**Python Password Generator Documentation**

**Overview**

This script generates strong passwords of specified lengths. Each password contains a mix of lowercase letters, uppercase letters, digits, and symbols, ensuring high security. The user can specify how many passwords to generate and the length of each password.

## Code Structure

The script is divided into three main parts:

1. **Importing Required Modules**
2. **Defining Functions**
3. **Main Script Execution**

**1. Importing Required Modules**

**Python Code:**

import random

import string

* random: This module implements pseudo-random number generators for various distributions.
* string: This module contains various string constants (e.g., ascii\_lowercase, ascii\_uppercase, digits, punctuation) which are used to build the character sets for password generation.

**2. Defining Functions**

**a. generate\_password(length)**

**Python Code:**

def generate\_password(length):

"""Generate a strong password with the given length."""

if length < 3:

raise ValueError("Password length should be at least 3 characters")

# Define possible characters for the password

lower = string.ascii\_lowercase

upper = string.ascii\_uppercase

digits = string.digits

symbols = string.punctuation

# Ensure the password has at least one of each type of character

all\_characters = lower + upper + digits + symbols

password = [

random.choice(lower),

random.choice(upper),

random.choice(digits)

]

if length > 3:

password += random.choices(all\_characters, k=length - 3)

# Shuffle the password list to avoid predictable patterns

random.shuffle(password)

# Convert list to string and return the password

return ''.join(password)

**Purpose**: Generate a strong password of the specified length.

* **Parameters**:
  + length (int): The length of the password.
* **Returns**:
  + (str): A randomly generated password.
* **Raises**:
  + ValueError: If the length is less than 3.

**Steps**:

1. Check if the length is less than 3 and raise an error if true.
2. Define character sets for lowercase, uppercase, digits, and symbols.
3. Ensure the password has at least one lowercase, one uppercase letter, and one digit.
4. Add additional random characters to meet the specified length.
5. Shuffle the password to ensure randomness.
6. Convert the list of characters to a string and return it.

**b. get\_password\_length()**

**Python Code:**

def get\_password\_length():

"""Get a valid password length from the user."""

while True:

try:

length = int(input("Enter the length of the password: "))

if length < 3:

raise ValueError("Password length should be at least 3 characters")

return length

except ValueError as e:

print(e)

**Purpose**: Prompt the user to enter a valid password length.

* **Returns**:
  + (int): A valid password length.
* **Steps**:

1. Continuously prompt the user to enter a password length until a valid length (an integer greater than or equal to 3) is entered.
2. Return the valid length.

**3. Main Script Execution**

**Python Code:**

if \_\_name\_\_ == "\_\_main\_\_":

try:

num\_passwords = int(input("How many passwords do you want to generate? "))

except ValueError:

print("Please enter a valid number.")

exit()

print(f"Generating {num\_passwords} passwords")

passwords = []

for i in range(num\_passwords):

print(f"Generating Password #{i+1}:")

length = get\_password\_length()

try:

password = generate\_password(length)

passwords.append(password)

except ValueError as e:

print(e)

continue

for i, password in enumerate(passwords, 1):

print(f"Password #{i}: {password}")

**Purpose**: Execute the password generation process based on user inputs.

* **Steps**:

1. Prompt the user to enter the number of passwords to generate.
   * If the input is not a valid integer, print an error message and exit.
2. Print a message indicating the number of passwords to be generated.
3. For each password to be generated:
   * Print a message indicating the current password being generated.
   * Get a valid password length using get\_password\_length().
   * Generate the password using generate\_password(length) and add it to the list of passwords.
4. Print each generated password.

**CONCLUSION:**

This documentation covers the purpose, parameters, returns, and steps for each function, as well as the overall flow of the script. It should help you understand and modify the script as needed.

Top of Form

Bottom of Form