SOFTWARE ENGINEERING UNIT –4 LAB TASKS

TEAM:

RAKSHIKA S – PES2UG20CS264

SHRAVYA U – PES2UG20CS326

Problem Statement–1:

Unit Testing:

A unit is the smallest block of code that functions individually. The first level of testing is Unit testing and this problem statement is geared towards the same.

•Discuss with your teammates and demarcate units in your code base

Note: discuss why the code snippet you have chosen can be classified as a unit

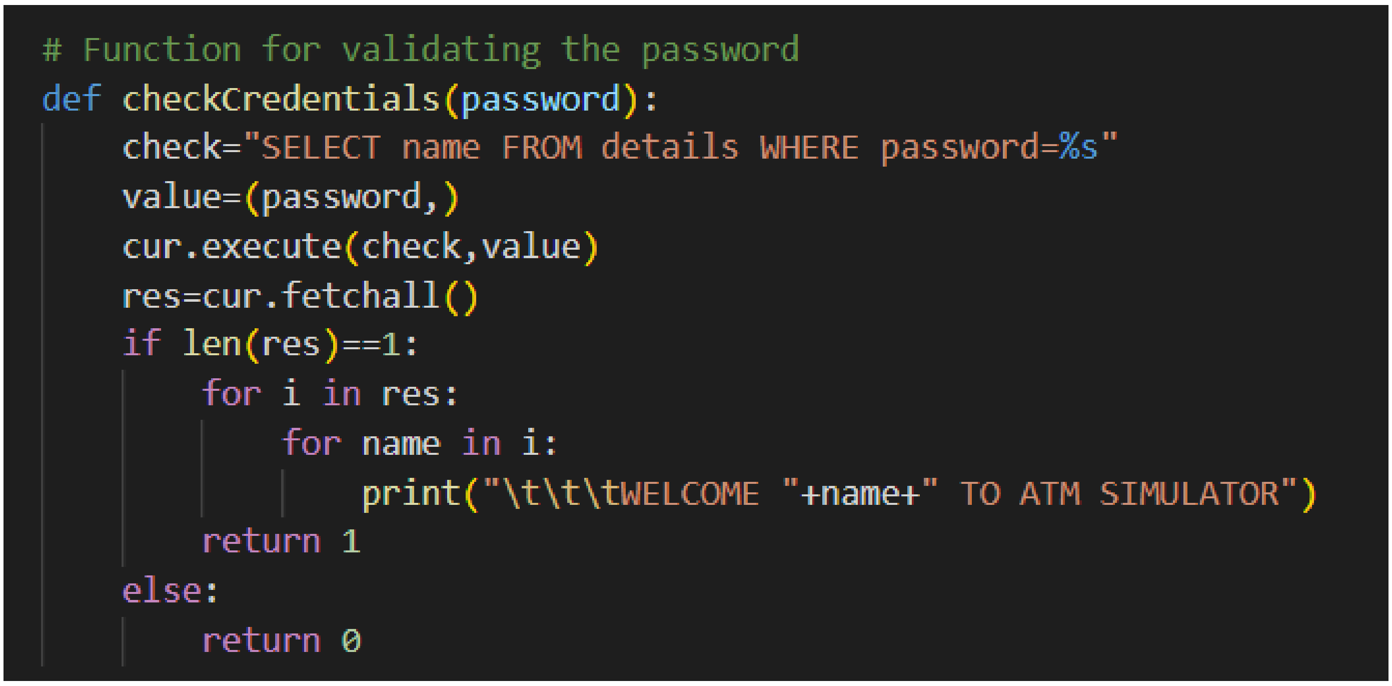
•Develop test cases for both valid and invalid data

•Ideate how you could further modularize larger blocks of code into compact units with your teammates

Solution:

- We can modularize our code into small units as create account, check credentials, check account number, validating existing customer, password generation, current date and time, checking bank statement.

- For this task of unit testing, we shall take the check credentials unit. This code snippet can be considered as a single unit as it is the smallest block of code that functions individually. It is a user defined function that independently check and validates the credentials of the user while logging into the bank account. It will cross check if the account number and the password match the credentials in the database.



```python
# Function for validating the password
def checkCredentials(password):
    check="SELECT name FROM details WHERE password=%s"
    value=(password,)
    cur.execute(check,value)
    res=cur.fetchall()
    if len(res)==1:
        for i in res:
            for name in i:
                print("\t\t\tWELCOME "+name+" TO ATM SIMULATOR")
        return 1
    else:
        return 0
```

- Test cases considering valid and invalid data for the checkCredentials unit:

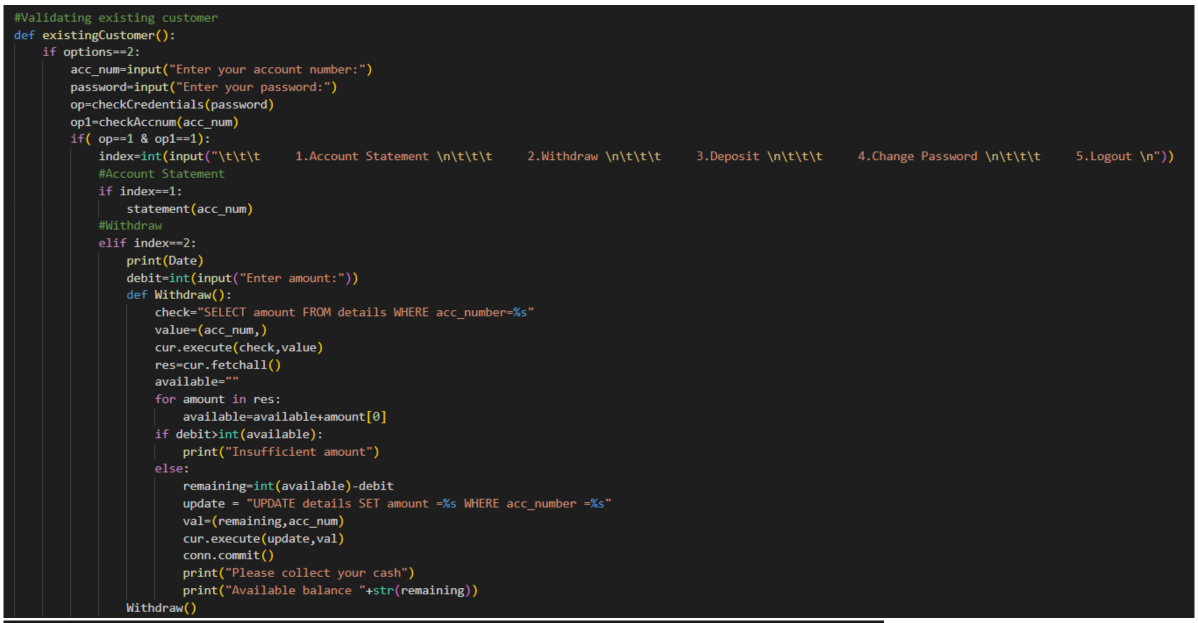
1. Invalid data:
   Test case description: To test the login functionality when the password is invalid.
   Pre-conditions: Account number should be entered
   Test Steps:
   1) Navigate to existing account
   2) Enter account number
   3) Enter invalid password
   4) Click enter
   Test data:
   Account number: 123
   Password: 67809145
   Expected Results: Login should not be Successful. Display wrong password.
   Actual Result: Login is not successful. Displays wrong password.
   Test Result: PASS

2. Valid data:
   Test case description: To test the login functionality when the password is valid.
   Pre-conditions: Account number should be entered
   Test Steps:
   1) Navigate to existing account
   2) Enter account number

