[**Final Report**](https://www.studocu.com/en-us/document/dr-apj-abdul-kalam-technical-university/btech/report-final/25902469?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=report-final)

*Title: Password Generator using Python*



A

Project Report

On

**Random password Generator Using Python**

Submitted

In partial fulfillment of requirements for the degree of

**Bachelor of Technology In Computer Science & Engineering**

Submitted by

**B. THARUN KUMAR REDDY (192G1A0509)**

**Department of Computer Science and Engineering**

**Contents**

[DECLARATION III](#_Toc31737)

[ACKNOWLEDGEMENT iV](#_Toc31739)

[ABSTRACT VI](#_Toc31740)

**CHAPTER 1 (INTRODUCTION)**

**CHAPTER 2 (PROJECT OVERVIEW)**

**CHAPTER 3 (OBJECTIVE)**

**CHAPTER 5 (TYPES AND STRENGTH OF PASSWORD GENERATED)**

**CHAPTER 6 (PROJECT PREQUISITE)**

**CHAPTER 7 (IMPLEMENTATION)**

**SOURCE CODE & OUTPUT**

**CHAPTER 8 (TESTING)**

**CONCLUSION**

**BIBLIOGRAPHY**

# *DECLARATION*

*I hereby declare that this submission is our own work and that to the best of our knowledge and beliefs. It contains no material l previously published or written by neither any person nor material which to a substant till extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning , except here due acknowledgement has been made in the text.*

*Signature:*

*Name: B.Tharun Kumar Reddy*

*Roll No:192G1A0509*

*Date: 06/07/2022*

# *ACKNOWLEDGEMENT*

*In completing this project we have been fortunate enough to have help, support and encouragement from many people. I would like to acknowledge them for their cooperation.*

*Firstly, we would like to thank* ***Mr. A.V.L.N.Sujith****, from Department of Computer Science & Engineering, A.I.M.T; for guiding us through each and every step of the process with knowledge and support. His thoughts have been a constant source of inspiration for us.*

*We would also like to acknowledge the contribution of all faculty members of the department for their kind assistance, suggestions and cooperation throughout the development of the project.*

*Finally, we would like to thank our classmates for the encouragement and help during the project.*

*Signature:*

*Name: Tharun Kumar Reddy*

*Roll No:192G1A0509*

*Date: 06/07/2022*

# ABSTRACT

*A* ***random password generator*** *is* [*software*](https://en.wikipedia.org/wiki/Computer_software) *program or* [*hardware*](https://en.wikipedia.org/wiki/Computer_hardware) *device that takes input from a* [*random*](https://en.wikipedia.org/wiki/Random) *or* [*pseudo-random*](https://en.wikipedia.org/wiki/Pseudo-random) *number generator and automatically generates a* [*password*](https://en.wikipedia.org/wiki/Password)*. Random passwords can be generated manually, using simple sources of randomness such as dice or coins, or they can be generated using a computer.*

*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. Use for education purpose only! For the project demo, look at the picture slider listed below.*

**1. INTRODUCTION**

With growing technology, everything has relied on data and securing these data is the main concern. Passwords are meant to keep the data safe that we upload on the Internet.

An easy password can be hacked easily and all the personal information can be misused. In order to prevent such things and keep the data safe, it is quite necessary to keep our passwords very strong.

A password generator is a software application device that creates arbitrary or tailored passwords for individuals. It assists individuals to produce more powerful passwords that offer greater protection for a provided sort of access. Some password generators are merely random password generators. These programs produce complex/strong passwords with mixes of numbers, uppercase and also lowercase letters, and also unique personalities such as dental braces, asterisks, slashes, and so on. It is a tool that generates passwords based on the given guidelines that you set to create an unpredictable strong password for your accounts.

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](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Password_manager). When a [password policy](https://en.wikipedia.org/wiki/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.

1. **PROJECT OVERVIEW**

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. Use for education purpose only! For the project demo, look at the picture slider listed below.

3.[**Python**](https://en.wikipedia.org/wiki/Python_(programming_language))

Until version 3.5.10, the random module includes a SystemRandom class that obtains cryptographic grade random bits from /dev/urandom on a Unix-like system, including Linux and macOS, while on Windows it uses CryptGenRandom.

From version 3.6 however, the usage of random.SystemRandom() is not recommended anymore, and the secrets module (that has a similar syntax) must be preferred.

