[Password Generator](https://www.studocu.com/in/document/indian-institute-of-technology-madras/btech-project/password-generator/40435821?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=password-generator)

# PASSWORD GENERATOR

#### A PROJECT REPORT

|  |  |
| --- | --- |
| S.NO | CONENT NAME |
| **1** | **ABSTRACT** |
| **2** | **INTRODUCTION** |
| 2.1 | **PROJECT** **DEFINITION** |
| 2.2 | **OBJECTIVE** |
| **3.** | **LITERATURE** **REVIEW** |
| **4**. | **DESIGN** **METHODOLOGY** |
| 4.1 | **PREREQUISITES** |
| 4.2 | **TO** **CREATE** **A** **TKINTER** |
| 4.3 | **FLOWCHART** **DIAGRAM** |
| 4.4 | **STEPS** **TO** **BUILD** **CURRENCY** **CONVERTER** |

4

|  |  |
| --- | --- |
| 4.5 | **OUTPUT** |
| **5.** | **PROBLEM** **STATEMENT** |
| 5.1 | **ARCHITECTURE** |
| **6.** | **RESULT** **AND** **DISCUSSION** |
| 6.1 | **ADVANTAGES** |
| 6.2 | **CONCLUSION** |
| 6.3 | **REFERENCESS** |

## INTRODUCTION:

We are developing a python project named Password generator. A Password generator is a very useful application. This application can generate a random password, with the combination of letters, numerics, and special characters. One can mention the length of the password based on requirement and can also select the strength of the password.

The Password generator tool creates a random and customized password for users that helps them to create a strong password which provides greater security. While there are many examples of "random" password generator programs available on the Internet, generating randomness can be tricky and many programs do not generate random characters in a way that ensures strong security.

A common recommendation is to use open source security tools where possible since they allow independent checks on the quality of the methods used. Note that simply generating a password at random does not ensure the password is a strong password, because it is possible, although highly unlikely, to generate an easily guessed or cracked password. In fact, there is no need at all for a password to have been produced by a perfectly random process: it just needs to be sufficiently difficult to guess.

A password generator can be part of a password manager. When a password policy enforces complex rules, it can be easier to use a password generator based on that set of rules than to manually create passwords.

### Project Definition:

Speaking regarding the system, the user can create a random password according to various sizes.It additionally presents with an aesthetic color-coded system which indicates the stamina of the password, beginning from Very Weak to Superb password strength. After creating a random password, the system presents it in the clipboard where the user can copy and paste easily.

This GUI based Password Generator supplies the most basic method for generating a solid password for the individuals. In short, this job just concentrates on producing arbitrary passwords. In order to run the task, you must have set up Python, on your PC. This is a basic GUI Based system, specially composed for the beginners. Password Generator in Python with source code is complementary to download.

Good password generators do the following:

* Adjust guidelines to fit different sites' unique password requirements
* Generate strong passwords using secure technology with built-in randomness
* Are integrated into a password manager like Dashlane to create, manage, and easily use all of your strong passwords

A password is your personal key to a computer system. Passwords help to ensure that only authorized individuals access computer systems. Passwords also help to

6

determine accountability for all transactions and other changes made to system resources, including data.

Password generators are tools that allow the user to create random and customized strong passwords based on preferences.

## OBJECTIVE:

The main objective of the study is to develop a password generator. For a secure and strong password remainder the following:-

* Never use personal information.
* Include a combination of letters, numbers, and characters.
* Prioritize password length.
* Never repeat passwords.
* Avoid using real words. Strong password:-
* **Long:** The longer a password, the more secure it is. A strong password should be at least 12 characters long.
* **Random:** Strong passwords use a combination of letters, numbers, cases, and symbols to form an unpredictable string of characters that doesn't resemble words or names.
* **Unique:** A strong password should be unique to each account to reduce vulnerability in the event of a hack.

