class BasePasswordManager:  
 old\_passwords = ["Hello","Kishan","Trainer","Simplilearn"] # List holds user’s past passwords  
 fast , \*others , last = old\_passwords  
 users\_current\_password = last # last item of the list is the user's current password  
  
 def get\_password(self):  
 return self.users\_current\_password # returns the current password as a string  
  
 def is\_correct(self,password): # method that receives a string  
 if self.get\_password() == password: # If the string is equal to the current password or not.  
 return True  
 else:  
 return False  
  
class PasswordManager(BasePasswordManager): #This class inherits from BasePasswordManager  
  
 def set\_password(self, new\_password): #methods that set the user's password and  
 # Password change is successful only if:  
 # Security level of the new password is greater.  
 # Length of the new password is a minimum of 6  
  
 if self.get\_level(new\_password) > self.get\_level() and len(new\_password) >= 6:  
 self.old\_passwords.append(new\_password)  
 print("Password changed Successfully.")  
 else:  
 print("Password can't be changed.")  
  
 def get\_level(self, password=None): # returns the security level of the current password.  
  
 if password == None:  
 password = self.get\_password()  
 if password.isalpha() or password.isnumeric():  
 level = 0  
 elif password.isalnum():  
 level = 1  
 else:  
 level = 2  
 return level  
  
Pass= BasePasswordManager()  
new\_password = input("Enter a password:")  
print(f"Is current password same as a new password: {Pass.is\_correct(new\_password)}")  
  
manage= PasswordManager()  
manage.set\_password(new\_password)  
print(f"Security Level of Password: {manage.get\_level(new\_password)}")