Here is a simple Python script that shows password generation before and after the secrets module appearance :

|  |
| --- |
| *#!/usr/bin/env python3*  **import** **sys import** **string**  **if** sys.version\_info < (3, 6): *# Python 3.5.10 or lower*  **import** **random else**: *# Python 3.6 and above*  **import** **secrets**  **def** getRandPwd(length):  alphabet = string.ascii\_letters + string.digits *# [a-zA-Z0-9]*  **if** sys.version\_info < (3, 6):  rng = random.SystemRandom() **return** ''.join(rng.choice(alphabet) **for** \_ **in** range(length)) **else**: **return** ''.join(secrets.choice(alphabet) **for** \_ **in** range(length))  password = getRandPwd(32) |

print(password)

**4. TYPES AND STRENGTH OF PASSWORD GENERATED**

Random password generators normally output a string of symbols of specified length. These can be individual characters from some character set, syllables designed to form pronounceable passwords, or words from some word list to form a [passphrase](https://en.wikipedia.org/wiki/Passphrase). The program can be customized to ensure the resulting password complies with the local password policy, say by always producing a mix of letters, numbers and special characters. Such policies typically reduce strength slightly below the formula that follows, because symbols are no longer independently produced.

The [Password strength](https://en.wikipedia.org/wiki/Password_strength) of a random password against a particular attack ([brute-force search)](https://en.wikipedia.org/wiki/Brute-force_search), can be calculated by computing the [information entropy](https://en.wikipedia.org/wiki/Information_entropy) of the random process that produced it. If each symbol in the password is produced independently and with uniform probability, the entropy in bits



where *N* is the number of possible symbols and *L* is the number of symbols in the

password. The function log2 is the [base-2 logarithm](https://en.wikipedia.org/wiki/Binary_logarithm). *H* is typically measured in [bits](https://en.wikipedia.org/wiki/Bit).

|  |  |  |
| --- | --- | --- |
| **Entropy per symbol for different symbol sets** | | |
| **Symbol set** | **Symbol count *N*** | **Entropy per symbol *H*** |
| [Arabic numerals](https://en.wikipedia.org/wiki/Arabic_numerals) (0–9) (e.g. [PIN)](https://en.wikipedia.org/wiki/Personal_identification_number) | 10 | 3.32 bits |
| [Hexadecimal](https://en.wikipedia.org/wiki/Hexadecimal) numerals (0–9, A–F) (e.g. [WEP key](https://en.wikipedia.org/wiki/Wired_Equivalent_Privacy)) | 16 | 4.00 bits |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [Case insensitive](https://en.wikipedia.org/wiki/Case_sensitivity) [Latin alphabet](https://en.wikipedia.org/wiki/Latin_alphabet) (a–z or A–Z) | | | | | | 26 | | | 4.70 bits | | |
| Case insensitive [alphanumeric](https://en.wikipedia.org/wiki/Alphanumeric) (a–z or A–Z, 0–9) | | | | | | 36 | | | 5.17 bits | | |
| [Case sensitive](https://en.wikipedia.org/wiki/Case_sensitivity) Latin alphabet (a–z, A–Z) | | | | | | 52 | | | 5.70 bits | | |
| Case sensitive alphanumeric (a–z, A–Z, 0–9) | | | | | | 62 | | | 5.95 bits | | |
| All [ASCII printable characters](https://en.wikipedia.org/wiki/ASCII#ASCII_printable_characters) | | | | | | 95 | | | 6.55 bits | | |
| [Diceware](https://en.wikipedia.org/wiki/Diceware) word list | | | | | | 7776 | | | 12.9 bits | | |
| **Lengths *L* of truly randomly generated passwords required to achieve desired a password entropy *H* for symbol sets containing *N* symbols.** | | | | | | | | | | | |
| **Desi red**  **pass wor d**  **entr opy**  ***H*** | [**Ara bic num eral s**](https://en.wikipedia.org/wiki/Arabic_numerals) | [**Hexad ecimal**](https://en.wikipedia.org/wiki/Hexadecimal) | [**Case insensiti ve**](https://en.wikipedia.org/wiki/Case_sensitivity) [**Latin**](https://en.wikipedia.org/wiki/Latin_alphabet)  [**alphabe**](https://en.wikipedia.org/wiki/Latin_alphabet)  [**t**](https://en.wikipedia.org/wiki/Latin_alphabet) | **Case insensitive** [**a**](https://en.wikipedia.org/wiki/Alphanumeric)  [**lphanumeri c**](https://en.wikipedia.org/wiki/Alphanumeric) | [**Case sensiti**](https://en.wikipedia.org/wiki/Case_sensitivity)  [**ve**](https://en.wikipedia.org/wiki/Case_sensitivity) **Lati n**  **alphab et** | | **Case sensiti ve**  **alphan**  **umeri c** | **All** [**A SCII print able**](https://en.wikipedia.org/wiki/Printable_characters)  [**char**](https://en.wikipedia.org/wiki/Printable_characters)  [**acter s**](https://en.wikipedia.org/wiki/Printable_characters) | | **All** [**ex tende d**](https://en.wikipedia.org/wiki/Extended_ASCII)  [**ASCI**](https://en.wikipedia.org/wiki/Extended_ASCII)  [**I**](https://en.wikipedia.org/wiki/Extended_ASCII)  [**printa ble**](https://en.wikipedia.org/wiki/Extended_ASCII)  [**chara cters**](https://en.wikipedia.org/wiki/Extended_ASCII) | [**Dicewa re**](https://en.wikipedia.org/wiki/Diceware) **wor d list** |
| 32 bits | 10 | 8 | 7 | 7 | 6 | | 6 | 5 | | 5 | 3 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 40 bits | 13 | 10 | 9 | 8 | 8 | 7 | 7 | 6 | 4 |
| 64 bits | 20 | 16 | 14 | 13 | 12 | 11 | 10 | 9 | 5 |
| 80 bits | 25 | 20 | 18 | 16 | 15 | 14 | 13 | 11 | 7 |
| 96 bits | 29 | 24 | 21 | 19 | 17 | 17 | 15 | 13 | 8 |
| 128 bits | 39 | 32 | 28 | 25 | 23 | 22 | 20 | 17 | 10 |
| 160 bits | 49 | 40 | 35 | 31 | 29 | 27 | 25 | 21 | 13 |
| 192 bits | 58 | 48 | 41 | 38 | 34 | 33 | 30 | 25 | 15 |
| 224 bits | 68 | 56 | 48 | 44 | 40 | 38 | 35 | 29 | 18 |
| 256 bits | 78 | 64 | 55 | 50 | 45 | 43 | 39 | 33 | 20 |
| 384 | 116 | 96 | 82 | 75 | 68 | 65 | 59 | 50 | 30 |
| bits |  |  |  |  |  |  |  |  |  |
| 512 bits | 155 | 128 | 109 | 100 | 90 | 86 | 78 | 66 | 40 |
| 1024 bits | 309 | 256 | 218 | 199 | 180 | 172 | 156 | 132 | 80 |