7

### Literature Review

Entropy in software engineering is the haphazardness created by an application, with the end goal of cryptography. In software engineering, we consider entropy because of the accompanying reasons. Entropy is basic for PCs to recreate different physical, compound and organic marvel that are regularly irregular. Entropy is important for cryptography, and a few plan methods, e.g., inspecting and property testing. Despite the fact that proficient deterministic calculations are obscure for some issues, straightforward, exquisite and quick randomized calculations are known. We need to know whether these randomized calculations can be DE randomized without losing proficiency. Irregularity is an effective device for scrambling delicate information. For example, the event that utilizes an irregular secret key generator 5, 10, 100 or 1000 times with similar parameters, there's a little shot that the generator will make a similar outcome twice since it should be arbitrary, which means the outcomes will be capricious, won't take after a set-design, and a past outcome will have no impact on any of the accompanying results.

The outcome saw in such a situation are pseudo irregular, i.e., they are not generally arbitrary yet they seem, by all accounts, to be irregular.

Conventional PC frameworks are not awesome at creating irregular outcomes. They are deterministic, which implies that on the off chance that we ask a similar inquiry we will find a similar solution without fail. Indeed, such machines are particularly and deliberately modified to dispense with irregularity in comes about. They do this by following standards and depending on calculations when they process. A very deterministic machine cannot create genuinely irregular number arrangements since it takes after a similar calculation to deliver its outcomes. Ordinarily, that implies it begins with a typical 'seed' number and after that takes after an example.Therefore the randomized passwords are not genuinely irregular since the same algorithm made them. Priti Jadhao and Lalit Dole present a survey on authentication password techniques. They discussed about many techniques but do not ensure the security of a user, since there is a very high chance of a user, coming across a previously used password, thus making the password generators unreliable. Zhou et. Al. about onetime password generating method.In spite of the fact that being very arbitrary, pseudo-irregular secret key generators are not 100% dependable. They normally do not; however, can furnish us with a formerly utilized secret key, accordingly making the whole thought of an irregular watchword generator outdated. In this manner, the ideal approach to enhance the execution and security of pseudorandom generators is high entropy, i.e., eccentrics of assembled information utilized for cryptographic and security capacities.

This is typically accomplished by utilizing cryptographically secure pseudo-

arbitrary number generator. The entropy in a cryptographically Secure Pseudo-Random Number Generator is higher than a customary Pseudo- Random Number Generator, on the grounds that it is generally created by

8

an eccentric wonder or a physical action, for example, organize action, hard drive action, console strokes, mouse developments, no. of dynamic procedures, time stamp, and so forth., i.e., everything that is continuously.

This information is then used to make the 'seed' number, which makes our new irregular secret key. In entirety, the higher the entropy, the harder an arbitrary number or secret word ought to be to foresee. A. Nitin et. al. a new technique for generating a complex password based on real components.

They proposed a new technique based on entropy to generate the password. An entropy based password generator, will use components that change in real time, like time stamp, to give us a new password, in every single hit of a button.

## DESIGN METHODOLOGY:

**Steps** **required** **for** **building** **this** **project:**

* All characters should be stored as a list. This can be done with the string module of Python or by typing each character individually.
* Ask the user for the length of the password.
* Use random.shuffle to shuffle the characters.
* Create an empty list to store the password.
* Iterate length times to generate the password.

9

* Choose a random character from all the characters using the random.choice method.
* Add the random character to the password.
* Randomize the resultant list of passwords.
* Use the join method to create a string from the list of passwords.
* Print the password.

**Project** **Prerequisites**

To build this project we will use the basic concept of python and libraries – Tkinter, pyperclip, random, string.

* **Tkinter** is a standard GUI library and is one of the easiest ways to build a GUI application.
* **pyperclip** module allows us to copy and paste text to and from the clipboard to your computer
* **The** **random** module can generate random numbers
* **string** module contains a number of functions to process the standard python string

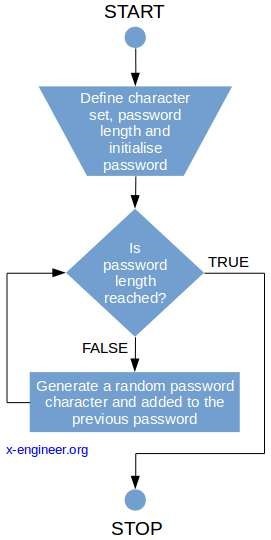
**Prerequisites** **:**

Python offers multiple environments for developing a python program. Out of all the pycharm is the most commonly IDE. It is user friendly and fast & safe.

