ACTA SCIENTIAE

ISSN: 2178-7727

Designing a Random Password Generator Using Python Programming Language

Rakesh Kumar Giri

* Associate Professor, Department of Computer Science & Engineering,
Saisha Institutions,Chennai, India
Corresponding Email: rakeshgiriap@gmail.com

ABSTRACT

Every individual who uses various online services, is concerned about security and privacy to safeguard personal information and data from hackers. The password-generating system is one of several generating systems available for the security of users' data. Because of the rise in sharing of information online, internet usage, electronic commerce transactions, and data transmission, both password security and authenticity have become grave issues. However, this shows that the password's strength(or length) is also necessary to be strong. As a result, cyber security experts generally advise using complex password combinations. However, due to difficult patterns, individuals frequently forget their passwords. Unlike previous random password generators, the authors are putting forth a novel approach in this paper that will produce a strong password. The Python programming language has been used to create the password-generating system. The authors have used pyperclip, tkinter, random and string libraries in the code design. According to various tests of the system carried out, the trust worthiness of the generated passwords is one hundred percent.

KEYWORDS: Python, Password, Generator, Cyber Security, Random, Interface

INTRODUCTION

During ancient times, when technology did not exist, confidential information was stored in lockers. These lockers ensured maximum protection of the files and the chances of all the valuable information getting compromised were almost negligible , as owners of the data kept the key to the lockers safe with them everywhere they went. However, in modern times, the idea of data protection has transformed entirely. Confidential information is now stored in devices like computers and mobile phones. They are stored in various files and folders that are in built in these devices Thus, to provide greater protection for such data, the concept of passwords has been created.

Users can create passwords and security codes of their own and use them to protect their data. They often create passwords that are easy to remember such as names of their own or those of their close ones, birth dates or other important dates, etc. Unfortunately, hacking and breach of cyber security are on the rise today, and such comprehensible passwords are getting easily cracked by attackers. Hence, it has become extremely essential to strengthen cyber security and data protection. The creation of hard passwords for users to protect their data in browsers and applications is a major step toward the same. This ensures greater protection and makes it difficult for hackers to crack them.

The team's password generator, created using the Python programming language, contributes toward this mission by providing hard-to-crack password combinations to users for better security of their data. Using the Jupyter Notebook coding platform, the password-generating system has been made from carefully written code that makes use of the tkinter, random, pyperclip, and string libraries of Python. The team has carefully analysed these libraries, and how they can be used to create a system that provides unique combinations of passwords.

This paper explains in detail the design of the generator's code and interface, the planning and implementation, the constraints faced during the making of the system, and the final successful output that has been obtained.

METHODOLOGY

The team has researched and created the Python password generator, as it is not only a concept that is essential

ISSN: 2178-7727

for tackling the world's cyber security issues, but it also uniquely explores the features and libraries of the Python programming language. The concept of passwords has been a secure way of maintaining cyber security for many years and continues to be so. Similarly, the Python programming language has been a unique language with simple syntax, and wonderfully designed inbuilt libraries and functions for performing various kinds of tasks. Upon detailed research, the team found out about tkinter, pyperclip, random and string libraries of Python and discovered how essential these libraries are for designing a random password generator. The complete code was designed and executed in the Jupyter Notebook coding platform, as mentioned earlier.

The core feature of the entire project is Python. It is a high-level programming language with concise syntax, which results in short codes that are easy to read and write. Python has also been a trusted language among software developers for years, which is why the team could successfully make use of it for making the password-generating system. The next three important features planned for and used are the tkinter, random and pyperclip modules of Python. The tkinter library helps programmers create a well-designed graphical user interface (GUI) in Python, and it is the only inbuilt GUI library in the programming language. Tkinter has played an important role by providing a suitable interface for the user to input the password length and generate a suitable password for copying. The Random library, as the name suggests, is used to generate random integers at the output and it is specifically used for tasks like generating random numbers, outputting a random value for a list, and so on.Thus, to generate a random password, the team had to make use of the random library. The Pyperclip library is primarily used for copy-and-paste functions in Python. As a result, it was compulsorily used to complete the generating system. Our generator also gives the facility to the users to select the length of the password as per their choice. This has been done because different websites and software have varied criteria for password length. Some may require long passwords, while some require short ones.