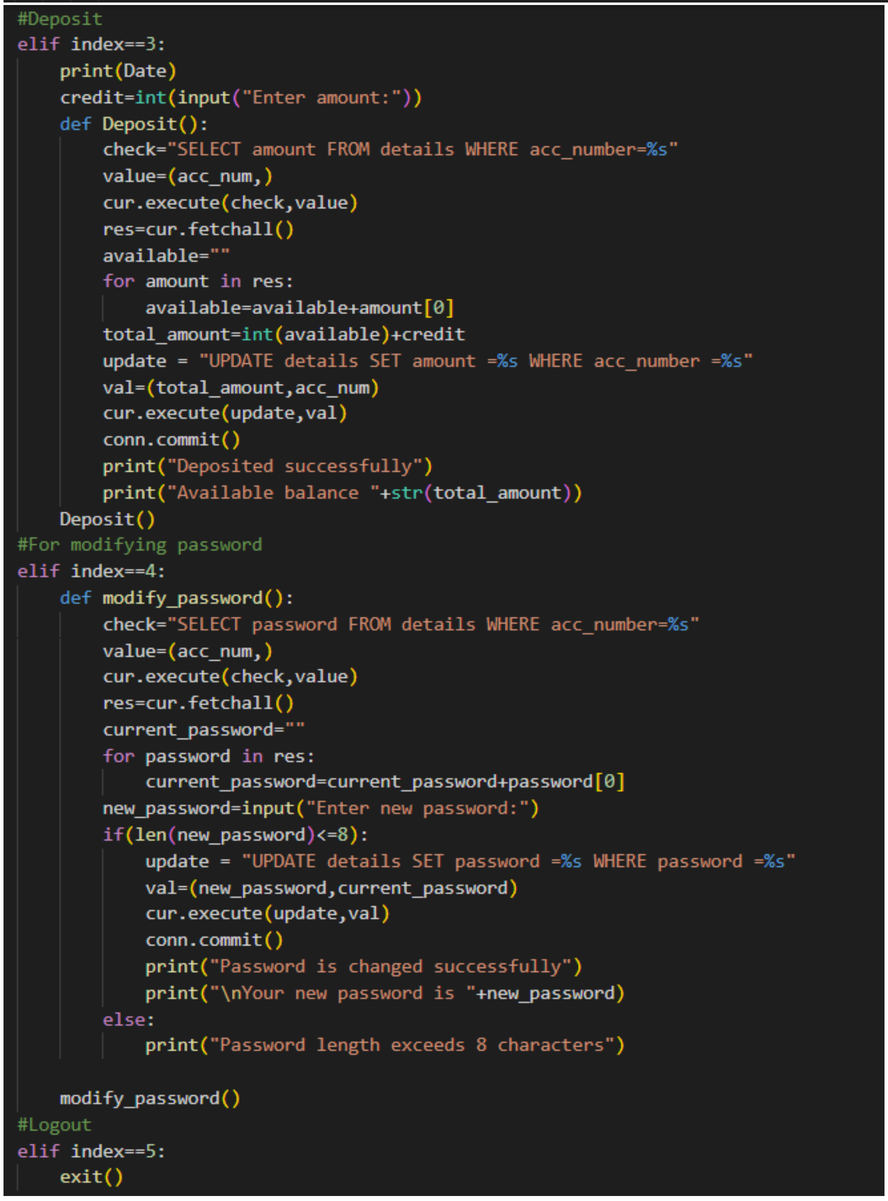
Test data:

Account number: 123
Password: 12345678
Expected Results: Login should be Successful. Login to account.
Actual Result: Login is successful. Logged into account.
Test Result: PASS

- We could modularize validating existing customer further into withdraw, deposit and modify password. We can define these functions individually and call them in the existingCustomer() function. This will divide the function into even simpler modules. This might be easier to test and debug the modules.



```
#Validating existing customer
def existingCustomer():
    if options==2:
        acc_num=input("Enter your account number:")
        password=input("Enter your password:")
        op=checkCredentials(password)
        op1=checkAccnum(acc_num)
        if( op==1 & op1==1):
            index=int(input("\t\t\t     1.Account Statement \n\t\t\t     2.Withdraw \n\t\t\t     3.Deposit \n\t\t\t     4.Change Password \n\t\t\t     5.Logout \n"))
            #Account Statement
            if index==1:
                statement(acc_num)
            #Withdraw
            elif index==2:
                print(Date)
                debit=int(input("Enter amount:"))
                def Withdraw():
                    check="SELECT amount FROM details WHERE acc_number=%s"
                    value=(acc_num,)
                    cur.execute(check,value)
                    res=cur.fetchall()
                    available=""
                    for amount in res:
                        available=available+amount[0]
                    if debit>int(available):
                        print("Insufficient amount")
                    else:
                        remaining=int(available)-debit
                        update = "UPDATE details SET amount =%s WHERE acc_number =%s"
                        val=(remaining,acc_num)
                        cur.execute(update,val)
                        conn.commit()
                        print("Please collect your cash")
                        print("Available balance "+str(remaining))
                Withdraw()
```



```
#Deposit
elif index==3:
    print(Date)
    credit=int(input("Enter amount:"))
    def Deposit():
        check="SELECT amount FROM details WHERE acc_number=%s"
        value=(acc_num,)
        cur.execute(check,value)
        res=cur.fetchall()
        available=""
        for amount in res:
            available=available+amount[0]
        total_amount=int(available)+credit
        update = "UPDATE details SET amount =%s WHERE acc_number =%s"
        val=(total_amount,acc_num)
        cur.execute(update,val)
        conn.commit()
        print("Deposited successfully")
        print("Available balance "+str(total_amount))
    Deposit()
#For modifying password
elif index==4:
    def modify_password():
        check="SELECT password FROM details WHERE acc_number=%s"
        value=(acc_num,)
        cur.execute(check,value)
        res=cur.fetchall()
        current_password=""
        for password in res:
            current_password=current_password+password[0]
        new_password=input("Enter new password:")
        if(len(new_password)<=8):
            update = "UPDATE details SET password =%s WHERE password =%s"
            val=(new_password,current_password)
            cur.execute(update,val)
            conn.commit()
            print("Password is changed successfully")
            print("\nYour new password is "+new_password)
        else:
            print("Password length exceeds 8 characters")

    modify_password()
#Logout
elif index==5:
    exit()
```

<u>Problem Statement –2: Dynamic Testing</u>

Dynamic testing involves execution of your code to analyse errors found during execution. Some common techniques are Boundary Value Analysis and Mutation Testing.

<u>Problem Statement –2. a: Boundary Value Analysis</u>

When it comes to finding errors in your code base, they are often found at locations where a condition is being tested. Due to this, developers often use Boundary Value tests to reduce defect density.

•How would you define a boundary test?

Note: Simple relational conditions are a basic example

•Build your boundary test cases and execute them

Solution:

1. How would you define a boundary test?

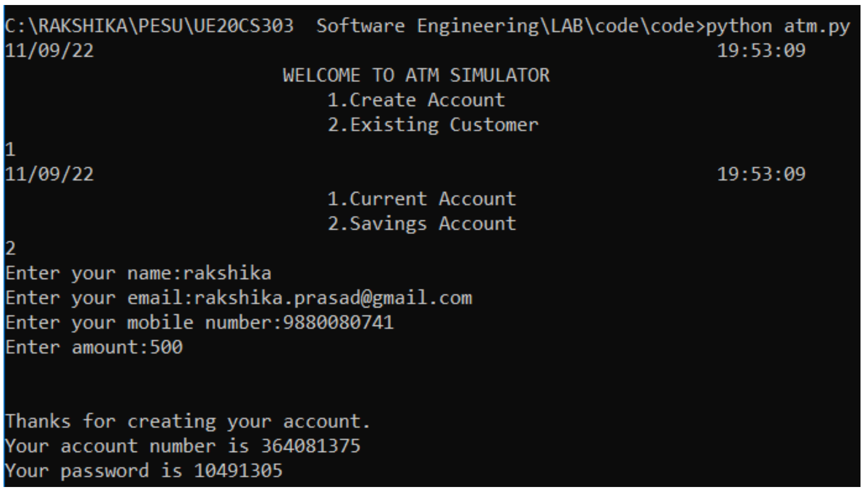
   Boundary Value Analysis is based on testing the boundary values of valid and invalid partitions. The behaviour at the edge of the equivalence partition is more likely to be incorrect than the behaviour within the partition, so boundaries are an area where testing is likely to yield defects. It checks for the input values near the boundary that have a higher chance of error. Every partition has its maximum and minimum values and these maximum and minimum values are the boundary values of a partition.

