Name: HAMMAD TAHIR

Intern ID: TN/IN01/PY/006

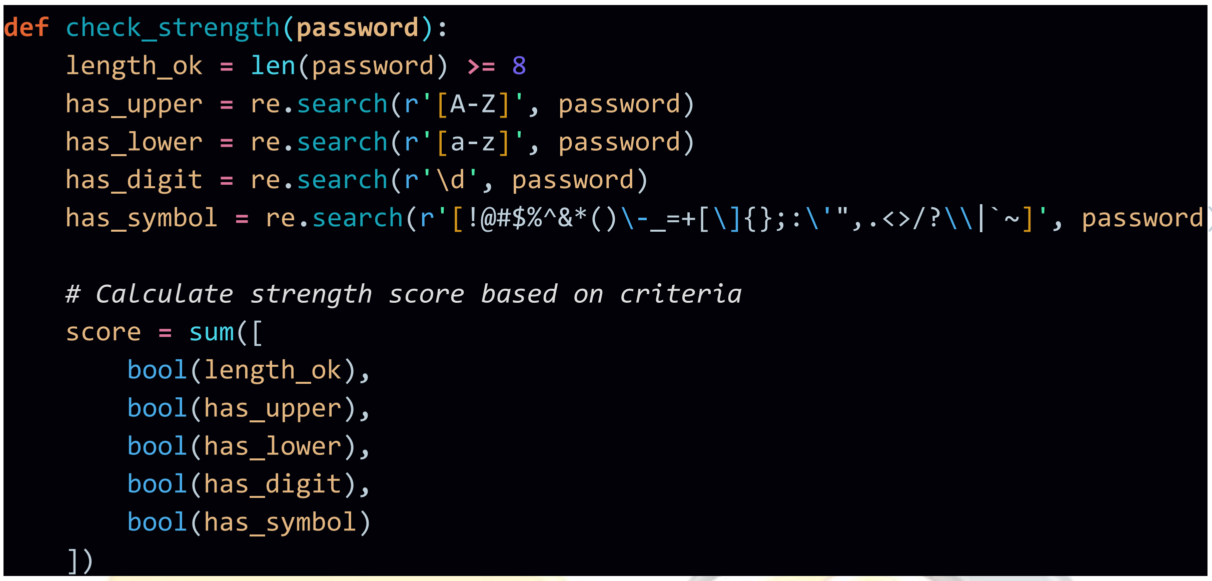
Email ID : BEINGHAMMAD345@GMAIL.COM

Internship Domain : PYTHON

Project Report

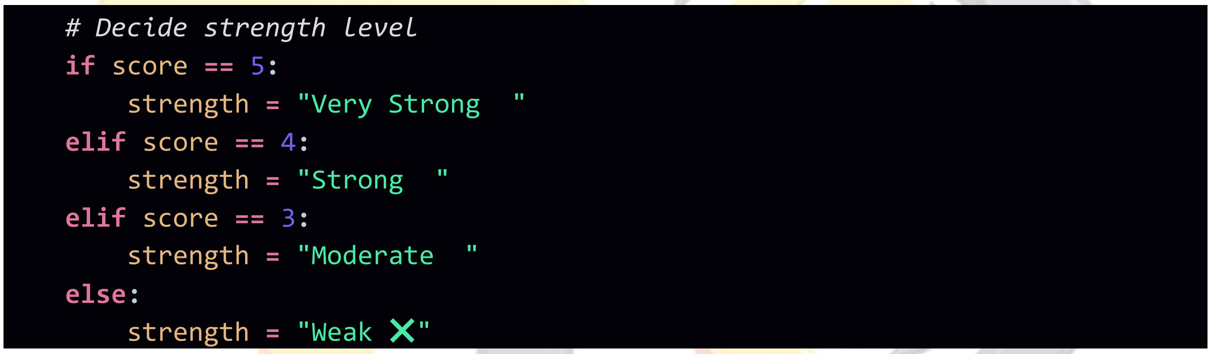
”Password Strength Checker And Breach Detection Tool”

Project Code for password making tool :

