./



GENESIS –Advanced Python

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
| 1.0 | 11-12-20 | 1. Megha Annie George |  | Srinivas K |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Details**

Advanced Python

Contents

[Contents 3](#_Toc58581487)

[List of Figures 3](#_Toc58581488)

[1 PYTHON CODE 3](#_Toc58581489)

[2 PEP8 SCREENSHOT 4](#_Toc58581490)

[3 INPUT FILE SCREENSHOT 5](#_Toc58581491)

[4 OUTPUT FILE SCREENSHOT 6](#_Toc58581492)

# List of Figures

Figure 1 PEP8 SCREENSHOT 5

Figure 2 INPUT FILE SCREENSHOT 6

Figure 3 OUTPUT FILE SCREENSHOT 6

# PYTHON CODE

# Importing libraries

import random

import string as s

# To generate password

class password\_generator:

# Constructor

def \_\_init\_\_(self):

self.pwd\_len = 0

self.pwd\_stren = "weak"

self.gen\_pwd = ""

# To accept length

def get\_length(self, length):

try:

self.pwd\_len = int(length)

assert (self.pwd\_len > 4 and self.pwd\_len < 20)

except ValueError:

print("Password Length: Invalid! Input is not an integer")

except AssertionError:

print("Password Length: Invalid! Input must be between 4 and 20")

else:

print("Password Length: ", self.pwd\_len)

# To calculate password strength

def calc\_strength(self):

if (self.pwd\_len < 4 or self.pwd\_len > 20):

self.pwd\_stren = "Invalid"

elif (self.pwd\_len >= 4 and self.pwd\_len <= 7):

self.pwd\_stren = "Weak"

elif (self.pwd\_len >= 8 and self.pwd\_len <= 11):

self.pwd\_stren = "Moderate"

else:

self.pwd\_stren = "Strong"

print("Password Strength: ", self.pwd\_stren, '\n')

# To generate random password

def generate\_password(self):

if (self.pwd\_len < 4 or self.pwd\_len > 20):

self.gen\_pwd = "Invalid"

else:

self.gen\_pwd = random.choice(s.ascii\_uppercase) + \

random.choice(s.ascii\_lowercase) + \

random.choice(s.digits) + random.choice(s.punctuation)

for a in range(self.pwd\_len - 4):

self.gen\_pwd = self.gen\_pwd + random.choice(s.ascii\_uppercase +

s.ascii\_lowercase +

s.digits +

s.punctuation)

password\_list = list(self.gen\_pwd)

random.shuffle(password\_list)

self.gen\_pwd = ''.join(password\_list)

print("Generated password: ", self.gen\_pwd)

# To handle input and output files

class input\_output\_file\_handling:

def read\_write\_file(self):

object1 = password\_generator()

fr = open("input.txt", "r")

fw = open("output.txt", "a")

for each\_line in fr:

length = each\_line

object1.get\_length(length)

object1.generate\_password()

object1.calc\_strength()

fw.write("Password Length: " + each\_line +

"Generated Password: " + object1.gen\_pwd + '\n' +

"Password Strength: " + object1.pwd\_stren + '\n\n')

fr.close()

fw.close()

# To display password criteria

print("Welcome to Random Password Generator!")

print("\nThe password strength is as follows:")

print("\nWeak\t\t:\t4 - 7 Characters")

print("Moderate\t:\t8 - 11 Characters")

print("Strong\t\t:\t12+ Characters")

print("\nNote : Minimum password length should be 4\n")

in\_out\_obj = input\_output\_file\_handling()

in\_out\_obj.read\_write\_file()

# PEP8 SCREENSHOT

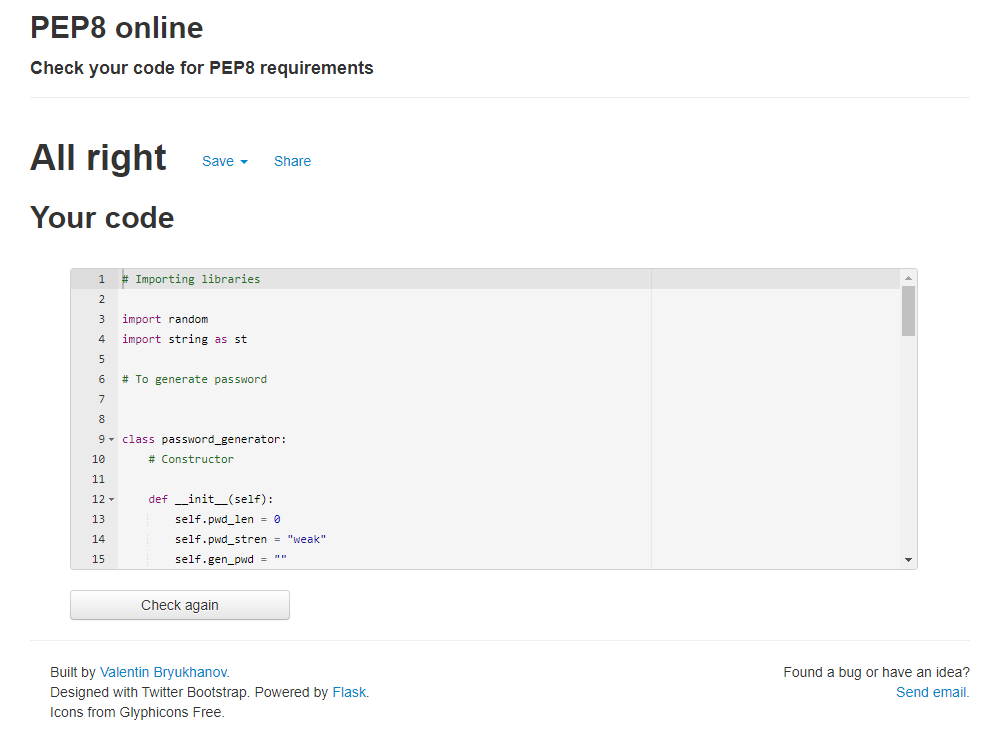


Figure 1 PEP8 SCREENSHOT

# INPUT FILE SCREENSHOT

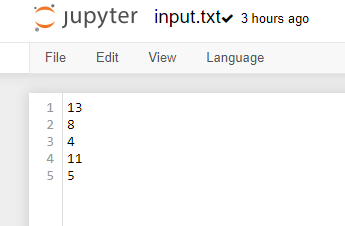


Figure 2 INPUT FILE SCREENSHOT

# OUTPUT FILE SCREENSHOT

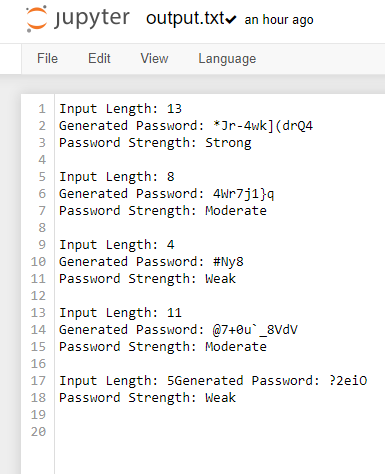


Figure 3 OUTPUT FILE SCREENSHOT