   - A boundary value for a valid partition is a valid boundary value.
   - A boundary value for an invalid partition is an invalid boundary value.
   - For each variable we check-
     - Minimum value.
     - Just above the minimum.
     - Nominal Value.
     - Just below Max value.
     - Max value

2. The boundary testing will be applied to the create account function:

a) Amount to be deposited while creating account should be between 500 and 100000.
   Valid test cases:

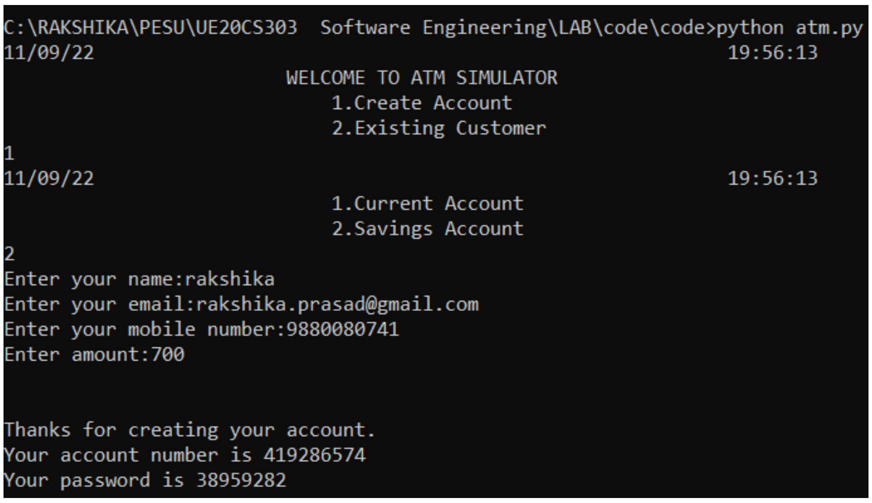
- Minimum value:



```
C:\RAKSHIKA\PESU\UE20CS303  Software Engineering\LAB\code\code>python atm.py
11/09/22                                                        19:53:09
                         WELCOME TO ATM SIMULATOR
                             1.Create Account
                             2.Existing Customer
1
11/09/22                                                        19:53:09
                             1.Current Account
                             2.Savings Account
2
Enter your name:rakshika
Enter your email:rakshika.prasad@gmail.com
Enter your mobile number:9880080741
Enter amount:500


Thanks for creating your account.
Your account number is 364081375
Your password is 10491305
```

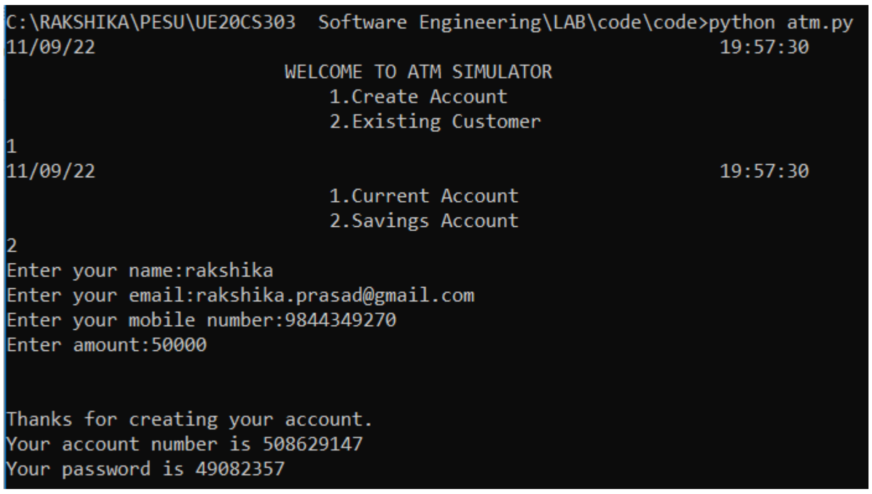
  - Just above the minimum



```
C:\RAKSHIKA\PESU\UE20CS303  Software Engineering\LAB\code\code>python atm.py
11/09/22                                                        19:56:13
                         WELCOME TO ATM SIMULATOR
                             1.Create Account
                             2.Existing Customer
1
11/09/22                                                        19:56:13
                             1.Current Account
                             2.Savings Account
2
Enter your name:rakshika
Enter your email:rakshika.prasad@gmail.com
Enter your mobile number:9880080741
Enter amount:700


Thanks for creating your account.
Your account number is 419286574
Your password is 38959282
```

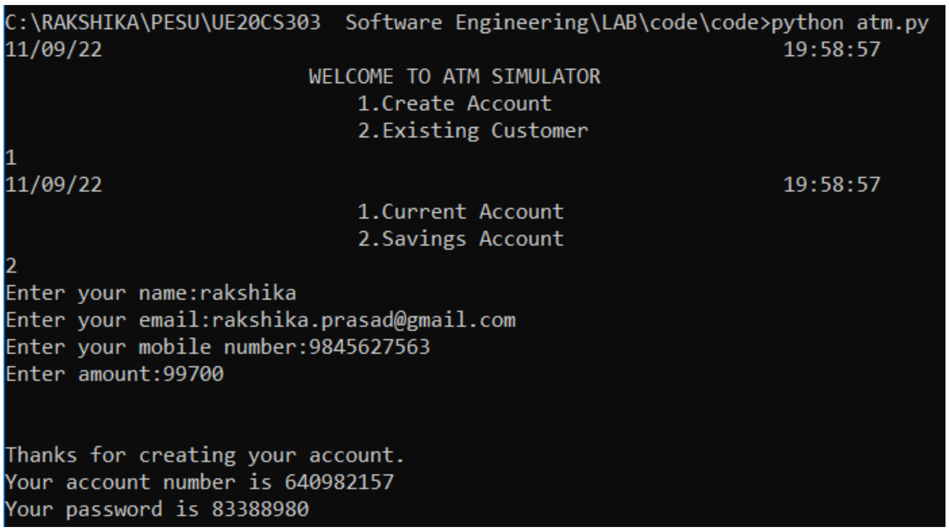
  - Nominal Value



```
C:\RAKSHIKA\PESU\UE20CS303  Software Engineering\LAB\code\code>python atm.py
11/09/22                                                        19:57:30
                         WELCOME TO ATM SIMULATOR
                             1.Create Account
                             2.Existing Customer
1
11/09/22                                                        19:57:30
                             1.Current Account
                             2.Savings Account
2
Enter your name:rakshika
Enter your email:rakshika.prasad@gmail.com
Enter your mobile number:9844349270
Enter amount:50000


Thanks for creating your account.
Your account number is 508629147
Your password is 49082357
```