**To** **install** **pycharm:**

* Simply text pycharm on google.
* Click on the top most site.
* Click on install button to install pycharm.

# FLOWCHART DIAGRAM:



1

## PROBLEM STATEMENT:

In order to create a strong password, users can use this password generator to generate a random and customized password.

## RESULT AND DISCUSSION: ADVANTAGES:

1. Minimal Effort.
2. Secure & strong password.
3. Less time consuming.
4. Fast response.
5. complex for hackers.

## CONCLUSION:

The password generated using alpha-numerical random password mechanism that was illustrated above is practical and can be used with great results. When the password is selected manually, most of the time, the users select the

password that are related to himself or herself and related to any of the event. This gives the space for the intruders to deploy various attacks in breaking the passwords. The random generated passwords avoid this particular situation. One of the drawbacks could be the difficulty in memorizing the randomly generated password. But when comparing the security achieved through the randomly generated password, it is much preferable than the manually chosen password.

The encryption and decryption standard provided here also strengthens the security measures. Since, the encryption and decryption standards are simple, it is cost- effective. The above done work also creates awareness and interest to start exploring this field more.

14

Code

#Before Copy this Source, Please, Take Owner Permission and Give Credits.

#Thanks.

try:

    from string import ascii\_letters, digits, punctuation, join

except ImportError:

    from string import ascii\_letters, digits, punctuation

from random import choice, sample, randint

# specific imports to make this small, fast, efficient

# to generate a password with specific restrictions: \

# NO numbers or special characters in 1st position

# NO space at the end of the string

#MainFunc

def isEven(integer):

    """Return Boolean: True if input is even, False if not."""

    return integer % 2 == 0

def RandPass(size = 8):

    """This is the password generator"""

    s0 = "!@#$%^&\*- \_~+-=" # this set of special characters contains a space

    s1 = ascii\_letters # upper AND lower cases

    s3 = digits

    # 's0' can be CUSTOMIZED to include only allowed characters

    s = s0 + s1

    s\_full = s + s3

    passlen = size.get()

    new\_password = ""

    # assigning specific sizes for each section of the pw generated

    if isEven(passlen) == True:

        front = passlen // 5

    else:

        front = passlen // 2

    mid = 2

    previous = passlen - (front + mid) - 1

    pass0 = "".join(choice(s0)) # forces a minimum number of punctuations in the middle

    pass1 = "".join(sample(s\_full,front ))

    pass2 = "".join(sample(s3,mid))

    # NO punctuations as 1st character!!!

    pass3 = "".join(sample(s, previous ))

    # sometimes integer division reduces the size of the desired password length, the following adjusts it back

    if passlen != len(pass0 + pass1 + pass2 + pass3):

        pass2 = "".join(sample(s3,passlen - (front+previous+1) ))

    if pass3[:-1] == ' ': # to avoid having an empty space at the end of password

        temp = list(pass3)

        temp[:-1] = choice(s)

        pass3 = ''.join(str(e) for e in temp)

    new\_password = pass0 + pass1 + pass2 + pass3

    if passlen <= 8:

        msg = 'VERY WEAK'

        colorVal = "#6d0001"

    elif passlen <=10:

        msg = 'WEAK'

        colorVal = "#cc0000"

    elif passlen <=12:

        msg = 'DECENT'

        colorVal = "#fc8600"

    elif passlen <=14:

        msg = 'GOOD'

        colorVal = "#eae200"

    elif passlen <=16:

        msg = 'STRONG'

        colorVal = "#9ff400"

    elif passlen <=18:

        msg = 'VERY STRONG'

        colorVal = "#007715"

    elif passlen >18:

        msg = 'EXCELLENT'

        colorVal = "#001fef"

    else:

        pass

    return new\_password, msg, colorVal

main.py

try:

    from tkinter import \*

except ImportError:

    from Tkinter import \*

import time

from pwgenfunc import RandPass

import pyperclip

# Main

def pwGenerator(size=8):

    data = RandPass(size)

    new\_password = data[0]

    pw\_strength = data[1]

    pw\_color = data[2]

