Python Training Certification Course

Name: Sowmya P

Project Title: **Password Manager**

DESCRIPTION

Zara is developing a new version of password manager. Earlier, she was using some third-party password manager, but she figured out that it can't keep track of all the passwords which has been set for the respective account. As she is very concerned about the security, she decided to develop her own version of the password manager  
  
**Objective:**To develop a custom password manager using Python

**Domain:** Security

**Source code:**

# Python Training certificate course  
# Project 2 : password manager  
  
class BasePasswordManager:  
 old\_password = ["parth", "Hello", "SowmyaP"]  
  
 def get\_password(self):  
 # method that returns the current password as a string.  
 return self.old\_password[-1]  
  
 def is\_correct(self, stringg):  
 # that receives a string and returns a boolean  
 if stringg == self.old\_password[-1]:  
 print("Password Is Correct")  
 return True  
 else:  
 print("Password Is Wrong")  
 return False  
  
  
# This class inherits from BasePasswordManager  
class PasswordManager(BasePasswordManager):  
  
 def \_\_init\_\_(self):  
 self.level\_old = 0  
 self.level\_new = 0  
 BasePasswordManager.\_\_init\_\_(self)  
  
 def get\_level(self, string):  
 self.password = string  
 a, n, d = 0, 0, 0  
 for i in self.password:  
 if i.isalpha():  
 a = a+1  
 elif i.isdigit():  
 n = n+1  
 else:  
 d = d+1  
 if len(self.password) == a:  
 return 0  
 if len(self.password) == a+n:  
 return 1  
 if len(self.password) == a+n+d:  
 return 2  
  
 def set\_password(self, stringg):  
 Previous\_password = BasePasswordManager.get\_password(self)  
 self.level\_new = self.get\_level(stringg)  
 self.level\_old = self.get\_level(Previous\_password)  
  
 if self.level\_new >= self.level\_old and len(stringg):  
#self.old\_password.append(stringg)  
 print("New password sucessfully added and securitylevel ", self.level\_new)  
 print("\n")  
 print("New password is",stringg)  
 else:  
 print("Password is weak and lower security level than past password")  
  
  
current = PasswordManager()  
  
print("\n")  
print("Current Password is " + current.get\_password())  
print("\n")  
current.is\_correct("SowmyaP")  
print("\n")  
current.set\_password("Som\_partha\_1234")  
print("\n")

**Steps performed:**

1. Implemented the following design

class **BasePasswordManager**

    members

**old\_passwords**: is a list that holds all of the user's past

                           passwords.

                           The last item of the list is the user's current password.

    methods

**get\_password** method that returns the current password as a string.

**is\_correct** method that receives a string and returns a boolean

             True or False depending on whether the string is equal to

             the current password or not.

class **PasswordManager**

    This class inherits from **BasePasswordManager**

        methods

**set\_password** method that sets the user's password.

             Password change is successful only if:

                    - Security level of the new password is greater.

                    - Length of new password is minimum 6

             However, if the old password already has the highest security level,

             new password must be of the highest security level for a successful password change.

**get\_level** method that returns the security level of the current password.

             It can also check and return the security level of a new password passed as a string.

*Security levels:*

*level 0 - password consists of alphabets or numbers only.*

*level 1 - Alphanumeric passwords.*

*level 2 - Alphanumeric passwords with special characters.*

**Output:**







