Password Generator Report

Thomas Parker

Oxford Brookes

Contents

Introduction:

Websites that handle users personal and confidential information require a secure password so that any user logging onto the website are validated securely before allowing them entry into the site.

The programme created here will be designed to generate a secure password so that it can be used for these types of websites.

This password will be between seven and fourteen characters long consisting of only alphanumeric characters that are both upper and lower case. The password generated will also follow no discernible pattern and will not allow two identical characters next to each other.

Informal Design and Explanation:

First there will be an empty list created in which to insert the generated password into.

```
password = [ ]
```

After this there will be a function created which generates a list of random characters that consist only of upper-case & lower-case characters and digits. A variable named ranNum will be generated prior to this which will ensure that the password created will be a random size between seven and fourteen characters long.

```
for i in range(0, random number):
    code = random numbers + random lower & upper characters
```

This will then be appended to our empty list.

```
password.append(code)
```

While the password is still in the form of a list, a function will then be ran in order to check that there are no duplicate characters whether upper-case or not.

```
charRemove(password)
```

This function uses a for loop to cycle through the code and replaces any characters that are the same as each other and replaces them with a random letter using a previously assigned variable.

```
def charRemove(password):
  p = 1
  for i in range (len(password)-1):
    if password[i] == password[p].lower() or password[i] == password[p].upper() or password[i] == password[p]:
      password.remove(password[p])
      password.append(randomletter)
```

Should a random letter appended to the list be the duplicate of another consecutive character, this next loop will append another random letter and replace it as a failsafe.

```
      if password[i] == password[p].lower() or password[i] == password[p].upper() or password[i] == password[p]:
        password.remove(password[p])
        password.append(randomletter)
    p+=1
    if p > (len(password)):
      return password
```

Once this has been completed, the password will be returned to the password generator function where it will check that the password has upper-case & lower-case characters and digits. This will print whether the password meets this criterion. Failing to do this, the list will be emptied and re-ran again to generate a new password.

```
rules = [must contain uppers),
    must contain lower),
    must contain digit),
    ]
  if all(rule(password) for rule in rules):
    print("PASS: Password contains upper case and lower case characters & digits")
    return password
  else:
    delete password[:]
    generatePassword()
```

Once the password has been generated, it will then be converted into a string.

```
password = (Convert into string)
```

This string will then go through the process of being validated and printed off.

```
print(validatePassword(password))
```

The first part of the validation process will check that the string is equal to or greater than seven characters long and equal to and less than fourteen characters long. This will print whether the password meets this criterion.

```
def validatePassword(password):
    length = len(password)
    # Tests that password is correct length
    if length >=7 <= 14:
        print("PASS: Password is in correct length")
    else:
        print("FAIL: Password is not in correct length")
```

The second part will check that the password only contains alphanumeric characters. This will print whether the password meets this criterion.

```
if password.isalnum() == True:
        print("PASS: Password only contains alphanumeric characters")
    else:
        print("FAIL: Password does not only contain alphanumeric characters")
```

Lastly, the validate function will confirm whether there are any duplicate characters and print whether this has been successful or not.

```
for i in range (len(password)-1):
        if password[i] == password[p].lower() or password[i] == password[p].upper() or password[i] ==
password[p]:
            print(password[i])
            print(password[p])
            print("FAIL: Duplicates found")
        p+=1
        if p > (len(password)):
            print("PASS: No duplicates found")
            return password
```

```
else:
    print("PASS: No duplicates found" - return password
```

Finally, the validated password will be returned and printed off, also printing whether the password meets the specification of each criterion.

Validate Password Test Case:

Code	Output	Explanation

length = len(password)
  if length >=7 <= 14:
  print("PASS: Password is in correct length")
  else:
  print("FAIL: Password is not in correct length")
	If the validation is successful it will print “PASS: Password is in correct length” Otherwise it will print "FAIL: Password is not in correct length")	This code tests that the password is equal to or greater the seven characters and equal to or less than fourteen characters

password.isalnum()
if password.isalnum() == True:
  print("PASS: Password only contains alphanumeric characters")
  else:
  print("FAIL: Password does not only contain alphanumeric characters")
	If the validation is successful it will print “PASS: Password only contains alphanumeric characters” Otherwise it will print “FAIL: Password does not only contain alphanumeric characters”	This code tests that the code only contains alphanumeric characters.