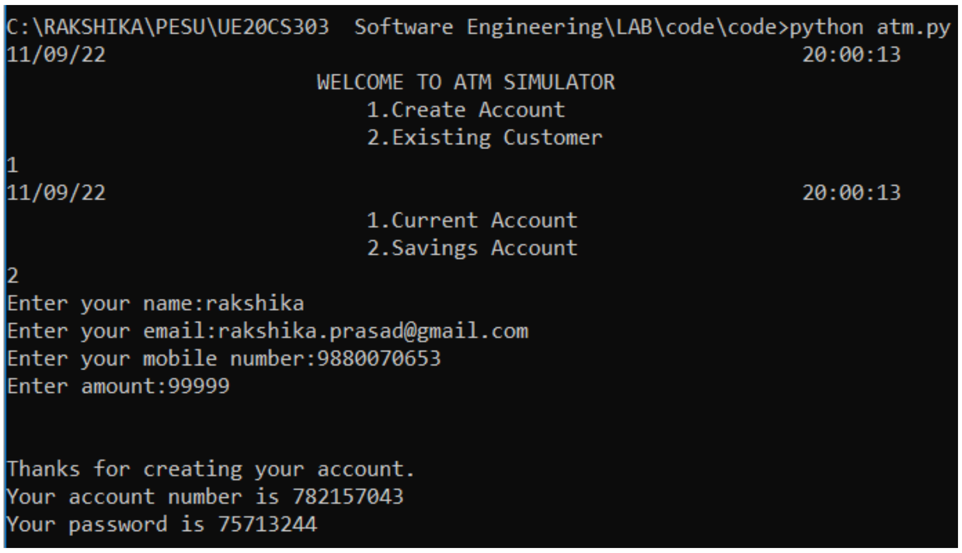
  - Just below Max value



```
C:\RAKSHIKA\PESU\UE20CS303  Software Engineering\LAB\code\code>python atm.py
11/09/22                                                        19:58:57
                         WELCOME TO ATM SIMULATOR
                             1.Create Account
                             2.Existing Customer
1
11/09/22                                                        19:58:57
                             1.Current Account
                             2.Savings Account
2
Enter your name:rakshika
Enter your email:rakshika.prasad@gmail.com
Enter your mobile number:9845627563
Enter amount:99700


Thanks for creating your account.
Your account number is 640982157
Your password is 83388980
```

- Max value

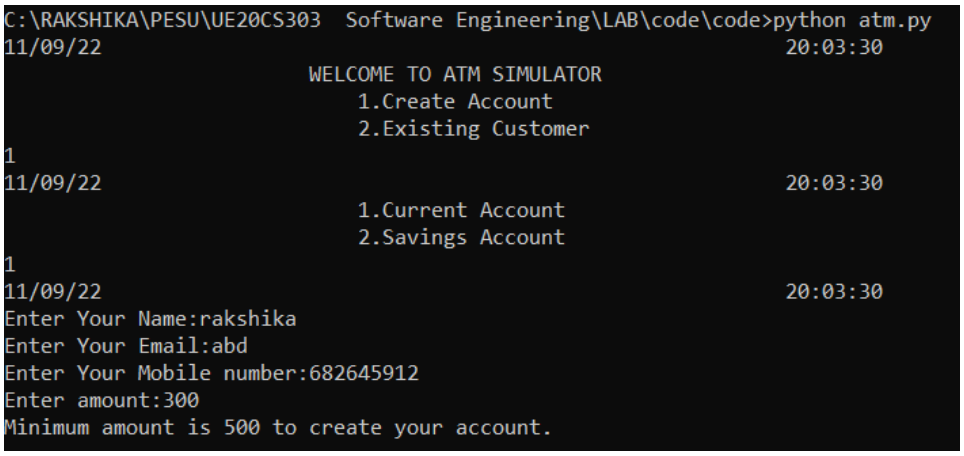


```
C:\RAKSHIKA\PESU\UE20CS303  Software Engineering\LAB\code\code>python atm.py
11/09/22                                                        20:00:13
                         WELCOME TO ATM SIMULATOR
                             1.Create Account
                             2.Existing Customer
1
11/09/22                                                        20:00:13
                             1.Current Account
                             2.Savings Account
2
Enter your name:rakshika
Enter your email:rakshika.prasad@gmail.com
Enter your mobile number:9880070653
Enter amount:99999


Thanks for creating your account.
Your account number is 782157043
Your password is 75713244
```

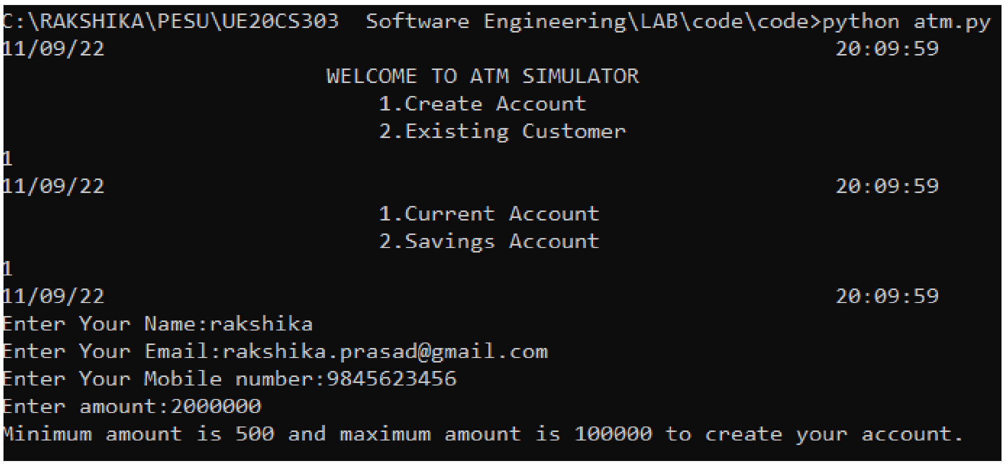
Invalid test cases:

- Below minimum value



```
C:\RAKSHIKA\PESU\UE20CS303  Software Engineering\LAB\code\code>python atm.py
11/09/22                                                        20:03:30
                         WELCOME TO ATM SIMULATOR
                             1.Create Account
                             2.Existing Customer
1
11/09/22                                                        20:03:30
                             1.Current Account
                             2.Savings Account
1
11/09/22                                                        20:03:30
Enter Your Name:rakshika
Enter Your Email:abd
Enter Your Mobile number:682645912
Enter amount:300
Minimum amount is 500 to create your account.
```

- Above maximum value



```
C:\RAKSHIKA\PESU\UE20CS303  Software Engineering\LAB\code\code>python atm.py
11/09/22                                                      20:09:59
                         WELCOME TO ATM SIMULATOR
                             1.Create Account
                             2.Existing Customer
1
11/09/22                                                      20:09:59
                             1.Current Account
                             2.Savings Account
1
11/09/22                                                      20:09:59
Enter Your Name:rakshika
Enter Your Email:rakshika.prasad@gmail.com
Enter Your Mobile number:9845623456
Enter amount:2000000
Minimum amount is 500 and maximum amount is 100000 to create your account.
```

__Problem Statement –2. b: Mutation Testing__

•Using your isolated units from the first problem statement, ideate with your team mates on how to mutate the code

•Develop at least 3 mutants of the functioning code and test all 4 code bases using the test case from the first problem statement

Mutation Testing is a type of software testing in which certain statements of the source code are changed/mutated to check if the test cases are able to find errors in source code. The goal of Mutation Testing is ensuring the quality of test cases in terms of robustness that it should fail the mutated source code.

Step 1: Faults are introduced into the source code of the program by creating many versions called mutants. Each mutant should contain a single fault, and the goal is to cause the mutant version to fail which demonstrates the effectiveness of the test cases.