The team faced several constraints with respect to the design of the code. Using the tkinter library to create the favourable interface was slightly challenging as the knowledge about the module was obtained recently. As a result, the team took time to create the proper outer appearance of the interface for the system. However, this problem was tackled with after the team researched more about the module and read about the different ways in which tkinter can be put to efficient use. The team also faced the issue of creating the‘ Copy to Clipboard' button as it encountered repeated errors during the import of the pyperclip module in the code. Multiple compilation errors were faced when the code was getting compiled, and despite a few corrections done later, an improper output was obtained.

But after a thorough investigation of the problem, which mainly arose due to a small discrepancy in the syntax and the values used, the code was redesigned with the proper use of all the necessary libraries, and the correct output was obtained. At the output, the team obtained a window that served as the interface of the password generator.

Figure 1. shows the final interface obtained after running the code and creating the buttons for the password generator.

Fig 1: The Resultant Output of the Python Password Generator (with the minimum password length being eight by default)

ISSN: 2178-7727

The procedure for using the password generator is as follows:

- First, the length of the password will be asked, with the minimum length being eight so the user will always obtain a strong password. The user can enter the password either manually or by clicking on the drop-up and drop-down arrows on the right of the typing space for increasing and decreasing the length respectively (figure 2).
- The user will then click on the‘Create Password’option.
- After clicking on the button, a password of the desired length will be obtained in the space below. This password will consist of unique combinations of alphanumeric characters and special symbols (figure 2).

The user will then click on the ‘Click to Copy’button, after which the generated password will be copied to the desired space where the password is needed.

Passwords are critical for authentication. This prevents unauthorized access to any website or portal where a possibility for misuse of information or identity theft persists.

A password is a combination of lowercase, uppercase, symbols, and digits. An ideal or a strong password contains a combination of these four. The length is also crucial in determining a password’s strength with the suitable length being above 10 characters unless specified otherwise.

Python Password Generator:
The python implementation of password generator project is using random and tkinter modules.

Project Prerequisites:
Python password generator requires no extra installation of modules. Random and string modules are predefined. Tkinter should already be available on your system since it is built-in. If importing generates an error, either reinstall python or use the command (for linux systems):
sudo apt-get install python3-tkinter

Project File Structure:
These are the steps to build password generator python project:

1. Import random and tkinter modules
2. Define password generator function
3. Define character string
4. Create the user interface
5. Add input widgets
6. Button to call the translate function

1. Import random and tkinter modules

```
#PythonGeeks program to generate a random password
#Import the necessary modules
import random
from tkinter import messagebox
from tkinter import *
```

Code Explanation:

- **Import random:** To randomly generate a password we use this module. Random is a built-in module used to generate a random subset or a substring of a list, array or string. In our case, it will generate a random string containing characters from the original string.
- **from tkinter import messagebox:** In case the user fails to provide inputs, we must raise a prompt. Thus we use messagebox to raise such prompts
- **from tkinter import *:** To use widgets and define the application interface, we import tkinter

ISSN: 2178-7727

2. Define password generator function

```
#Password generator function
def generate_password():
try:
repeat = int(repeat_entry.get())
length = int(length_entry.get())
except:
messagebox.showerror(message="Please key in the required inputs")
return
#Check if user allows repetition of characters
if repeat == 1:
password = random.sample(character_string,length)
else:
password = random.choices(character_string,k=length)
#Since the returned value is a list, we convert to a string using join
password=''.join(password)
#Declare a string variable
password_v = StringVar()
password="Created password: "+str(password)
#Assign the password to the declared string variables
password_v.set(password)
#Create a read only entry box to view the output, position using place
password_label = Entry(password_gen, bd=0, bg="gray85", textvariable= password_v, state="readonly")
password_label.place(x=10, y=140, height=50, width=320)
```

Code explanation:

- **def generate_password:** Declaration of a function to generate a random password in python
- **Try…except block:** Read the user inputs, length and repeat from the entry widgets using get(). In case the user fails to enter both the inputs, the code will enter the except block and display an error prompt. Using messagebox.showerror, display the error message, which is set to the message variable. If the code enters the except block and to prevent the code after except block from executing, add a return statement
- **repeat == 1 test condition:** Checks if the user allows for repetition of characters in the password. If repeat==1, then use sample for unique characters without repetition and choice if repetition is allowed. Both functions of random take the string to obtain a substring of and k = length, mentioning the length of the substring. A password made with sample may contain similar letters of uppercase and lowercase. This is due to the difference in ASCII values
- **Password = ''.join(password):** Both the random functions return a list. Hence join is used to convert the list to string using '' (no space or empty).
- **password_v = StringVar():** Declare a string variable to substitute the text in the entry widget
- **password="Created password: "+str(password):** Concatenate the string with a text. It is optional
- **password_v.set(password):** Assign the concatenated password to the declared string variable
- **password_label:** Create a read-only (non editable) Entry widget to display the password. The parameters are: 1. window of the application where the widget will be. 2. bd=0 the border of the entry widget (0 implies no border) 3. bg="gray85": The default background colour of the widget is white. To match with the window background colour, bg=gray85 is set. 4. Textvariable = password_v: Assign the text variable password_v to display the text. 5. state="readonly": Makes the widget readonly and not editable
-