Any password generator is limited by the state space of the pseudo-random number generator used if it is based on one. Thus a password generated using a 32-bit generator is limited to 32 bits entropy, regardless of the number of characters the password contains.

Note, however, that a different type of attack might succeed against a password evaluated as 'very strong' by the above calculation.

**5. PROJECT PREQUISITES**

**Python** is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) [general-purpose programming language.](https://en.wikipedia.org/wiki/General-purpose_programming_language) Its design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with its use of [significant indentation.](https://en.wikipedia.org/wiki/Off-side_rule) Its [language constructs](https://en.wikipedia.org/wiki/Language_construct) as well as its [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help [programmers](https://en.wikipedia.org/wiki/Programmers) write clear, logical code for small and largescale projects.

Python is [dynamically-typed](https://en.wikipedia.org/wiki/Type_system#DYNAMIC) and [garbage-collected.](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [structured](https://en.wikipedia.org/wiki/Structured_programming) (particularly, [procedural)](https://en.wikipedia.org/wiki/Procedural_programming), object-oriented and [functional programming.](https://en.wikipedia.org/wiki/Functional_programming) It is often described as a "batteries included" language due to its comprehensive [standard library.](https://en.wikipedia.org/wiki/Standard_library)

[Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) began working on Python in the late 1980s, as a successor to the [ABC programming language](https://en.wikipedia.org/wiki/ABC_(programming_language)), and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as [list comprehensions](https://en.wikipedia.org/wiki/List_comprehension) and a [cycle-detecting](https://en.wikipedia.org/wiki/Cycle_detection) garbage collection system (in addition to [reference counting](https://en.wikipedia.org/wiki/Reference_counting)). Python 3.0 was released in 2008 and was a major revision of the language that is not completely [backward-compatible](https://en.wikipedia.org/wiki/Backward_compatibility). Python 2 was discontinued with version 2.7.18 in 2020.

