Password Generator - Python

#### **About Password Generator:**

A password is 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 Project:**

The python implementation of this project is using random and tkinter modules. This project is suitable for beginners who are starting with python.

#### **Project Prerequisites:**

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.

#### **Download Password Generator Code:**

Refer to the code from the following link: [Password Generator Python Code](https://drive.google.com/file/d/1BioKlCfkfsUXd-BObOHyFCvJf-X4LQxk/view?usp=sharing)

#### **Project File Structure:**

These are the steps to build the 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
7. **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

1. **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
* **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. 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

1. **Define character string:**

#Define a string containing letters, symbols and numbers

character\_string="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!#$%&'()\*+,-./:;<=>?@[\]^\_`{|}~"

**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.

1. **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.

1. **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 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 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.

1. **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

password\_gen.mainloop()

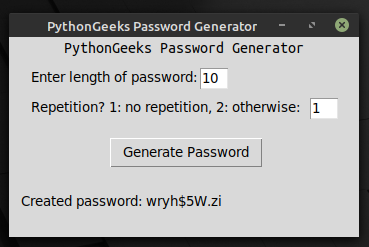
**Code explanation:**

* **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

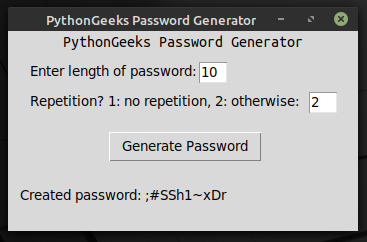
#### **Project output:**

Enter the inputs and view the generated password:

1. Unique characters in password:



1. Repetition allowed in password:



Here, we observe that character ‘S‘ is repeated

#### **Summary**

Thus using python, we created a simple password generator. This project also provided an introduction to a random module which can be used to generate a random number or a substring. An introduction to Tkinter using simple widgets is also provided