./

GENESIS – Advanced Python Learning Outcome & Mini-project Summary Report



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
| 1.0 | 11-12-2020 | Hareesh Unnikrishnan |  |  |  |
| 2.0 | 12-12-2020 | Hareesh Unnikrishnan |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Details**

Contents

[Contents 3](#_Toc58755172)

[1 ATM MANAGEMENT 4](#_Toc58755173)

[1.1 Source Code 4](#_Toc58755174)

[1.2 Program Output : 6](#_Toc58755175)

[2.1 Source Code: 9](#_Toc58755176)

[2.2 Program Output : 11](#_Toc58755177)

# 1 ATM MANAGEMENT

# 1.1 Source Code

import os

from random import randint

class SavingsAccount():

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

# [key][0] name ; [key][1] balance

self.savingsAccounts = {} #creating dictionary to store the user name and amount

def createAccount(self, name, initialDeposit):

print()

self.secretPin = randint(1000, 9999)

self.savingsAccounts[self.secretPin] = [name, initialDeposit]

print("Account has been created successfully.Your secret pin is ",self.secretPin)

def authenticate(self, name, secretPin):

if secretPin in self.savingsAccounts.keys():

if self.savingsAccounts[secretPin][0] == name:

print("Authentication Successful")

self.secretPin = secretPin

return True

else:

print("Authentication Failed")

return False

else:

print("Authentication Failed")

return False

def withdraw(self, withdrawalAmount):

print()

if withdrawalAmount > self.savingsAccounts[self.secretPin][1]:

print("Insufficient balance to withdraw.")

else:

self.savingsAccounts[self.secretPin][1] -= withdrawalAmount

print("Withdrawal was successful.")

self.displayAccBalance()

def deposit(self, depositAmount):

print()

self.savingsAccounts[self.secretPin][1] += depositAmount

print("Deposit was successful.")

output=open("op.txt","a+")

output.write("Deposit was successful.")

self.displayAccBalance()

print()

def displayAccBalance(self):

ans1=print("Avaialble balance: ",self.savingsAccounts[self.secretPin][1])

output=open("op.txt","w") #writing account balance to the file

ans=str("Avaialble balance: " + str(self.savingsAccounts[self.secretPin][1])+"\n")

output.write(ans)

output.close()

#self.send\_sms(self.savingsAccounts[self.secretPin][1])

bank\_name=open("test.txt","r") #getting bank name as input from file

Name=bank\_name.read()

print("Welcome to {} Bank".format(Name))

savingsAccount = SavingsAccount()

while True:

print("Enter 1 to Create a new Saving Account")

print("Enter 2 to Access an existing account")

print("Enter 3 to exit")

userChoice = int(input())

if userChoice == 1:

print("Enter your name: ")

name = input()

print("Enter the initial deposit: ")

deposit = int(input())

savingsAccount.createAccount(name, deposit)

elif userChoice == 2:

print("Enter your name: ")

name = input()

print("Enter your secret Pin: ")

secretPin = int(input())

authenticationStatus = savingsAccount.authenticate(name, secretPin)

if authenticationStatus is True:

while True:

print()

print("Enter 1 to withdraw")

print("Enter 2 to deposit")

print("Enter 3 to display avialable balance")

print("Enter 4 to go back to the previous menu")

userChoice = int(input())

if userChoice == 1:

print("Enter a withdrawal amount")

withdrawalAmount = int(input())

savingsAccount.withdraw(withdrawalAmount)

elif userChoice == 2:

print("Enter an amount to be deposited")

depositAmount = int(input())

savingsAccount.deposit(depositAmount)

elif userChoice == 3:

savingsAccount.displayAccBalance()