3. Define character string:

```
#Define a string containing letters, symbols and numbers
character_string="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!#$
```

ISSN: 2178-7727

%&'()*+,-./:;<=>?@[\]^_`{|}~"

Code explanation:

- character_string: A password must contain lowercase and uppercase characters, numbers and symbols for better security. Thus we define a character string containing lowercase, uppercase letters, numbers and symbols.

4. Create the user interface

```
#Define the user interface
password_gen = Tk()
password_gen.geometry("350x200")
password_gen.title("PythonGeeks Password Generator")
```

Code explanation:

- **password_gen:** Invoke the class tkinter using Tk(). This contains provisions to handle user events and widgets.
- **password_gen.geometry():** Define the dimensions of the frame of the application in the format lengthxbreadth
- **password_gen.title():** An optional parameter to set the title of the application.

5. Add input widgets

```
#Mention the title of the app
title_label = Label(password_gen, text="PythonGeeks Password Generator", font=('Ubuntu Mono',12))
title_label.pack()
#Read length
length_label = Label(password_gen, text="Enter length of password: ")
length_label.place(x=20,y=30)
length_entry = Entry(password_gen, width=3)
length_entry.place(x=190,y=30)
#Read repetition
repeat_label = Label(password_gen, text="Repetition? 1: no repetition, 2: otherwise: ")
repeat_label.place(x=20,y=60)
repeat_entry = Entry(password_gen, width=3)
repeat_entry.place(x=300,y=60)
```

Code explanation:

- **title_label, length_label, repeat_label:** Use labels to add non-editable text of an application. The parameters are window of the screen, text to display and an optional font styling.
- **repeat_entry, length_entry:** To read user input, we use Entry widget. Alternatively, Text can also be used. To use the widget, pass the following parameters: password_gen-the screen of the python password generator app and the width of the widget.
- **widget.place() and widget.pack():** Any widget or label will be visible only after positioning it on the window or screen of the app. To achieve this, we make use or pack() and place(). pack() center aligns text along a row. First widget/label is set on row 1, second on row 2 and so on. To enable customized positioning of the widget in python password generator and to set to or more labels and widgets on the same row, we can use place(). place() takes an x and y coordinate which is the row and the column.

6. Button to call the translate function:

```
#Generate password
password_button = Button(password_gen, text="Generate Password", command=generate_password)
password_button.place(x=100,y=100)
#Exit and close the app
```

