**PYTHON ASSIGNMENT -3**

**BANK MANAGEMENT SYSTEM**

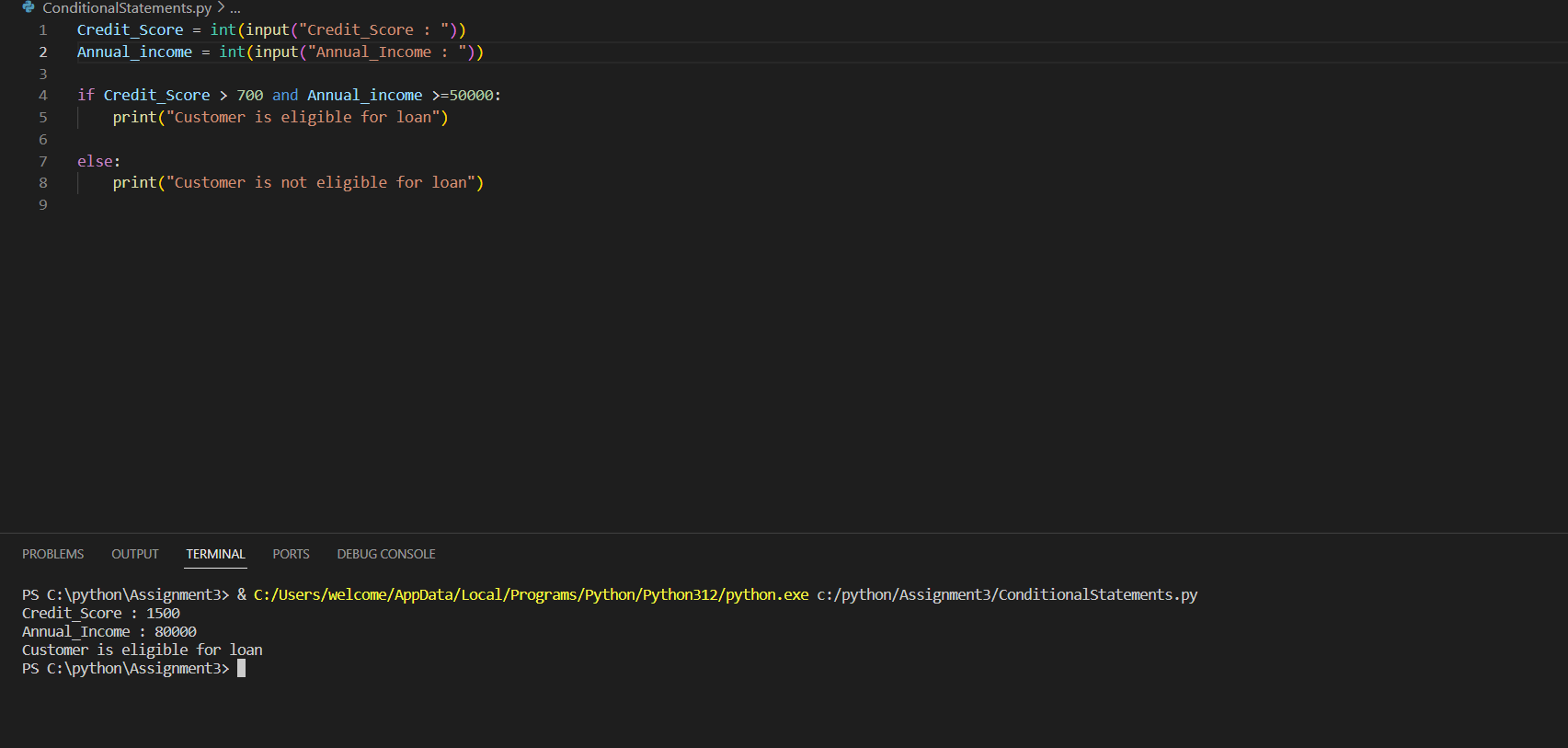
Control Structure

**Task 1**

1. Write a program that takes the customer's credit score and annual income as input.

2. Use conditional statements (if-else) to determine if the customer is eligible for a loan.

3. Display an appropriate message based on eligibility.



**Task 2:** Nested Conditional Statements Create a program that simulates an ATM transaction. Display options such as "Check Balance," "Withdraw," "Deposit

CODE:

print("Welcome to ATM...")

print("Select 1 to CheckBalance")

print("Select 2 to Withdraw")

print("Select 3 to Deposit")

enter =int(input("Select Something: "))