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.

To install the libraries we can use pip installer from the command line:

pip install tkinter pip install pyperclip pip install random pip install strings

**6. IMPLEMENTATION**

Let’s check the step to build a Password Generator using Python

* Import modules
* Initialized Window
* Select Password Length
* Define Functions

*Steps to create random password generator*

## *1. Import Libraries*

The first step is to import libraries

from tkinter import \* import random, string import pyperclip

## *2. Initialize Window*

root = Tk() root.geometry("400x400") root.resizable(0,0)

root.title("PASSWORD GENERATOR")

* **Tk()** initialized tkinter which means window created
* **geometry()** set the width and height of the window
* **resizable(0,0)** set the fixed size of the window
* **title()** set the title of the window

Label(root, text = 'PASSWORD GENERATOR' , font ='arial 15 bold').pack()

Label(root, font ='arial 15 bold').pack(side = BOTTOM)

**Label()** widget use to display one or more than one line of text that users can’t able to modify.

* **root** is the name which we refer to our window
* **text** which we display on the label
* **font** in which the text is written
* **pack** organized widget in block

## *3. Select Password Length*

pass\_label = Label(root, text = 'PASSWORD LENGTH', font = 'arial 10 bold').pack() pass\_len = IntVar()

length = Spinbox(root, from\_ = 8, to\_ = 32 , textvariable = pass\_len , width = 15).pack()

* **pass\_len** is an integer type variable that stores the length of a password.
* To select the password length we use **Spinbox()** widget.
* **Spinbox()** widget is used to select from a fixed number of values. Here the value from 8 to 32

## *4. Function to Generate Password*

pass\_str = StringVar()  **def** Generator(): password = ''  **for** x **in** range (0,4):

password = random.choice(string.ascii\_uppercase) + random.choice(string.ascii\_lowercase) + random.choice(string.digits) + random.choice(string.punctuation)  **for** y **in** range(pass\_len.get()- 4): password = password + random.choice(string.ascii\_uppercase + string.ascii\_lowercase + string.digits + string.punctuation) pass\_str.set(password)

* **pass\_str** is a string type variable that stores the generated password
* **password** = “” is the empty string
* First loop will generate a string of length 4 which is a combination of an uppercase letter, a lowercase letter, digits, and a special symbol and that string will store in password variable.
* The second loop will generate a random string of length entered by the user – 4 and add to the password variable. Here we minus 4 to the length of the user because we already generate the string of length 4.

We have done this because we want a password which must contain an uppercase, a lowercase, a digit, and a special symbol.

Now the password is set to the **pass\_str()** variable.

Button(root, text = "GENERATE PASSWORD" , command = Generator ).pack(pady= 5) Entry(root , textvariable = pass\_str).pack()

* **Button()** widget used to display button on our window
* **command** is called when the button is click
* **Entry()** widget used to create an input text field
* **textvariable** used to retrieve the current text to the entry widget

## *5. Function to Copy Password*

def Copy\_password():

pyperclip.copy(pass\_str.get())

Button

(

root, text =

'COPY TO CLIPBOARD'

, command = Copy\_password

)

.pack

(

pady=5

)

**pyperclip.copy()**

used to copy the text to clipboard

**SOURCE CODE & OUTPUT**

from tkinter import \*

import random, string

import pyperclip

root =Tk()

root.geometry("800x400")

root.resizable(0,0)

root.title("MY - PASSWORD GENERATOR")

#heading

heading = Label(root, text = 'PASSWORD GENERATOR' , font ='arial 15 bold').pack()

Label(root, font ='arial 15 bold').pack(side = BOTTOM)

###select password length

pass\_label = Label(root, text = 'PASSWORD LENGTH', font = 'arial 10 bold').pack()

pass\_len = IntVar()

length = Spinbox(root, from\_ = 8, to\_ = 32 , textvariable = pass\_len , width = 15).pack()

pass\_str = StringVar()

def Generator():

password = ''

for x in range (0,4):

password = random.choice(string.ascii\_uppercase)+random.choice(string.ascii\_lowercase)+random.choice(string.digits)+random.choice(string.punctuation)

for y in range(pass\_len.get()- 4):

password = password+random.choice(string.ascii\_uppercase + string.ascii\_lowercase + string.digits + string.punctuation)

pass\_str.set(password)

Button(root, text = "GENERATE PASSWORD" , command = Generator ).pack(pady= 5)