ACTA SCIENTIAE

ISSN: 2178-7727

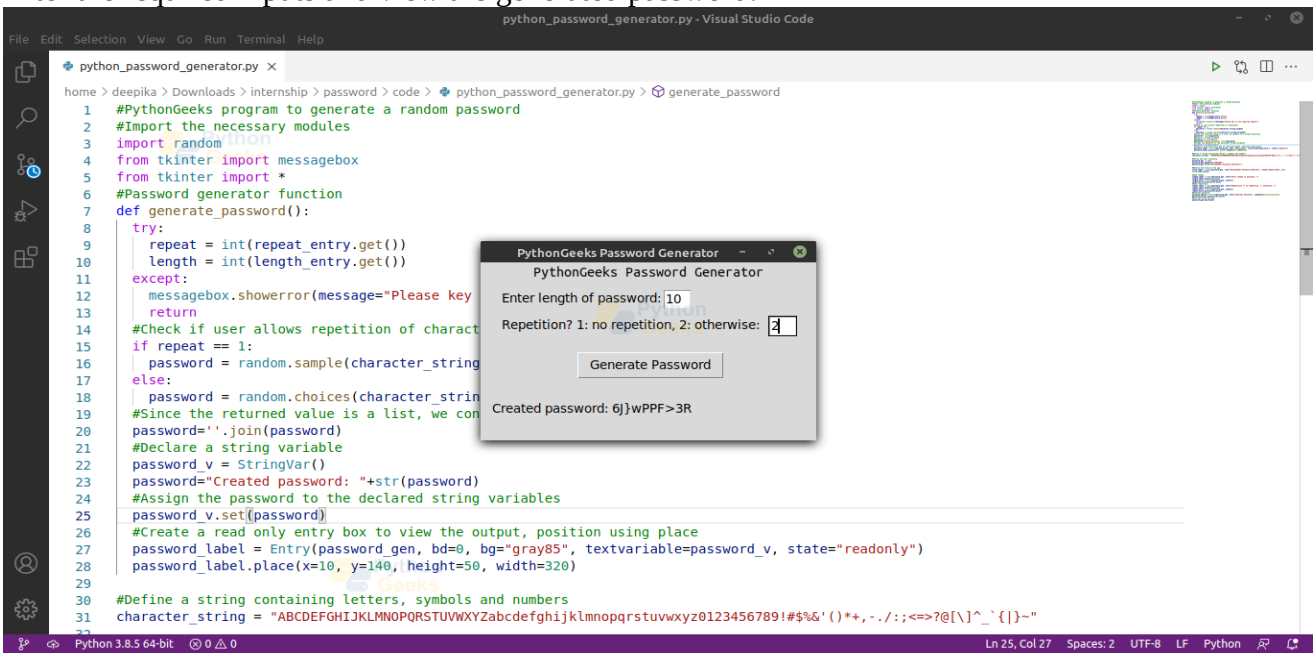
```
password_gen.mainloop()
```

Code explanation:

- **Password_button:** Buttons perform a certain function when the user clicks on it. Link the generate_password function to the button using command argument along with password_gen and name of the button. Place the widget using place().
- **password_gen.mainloop():** To close the app when the user clicks on exit

Python Password Generator Output:

Enter the required inputs and view the generated password:



Summary

We have successfully created python password generator. This python project also provided an introduction to a random module that can be used to generate a random number or a substring. An introduction to Tkinter using simple widgets is also provided.

MODELING AND ANALYSIS

The Python programming language has been used to create the password-generating system. Developed in 1991,this language is object-oriented, and its syntax is concise and short. This helps programmers create a wide range of software using clear codes.The Python code for the system has been compiled in the compiler available in the online coding platform, Jupyter Notebook. This platform is a part of Project Jupyter, which was created in 2014 with the aim of helping programmers and data scientists worldwide create codes, software, or websites. The most promising feature of Jupyter is its compatibility with all browsers and its support for all programming languages. Jupyter also provides options for creating a suitable interface for the code that has been written by the user.

RESULT

The code written is shown in the window that has been shown in Figure 1. The minimum password length i.e., eight will already be shown in the typing space, as it is the default length. Users can keep the same length or increase the length using the arrows on the right of the typing space (figure 2). After entering the length, the user needs to click on the ‘Create Password’ button at the bottom of the typing space. A new password will be generated in the space below the button, which will consist of a unique combination of alphanumeric characters and special symbols. An example of the generated password is shown in Figure 3.

ISSN: 2178-7727

After the password has been created, the user needs to click on the 'Click to Copy' button to copy the generated password to the clipboard. The password can be used anywhere at the user's convenience. The resultant password is not only strong in length but also hard to crack for attackers. Thus, the user's data stays protected.