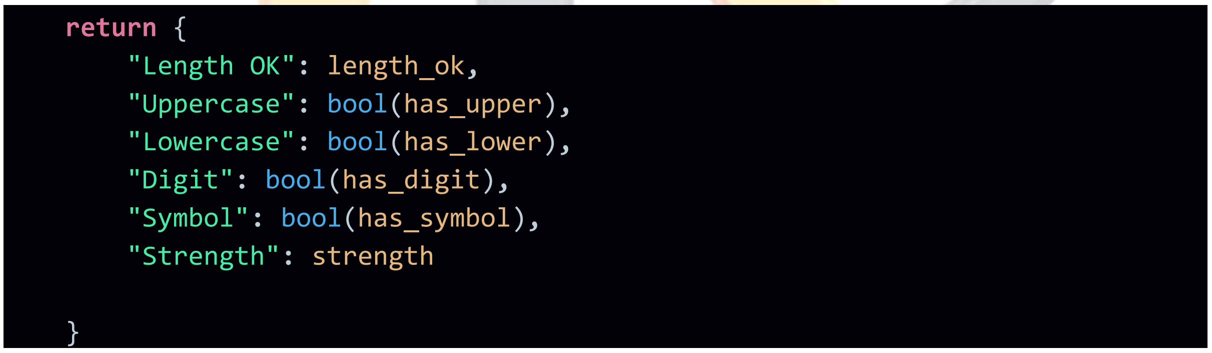


```python
def check_strength(password):
    length_ok = len(password) >= 8
    has_upper = re.search(r'[A-Z]', password)
    has_lower = re.search(r'[a-z]', password)
    has_digit = re.search(r'\d', password)
    has_symbol = re.search(r'[!@#$%^&*()\-_=+[\]{};:\'",.<>/?\\|`~]', password)

    # Calculate strength score based on criteria
    score = sum([
        bool(length_ok),
        bool(has_upper),
        bool(has_lower),
        bool(has_digit),
        bool(has_symbol)
    ])
```



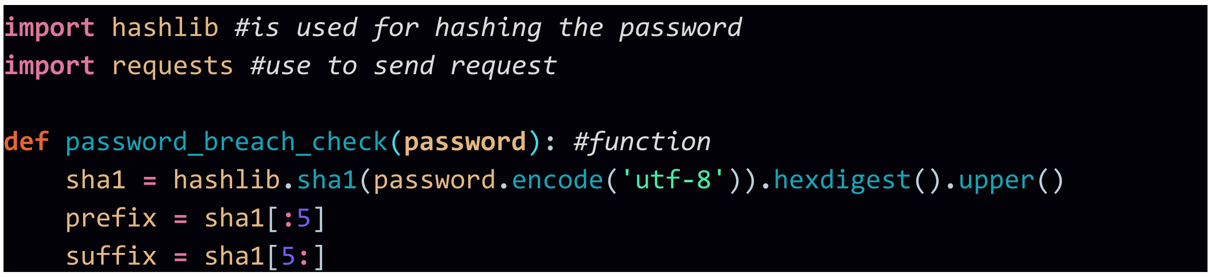
```python
    # Decide strength level
    if score == 5:
        strength = "Very Strong  "
    elif score == 4:
        strength = "Strong  "
    elif score == 3:
        strength = "Moderate  "
    else:
        strength = "Weak ❌"
```



```python
    return {
        "Length OK": length_ok,
        "Uppercase": bool(has_upper),
        "Lowercase": bool(has_lower),
        "Digit": bool(has_digit),
        "Symbol": bool(has_symbol),
        "Strength": strength

    }
```

Code For Breach Detection:



```python
import hashlib #is used for hashing the password
import requests #use to send request

def password_breach_check(password): #function
    sha1 = hashlib.sha1(password.encode('utf-8')).hexdigest().upper()
    prefix = sha1[:5]
    suffix = sha1[5:]
    url = f'https://api.pwnedpasswords.com/range/{prefix}'
    response = requests.get(url)
    if response.status_code != 200:
        raise RuntimeError(f"API Error: {response.status_code}")

    hashes = response.text.splitlines() #it is uses to split the lines we got
from the api
```

```
    for line in hashes: #for loop used to see if password leaked or not
        h_suffix, count = line.split(':')
        if h_suffix == suffix:
            return int(count)
```

```
    return 0
```

```
user_password = input("Enter the password: ")
breaches = password_breach_check(user_password)
```

```
if breaches:
    print(f"This password was found in {breaches} breaches! Please change it.")
else:
     print("This password was NOT found in any known breaches.")
```

