ABSTRACT

A **password generator** is software program or hardware device that takes input from a random or pseudo-random number generator and automatically generates a 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. The password generator tool using python program with help of kali linux.

The users of computer technology and internet are increasing day by day. As the users are growing , the need for security is also felt very much. The data assets and other valuable facts are stored in the computer systems. One of the way to safe guard the data assets is to have a proper authorization method to access the data. This is achieved by user identification and password mechanism.

The selection of password is important, since the entire authorization is dependent on the password. The password need to be strong enough to avoid brute force attack and other attacks. Here we discuss a method of generating random passwords, which are strong enough to combat the attacks.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No
	ABSTRACT	i
	LIST OF FIGURES	iii
1	**INTRODUCTION**	1
	1.1 About Password	1
	1.2 Random Generator Tool	1
2	**AIM AND SCOPE OF PASSWORD GENERATOR**	2
	2.1 Aim	2
	2.2 Scope for Generator Tool	2
3	**EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHMS USED**	3
	3.1 New Password Encryption Method	3
	3.2 New Decryption Method	4
	3.3 Stronger Methods of Python with sample code	5
	3.4 Algorithm	7
4	**RESULTS**	7
5	**SUMMARY AND CONCLUSIONS**	8
	REFERENCES	8
6	**APPENDIX**	9
	A.SCREENSHOTS	9
	B. SOURCE CODE	11

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO
Fig 6.1	To start the VM VirtualBox Manager	9
Fig 6.2	Enter in to kali linux	9
Fig 6.3	To open the program	10
Fig 6.4	Output of password generator tool	10

1) INTRODUCTION

1.1 About Password:

A password is a string of characters used to verify the identity of a user during the authentication process . Passwords are typically used in tandem with a username; they are designed to be known only to the user and allow that user to gain access to a device, application or website.

1.2 Random Generator Tool:

Random password generator is to produce random password with high security. Generally, random passwords have various benefits over user-chosen password where it

enhances security and confidentiality. The new methodology has been created to generate random password which consists of both upper- & lower-case letter and digits from 0 to 9, with fixed length. The alphanumeric is a simple algorithm that generates random password with predetermined length. The password generator algorithm selects a random character form random character list and forms the password, which is combination of numbers, lower & upper-case letters.

2) AND SCOPE OF PASSWORD GENERATOR

2.1 Aim:

To create a program on password generator tool using python programming language with the help of kali linux.

2.2 Scope for Generator Tool:

The scope of our project is to generate a password instead of making a secured password .hardware device, or online tool that automatically generates a password using parameters that a user sets, including mixed-case letters, numbers, symbols, pronounceability, length, and strength To analyze the performance of proposed work of Password Generator .

3) EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHMS USED

3.1 New Password Encryption Method :

Cryptography is a process of converting ordinary text or plain text to cipher text (encryption), then converting back to the original text (decryption). There are several ways to classify the various algorithms. The most common types are i) Secret Key Cryptography which is also known as Symmetric Key Cryptography and ii) Public Key

Cryptography which is also known as Asymmetric Key cryptography. The algorithm that we are adopting is the symmetric key cryptographic technique. The key that is used to encrypt the password is 4 digits binary number (>=1000). This algorithm maintains three separate variable lists that contain upper and lower case letters and numbers. To encrypt the password, the algorithm checks the alphabets in an inputted password, and finds out whether the alphabet is an upper-case/lower-case/number and then corresponding constant is used with the character.

The algorithm consists of three parts. The first part is to convert the password to decimal value. The second part is to perform binary manipulations. The third part is to converting binary value to alpha-numerical value. In the first step there are three cases to change alphanumeric into decimal value: (1) If the alphabet is upper-case letter, and then the ASCII value of alphabet and constant 50 are summed. (2) If the alphabet is lower-case letter, then the constant 20 is subtracted from the ASCII value of alphabet

(3) If the alphabet is number, and then the constant 10 is subtracted from the discovered ASCII value.

The second step converts the decimal value received from the previous procedure into binary and reverses the binary value. The reversed binary value is divided by the key. The key is selected by the user. The remainder and the quotient are formed as resultant binary value. The remainder in first 3 digits and quotient in next 5 digits are formed. The final step is to convert the resultant binary value to alphanumerical value.

3.2 *New* Decryption Method:

Encryption is the process of converting plaintext into cipher text. Decryption is the reverse of encryption where the cipher text is converted into plaintext. New type of decryption algorithm has to be generated to decrypt the encrypted password. The symmetric key cryptography is using same key for both encryption and decryption.

The decryption process will be taken according to the following step: This algorithm maintains the separate arrays such as upper case array list, lower case array list and numbers array list that hold certain values for both upper and lower case alphabets and digits from 0 to 9. The key used for decryption that must be same as key used for encrypting the password. The encrypted password is in hexadecimal format.

First step of decryption is to convert the each character to ASCII value. Each ASCII decimal value is converted to binary digits.The last 5 digits are multiplied by the key. The first 3 digits of the cipher text are added with the result produced through the multiplication. If the result produced is not an 8-bit number, then zeros are added at the left to make it 8- bit number. Now the 8-bit digits are reversed.

In the second step of decryption reversed binary digits are converted to a decimal value. The decimal value is checked against the array list using the different range of values for each array. For Upper-case array, the values start from 115 to 140, for Lower-case array, the values start from 77 to 102, and Number array start from 38 to 47. The range of the decimal value is checked to find the array list. If the decimal is in upper-case array list, subtract the constant 50 from the decimal value. This is the ASCII value of alphabet in random password. If the decimal is in lower-case array list, then add constant 20 with the decimal value. This process produces the ASCII value and then the alphabet for ASCII value is discovered. If the decimal is in number array list, then the constant 10 added with the decimal value. The resultant value is the ASCII value of the character. In the final step, the ASCII value is converted to character, which produces the decrypted form.

3.3 Stronger Methods of Python with sample code :

A variety of methods exist for generating strong, cryptographically secure random passwords. On Unix platforms /dev/random and /dev/unransom are commonly used, either programmatically or in conjunction with a program such as makepasswd. Windows programmers can use the Cryptographic Application Programming Interface function CryptGenRandom .The Java programming language includes a class called *SecureRandom*. Another possibility is to derive randomness by measuring some external phenomenon, such as timing user keyboard input.

Sample code:

```
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)
```

3.4 Algorithm:

Step 1: Start the process

Step 2: Create random character list with numbers, upper & lower-case letters.

Step 3: Password must be in fixed length of 12.

Step 4: Create Random Password Generator method to generate the password.

Step 5: Random Password Generator chooses any of the three character set.

Step 6: The index position of any one of the characters from the random character set is returned.

Step 7: Append the characters selected through the index, one by one.

Step 8: Print the password.

Step 9: End.

4) RESULTS

Since all the applications are protected with passwords, more research can be accomplished for secured automatic password generations. The proposed method uses only the alphabets and numerical values for random character list. Still special symbols could be considered for strengthening the password. The password length also can be extended to make the password strong. New encryption and decryption standard could be implemented with the randomly selected passwords. The experimental study can be done with large number of samples in future.

5) SUMMARY AND CONCLUSIONS

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 costeffective. The above done work also creates awareness and interest to start exploring this field more.

REFERENCES

1) Art Conklin, Glenn Dietrich, Diane Walz, "Password-Based Authentication: A System Perspective", Proceedings of the 37th Hawaii International Conference on System Sciences – 2004.

2) Thorsten Brantz and Alex Franz,The Google Web 1T 5-gram corpus, Technical Report LDC2006T13, Linguistic Data Consortium, 2006.

3) Levine, John R., Ed.: *Internet Secrets*, Second edition, page 831 ff. John Wiley and Sons.

4) Burr, W. E.; Dodson, D. F.; Polk, W. T. (2006). "ElectronicAuthenticationGuideline" (PDF).NIST. doi:10.6028/NIST.SP.800-63v1.0.2.

6) **APPENDIX**

A.) SCREENSHOTS:

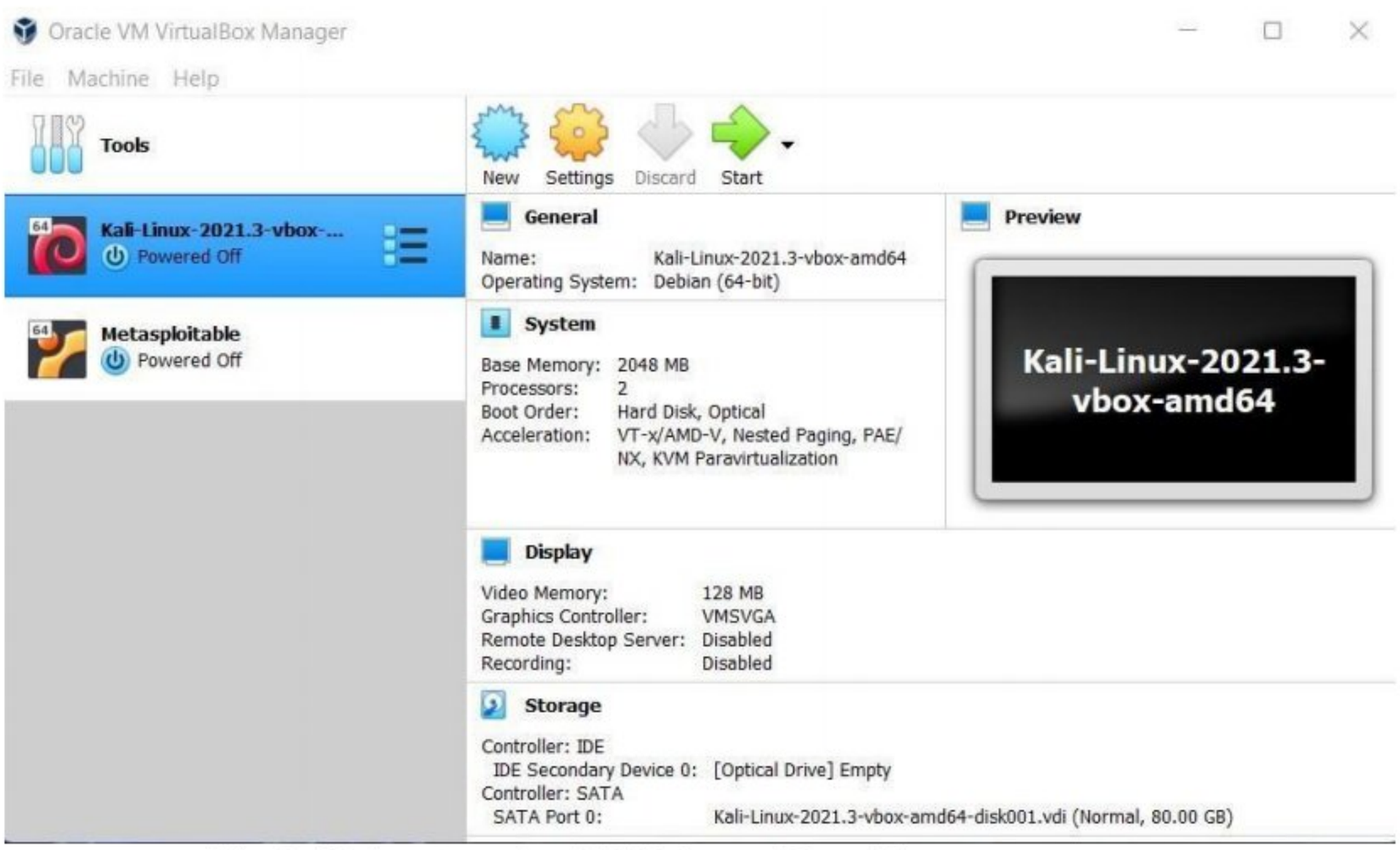

Fig 6.1 To start the VM VirtualBox Manager

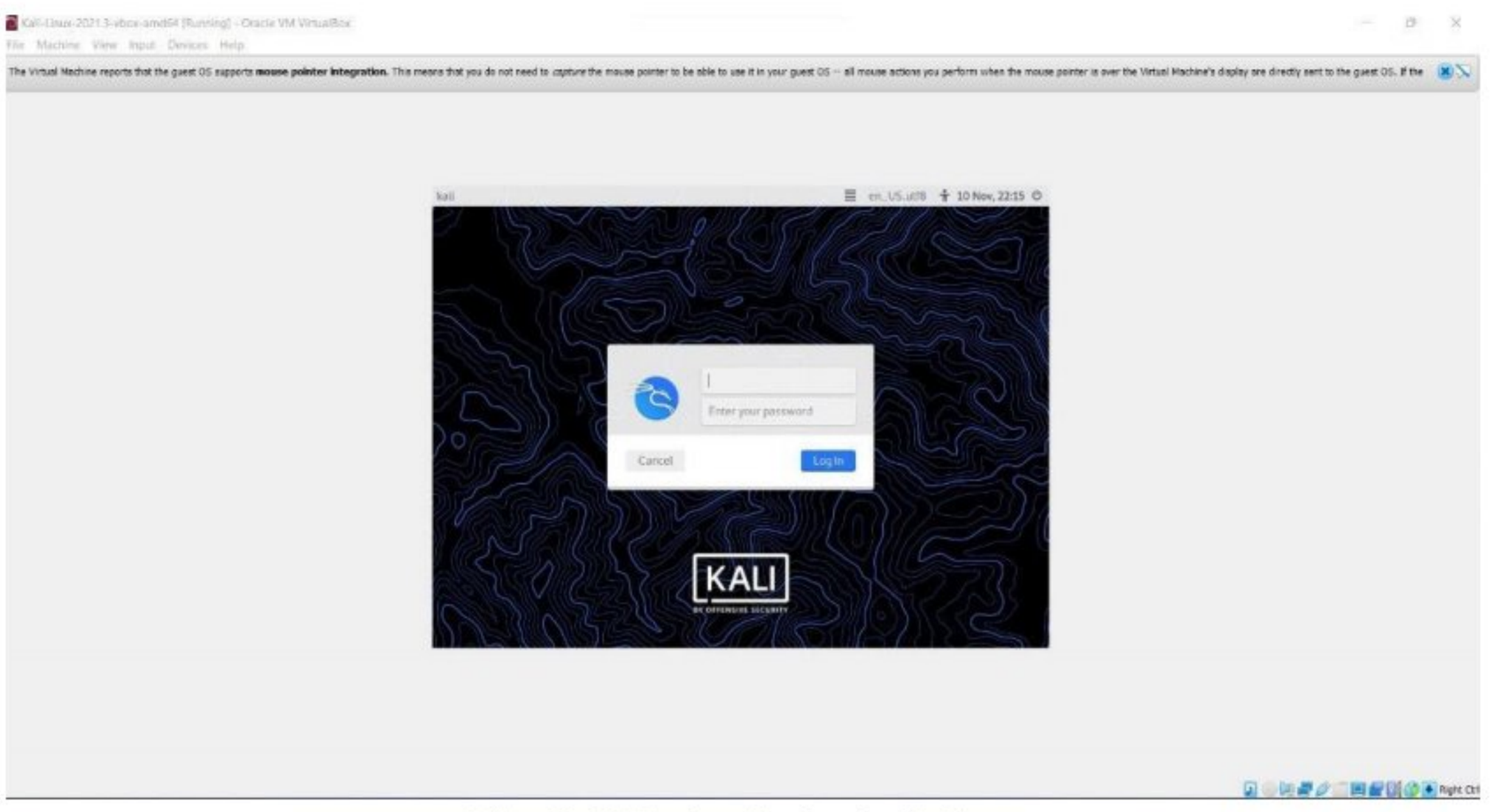

Fig 6.2 Enter in to kali linux

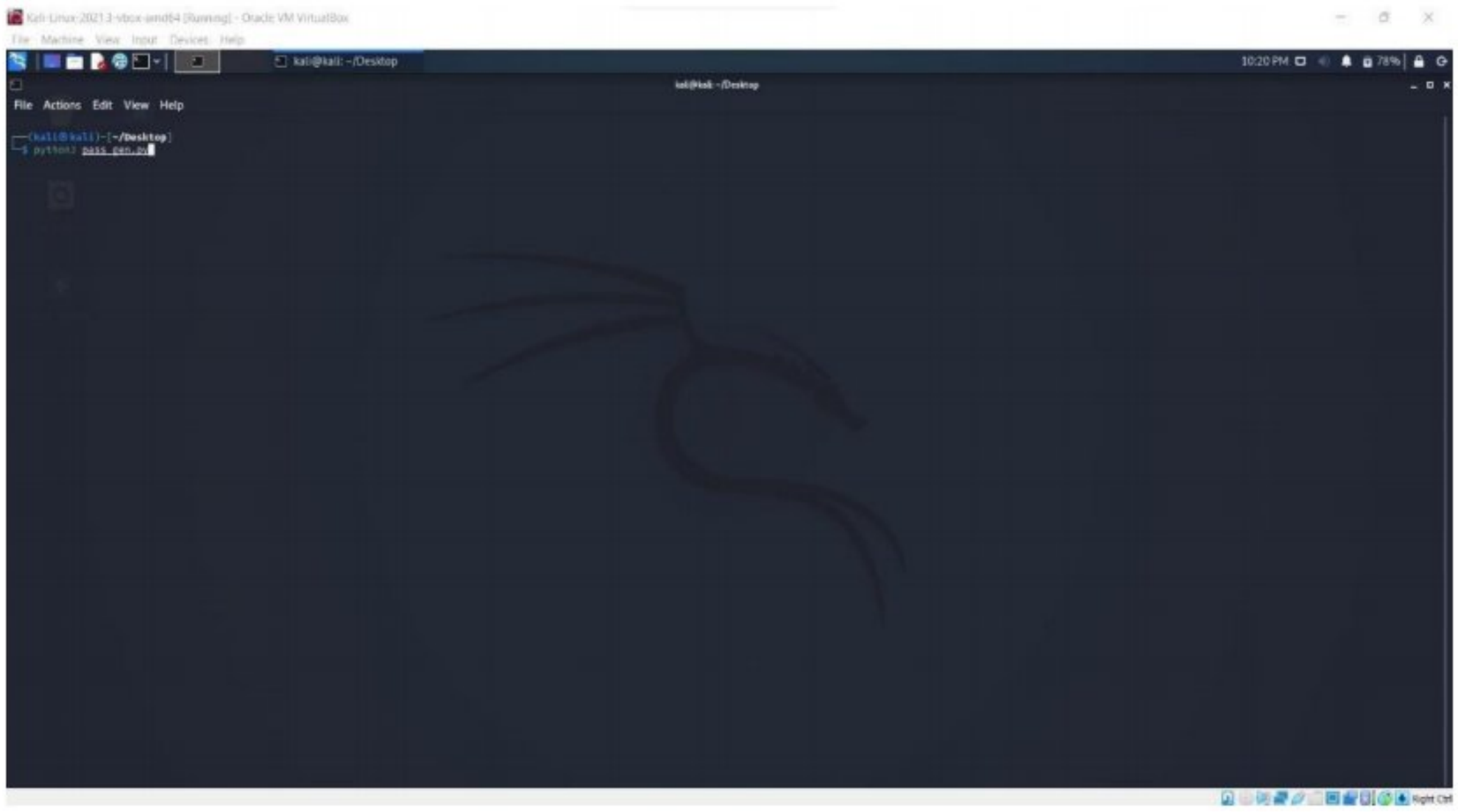

Fig 6.3 To open the program

Fig 6.4 Output of password generator tool

B. SOURCE CODE :

```
from tkinter import *
import random, string

 root = Tk()
root.geometry("500x300")
root.title("Password Generator Tool")

title = StringVar()
label = Label(root, textvariable=title).pack()
title.set("Select the strength of the password:")

  def selection():
   global selection
   selection = choice.get()

choice = IntVar()
R1 = Radiobutton(root, text="Low", variable=choice, value=1,
command=selection).pack(anchor=CENTER)
R2 = Radiobutton(root, text="Medium", variable=choice, value=2,
command=selection).pack(anchor=CENTER)
R3 = Radiobutton(root, text="High", variable=choice, value=3,
command=selection).pack(anchor=CENTER)
labelchoice = Label(root)
labelchoice.pack()

lenlabel = StringVar()
lenlabel.set(" Enter a Password length:")
```

```
lentitle = Label(root, textvariable=lenlabel).pack()

val = IntVar()
spinlenght = Spinbox(root, from_=4, to_=30, textvariable=val,
width=5).pack()
   def callback():
    lsum.config(text=passgen())
```

```
passgenButton = Button(root, text="GENERATE", bd=9,
height=1, command=callback, pady=1, bg='#0059b3')
passgenButton.pack()
password = str(callback)
lsum = Label(root, text="")
lsum.pack(side=BOTTOM)
poor= string.ascii_uppercase + string.ascii_lowercase
average= string.ascii_uppercase + string.ascii_lowercase +
string.digits
symbols = """`~!@#$%^&*()_-+={}[]\|:;"'<>,.?/"""
advance = poor+ average + symbols
def passgen():
    if choice.get() == 1:
        return "".join(random.sample(poor, val.get()))
    elif choice.get() == 2:
        return "".join(random.sample(average, val.get()))
    elif choice.get() == 3:
        return "".join(random.sample(advance, val.get()))
root.mainloop()
```