Current\_Balance = int(input("Enter Current Balance : $ "))

if enter == 1:

print(f'Current\_Balance is : ${Current\_Balance}')

elif enter ==2 :

Withdrawl\_Amount = int(input("Enter Withdrawal Amount : $ "))

if Withdrawl\_Amount > Current\_Balance:

print("Sorry...Your Current Balance is not Sufficient to withdraw your amount")

elif Withdrawl\_Amount % 100 !=0 or Withdrawl\_Amount % 500 !=0:

print("Sorry..Amount must be in multiples of 100s or 500s ")

else:

Current\_Balance -= Withdrawl\_Amount

print("Withdrawl Successful.Collect your cash...")

print(f'Your current\_balance is : $ {Current\_Balance} ')

elif enter == 3:

deposit\_amount = int(input("Enter deposit amount: $ "))

if deposit\_amount <= 0:

print("Invalid Amount...Enter proper amount")

else:

Current\_Balance += deposit\_amount

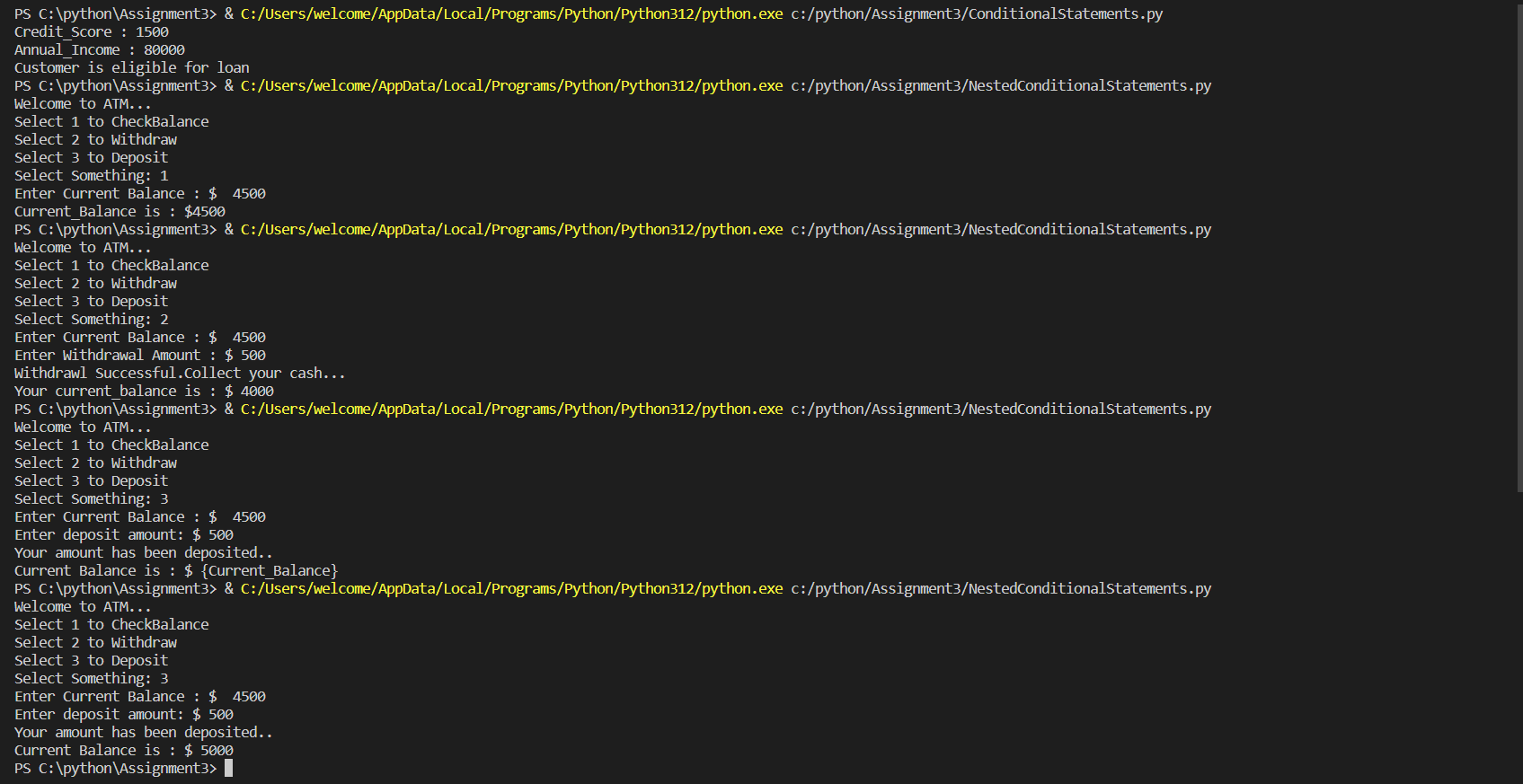
print("Your amount has been deposited..")

print('Current Balance is : $ {Current\_Balance}')

else:

print("Invalid...Enter again")

OUTPUT:



**Task 3:** Loop Structures You are responsible for calculating compound interest on savings accounts for bank customers. Create a program that calculates the future balance of a savings account.

