./



GENESIS – Advanced Python Learning Report

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 |  |
| 2.0 | 12-12-20 | 1. Megha Annie George |  | Srinivas K |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Details**

Random Password Generator

Contents

[Contents 3](#_Toc58753113)

[List of Figures 3](#_Toc58753114)

[ACTIVITY 1 – LEGACY CODE 3](#_Toc58753115)

[1 PYTHON CODE 4](#_Toc58753116)

[2 PEP8 SCREENSHOT 6](#_Toc58753117)

[3 INPUT FILE SCREENSHOT 6](#_Toc58753118)

[4 OUTPUT FILE SCREENSHOT 7](#_Toc58753119)

[ACTIVITY 2 – APPLYING OOP CONCEPTS 8](#_Toc58753120)

[1 PYTHON CODE 8](#_Toc58753121)

[2 PEP8 SCREENSHOT 10](#_Toc58753122)

[3 INPUT FILE SCREENSHOT 11](#_Toc58753123)

[4 OUTPUT FILE SCREENSHOT 11](#_Toc58753124)

[ACTIVITY 3 – REGEX PRACTICE 12](#_Toc58753125)

[1 PRACTICE CODE 12](#_Toc58753126)

[GITHUB LINK 14](#_Toc58753127)

# List of Figures

Figure 1 PEP8 SCREENSHOT – LEGACY CODE 6

Figure 2 INPUT FILE SCREENSHOT – LEGACY CODE 6

Figure 3 OUTPUT FILE SCREENSHOT – LEGACY CODE 7

Figure 4 PEP8 SCREENSHOT - APPLYING OOP 10

Figure 5 INPUT FILE - APPLYING OOP 11

Figure 6 OUTPUT FILE SCREENSHOT - APPYING OOP 11

Figure 7 REGEX PRACTICE 1 12

Figure 8 REGEX PRACTICE 2 13

# ACTIVITY 1 – LEGACY CODE

# 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

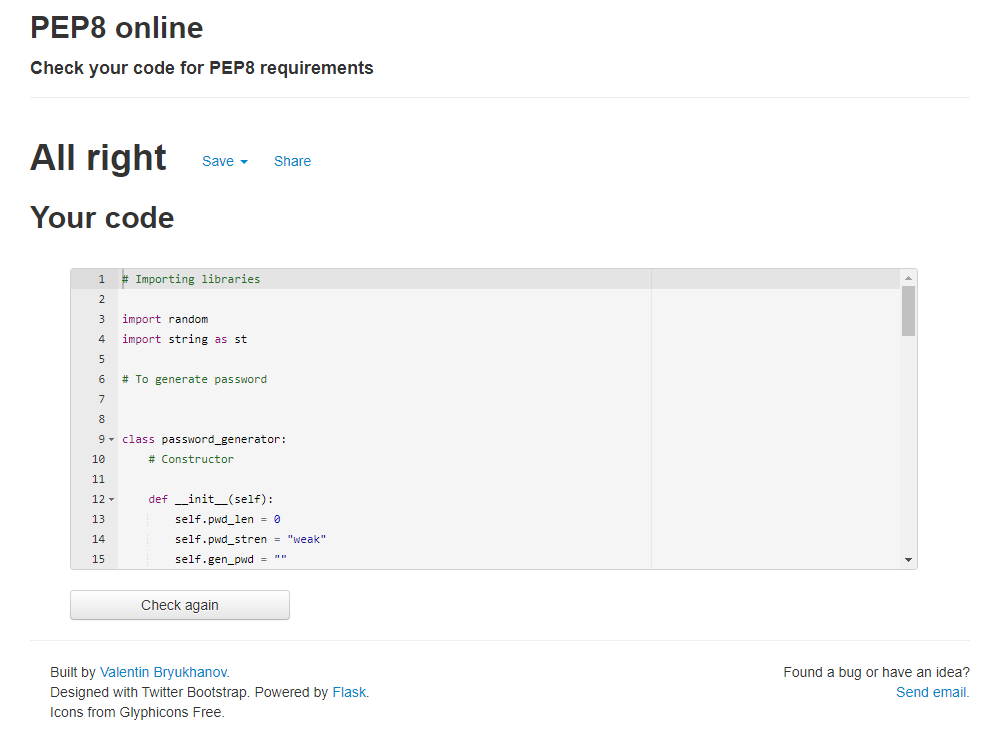


Figure PEP8 SCREENSHOT – LEGACY CODE

# INPUT FILE SCREENSHOT

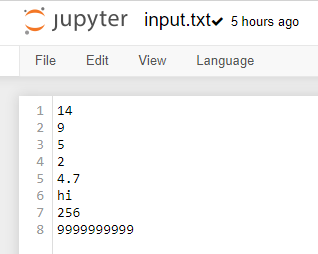


Figure INPUT FILE SCREENSHOT – LEGACY CODE

# OUTPUT FILE SCREENSHOT

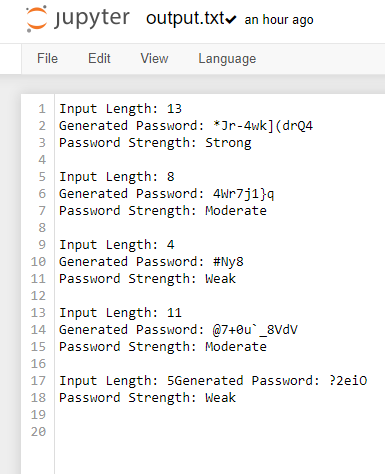


Figure OUTPUT FILE SCREENSHOT – LEGACY CODE

# ACTIVITY 2 – APPLYING OOP CONCEPTS

# PYTHON CODE

# Importing libraries

import random

import string as s

# Class to get length

class get\_length:

# Constructor

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

self.pwd\_len = 0

# To accept length

def get\_len(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 strength

class calculate\_strength(get\_length):

# Constructor

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

self.pwd\_stren = ""

super().\_\_init\_\_()

# To calculate password strength

def calc\_stren(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

class generate\_password(get\_length):

# Constructor

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

self.gen\_pwd = ""

super().\_\_init\_\_()

# To generate password

def generate\_pwd(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(calculate\_strength, generate\_password):

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

super().\_\_init\_\_()

def read\_write\_file(self):

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

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

for each\_line in fr:

length = each\_line

self.get\_len(length)

self.generate\_pwd()

self.calc\_stren()

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

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

"Password Strength: " + self.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