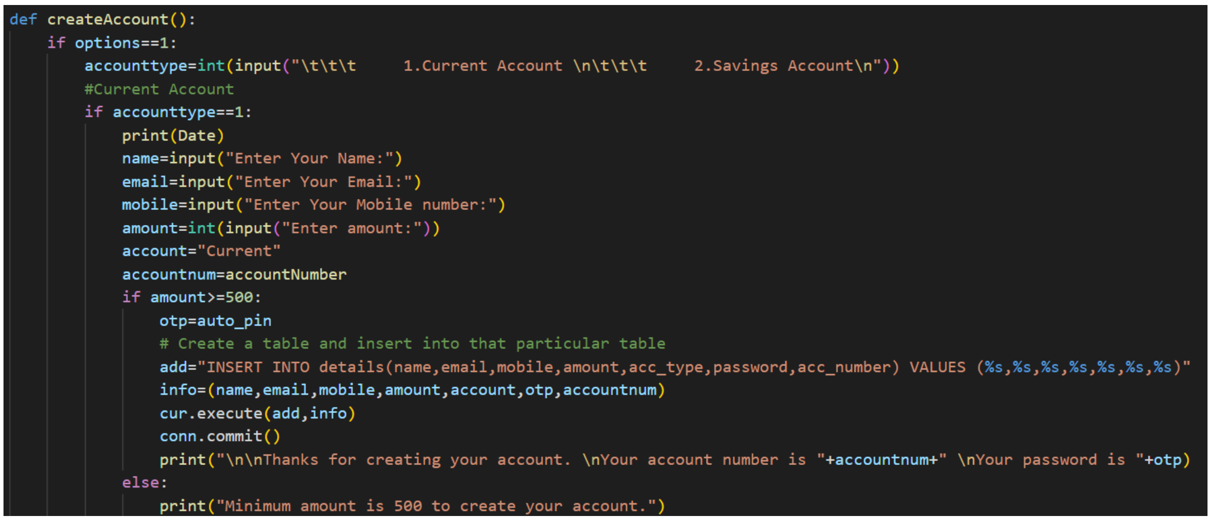
Step 2: Test cases are applied to the original program and also to the mutant program. A Test Case should be adequate, and it is tweaked to detect faults in a program.

Step 3: Compare the results of an original and mutant program.

Step 4: If the original program and mutant programs generate the different output, then that the mutant is killed by the test case. Hence the test case is good enough to detect the change between the original and the mutant program.

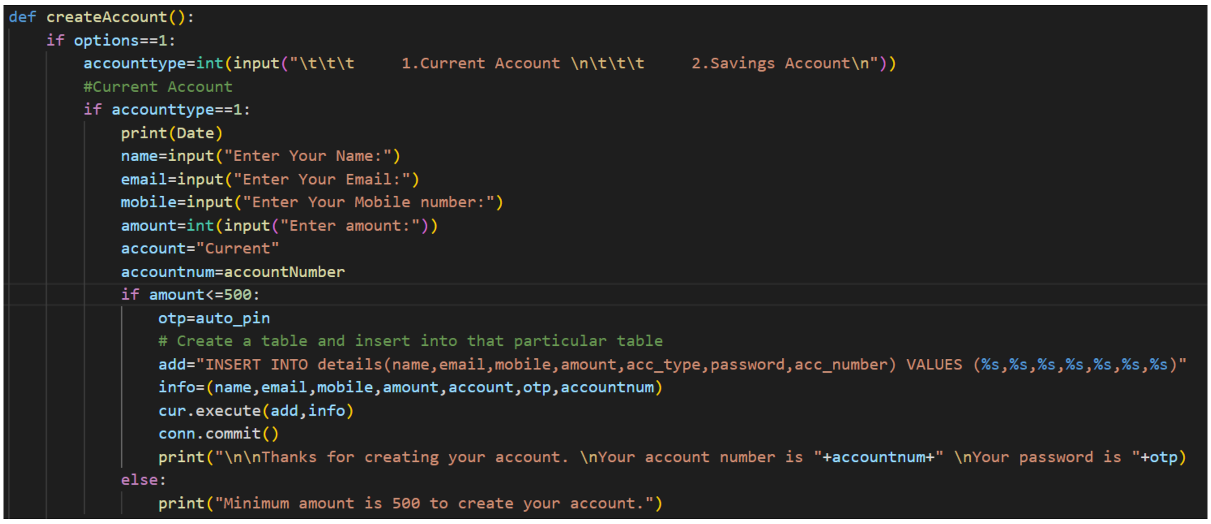
Step 5: If the original program and mutant program generate the same output, Mutant is kept alive. In such cases, more effective test cases need to be created that kill all mutants.

1) Original:



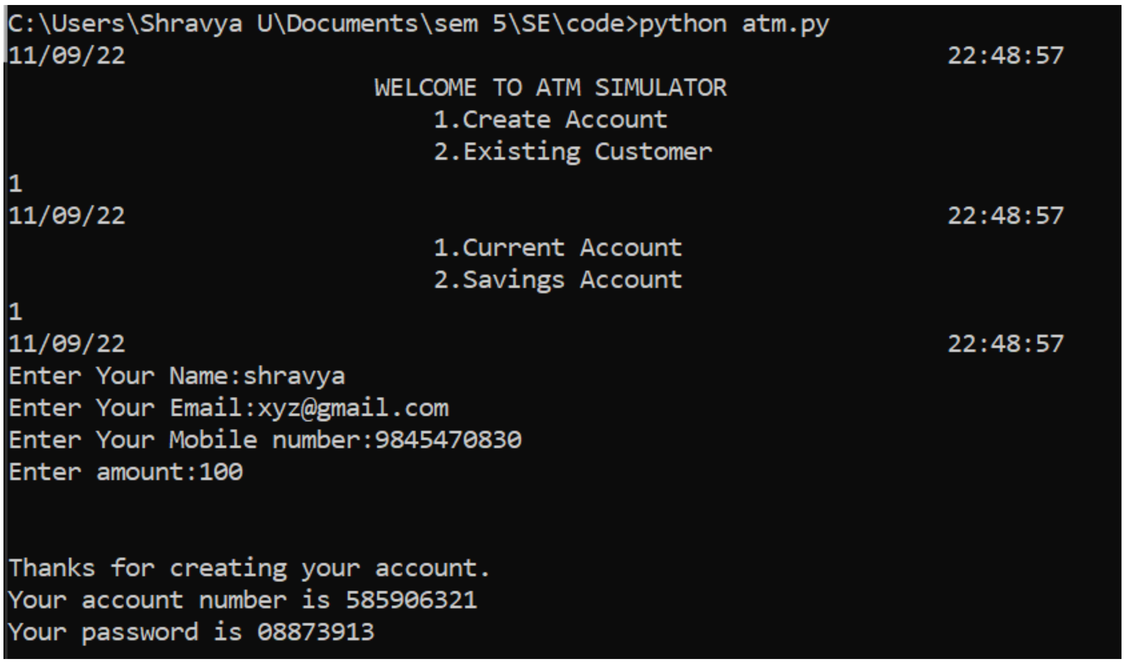
```
def createAccount():
    if options==1:
        accounttype=int(input("\t\t\t     1.Current Account \n\t\t\t     2.Savings Account\n"))
        #Current Account
        if accounttype==1:
            print(Date)
            name=input("Enter Your Name:")
            email=input("Enter Your Email:")
            mobile=input("Enter Your Mobile number:")
            amount=int(input("Enter amount:"))
            account="Current"
            accountnum=accountNumber
            if amount>=500:
                otp=auto_pin
                # Create a table and insert into that particular table
                add="INSERT INTO details(name,email,mobile,amount,acc_type,password,acc_number) VALUES (%s,%s,%s,%s,%s,%s,%s)"
                info=(name,email,mobile,amount,account,otp,accountnum)
                cur.execute(add,info)
                conn.commit()
                print("\n\nThanks for creating your account. \nYour account number is "+accountnum+" \nYour password is "+otp)
            else:
                print("Minimum amount is 500 to create your account.")
```