**CODE:**

no\_of\_customers = int(input(" No.of.Customers: "))

for customer in range(1,no\_of\_customers + 1):

print(f'Customer {customer}:')

intial\_balance = int(input("Enter initial balance: $ "))

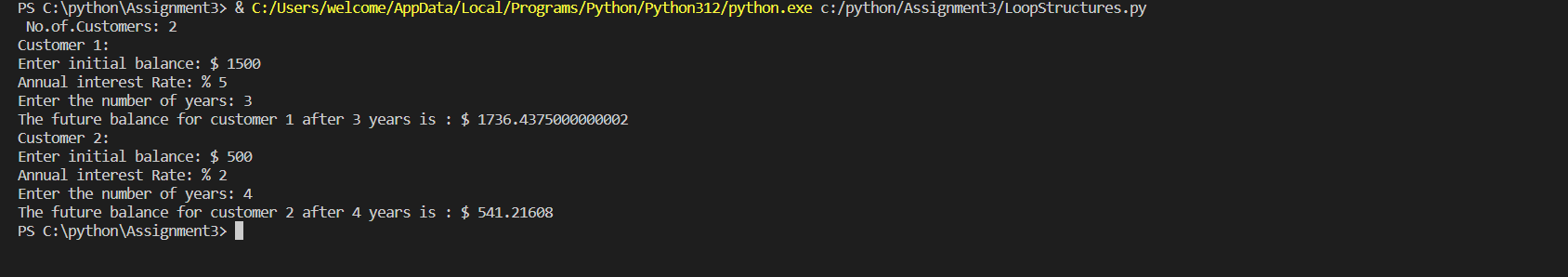
annual\_interest\_rate = int(input("Annual interest Rate: % "))

years = int(input("Enter the number of years: "))

future\_balance =intial\_balance \* (1 + annual\_interest\_rate/100) \*\* years

print(f'The future balance for customer {customer} after {years} years is : $ {future\_balance}')

**OUTPUT:**

****

**Task 4:** Looping, Array and Data Validation You are tasked with creating a program that allows bank customers to check their account balances

CODE:

Customer\_Accounts = [

{'AccountNo': 101101, 'Balance': 20000},

{'AccountNo': 392368, 'Balance': 5000},

{'AccountNo': 567432, 'Balance': 4500},

{'AccountNo': 655665, 'Balance': 10000},

{'AccountNo': 987678, 'Balance': 25000}

]

while True:

try:

AccountNo = int(input("Account Number: "))

AccountInfo = next((acc for acc in Customer\_Accounts if acc['AccountNo'] == AccountNo), None)

if AccountInfo:

Balance = AccountInfo['Balance']

print(f"Account Number: {AccountNo}, Balance: ${Balance}")

break

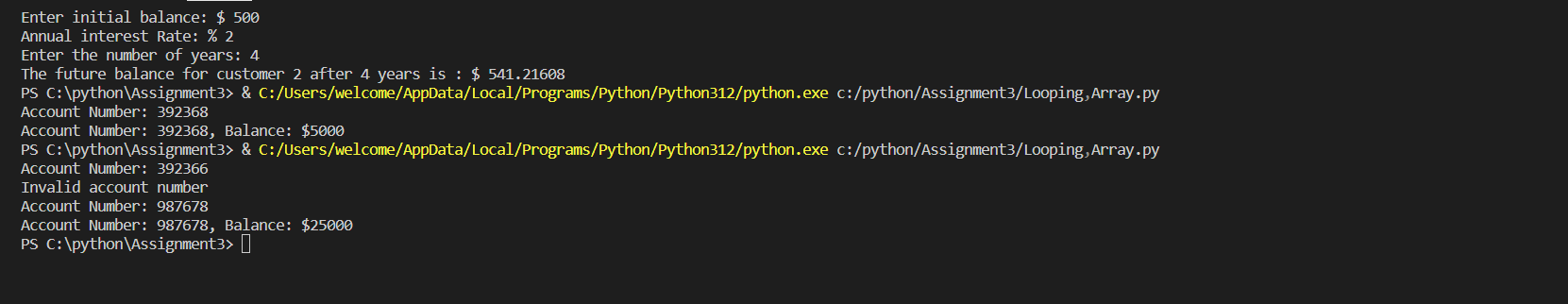
else:

print("Invalid account number")

except ValueError:

print(" Please enter a valid account number.")