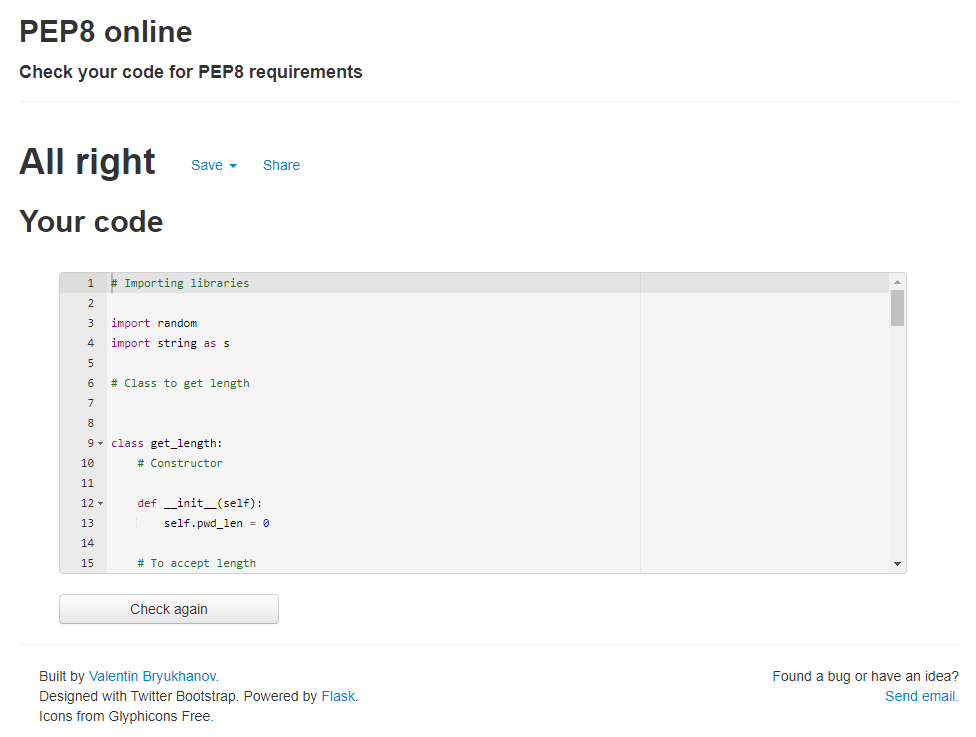


Figure PEP8 SCREENSHOT - APPLYING OOP

# INPUT FILE SCREENSHOT

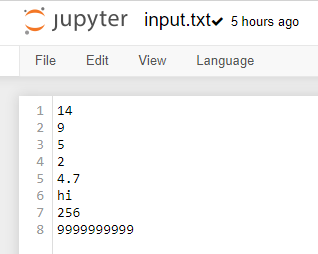


Figure INPUT FILE - APPLYING OOP

# OUTPUT FILE SCREENSHOT

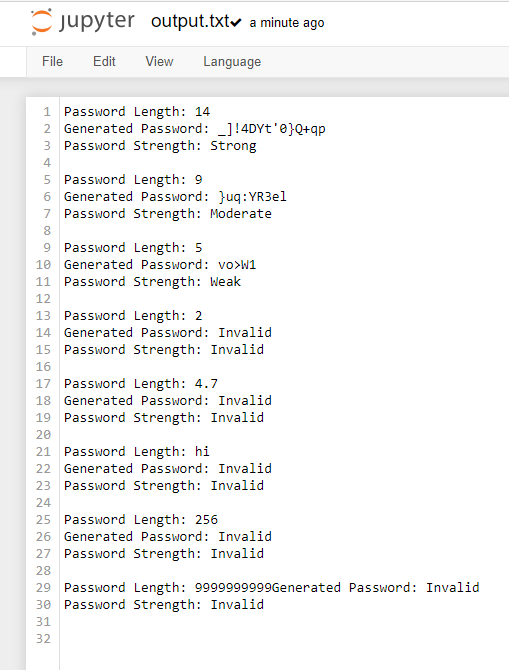


Figure OUTPUT FILE SCREENSHOT - APPYING OOP

# ACTIVITY 3 – REGEX PRACTICE

# PRACTICE CODE

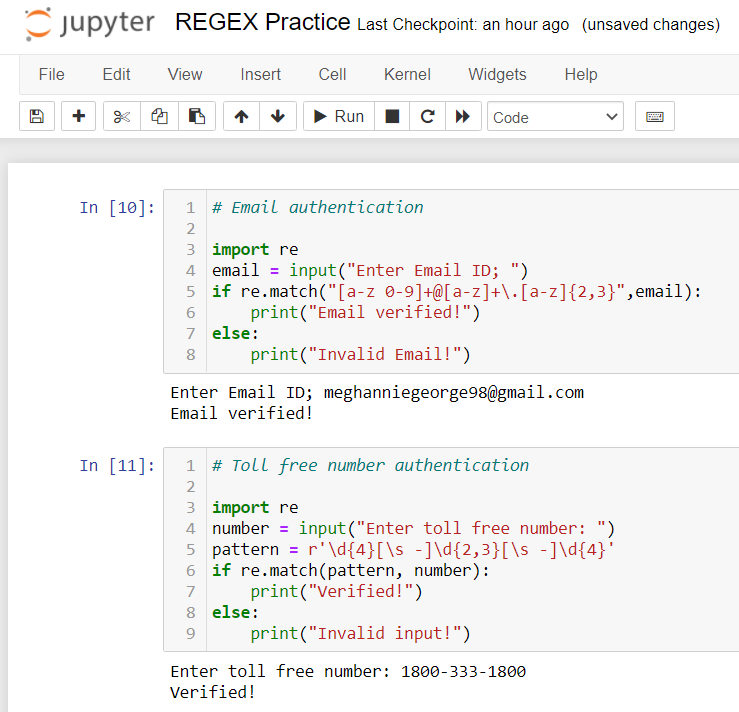


Figure REGEX PRACTICE 1

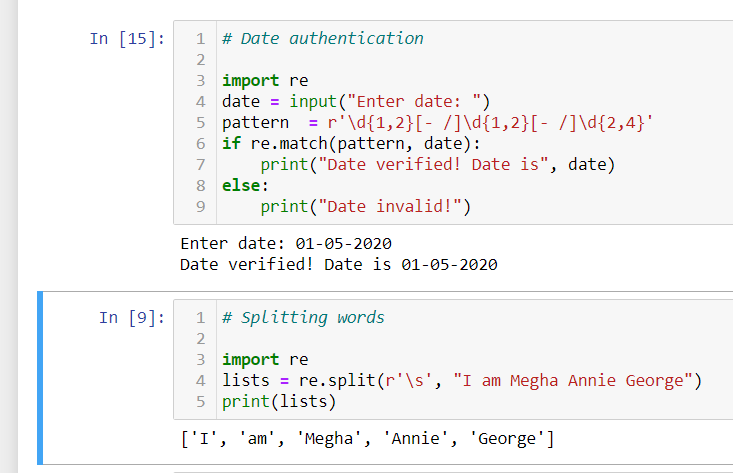


Figure REGEX PRACTICE 2

# GITHUB LINK

* <https://github.com/99003169/my_repo>