sets the title.

```
63     tk.Label(root, text="Enter your password:").pack()  
64     entry = tk.Entry(root, width=40, show="*")  
65     entry.pack()
```

Adds a label and a password input field (show="*" hides characters).

```
67     tk.Button(root, text="Check Strength", command=evaluate).pack(pady=10)
```

A button that runs the evaluate() function on click.

```
68 result_label = tk.Label(root, text="")
69 result_label.pack()
```

Label to display strength result (updates dynamically).

```
71     feedback_text = tk.Text(root, height=10, width=50)
72     feedback_text.pack()
```

Text box to show detailed feedback and suggestions.

```
74     root.mainloop()
```

Starts the Tkinter GUI event loop (keeps the window open and interactive).

Is a fully functional and educational GUI tool that:

- Checks real-time password strength using 5 key security rules
- Provides feedback and best-practice suggestions
- Offers strong, randomly generated passwords
- Uses clean modular Python code with Tkinter, Regex, and string