rules = [lambda password: any(x.isupper() for x in password),
  lambda password: any(x.islower() for x in password),
  lambda password: any(x.isdigit() for x in password),
  ]
  if all(rule(password) for rule in rules):
  print("PASS: Password contains upper case and lower case characters & digits")
  return password
  else:
  del password[:]
  generatePassword()
	If the validation is successful it will print “PASS: Password contains upper case and lower-case characters & digits” Otherwise it will delete the list and re-generate the password.	This code tests that the password contains upper case and lower-case characters and digits.

p = 1
  for I in range (len(password)-1):
  if password[i] == password[p].lower() or password[i] == password[p].upper() or password[i] == password[p]:
  print(password[i])
  print(password[p])
  print("FAIL: Duplicates found")
  p+=1
  if p > (len(password)):
	If the validation is successful it will print “PASS: No duplicates found” Otherwise it will print “FAIL: Duplicates found”	This code tests that two consecutive characters are not the same whether upper-case or lower-case.

print("PASS: No duplicates found") return password else: print("PASS: No duplicates found") return password		

Password Generator Programme Code:

```
# Impots librarys

import string

import random


#Creates empty lisst

password = [ ]


#Generates random letter

ranLet = random.choice(string.ascii_letters)


#Creates a random number

ranNum = (random.randint(7,14))


# Generates password

def generatePassword():

  for i in range(0, ranNum):

    code = random.choice(string.ascii_uppercase + string.digits + string.ascii_letters)

    password.append(code)

    charRemove(password)

    # Tests that there is upper case and lower case characters & digits

    rules = [lambda password: any(x.isupper() for x in password),

    lambda password: any(x.islower() for x in password),

    lambda password: any(x.isdigit() for x in password),

    ]
```

```
    if all(rule(password) for rule in rules):
        print("PASS: Password contains upper case and lower case characters & digits")
        return password
    else:
        del password[:]
        generatePassword()


# Removes duplicate characters
def charRemove(password):
    p = 1
    for i in range (len(password)-1):
        if password[i] == password[p].lower() or password[i] == password[p].upper() or password[i] ==
password[p]:
            password.remove(password[p])
            password.append(ranLet)
            # Extra failsafe
            if password[i] == password[p].lower() or password[i] == password[p].upper() or password[i] ==
password[p]:
                password.remove(password[p])
                password.append(ranLet)
        p+=1
        if p > (len(password)):
            return password


# Validates Password
def validatePassword(password):
    length = len(password)
    password.isalnum()
    # Tests that password is correct length
    if length >=7 <= 14:
        print("PASS: Password is in correct length")
    else:
```

```
        print("FAIL: Password is not in correct length")
    # Tests that password only contain alphanumeric characters
    if password.isalnum() == True:
        print("PASS: Password only contains alphanumeric characters")
    else:
        print("FAIL: Password does not only contain alphanumeric characters")
    # Tests there are no duplicates
    p = 1
    for i in range (len(password)-1):
        if password[i] == password[p].lower() or password[i] == password[p].upper() or password[i] ==
password[p]:
            print(password[i])
            print(password[p])
            print("FAIL: Duplicates found")
        p+=1
        if p > (len(password)):
            print("PASS: No duplicates found")
            return password
        else:
            print("PASS: No duplicates found")
            return password


# Generates password
generatePassword()


#Turns password into string
password = ''.join(password)


# Makes sure password is secure and then displays it
print(validatePassword(password))
```

Bibliography

Rooy, J. L., 2013. *https://stackoverflow.com.* [Online]
Available at: https://stackoverflow.com/questions/17140408/if-statement-to-check-whether-a-string-has-a-capital-letter-a-lower-case-letter
[Accessed 8th December 2017].

Unknown, Unknown. *http://www.practicepython.org.* [Online]
Available at: http://www.practicepython.org/solution/2014/06/06/16-password-generator-solutions.html
[Accessed 28th November 2017].

Unknown, Unknown. *https://www.tutorialspoint.com.* [Online]
Available at: https://www.tutorialspoint.com/python/string_isalnum.htm
[Accessed 8th December 2017].

Vazquez-Abrams, I., 2013. *https://stackoverflow.com.* [Online]
Available at: https://stackoverflow.com/questions/2257441/random-string-generation-with-upper-case-letters-and-digits-in-python
[Accessed 24 November 2017].