Mutant 1:



```
def createAccount():
    if options==1:
        accounttype=int(input("\t\t\t     1.Current Account \n\t\t\t     2.Savings Account\n"))
        #Current Account
        if accounttype==1:
            print(Date)
            name=input("Enter Your Name:")
            email=input("Enter Your Email:")
            mobile=input("Enter Your Mobile number:")
            amount=int(input("Enter amount:"))
            account="Current"
            accountnum=accountNumber
            if amount<=500:
                otp=auto_pin
                # Create a table and insert into that particular table
                add="INSERT INTO details(name,email,mobile,amount,acc_type,password,acc_number) VALUES (%s,%s,%s,%s,%s,%s,%s)"
                info=(name,email,mobile,amount,account,otp,accountnum)
                cur.execute(add,info)
                conn.commit()
                print("\n\nThanks for creating your account. \nYour account number is "+accountnum+" \nYour password is "+otp)
            else:
                print("Minimum amount is 500 to create your account.")
```

Creation of the account must be unsuccessful if the amount is less than 500.

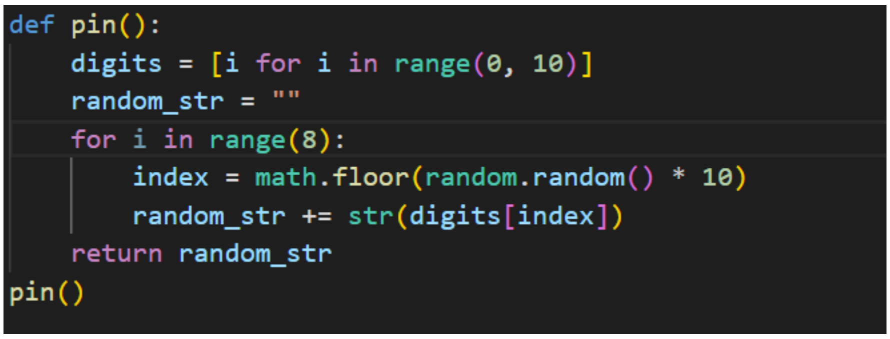


```
C:\Users\Shravya U\Documents\sem 5\SE\code>python atm.py
11/09/22                                                22:48:57
                         WELCOME TO ATM SIMULATOR
                             1.Create Account
                             2.Existing Customer
1
11/09/22                                                22:48:57
                             1.Current Account
                             2.Savings Account
1
11/09/22                                                22:48:57
Enter Your Name:shravya
Enter Your Email:xyz@gmail.com
Enter Your Mobile number:9845470830
Enter amount:100


Thanks for creating your account.
Your account number is 585906321
Your password is 08873913
```

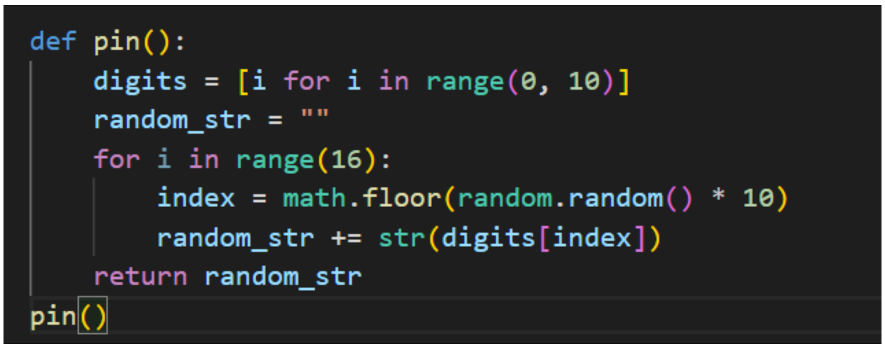
An account is created with amount less than 500. The test case fails. The test case is able to detect this error and the quality of the test case is ensured.

2) **Original:**



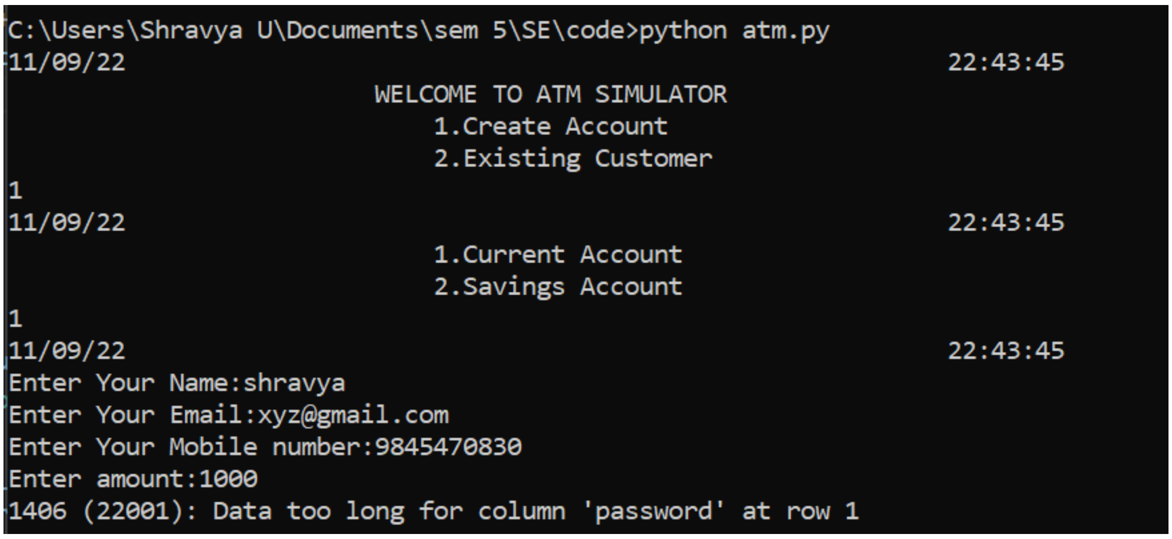
```
def pin():
    digits = [i for i in range(0, 10)]
    random_str = ""
    for i in range(8):
        index = math.floor(random.random() * 10)
        random_str += str(digits[index])
    return random_str
pin()
```

Mutant 2:



```
def pin():
    digits = [i for i in range(0, 10)]
    random_str = ""
    for i in range(16):
        index = math.floor(random.random() * 10)
        random_str += str(digits[index])
    return random_str
pin()
```

A 16 digits random password must be created on running the code.

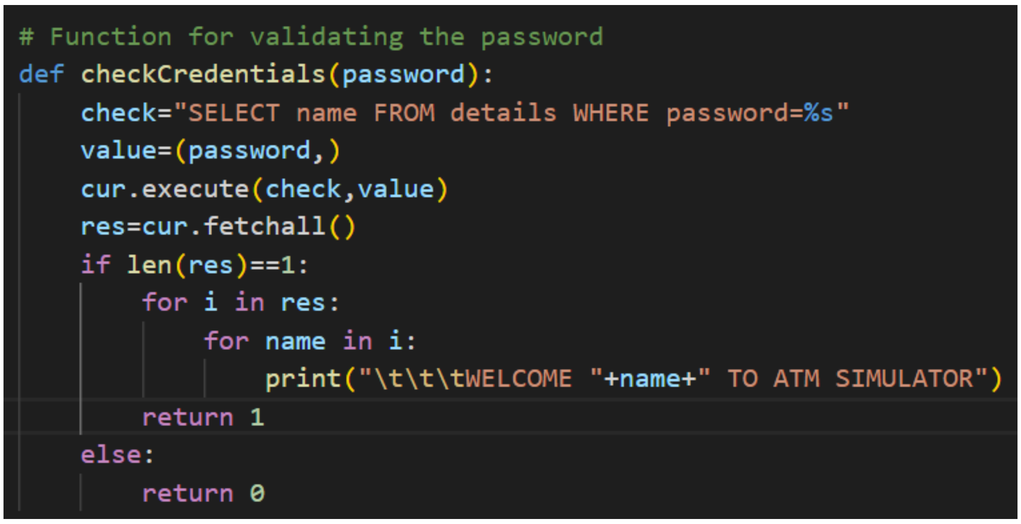


```
C:\Users\Shravya U\Documents\sem 5\SE\code>python atm.py
11/09/22                                                22:43:45
                         WELCOME TO ATM SIMULATOR
                             1.Create Account
                             2.Existing Customer
1
11/09/22                                                22:43:45
                             1.Current Account
                             2.Savings Account
1
11/09/22                                                22:43:45
Enter Your Name:shravya
Enter Your Email:xyz@gmail.com
Enter Your Mobile number:9845470830
Enter amount:1000
1406 (22001): Data too long for column 'password' at row 1
```

The generated password exceeds the password length limit.