    PASSWORD.set(new\_password)

    label\_strength.configure(foreground="white", background=pw\_color, text=pw\_strength, font=('Segoe UI', 10, 'bold'), bd=10, height=1, width=10)

    gui.clipboard\_clear()

    gui.clipboard\_append(new\_password)

    gui.update()

    time.sleep(.02)

    gui.update()

# MainWindow

gui = Tk()

gui.title("Password Generator")

gui.config(bg='#1A1A1A')

width = 600

height = 342

screen\_width = gui.winfo\_screenwidth()

screen\_height = gui.winfo\_screenheight()

x = (screen\_width / 2) - (width / 2)

y = (screen\_height / 2) - (height / 2)

gui.geometry("%dx%d+%d+%d" % (width, height, x, y))

# Variables

PASSWORD = StringVar()

PW\_SIZE = IntVar()

e1 = Entry(gui, text=PW\_SIZE)

PW\_SIZE.set(8)

# Window Frame

Top = Frame(gui, width=width)

Top.pack(side=TOP)

Top.config(bg='#1A1A1A')

Form = Frame(gui, width=width, background="#1A1A1A")

Form.pack(side=TOP)

Bot = Frame(gui, width=width)

Bot.pack(side=BOTTOM)

# Labels

label\_password = Label(Form, font=('Segoe UI', 18), text="Password", foreground="white", background="#1A1A1A", bd=10)

label\_password.grid(row=0, pady=10)

label\_strength = Label(Form, font=('Segoe UI', 10, 'bold'), foreground="white", background="white", text="Weak", bd=10, height=1, width=10)

label\_strength.grid(row=0, column=3, pady=10, padx=10)

label\_pw\_size = Label(Form, font=('Segoe UI', 18), text="Size", foreground="white", background="#1A1A1A", bd=10)

label\_pw\_size.grid(row=2, pady=10)

label\_instructions = Label(Bot, width=width, font=('Segoe UI', 12, 'bold'), text="Password Generated to your Clipboard!", foreground="white", background="#1A1A1A", bd=1, relief=SOLID)

label\_instructions.pack(fill=X)

# Button

password = Entry(Form, textvariable=PASSWORD, font=(18), width=24)

password.grid(row=0, column=1, columnspan=2)

pw\_size = Scale(Form, from\_=8, to=24, length=200, width=24, sliderlength=14, orient=HORIZONTAL, variable=PW\_SIZE, foreground="white", background="#1A1A1A", font=(16))

pw\_size.grid(row=2, column=1, columnspan=2)

# Copy to Clipboard

def Copy\_password():

    pyperclip.copy(PASSWORD.get())

Button(Top, text='COPY TO CLIPBOARD', foreground="white", background="#1A1A1A", command=Copy\_password).pack(pady=5)

btn\_generate = Button(Form, text="Generate Now", width=20, command=lambda: pwGenerator(PW\_SIZE.get()))

btn\_generate.grid(row=4, column=1, columnspan=2)

gui.resizable(False, False)

gui.mainloop()

import random

import string

def RandPass(size):

    # Define possible characters for the password

    characters = string.ascii\_letters + string.digits + string.punctuation

    # Generate a random password

    password = ''.join(random.choice(characters) for \_ in range(size))

    # Determine the strength of the password

    if size < 8:

        strength = "Weak"

        color = "red"

    elif 8 <= size < 12:

        strength = "Moderate"

        color = "yellow"

    else:

        strength = "Strong"

        color = "green"

    return password, strength, color

## REFERENCES:

* [http://geeksforgeeks.com](http://geeksforgeeks.com/)
* [http://github.com](http://github.com/)
* [http://linkedin.com](http://linkedin.com/)
* https://en.wikipedia.org

[](https://www.studocu.com/in?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=password-generator)