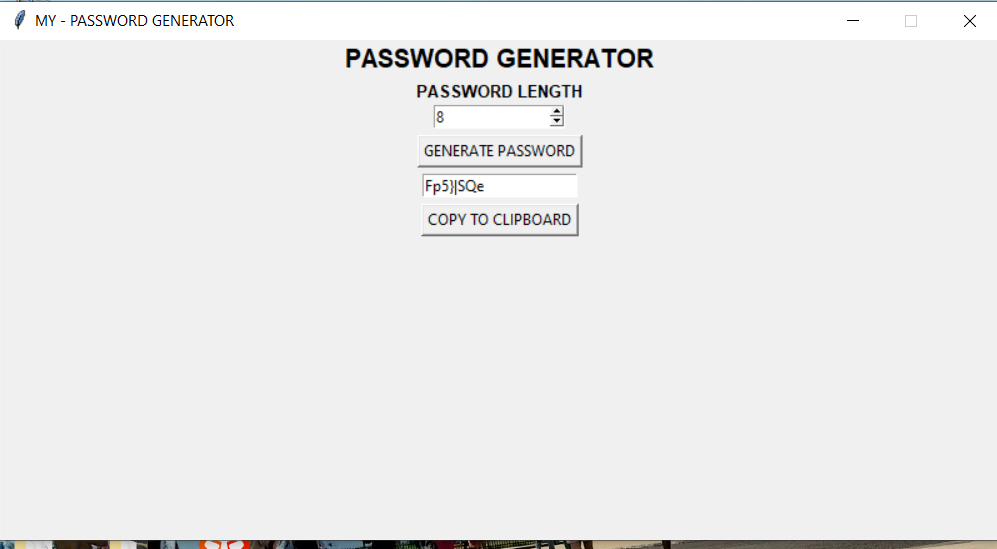
Entry(root , textvariable = pass\_str).pack()

def Copy\_password():

pyperclip.copy(pass\_str.get())

Button(root, text = 'COPY TO CLIPBOARD', command = Copy\_password).pack(pady=5)

root.mainloop()



**8. TESTING**

***T****esting Objectives:*

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say, o Testing is a process of executing a program with the intent of finding an error.

o A successful test is one that uncovers an as yet undiscovered error. o The tests are inadequate to detect possibly present errors. o The software more or less confirms to the quality and reliable standards.

**Unit Testing :**

▪ The purpose of the coding and unit testing phase of software development is to translate the software design into source code. Each component of the design is implemented as a program module. The end-product of this phase is a set of program modules that have been individually tested. To enable the engineers to write good quality programs, every software development organization normally formulates its own coding standard that suits itself. A coding standard addresses issues such as the standard ways of laying out the program codes, the template for laying out the function and module headers, commenting guidelines, variable and function naming conventions, the maximum number of source lines permitted in each module, and so forth.

▪ During this phase, each module is unit tested to determine the correct working of all the individual modules. It involves testing each module in isolation as this is the most efficient way to debug the errors identified at this stage. Another reason behind testing a module in isolation is that the other modules, with which this module has to be interfaced, may not be ready.

**Integration and System Testing**

Integration of different modules is undertaken once they have been coded and unit tested. During the integration and syste3m testing phase, the modules are integrated in a planned manner. The different modules making up a software product are almost never integrated in one shot. Integration is normally carried out incrementally over a number of steps. During each integration step, the partially integrated system is tested and a set of previously planned modules are added to it. Finally, when all the modules have been successfully integrated and tested, system testing is carried out. The goal of system testing is to ensure that the developed system conforms to its requirements laid out in the SRS document.

**CONCLUSIONS**

* That’s it!!!
* Here we are completed with our GUI Project using Python Tkinter.
* With these steps, we have successfully created a random password generator project using python. We used popular tkinter library to rendering graphics in our display window and we also learned about pyperclip and random library.
* We learned how to create buttons, input textfield, labels, and spinbox. In this way, we successfully created our password generator python project. Hope you enjoyed it.

**BIBLIOGRAPHY**

* + Books:

▪ **Python Programming: An Introduction to Computer Science (3rd Edition)**

* + Websites:

[**https://www.python.org/**](https://www.python.org/) [**https://www.geeksforgeeks.org/**](https://www.geeksforgeeks.org/) [**https://www.wikipedia.org/**](https://www.wikipedia.org/) [**https://github.com/**](https://github.com/)