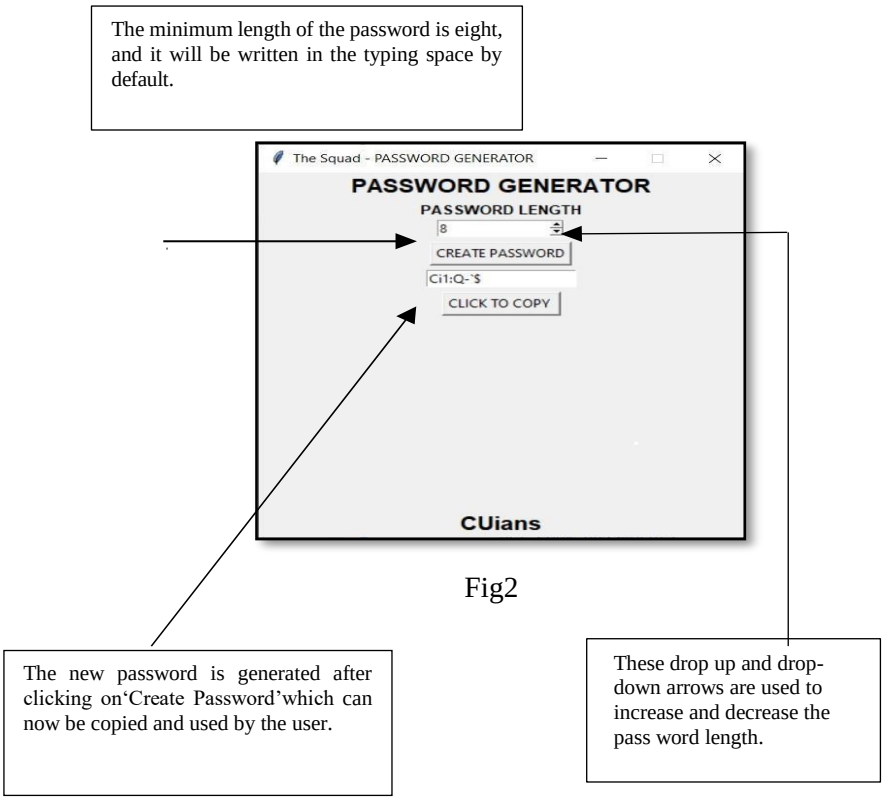


Fig2

CONCLUSION

The password-generating system created by the team using the Python programming language is a valuable tool and a part of the contribution towards protecting the world's cyber security. It not only serves to maintain the privacy of the user but also gives the users the complete liberty to choose a password length of their choice. The most promising feature of the password generator is the minimum length of the password that the user can choose i.e.,eight. As a result, the system will always output the strongest unique password combination to the user. A strong password with a difficult combination will be challenging for attackers to crack.As a result, the data of the users stay protected.

Previous research done on the same has highlighted the fact that it is challenging for users to memorise the passwords provided by the generator. Atif Ul Aftab et al. (2019) [2] have mentioned a way of helping users memorise the passwords by inputs provided by the user. The system will ask the user to provide five texts and two numbers. The term 'text' refers to a single word or even multiple words merged with outspace. In such a case, the users will be able to provide the words and numbers that are convenient for them. After that, the generating system will generate a password by randomly choosing any two texts and one number from the seven choices entered by the user. This password will not only be secure but also easy to memorise by the user because of the convenient choices the user had entered. However, even if the user provides the words and the numbers to be added to their passwords, it will still be a challenge for them to memorise the passwords as the password combinations that will be formed as output will still be complex. Come what may, users will have to devise their own methods of remembering their passwords. Thus, the authors' pass word generators till proves to be a solution as effective as the other password generators that have been already created with the unique feature of providing only strong passwords to the users and providing them the choice of giving input of even stronger passwords through the length of their choice.

The way ahead lies in adding more features to the password generator. The users can even save the passwords

ISSN: 2178-7727

that have been generated, for smooth logging in after the first login. These passwords can be kept hidden in the cloud storage which the user can access by verification of identity. In this manner, the password stays saved and also protected. More such features can be worked upon in the future for better security.

REFERENCES

1. D.Muthulakshmi and A.Sandanasamy (2014) 'Alpha-Numerical Random Password Generator for Safeguarding the Data Assets'–International Journal of Engineering Research and Technology (IJERT) Vol. 3, Issue 12, pp. 435-437.
2. Atif Ul Aftab, Farhana Zaman Glory, Noman Mohammed and Olivier Tremblay-Savard(2019) 'Strong Password Generation Based on User Inputs' – 10th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 417-419.
3. Nitin Arora, Kamal Preet Singh and Ahatsham (2018) 'User Choice-Based Secure Password Generator using Python' –International Journal of Research in Engineering, IT and Social Sciences Vol. 8, Issue 8, pp. 150-151
4. Fatma Al Maqbali and Chris Mitchell (2016) 'Password Generators: Old Ideas and New' - International Federation for Information Processing (IFIP) International Conference on Information Security Theory and Practice, p.246
5. Lujo Bauer, Nicolas Christin, Lorrie Cranor, Patrick Kelley, Saranga Komanduri, Pedro Leon, Michelle Mazurek and Richard Shay (2010) 'Encountering Stronger Password Requirements:User Attitudes and Behaviors' – Proceedings of the Sixth Symposium on Usable Privacy and Security Article No.2, p.8
6. Michael Leonhard and V.N. Venkatakrishnan (2007) 'A Comparative Study of Three Random Password Generators'-Institute of Electrical and Electronics Engineers (IEEE) Conference on Electro Information Technology, pp. 271-272