<u>Code for Gradio app Interface:</u>

```
import re
import hashlib
import requests
import gradio as gr

# --- Phase 1: Password Strength Check ---
def check_password_strength(password):
    strength_report = {
        "Length OK": len(password) >= 8,
        "Uppercase": bool(re.search(r"[A-Z]", password)),
        "Lowercase": bool(re.search(r"[a-z]", password)),
        "Digit": bool(re.search(r"\d", password)),
        "Symbol": bool(re.search(r"[!@#$%^&*(),.?\":{}|<>]", password))
    }
```

```
    score = sum(strength_report.values())
```

```
    if score == 5:
        strength = "Very Strong  "
    elif score >= 4:
        strength = "Strong ✅"
    elif score == 3:
        strength = "Moderate  "
    else:
        strength = "Weak ❌"
```

```python
    strength_report["Strength"] = strength
    return strength_report

# --- Phase 2: Breach Check via HIBP ---
def check_password_breach(password):
    sha1pass = hashlib.sha1(password.encode('utf-8')).hexdigest().upper()
    prefix, suffix = sha1pass[:5], sha1pass[5:]

    url = f"https://api.pwnedpasswords.com/range/{prefix}"
    res = requests.get(url)

    if res.status_code != 200:
        return "Error contacting breach API"

    hashes = (line.split(':') for line in res.text.splitlines())
    for h, count in hashes:
        if h == suffix:
            return int(count)

    return 0

# --- Combined Gradio App ---
def analyze_password(password):
    if not password:
        return "❌ Please enter a password", None

    strength_info = check_password_strength(password)
    breaches = check_password_breach(password)

    strength_output = "\n".join([f"{k}: {v}" for k, v in
strength_info.items()])

    if breaches == "Error contacting breach API":
        breach_msg = "  Could not reach HaveIBeenPwned API"
    elif breaches > 0:
        breach_msg = f"❌ Password found in {breaches} breaches! Change it!"
    else:
        breach_msg = "✅ Password not found in known breaches"

    return strength_output, breach_msg

# --- Gradio Interface ---
with gr.Blocks(theme=gr.themes.Soft()) as app:
    gr.Markdown("#   Password Strength & Breach Detection Tool")
    gr.Markdown("Enter your password below to analyze its strength and check
if it has been exposed in data breaches.")

    with gr.Row():
        password_input = gr.Textbox(type="password", label="Enter Password",
placeholder="Your password...")
```

```
    with gr.Row():
        analyze_btn = gr.Button("  Analyze Password")
```

```
    with gr.Row():
        strength_output = gr.Textbox(label="  Strength Analysis", lines=6)
        breach_output = gr.Textbox(label="  Breach Check Result", lines=2)
```

```
    analyze_btn.click(analyze_password, inputs=password_input,
outputs=[strength_output, breach_output])
```

```
# --- Launch App ---
app.launch()
```

