1. **Problem Statement -**

Generate random passwords based on user-defined criteria (length)

1. **Motivation-**

**Improving Cybersecurity**

* Passwords are a critical aspect of digital security. Weak or predictable passwords are a common vulnerability in systems. A password generator to help users create strong, secure passwords that are resistant to attacks like brute force or dictionary attacks.

**Learning Opportunity**

* Writing a password generator is a practical coding project that helps:
  + Learn about randomness and pseudorandom number generators.
  + Understand string manipulation and character sets.
  + Gain experience in implementing user-friendly tools.

1. **Objective –**

The main goal of creating a random password generator is to boost digital security. It's designed to help people come up with strong, unpredictable passwords. These robust passwords are crucial for safeguarding sensitive information, accounts, and systems from unauthorized access. Essentially, a random password generator acts as your digital shield, ensuring that your online presence remains secure and protected from potential threats. Below are some specific objectives:

* **Strengthen Security-**Generate passwords that are resistant to attacks, such as brute force, dictionary attacks, or social engineering.
* **Automate and Simplify-**Save time and effort by automating the process of creating secure passwords.
* **Reduce Security Risks -**Mitigate risks associated with weak or reused passwords by enabling easy generation of unique passwords for every application.

1. **Methodology of Implementation**

**Explanation-**

The program starts with importing 2 libraries random (used for shuffling) and strings (collection of all letters, numbers and punctuation). Now 4 lists are created **s1,s2,s3,s4** comprising lowercase letters, uppercase letters, digits and punctuation respectively. Then the user must input the length of required password. The length must be greater than equal to 8 else the output will be **"The length should be greater than 8".**If the length is greater than equal to 8 the program now shuffles all 4 lists created before then it will calculate the number of letters and digits , punctuation in the password(part 1 and part 2).Then an empty list is created named result which will store the password . Now the code will iterate over the calculated values of part1 and will add upper and lowercase letters in the list result same will be the process for digits and punctuation. Now since the list is ready with all the appended digits and letters. The code will shuffle the list again because it was ordered. The last step is to join the list result using join function and then A diagram with text and images

AI-generated content may be incorrect.print it.

1. **Hardware/Software Used-**

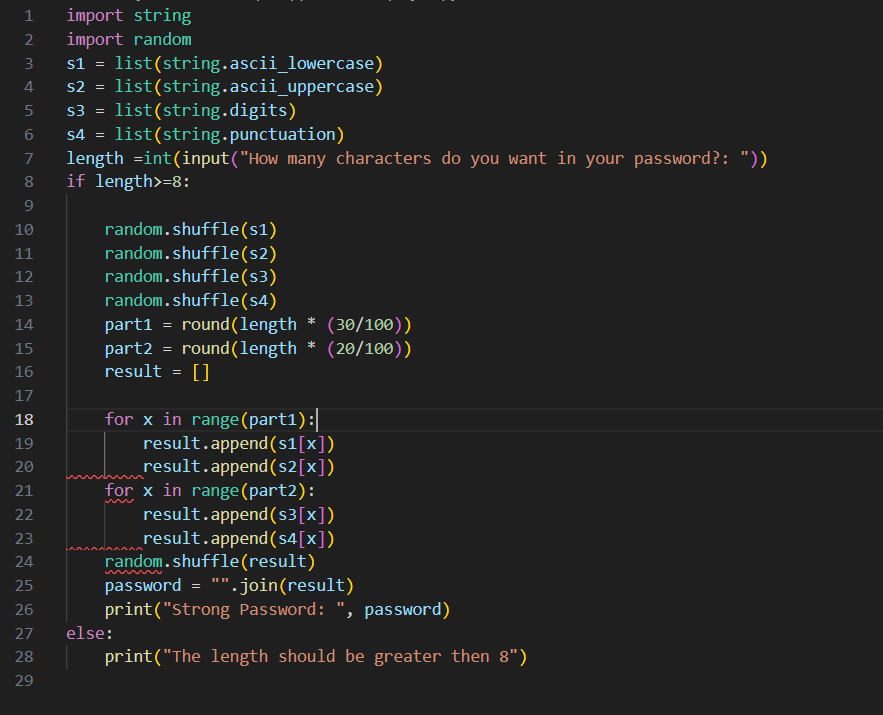
**Hardware Used:** Laptop

**Software Used:** VisualStudioCode

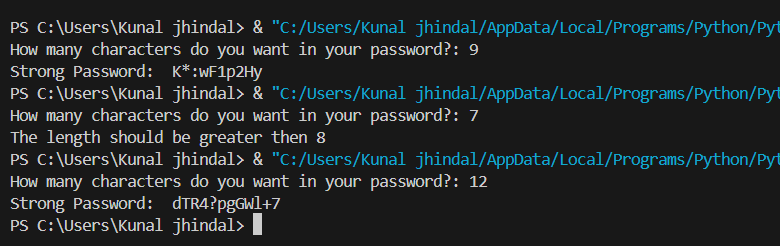
1. **Data Set Description-**

N/A

1. **Executable Code-**



1. **Result Analysis with Output Screen Shot-**



1. **Learning Outcome-**

**1. Understanding Randomness in Programming**

* Learn how to use libraries or modules (e.g., random in Python) to generate random numbers or characters.
* Understand the difference between pseudo-random and true randomness, and why pseudo-randomness is often used in software.

**2. String Manipulation**

* Practice operations like concatenation, slicing, and iteration over strings.
* Understand how to work with ASCII values and convert between characters and their numeric representations if needed.

**3. Modular Programming**

* Break down the task into smaller functions, e.g., one function for generating random characters and another for assembling them into a password.
* Gain experience in organizing code logically for readability and reusability.

**4. Working with Character Sets**

* Learn about character sets such as uppercase letters, lowercase letters, digits, and special characters.
* Use these sets to ensure password diversity and strength.

## References

1. D. V. Lindberg and H. K. H. Lee, “Optimization under constraints by applying an

asymmetric entropy measure,” *J. Comput. Graph. Statist.*, vol. 24, no. 2, pp. 379–393, Jun. 2015, doi: 10.1080/10618600.2014.901225.

1. B. Rieder, *Engines of Order: A Mechanology of Algorithmic Techniques*. Amsterdam, Netherlands: Amsterdam Univ. Press, 2020.