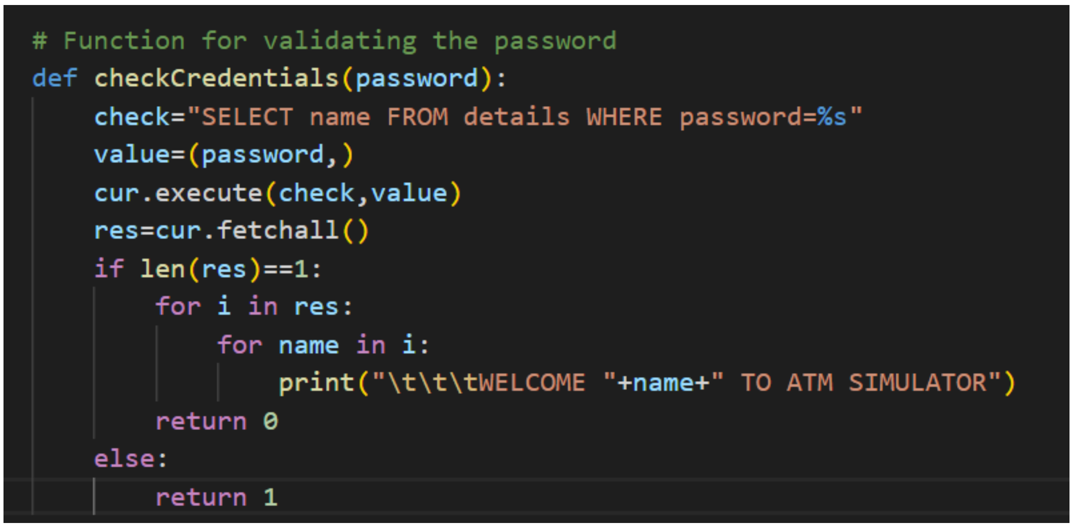
The test case is able to detect this error and the quality of the test case is ensured.

3) **Original:**



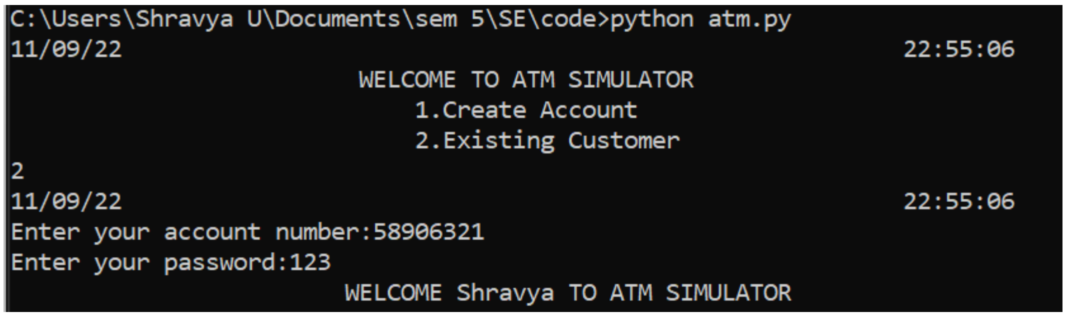
```python
# Function for validating the password
def checkCredentials(password):
    check="SELECT name FROM details WHERE password=%s"
    value=(password,)
    cur.execute(check,value)
    res=cur.fetchall()
    if len(res)==1:
        for i in res:
            for name in i:
                print("\t\t\tWELCOME "+name+" TO ATM SIMULATOR")
        return 1
    else:
        return 0
```

Mutant 3:



```python
# Function for validating the password
def checkCredentials(password):
    check="SELECT name FROM details WHERE password=%s"
    value=(password,)
    cur.execute(check,value)
    res=cur.fetchall()
    if len(res)==1:
        for i in res:
            for name in i:
                print("\t\t\tWELCOME "+name+" TO ATM SIMULATOR")
        return 0
    else:
        return 1
```

If a wrong password is entered, the account must not get logged in.



```
C:\Users\Shravya U\Documents\sem 5\SE\code>python atm.py
11/09/22                                                          22:55:06
                         WELCOME TO ATM SIMULATOR
                             1.Create Account
                             2.Existing Customer
2
11/09/22                                                          22:55:06
Enter your account number:58906321
Enter your password:123
                        WELCOME Shravya TO ATM SIMULATOR
```

On entering the wrong password. The atm simulator let's the user to login. The test case fails. The test case is able to detect this error and the quality of the test case is ensured.

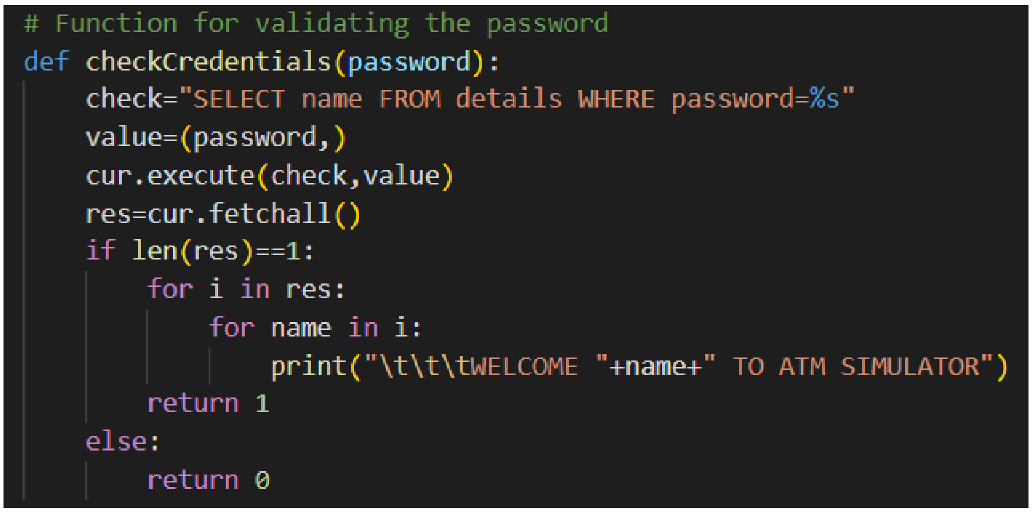
<u>Problem Statement –3: Static Testing</u>

Static testing involves validating your code without any execution. Under this problem statement, you will be expected to analyse and calculate the cyclomatic complexity of your code.