Overall Project :

```
#   Password Strength & Breach Detection Tool
# Author: Hammad Tahir
# Description: A CLI tool to evaluate password strength and check if it's
leaked in any known data breaches.

import re               # For pattern matching (e.g., checking for digits,
symbols)
import hashlib          # For hashing password using SHA-1
import requests         # For making API requests to HaveIBeenPwned
```

```
# ------------------------------
# Function to check password strength
# ------------------------------
def check_strength(password):
    length_ok = len(password) >= 8
    has_upper = re.search(r'[A-Z]', password)
    has_lower = re.search(r'[a-z]', password)
    has_digit = re.search(r'\d', password)
    has_symbol = re.search(r'[!@#$%^&*()\-_=+[\]{};:\'",.<>/?\\|`~]', password)
```

```
    # Calculate strength score based on criteria
    score = sum([
        bool(length_ok),
        bool(has_upper),
        bool(has_lower),
        bool(has_digit),
        bool(has_symbol)
    ])
```

```
    # Decide strength level
    if score == 5:
        strength = "Very Strong  "
```

```python
    elif score == 4:
        strength = "Strong  "
    elif score == 3:
        strength = "Moderate  "
    else:
        strength = "Weak ❌"
```

```python
    return {
        "Length OK": length_ok,
        "Uppercase": bool(has_upper),
        "Lowercase": bool(has_lower),
        "Digit": bool(has_digit),
        "Symbol": bool(has_symbol),
        "Strength": strength
    }
```

```python
# -------------------------------
# Function to check password breach using HaveIBeenPwned API
# -------------------------------
def check_password_breach(password):
    # Convert password to SHA-1 hash
    sha1 = hashlib.sha1(password.encode('utf-8')).hexdigest().upper()
    prefix = sha1[:5]     # First 5 characters of the hash
    suffix = sha1[5:]     # Remaining characters
```

```python
    # API URL for k-anonymity
    url = f"https://api.pwnedpasswords.com/range/{prefix}"
    res = requests.get(url)
```

```python
    # Check if API call was successful
    if res.status_code != 200:
        raise RuntimeError(f"API error: {res.status_code}")
```

```python
    # Process the response to find match
    hashes = res.text.splitlines()
    for line in hashes:
        h_suffix, count = line.split(":")
        if h_suffix == suffix:
            return int(count)  # Password found in breaches
```

```python
    return 0  # Password not found in breaches
```

```python
# -------------------------------
# Main Program Starts Here
# -------------------------------
print("  Password Strength & Breach Detection Tool  ")
password = input("Enter your password: ")
```

```python
# 1. Check strength
strength_report = check_strength(password)
```

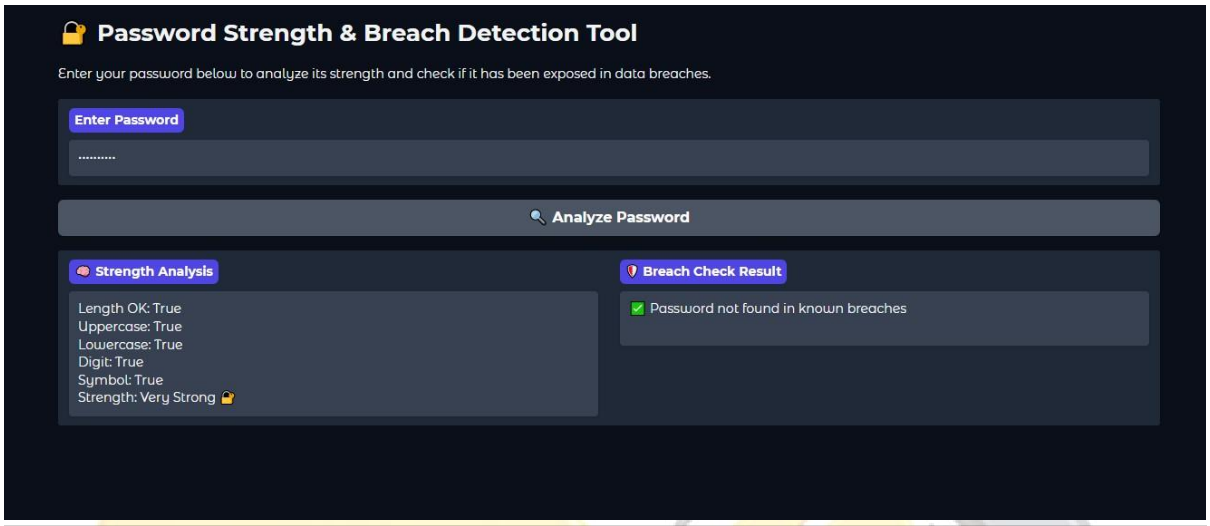
```python
# 2. Check breach
breach_count = check_password_breach(password)
```

```python
# ------------------------------
# Display Strength Report
# ------------------------------
print("\n  Password Analysis:")
for key, value in strength_report.items():
    print(f"{key}: {value}")
```

```python
print(f"\n  Overall Strength: {strength_report['Strength']}")
```

```python
# ------------------------------
# Display Breach Result
# ------------------------------
if breach_count:
    print(f"❌ WARNING: This password has been found in {breach_count} data breaches!")
    print("  Please avoid using this password.")
else:
    print("✅ This password was NOT found in any known breaches. Looks safe!")
```

Project Output :