OUTPUT:



**Task 5:** Password Validation Write a program that prompts the user to create a password for their bank account.

**CODE:**

while True:

password = input("Create a password for your bank account: ")

if len(password) >= 8 and any(char.isupper() for char in password) and any(char.isdigit() for char in password):

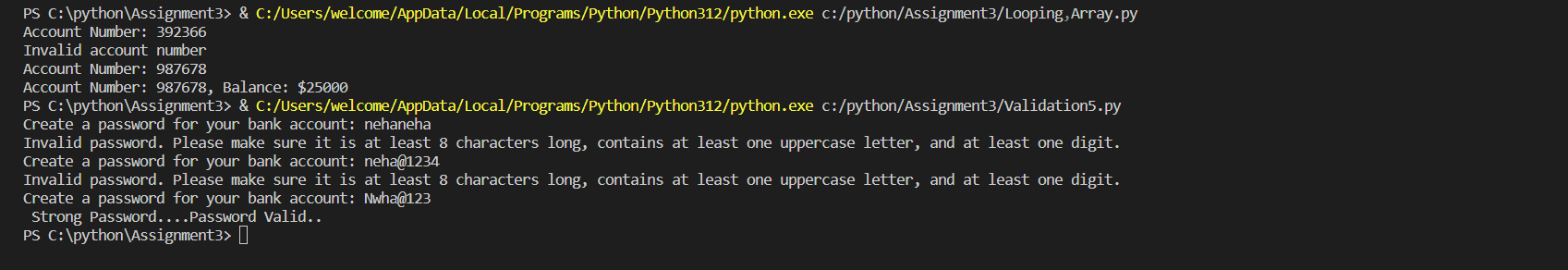
print(" Strong Password....Password Valid..")

break

else:

print("Invalid password. Please make sure it is at least 8 characters long, contains at least one uppercase letter, and at least one digit.")

**OUTPUT:**



**Task 6:** Password Validation Create a program that maintains a list of bank transactions (deposits and withdrawals) for a customer.

**CODE:**

transactions = []

while True:

print("Welcome to Bank...")

print("Select 1 to Deposit")

print("Select 2 to Withdrawl")

print("Select 3 to Transaction history")

print("Select 4 to Exit")

option = int(input("Enter options: "))

if option == 1:

deposit\_amount = int(input("Enter the deposit amount: "))

transactions.append(('Deposit', deposit\_amount))

print(" Amount Deposited!")

elif option == 2:

withdrawal\_amount = int(input("Enter amount for Withdrawal: "))

transactions.append(('Withdrawal', withdrawal\_amount))

print("Withdrawal successful!")

elif option == 3:

if transactions:

print(" Transaction History:")

for transaction\_type, amount in transactions:

print(f"{transaction\_type}: {amount}")

elif option == 4:

print("Exit....")

break

else:

print(" Enter 1, 2, or 3 options to proceed")

**OUTPUT:**

****

**BANKING MANGEMENT SYSTEM**

**CODE**

import mysql.connector as con

from mysql.connector import Error

connect = con.connect(host = "localhost",user = "root",password = "root",database = "bank2")

# get account\_balance

def get\_balance():

AccNo = input("Enter Account No: ")

data =(AccNo,)

sql = "select balance from accounts where Accno = %s;"

c = connect.cursor()

c.execute(sql,data)

list1 = c.fetchone()

print(f" The balance of the Account {AccNo} is:", list1[0])

menu()

# deposit

def deposit\_amount():

AccNo = input("Enter Account No: ")

Amount = int(input("Enter amount: "))

data = (AccNo,)

sql = "Select balance from accounts where AccNo = %s; "

c = connect.cursor()

c.execute(sql, data)

list1 = c.fetchone()

deposit = sum(list1, Amount)

data1 = (deposit, AccNo)

sql1 = " update Accounts set balance = %s where AccNo = %s;"

c.execute(sql1,data1)

print("Amount deposited...")

print("The Current balance of the Account {Accno} is: ",deposit)

menu()

#withdraw

def withdraw\_amount():