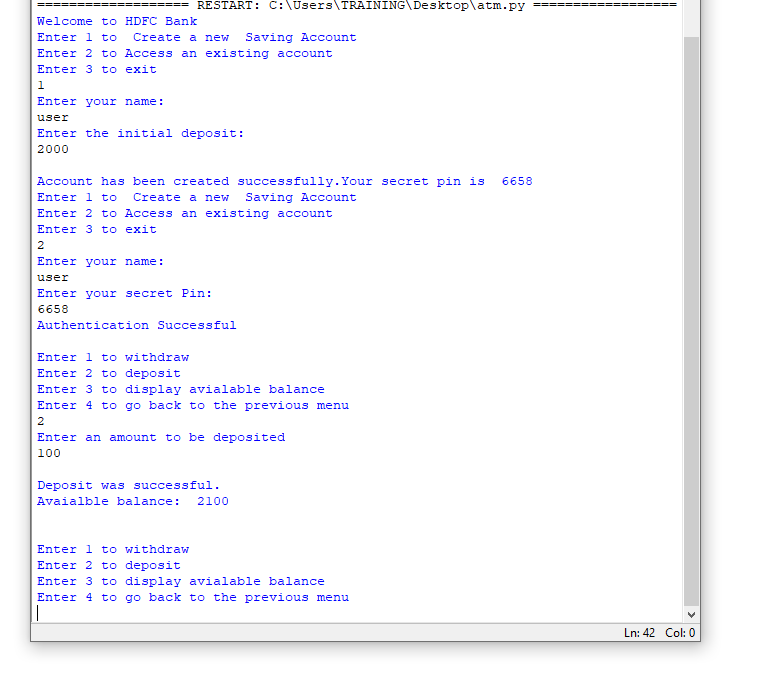
elif userChoice == 4:

break

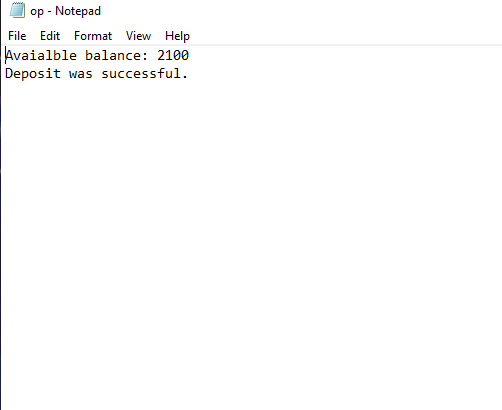
elif userChoice == 3:

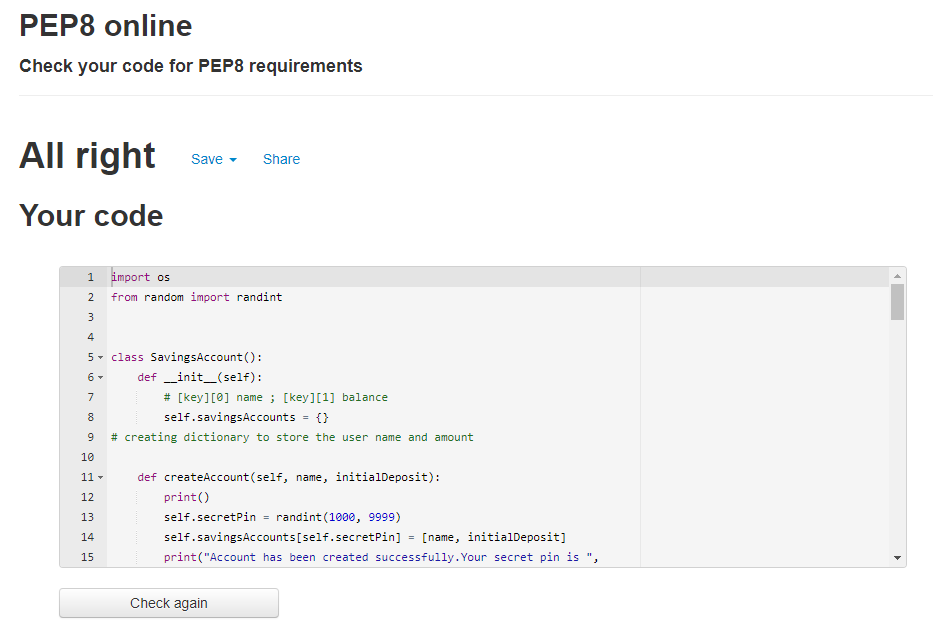
quit()

# 1.2 Program Output :



**1.3 Output File**





# 2.1 Source Code:

Features included

* Inheritance
* Exception Handling
* Regular expression

import os

import re #import Regular expression library

from random import randint

class Account():

def createAccount(self, name, initialDeposit):

def createAccount(self, name, initialDeposit, email\_ID):

print()

self.secretPin = randint(1000, 9999)

self.savingsAccounts[self.secretPin] = [name, initialDeposit, email\_ID]

print("Account has been created successfully.Your secret pin is ",self.secretPin)

class SavingsAccount(Account): #creating Saving Account from Account

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

# [key][0] name ; [key][1] balance

self.savingsAccounts = {} #creating dictionary to store the user name and amount

def authenticate(self, name, secretPin):

if secretPin in self.savingsAccounts.keys():

if self.savingsAccounts[secretPin][0] == name:

print("Authentication Successful")

self.secretPin = secretPin

return True

else:

print("Authentication Failed")

return False

else:

print("Authentication Failed")

return False

def withdraw(self, withdrawalAmount): #calculate withdraw function and return acc balance

print()

if withdrawalAmount > self.savingsAccounts[self.secretPin][1]:

print("Insufficient balance to withdraw.")

else:

self.savingsAccounts[self.secretPin][1] -= withdrawalAmount

print("Withdrawal was successful.")

self.displayAccBalance()

def deposit(self, depositAmount):

print()

self.savingsAccounts[self.secretPin][1] += depositAmount

print("Deposit was successful.")

output=open("op.txt","a+")

output.write("Deposit was successful.")

self.displayAccBalance()

print()

def displayAccBalance(self):

ans1=print("Avaialble balance: ",self.savingsAccounts[self.secretPin][1])

output=open("op.txt","w") #writing account balance to the file

ans=str("Avaialble balance: " + str(self.savingsAccounts[self.secretPin][1])+"\n")

output.write(ans)

output.close()

#self.send\_sms(self.savingsAccounts[self.secretPin][1])

def email\_Id\_verification(self,email\_ID):

pattern = '[a-z 0-9]+@[a-z]+\.[a-z]{3}'

test\_string = email\_ID

result = re.match(pattern, test\_string)

if result:

print("email ID verification successful.")

return True

else:

print("email ID verification unsuccessful.")

return False

bank\_name=open("test.txt","r") #getting bank name as input from file

Name=bank\_name.read()

print("Welcome to {} Bank".format(Name))

savingsAccount = SavingsAccount()

while True:

print("Enter 1 to Create a new Saving Account")

print("Enter 2 to Access an existing account")

print("Enter 3 to exit")

userChoice = int(input())

if userChoice == 1:

print("Enter your name: ")

name = input()

print("Enter you Email\_ID")

email\_ID = input()

print("Enter the initial deposit: ")

deposit = int(input())

savingsAccount.createAccount(name, deposit, email\_ID)

elif userChoice == 2:

print("Enter your name: ")

name = input()

print("Enter your secret Pin: ")

secretPin = int(input())

email\_id = savingsAccount.email\_Id\_verification(email\_ID)

authenticationStatus = savingsAccount.authenticate(name, secretPin)

if authenticationStatus and email\_id is True:

while True:

print()

print("Enter 1 to withdraw")

print("Enter 2 to deposit")

print("Enter 3 to display avialable balance")

print("Enter 4 to go back to the previous menu")

userChoice = int(input())

if userChoice == 1:

print("Enter an withdrawal amount “)

try:

withdrawalAmount = int(input())

savingsAccount.withdraw(withdrawalAmount)

except ValueError:

print("Invalid Input for withraw!Give input as Integer")

elif userChoice == 2:

print("Enter an amount to be deposited")

try:

depositAmount = int(input())

savingsAccount.deposit(depositAmount)

except ValueError:

print("Invalid Input for deposit!Give input as Integer")

elif userChoice == 3:

savingsAccount.displayAccBalance()

elif userChoice == 4:

break

elif userChoice == 3:

quit()

# 2.2 **Program Output :**

