RANDOM PASSWORD GENERATOR

1. PROJECT OVERVIEW :

A Random Password Generator is a Python-based application that automatically creates secure and unpredictable passwords. Passwords are essential for protecting user accounts, devices, and confidential information. Instead of manually creating passwords, which can be weak or repetitive, this project generates strong combinations of characters (letters, digits, symbols) based on user preferences.

The project allows the user to specify the password length and optional custom settings such as including uppercase letters, lowercase letters, numbers, and special characters. The program then randomly selects characters from these sets using Python's random and string modules and displays the final password.

2. SIGNIFICANCE OF THE PROJECT:

a) Security Improvement

Weak passwords are easy to guess or crack. This tool generates strong and complex passwords, making it difficult for hackers to break them.

(b) Automation

It saves time by automatically producing passwords rather than creating them manually.

(c) User-Friendly

Even users with no technical knowledge can generate secure passwords within seconds.

(d) High Customization

The user can choose password length and components, making the tool suitable for websites, personal accounts, email IDs, or system login.

(e) Practical Real-World Use

Random password generation is widely used in:

- Banking applications
- Online account creation
- System security
- Password managers

Thus, this project is simple but highly useful in real-world cyber security.

3. FUNCTIONALITY AND FEATURES OF PROJECT:

Character Set Preparation:

It creates a combined set of characters containing:

- Uppercase letters (A–Z)
- Lowercase letters (a–z)
- Digits (0–9)
- Special symbols (!@#$%^&* etc.)

Random Selection:

Using the random.choice() or random.sample() method, the program picks characters randomly from the set.

Password Construction:

The selected characters are joined together to form the final password.

Output:

The secure generated password is displayed to the user.

4. PYTHON CODE:

```
import random

import string

print("----- Random Password Generator -----")

# Step 1: User enters desired password length
```

```
length = int(input("Enter password length: "))
# Step 2: Options for character types
use_upper = input("Include uppercase letters? (y/n): ").lower() == 'y'
use_lower = input("Include lowercase letters? (y/n): ").lower() == 'y'
use_digits = input("Include digits? (y/n): ").lower() == 'y'
use_symbols = input("Include special characters? (y/n): ").lower() == 'y'
# Step 3: Create character pool based on user choices
characters = ""
if use_upper:
    characters += string.ascii_uppercase    # A–Z
if use_lower:
    characters += string.ascii_lowercase    # a–z
if use_digits:
    characters += string.digits             # 0–9
if use_symbols:
    characters += string.punctuation        # special characters
# Step 4: If user selected nothing
if characters == "":
    print("Error! No character type selected. Using all types by default.")
    characters = string.ascii_letters + string.digits + string.punctuation
# Step 5: Generate password
password = ''.join(random.choice(characters) for _ in range(length))
# Step 6: Display result
print("\nGenerated Strong Password:", password)
```

5. CODE EXPLANATION :

1.Importing Modules

- random → selects random characters
- string → contains predefined sets of characters like ascii_letters, digits, punctuation

2.User Input

- The program takes the desired password length from the user.
- Character Set Creation
- Depending on the user's choices, the program adds character groups to the characters string.
- Password Generation

''.join(random.choice(characters) for _ in range(length))

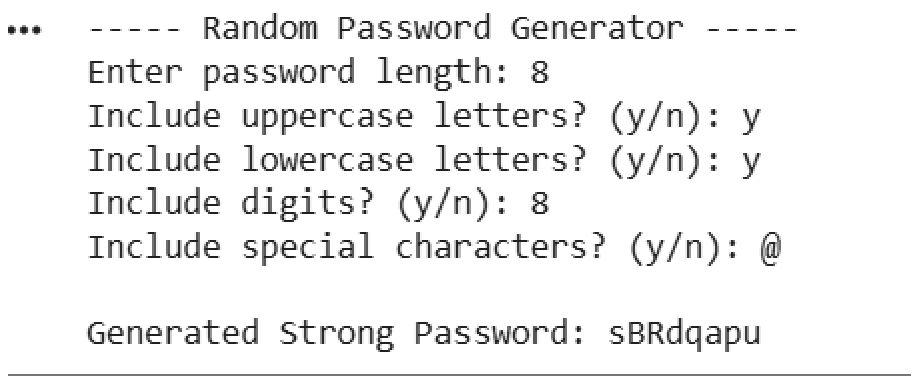
3.This line:

- Chooses a random character length times
- Joins them to form a complete password

4.Final Output

The program prints the generated password.

6. SAMPLE OUTPUT:



```
•••   ----- Random Password Generator -----
      Enter password length: 8
      Include uppercase letters? (y/n): y
      Include lowercase letters? (y/n): y
      Include digits? (y/n): 8
      Include special characters? (y/n): @

      Generated Strong Password: sBRdqapu
```

7. SUMMARY :

This project demonstrates:

- Python’s ability to automate daily tasks
- Importance of secure password creation
- Working with important modules like random and string

It is a perfect mini-project for beginners in Python and cybersecurity.