AccNo = input("Enter Account No: ")

Amount = int(input("Enter amount: "))

data = (AccNo,)

sql = "Select balance from accounts where AccNo = %s; "

c = connect.cursor()

c.execute(sql, data)

list1 = c.fetchone()

if(Amount > list1[0]):

print("Withdrawl not possible....\n Insufficient funds...")

menu()

else:

withdraw = list1[0] - Amount

data1 = (withdraw, AccNo)

sql1 = " update Accounts set balance = %s where AccNo = %s;"

c.execute(sql1, data1)

print("Amount Withdrawl successful....")

print("The Current balance of the Account {Accno} is: ", withdraw)

menu()

#transfer

def transfer():

pass

'''Sender = input("Sender Account No: ")

Receiver = input("Receiver Account No: ")

Amount= input("Enter Amount: ")

data = (Sender,)

data1 = (Receiver,)

query1 = "select balance from accounts where AccNo = %s; "

c = connect.cursor()

c.execute(query1, data1)

receive = c.fetchall()

deposit = receive + Amount

print(list)'''

menu()

#get transactions

def get\_transactions():

print("....View Transactions...")

AccNo = input("Enter Acc No: ")

data = (AccNo,)

query = " select \* from transactions where account = %s;"

c = connect.cursor()

c.execute(query, data)

list = c.fetchall()

for i in list:

print("Account Number: ", i[0])

print("Description: ", i[1])

print("Date and Time: ", i[2])

print("Transaction Type: ", i[3])

print("Amount: ", i[4])

print("-----------------------------")

menu()

#create account

def create\_account():

AccNo = input("Enter Account No: ")

if(check\_account(AccNo) == True):

print("Account already exists....")

menu()

else:

AccType = input("Enter Account type: ")

Balance = input("Enter Balance: ")

CusName = input("Enter Customer Name: ")

data = (AccNo, AccType, Balance, CusName)

sql = " insert into accounts values (%s,%s,%s,%s);"

c = connect.cursor()

c.execute(sql, data)

connect.commit()

print(".......Account created successfully....")

menu()

# check accountno

def check\_account(AccNo):

sql = 'select \* from accounts where AccNo=%s'

c = connect.cursor(buffered=True)

data = (AccNo,)

c.execute(sql, data)

r = c.rowcount

if r == 1:

return True

else:

return False

#list accounts

def list\_accounts():

print("....View Accounts Details...")

AccNo = input("Enter Acc No: ")

data = (AccNo,)

query = " select \* from accounts where AccNo = %s;"

c = connect.cursor()

c.execute(query, data)

list = c.fetchall()

for i in list:

print("Account Number: ", i[0])

print("Account Type: ", i[1])

print("Balance: ", i[2])

print("Customer Name : ", i[3])

print("-----------------------------")

menu()

#get account details

def get\_customer\_details():

print("....View Customer Details...")

CusId = input("Enter Customer Id: ")

data = (CusId,)

query = " select \* from customers where customerId = %s; "

c = connect.cursor()

c.execute(query, data)

list = c.fetchall()

for i in list:

print("Customer Id: ", i[0])

print("First\_Name: ", i[1])

print("Last\_Name: ", i[2])

print("Email : ", i[3])

print("Phone Number : ", i[4])

print("Address : ", i[5])

print("-----------------------------")

menu()

#calculate interest

def calculate\_interest():

Principal = int(input("Enter Principal Amount: "))

Rate = int(input("Enter rate of interest : %"))

Time = int(input("Enter years: "))

Interest = Principal\* Rate \*Time

print(f"The calculated Interest of Amount {Principal} is : ", Interest)

menu()

#creating menu

def menu():

print("Welcome to bank")

print ("Select an option:")

print("1.To create new account")

print("2.Deposit amount")

print("3.Withdraw amount")

print("4.Get balance Amount")

print("5.Get Customer details")

print("6.Transfer")

print("7.List Accounts")

print("8.Get Transactions ")

print("9.Calculate interest ")

print("10.Exit")

option = input("Enter option: ")

if option == '1':

create\_account()

elif option == '2':

deposit\_amount()

elif option== '3':

withdraw\_amount()

elif option == '4':

get\_balance()

elif option == '5':

get\_customer\_details()

elif option == '6':

transfer()

elif option == '7':

list\_accounts()

elif option == '8':

get\_transactions()

elif option == '9':

calculate\_interest()

elif option == '10':

print("....Exit....")

else:

print("Invalid option...\n Try again...")

menu()

menu()

**OUTPUT:**

