Password generator

INDEX

CHAPTER 1

ABSTRACT

Security is one of the most crucial parts of our lives. The importance of security is increasing day by day as most things are going online. Passwords come into light as we talk about security. In this project, we will create a password generator that helps us generate random and strong passwords quickly.
The best thing about creating our own password generator is that we can customize it as we like.

First, we will create a password generator that asks the length of the password and generates a random password containing digits, alphabets, and special characters. Next, we will improve it by asking the number of each type of character, like the number of digits, alphabets, and special characters. We try to guarantee that the program will include digits and special characters by asking the user to enter the number of digits, alphabets, and special characters they want. When the user enters the number of characters for each type, the program will include the respective number of character types into the password.

CHAPTER 2

REQUIREMENTS

System requirements:

1. Requirements at developer's end:

    Hardware: operating system: MS Windows XP or Ubuntu

    Language: Python

    RAM: 512 MB

    Hard disk: 5GB

    Software: Python 3

2. Requirements at client's end:

    Hardware

    Software

CHAPTER 3

DESIGN

1. Start

2. Store all the characters as a list. Use the sting module of Python to store them all.

3. Ask the user to enter the **length** of the password.

4. Shuffle the characters using **random.shuffle** method

5. Initialize an empty list to store the password.

6. Write a **loop** that iterates length times

- Pick a random character from all the characters using **random.choice** method
- Append the random character to the password

7. Shuffle the resultant password list to make it random.

8. Convert the password list to string using the **join** method.

9. Print the password.

CHAPTER 4

IMPLEMENTATION

During the implementation process, developers must write enough comments inside the code so that if anybody starts working on the code later, he/she can understand what has already been written. Writing good comments is very important as all other documents, no matter how good they are, will be lost eventually. Ten years after the initial work, you may find only that information which is present inside the code in the form of comments.

Development tools also play an important role in this phase of the project. Good development tools save a lot of time for the developers, as well as saving money in terms of improved productivity. The most important development tools for time saving are editors and debuggers. A good editor helps a developer to write code quickly. A good debugger helps make the written code operational in a short period. Before starting the coding process, you should spend some time choosing good development tools.

IMPLEMENTATION:

```
import string
import random

## characters to generate password from
alphabets = list(string.ascii_letters)
digits = list(string.digits)
special_characters = list("!@#$%^&*()")
characters = list(string.ascii_letters + string.digits + "!@#$%^&*()")

def generate_random_password():
        ## length of password from the user
        length = int(input("Enter password length: "))

## number of character types
        alphabets_count = int(input("Enter alphabets count in password: "))
        digits_count = int(input("Enter digits count in password: "))
        special_characters_count = int(input("Enter special characters count in password: "))

        characters_count = alphabets_count + digits_count + special_characters_count

        ## check the total length with characters sum count
        ## print not valid if the sum is greater than length
        if characters_count > length:
                print("Characters total count is greater than the password length")
                return


        ## initializing the password
        password = []

## picking random alphabets
        for i in range(alphabets_count):
                password.append(random.choice(alphabets))


        ## picking random digits
        for i in range(digits_count):
                password.append(random.choice(digits))
```

```
## picking random alphabets
        for i in range(special_characters_count):
                password.append(random.choice(special_characters))


        ## if the total characters count is less than the password length
        ## add random characters to make it equal to the length
        if characters_count < length:
                random.shuffle(characters)


for i in range(length - characters_count):
                        password.append(random.choice(characters))


        ## shuffling the resultant password
        random.shuffle(password)

        ## converting the list to string
        ## printing the list
        print("".join(password))



## invoking the function
generate_random_password()
```

CHAPTER 5
TESTING

Testing is probably the most important phase for long-term support as well as for the reputation of the company. If you don't control the quality of the software, it will not be able to compete with other products on the market. If software crashes at the customer site, your customer loses productivity as well money and you lose credibility. Sometimes these losses are huge. Unhappy customers will not buy your other products and will not refer other customers to you. You can avoid this situation by doing extensive testing.

Example test case:

Test case 1:

Sample Input:

```
Enter password length:
Enter alphabets count in password:
Enter digits count in password:
Enter special characters count in password:
```

Expected output:

A randomly generated password with the number of respective characters.

Supplied Input:

```
Enter password length: 10
Enter alphabets count in password: 3
Enter digits count in password: 2
Enter special characters count in password: 3
```

Obtained output:

```
V2(&#XlQq1
```

If you see the password generated this time, it has the minimum number of characters that the user wants. And the program has included 2 more random characters to make the password length equal to user input.

Hurray! We have a complete strong password generator.

CHAPTER 6

DEVELOPMENT AND MAINTENANCE

The program should be opened in Python 3 software and it is executed by the user. The input screen appears where the user inputs the required information. The output is automatically displayed on the screen i.e.; the generated password is displayed on the screen.

Since the password can be generated as per the given numbers of characters. It will be easy for the user to customize it as required.

CHAPTER 7

OUTPUT

```
Enter password length: 12
Enter alphabets count in password: 3
Enter digits count in password: 3
Enter special characters count in password: 6
1$*4q$()7G$p
```

```
Enter password length: 10
Enter alphabets count in password: 3
Enter digits count in password: 2
Enter special characters count in password: 3
V2(&#XlQq1
```

CHAPTER 8

CONCLUSION

With the help of this project one can easily generate a random password anytime he/she wants. This project is very useful for those who are always in a perplexed state about what password to use for a tight security purpose.

As these passwords are randomly generated, no one can ever guess them and therefore it provides a tight security be it for lockers or any software etc. which requires protection from outsiders. We can also add many other features to improvise the code as per our requirement and make sure that the resultant password is strong enough.

CHAPTER 9
BIBLIOGRAPHY

Text books referred:

Learning Python-Mark Lutz

Websites referred: