



Project Report

PROJECT TITLE :- Random Password Generator

SUBMITTED TO:-

Mr. Mayur Dev Sewak
General Manager, Operations
Eisystems Services

&

Ms. Mallika Srivastava
Trainer, Data Analytics Domain
Eisystems Services

SUBMITTED BY:-

Devanshu Bareley
Email id:-devanshubareley5@gmail.com
Samrat Ashok Technological Institute
(Engineering College), Vidisha

DATE OF SUBMISSION:- JUNE 20,2020

Content Table

Serial Number	Title	Page Number
1	Cover Page	1
2	Content Table	2
3	List of Figures	2
4	Abstract of Project	3
5	Project Summary	4
6	Objectives of Project	5
6.1	Introduction	6
6.2	Preamble	6
6.3	Motivation	7
6.4	Methodologies	8
7	Details of Process	9
8	Components	10
9	Algorithm	11
10	Data Flow Diagram	12
11	Program Code	13
12	Output	16
13	References	17

List of Figures

Figure Number	Caption
Fig 1.1	Data Flow Diagram
Fig 2.1	Code input

Abstract of Project

Random Password Generator:

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. With growing technology, everything has relied on data and securing these data is strong.

Let's create a simple application which can randomly generate strong passwords using Python module.

This application can generate random password, with the combination of letters, numerics, and special characters. One can mention length of the password based on requirement and can also select the strength of the password.

Project Summary

Having a weak password is not good for a system which demands high confidentiality and security of user credentials. It turns out that people find it difficult making up a strong password which is strong enough to prevent unauthorized users from memorizing it.

This article use a mixture of numbers, alphabets and other symbols found on the computer keyboard to form a 12-character password which is unpredictable and cannot easily be memorized.

- The components of the password are represented in the form of arrays.
- Use the random method to select at least one character from each array of characters.
- Since 12-character password is required, so fill the rest of the length of the password left with randomly selected characters from an array formed from the combination of all the character needed in the final password
anytime the password is generated, shuffle the password using `random.shuffle()` to ensure that the final password does not follow a particular pattern.

Objectives of Project

A random 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.

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 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.

Introduction

Some password generators are simply random password generators. These programs produce complex/strong passwords with combinations of numbers, uppercase and lowercase letters, and special characters such as braces, asterisks, slashes, etc.

Preamble

Generate strong, random passwords

Passwords are a real security threat. Over 80% of hacking-related breaches are due to weak or stolen passwords, a recent report shows . So if you want to safeguard your personal info and assets, creating secure passwords is a big first step. And that's where the LastPass Password Generator can help. Impossible-to-crack passwords are complex with multiple types of characters (numbers, letters, and symbols). Making your passwords different for each website or app also helps defend against hacking. This password generator tool runs locally on your Windows, Mac or Linux computer, as well as your iOS or Android device. The passwords you generate are never sent across the web.

1. Always use a unique password for each account you create. The danger with reusing passwords is that as soon as one site has a security issue, it's very easy for hackers to try the same username and password combination on other websites.
2. Don't use any personally identifiable information in your passwords. Names, birthdays, and street addresses may be easy to remember but they're also easily found online and should always be avoided in passwords to ensure the greatest strength.
3. Make sure your passwords are at least 12 characters long and contain letters, numbers, and special characters. Some people prefer to generate passwords which are 14 or 20 characters in length.
4. If you're creating a master password that you'll need to remember, try using phrases or lyrics from your favorite movie or song. Just add random characters, but don't replace them in easy patterns.

5. Use a password manager like LastPass to save your passwords. We keep your information protected from attacks or snooping.
6. Avoid weak, commonly used passwords like asd123, password1, or Temp!. Some examples of a strong password include: S&2x4S12nLS1*, JANa@sx3l2&s\$, 49915w5\$0YmH.
7. Avoid using personal information for your security questions, instead, use LastPass to generate another "password" and store it as the answer to these questions. The reason? Some of this information, like the name of the street you grew up on or your mother's maiden name, is easily found by hackers and can be used in a brute-force attack to gain access to your accounts.
8. Avoid using similar passwords that change only a single word or character. This practice weakens your account security across multiple sites.
9. Change your passwords when you have reason to, such as after you've shared them with someone, after a website has had a breach, or if it's been over a year since you last rotated it.
10. You should never share your passwords via email or text message. The secure way to share is with a tool like LastPass that gives you the ability to share a hidden password and even revoke access when the time comes.

Motivation

Change up your passwords: If you are still using your password from 2010, it's time to change it up. Try using my tips for creating a secure and memorable password here.

Next you want to generate a password. To do this you can use a random password generator (this one has a bookmarklet so you can do it while you are on the site you are creating a password for!) that will take your goal or reminder and turn it into a secure password, or you can simply add your own random characters. For instance:

- #!@Lose5<>Nov09-1
- rewq-5/11-09uiop
- 4thLOSE5by11/09ht4
- 678LOSE5Nov09zxc

The possibilities are pretty endless. Giving you both a motivating password, and a means to make it difficult to hack.

Methodologies

1: Using OpenSSL

2: Using the pwgen utility

3: Using the GPG utility

4: Using the perl utility

5: Using the Revelation UI Application

6: Using the UI Keepassx application

Details of Process

A random 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.

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.

Long strings of random characters are difficult for most people to memorize. Mnemonic hashes, which reversibly convert random strings into more memorable passwords, can substantially improve the ease of memorization. As the hash can be processed by a computer to recover the original 60-bit string, it has at least as much information content as the original string.^[1] Similar techniques are used in memory sport.

Components

Having a weak password is not good for a system which demands high confidentiality and security of user credentials. It turns out that people find it difficult making up a strong password which is strong enough to prevent unauthorized users from memorizing it.

This article use a mixture of numbers, alphabets and other symbols found on the computer keyboard to form a 12-character password which is unpredictable and cannot easily be memorized.

- The components of the password are represented in the form of arrays.
- Use the random method to select at least one character from each array of characters.
- Since 12-character password is required, so fill the rest of the length of the
- password left with randomly selected characters from an array formed from the
- combination of all the character needed in the final password
- anytime the password is generated, shuffle the password using
- `random.shuffle()` to ensure that the final password does not follow a particular pattern.

Algorithm of random password generator

Procedure:

Step 1: Start the process.

Step 2: Store upper-case letters, lower-case letters and numbers as separate variable lists.

Step 3: Random password is used as input.

Step 4: Conversion from the alphabets to decimal notations.

4.1. Check the alphabet is upper-case. Find out its ASCII value and add with constant 50.

4.2 Check the alphabet is lower-case. Find out its ASCII value and subtract the constant 20 from the ASCII value.

4.3 Check the alphabet is Number. Find out its ASCII value and subtract the constant 10 from the ASCII value.

Step 5: Convert the resultant decimal value to binary digits.

Step 6: Reverse the binary digits.

Step 7: Get the key from the user.

Step 8: Divide the reversed binary digit by the key.

Step 9: Store the remainder in first 3 digits & quotient in next 5 digits (remainder and quotient wouldn't be more than 3 digits and 5 digits long respectively. If any of these are less than 3 and 5 digits respectively we need to add required number of 0s (zeros) in the left hand side.

Step 10: Convert the binary to decimal value.

Step 11: Consider the decimal value as the ASCII value and find the related character as encrypted value.

Data Flow Diagram of Random Password Generator



Fig 1.1 Data Flow Diagram

Program Code

Input code of random password generator is:-

Generate a random string of fixed length

```
import random
import string

def randomString(stringLength=8):
    letters = string.ascii_lowercase
    return ''.join(random.choice(letters) for i in range(stringLength))

print("Random String is ", randomString())
print("Random String is ", randomString(8))
print("Random String is ", randomString(8))
```

Generate a random string of lower case and upper case letters

```
def randomString(stringLength):
    letters = string.ascii_letters
    return ''.join(random.choice(letters) for i in range(stringLength))

print("Random String with the combination of lowercase and uppercase letters")
print("First Random String is ", randomString(8))
print("second Random String is ", randomString(8))
```

Generate a random string of specific letters only

```
def randString(length=5):
    # put your letters in the following string
    your_letters='abcdefghi'
    return ''.join((random.choice(your_letters) for i in range(length)))

print("Random String with specific letters ", randString())
print("Random String with specific letters ", randString(5))
```

Generate a random string without repeating characters

```
def randomString2(stringLength=8):
    letters = string.ascii_lowercase
    return ''.join(random.sample(letters, stringLength))

print("Random String is ", randomString2())
print("Random String is ", randomString2(8))
```

Generate a random alphanumeric string of letters and digits

```
def get_random_alphaNumeric_string(stringLength=8):
```

```
lettersAndDigits = string.ascii_letters + string.digits
return ''.join((random.choice(lettersAndDigits) for i in range(stringLength)))
```

```
print("First alphaNumeric Random String is ", get_random_alphaNumeric_string(8))
print("Second alphaNumeric Random String is ", get_random_alphaNumeric_string(8))
print("Third alphaNumeric Random String is ", get_random_alphaNumeric_string(8))
```

Generate a random alphanumeric string with a fixed count of letters and digits

```
def random_alphaNumeric_string(lettersCount, digitsCount):
    sampleStr = ''.join((random.choice(string.ascii_letters) for i in range(lettersCount)))
    sampleStr += ''.join((random.choice(string.digits) for i in range(digitsCount)))
```

Convert string to list and shuffle it to mix letters and digits

```
sampleList = list(sampleStr)
random.shuffle(sampleList)
finalString = ''.join(sampleList)
return finalString
```

```
print("First Random alphaNumeric String is ", random_alphaNumeric_string(5, 3))
print("Second Random alphaNumeric String is ", random_alphaNumeric_string(6, 2))
```

Generate a random password string with Special characters, letters, and digits

```
def randomStringwithDigitsAndSymbols(stringLength=10):
    password_characters = string.ascii_letters + string.digits + string.punctuation
    return ''.join(random.choice(password_characters) for i in range(stringLength))
```

```
print("Generating Random String password with letters, digits and special characters ")
print ("First Random String ", randomStringwithDigitsAndSymbols() )
print ("Second Random String", randomStringwithDigitsAndSymbols(10) )
print ("Third Random String", randomStringwithDigitsAndSymbols(10))
```

Approach Second

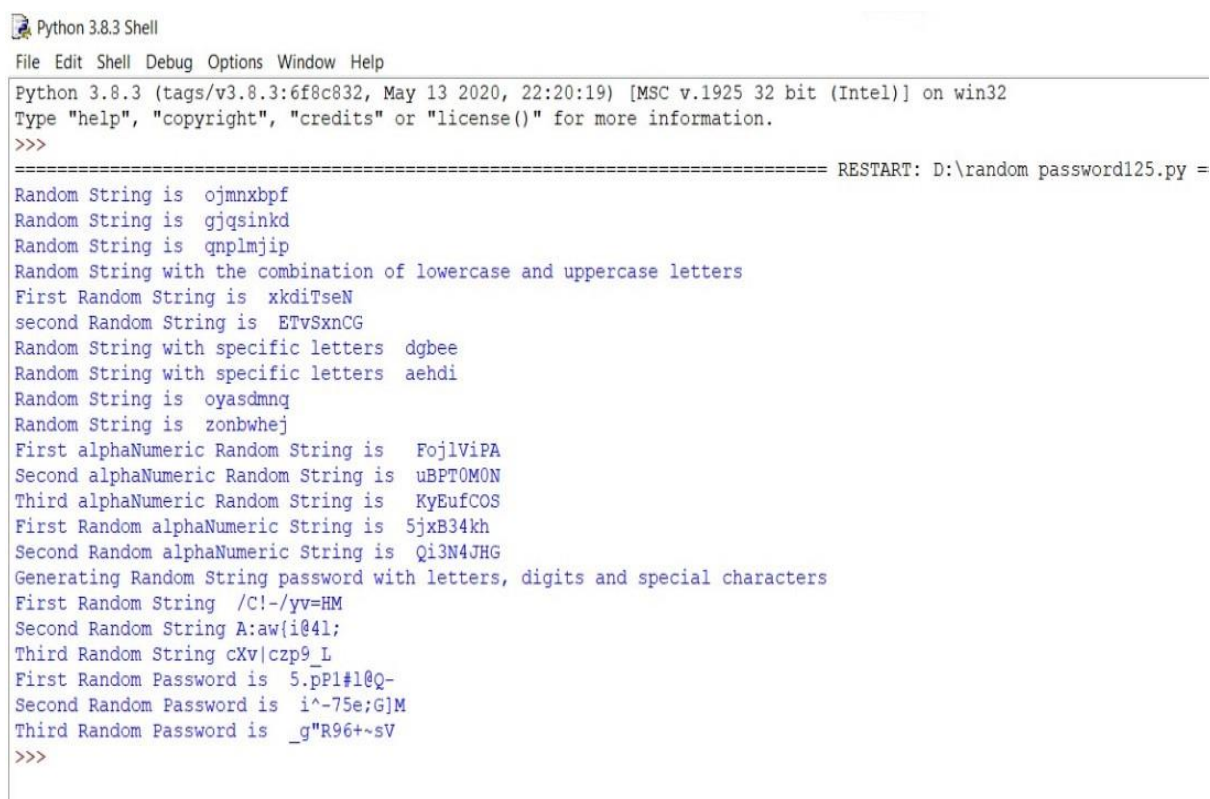
```
def randomPassword():
    randomSource = string.ascii_letters + string.digits + string.punctuation
    password = random.choice(string.ascii_lowercase)
    password += random.choice(string.ascii_uppercase)
    password += random.choice(string.digits)
    password += random.choice(string.punctuation)

    for i in range(6):
        password += random.choice(randomSource)

    passwordList = list(password)
    random.SystemRandom().shuffle(passwordList)
    password = ''.join(passwordList)
    return password

print ("First Random Password is ", randomPassword())
print ("Second Random Password is ", randomPassword())
print ("Third Random Password is ", randomPassword())
```

Output of above code



```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\random password125.py =
Random String is ojmnbxpf
Random String is gjqsinkd
Random String is qnplmjip
Random String with the combination of lowercase and uppercase letters
First Random String is xkdiTseN
second Random String is ETvSxnCG
Random String with specific letters dgbee
Random String with specific letters aehdi
Random String is oyasdmng
Random String is zonbwhej
First alphaNumeric Random String is FojlViPA
Second alphaNumeric Random String is uBPTOMON
Third alphaNumeric Random String is KyEufCOS
First Random alphaNumeric String is 5jxB34kh
Second Random alphaNumeric String is Qi3N4JHG
Generating Random String password with letters, digits and special characters
First Random String /C!-/yv=HM
Second Random String A:aw{i@4l;
Third Random String cXv|czp9_L
First Random Password is 5.pPl#l@Q-
Second Random Password is i^-75e;G]M
Third Random Password is _g"R96+~sV
>>>
```

Fig 2.1 Code input

References

- [1] US Dept. of Defense. Password Management Guideline, CSC-STD-002-85, 1985.
- [2] Cynthia Kuo, Sasha Romanosky, and Lorrie Faith Cranor. Human Selection of Mnemonic Phrase-based Passwords. Symposium On Usable Privacy and Security (SOUPS) 2006, Pittsburgh, Pennsylvania, USA, July 2006.
- [3] S. Gaw and Edward W. Felten. Password Management Strategies for Online Accounts. Symposium On Usable Privacy and Security (SOUPS) 2006, Pittsburgh, Pennsylvania, USA, July 2006.
- [4] J. Yan, A. Blackwell, R. Anderson, and A. Grant. Password Memorability and Security: Empirical Results. IEEE Security and Privacy, 2
- (5), pp. 25—31, 2004. [5] J. Bunnell, J. Podd, R. Henderson, R. Napier, and J. Kennedy-Moffat. Cognitive, associative and conventional passwords: Recall and guessing rates. Computers and Security, Vol. 16, No. 7, pp. 645—657, 1997.
- [6] Arnold G. Reinhold. The Diceware Passphrase Homepage, <http://www.diceware.com/>, accessed 14 October, 2006.
- [7] Arnold G. Reinhold. Diceware English Wordlist, <http://world.std.com/~reinhold/diceware.wordlist.asc>, accessed 14 October, 2006.
- [8] R. Ganesan and C. Davies. A New Attack on Random Pronounceable Password Generators. Proceedings of the 17th NIST-NCSC National Computer Security Conference, 1994. pp. 184—197.