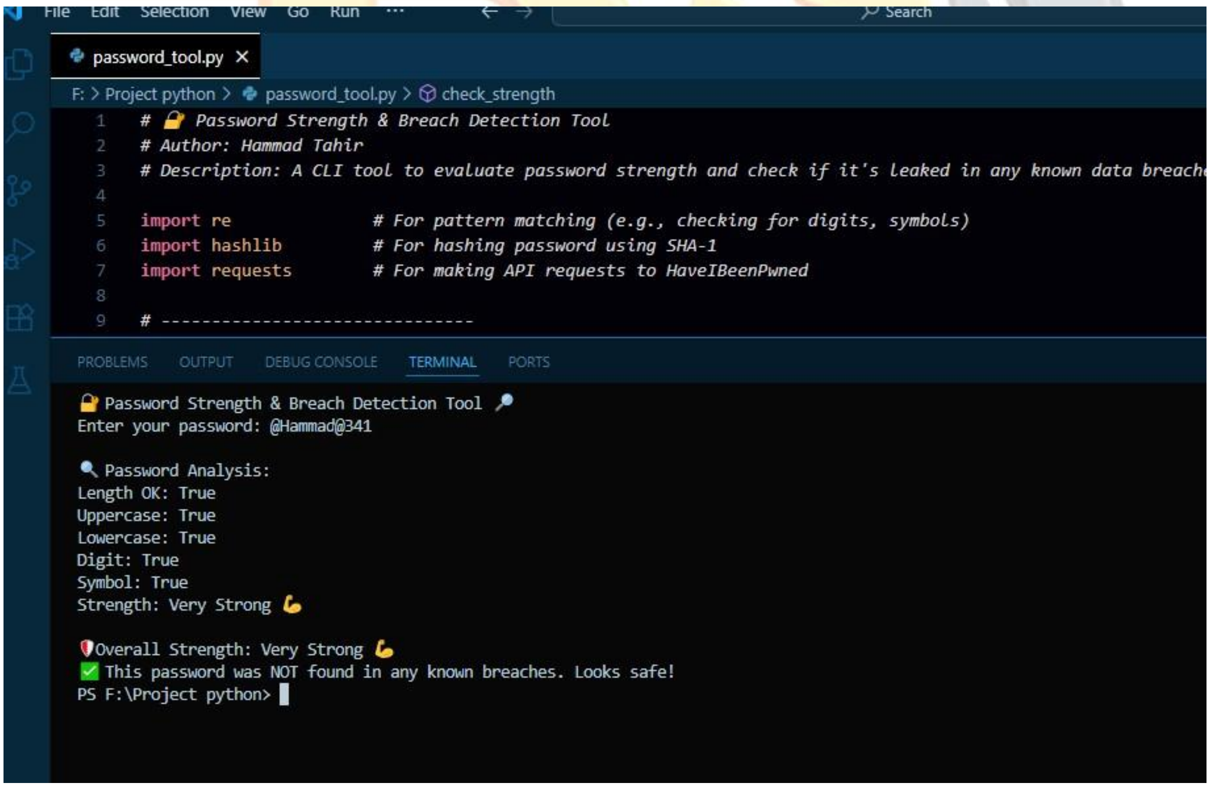


🔐 **Password Strength & Breach Detection Tool**

Enter your password below to analyze its strength and check if it has been exposed in data breaches.

```
Length OK: True
Uppercase: True
Lowercase: True
Digit: True
Symbol: True
Strength: Very Strong 🔐
```

```
✅ Password not found in known breaches
```



File Edit Selection View Go Run ...
Search
password_tool.py
F: > Project python > password_tool.py > check_strength
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```python
# 🔐 Password Strength & Breach Detection Tool
# Author: Hammad Tahir
# Description: A CLI tool to evaluate password strength and check if it's leaked in any known data breach

import re              # For pattern matching (e.g., checking for digits, symbols)
import hashlib         # For hashing password using SHA-1
import requests        # For making API requests to HaveIBeenPwned

# ------------------------------
```

```
🔐 Password Strength & Breach Detection Tool 🔎
Enter your password: @Hammad@341

🔍 Password Analysis:
Length OK: True
Uppercase: True
Lowercase: True
Digit: True
Symbol: True
Strength: Very Strong 💪

🛡️Overall Strength: Very Strong 💪
✅ This password was NOT found in any known breaches. Looks safe!
PS F:\Project python> █
```