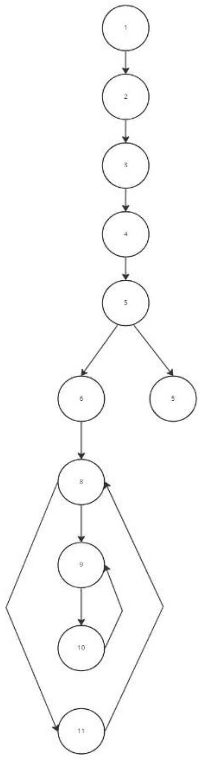
•Using the unit you selected in the first problem statement as an example, develop the control flow graph of your problem statement.

•Using the Control flow graph, calculate the cyclomatic complexity of your code.

•Using the cyclomatic complexity as an indicator, Ideate and code your unit again to reduce complexity



```python
# Function for validating the password
def checkCredentials(password):
    check="SELECT name FROM details WHERE password=%s"
    value=(password,)
    cur.execute(check,value)
    res=cur.fetchall()
    if len(res)==1:
        for i in res:
            for name in i:
                print("\t\t\tWELCOME "+name+" TO ATM SIMULATOR")
        return 1
    else:
        return 0
```



E: 12

N: 11

P: 1

Cyclomatic complexity: E – N + 2P

= 12 – 11 + 2

= 3

The cyclomatic complexity of this unit is 3 which is between 1-10. This means that the unit is a Structured and well written code, it is High Testable and utilizes less Cost and Effort.

Therefore, the unit need not be recoded to reduce the complexity.

Problem Statement – 4: Acceptance Testing

Assume your neighbouring team is the client for your code. Give them an idea of what your product is and the software requirements for the product.

• Exchange your code base and test each other's projects to see if it meets user requirements

• If you identify a bug in the project you are testing, inform the opposing team of the bug

• As a team, based in clients experience, ideate modifications to the existing project that could improve client experience.

Solution:

The aim of our ATM Simulator product is to build a Python based ATM (Automated Teller Machine) Simulation System. This ATM Simulator requires the constant updating of records of the bank. This product will have a User Interface which will make the whole process user friendly.

The product provides the access to the customer to create an account, deposit/withdraw the cash from his account, also to view reports of all accounts present. The customers can access their Account details and perform the transactions on account as per their requirements. This project has been developed to carry out the processes easily and quickly, which is not possible with the manuals systems, which are overcome by this software.

Software requirements for the product:

Functional Requirements:

1) The banking system provides services to a large number of users. The system should identify individual users of the banking system by the account number and account pin.

Input: Account Number, Account password

Output: Main Menu

2) The system should be able to perform banking transactions from the main menu that the user selects at the ATM.

Input: Main Menu Option

Output: Executed Banking transaction

3) The system should be able to verify the withdrawals and deposits.

Input: withdrawal or deposit

Output: ATM proceeds to the next step of transaction or displays message of failure to proceed to the next step.

4) The system should log out the user to control access the users bank account.

Input: exit the system.

Output: Thank you message, then display the welcome message for the next user.

Other Non-functional Requirements:

Performance Requirements:

- It must be able to perform in adverse conditions
- Must have high data transfer rate.

Safety Requirements:

- Must be safe in physical aspect, say in a booth
- Must be sealed to the floor to prevent any kind of theft
- There must be a guard just outside the booth for man power security

Security Requirements:

- Users are advised to change their PIN on first use
- Users are advised not to tell their PIN to anyone

User Requirements:

Upon first approaching the ATM, the user should experience the following sequence of events:

- The screen displays a welcome message and prompts the user to enter an account number.
- The user enters an account number.
- The screen prompts the user to enter the password associated with the specified account number.
- The user enters a password.
- If the user enters a valid account number and the correct password for that account, the screen displays main menu.
- If the user enters 2 to make a withdrawal the screen displays the menu.

As a client, the product meets all the user requirements. We have done user acceptance testing. User acceptance testing is used to determine whether the product is working for the user correctly. Specific requirements which are quite often used by the customers are primarily picked for the testing purpose. This is also termed as End-User Testing. The product is tested against the SRS provided by the product manager.

Bugs found:

- Input validation
- Encryption of password

As a team, based on clients experience we would suggest a few modifications in the product.

- The product can have an input validation.

- The product can have a use of encrypted password.

<u>**Problem Statement – 5:**</u>

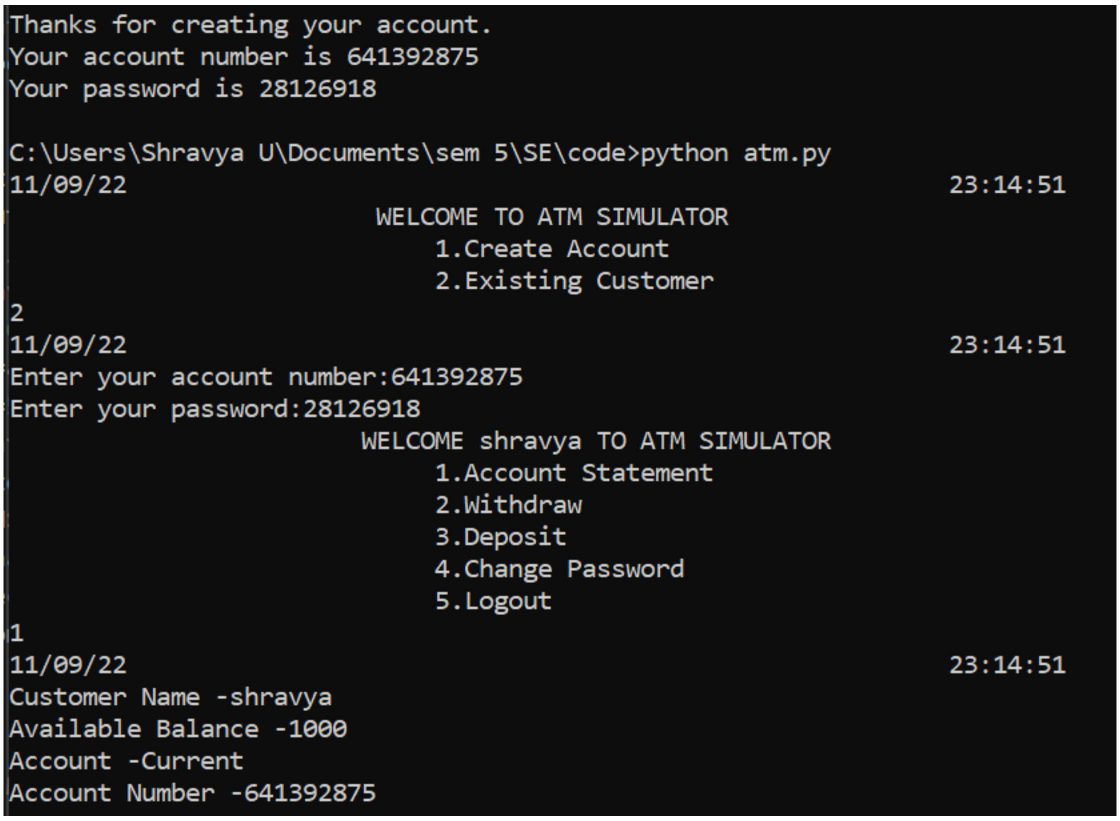
Maintenance Activities Once a product is completed, it is handed off to a service-based company to ensure all maintenance activities are performed without the added expenditure of skilled developers. However, a few tasks are performed by the maintenance team to gauge the product better. In this problem statement, you will be asked to experiment with your code.

• Exchange code bases with your neighbouring teams and reverse engineer a block of code in order to understand it's functionality

• After understanding the code block, Re-Engineer the code. Ideate how to refactor the code and the portion of the code base you would have to change. Discuss how the new changes would impact the time and space complexity of the project during execution

• After Reverse Engineering and Re-Engineering, the code, perform acceptance testing between the teams.

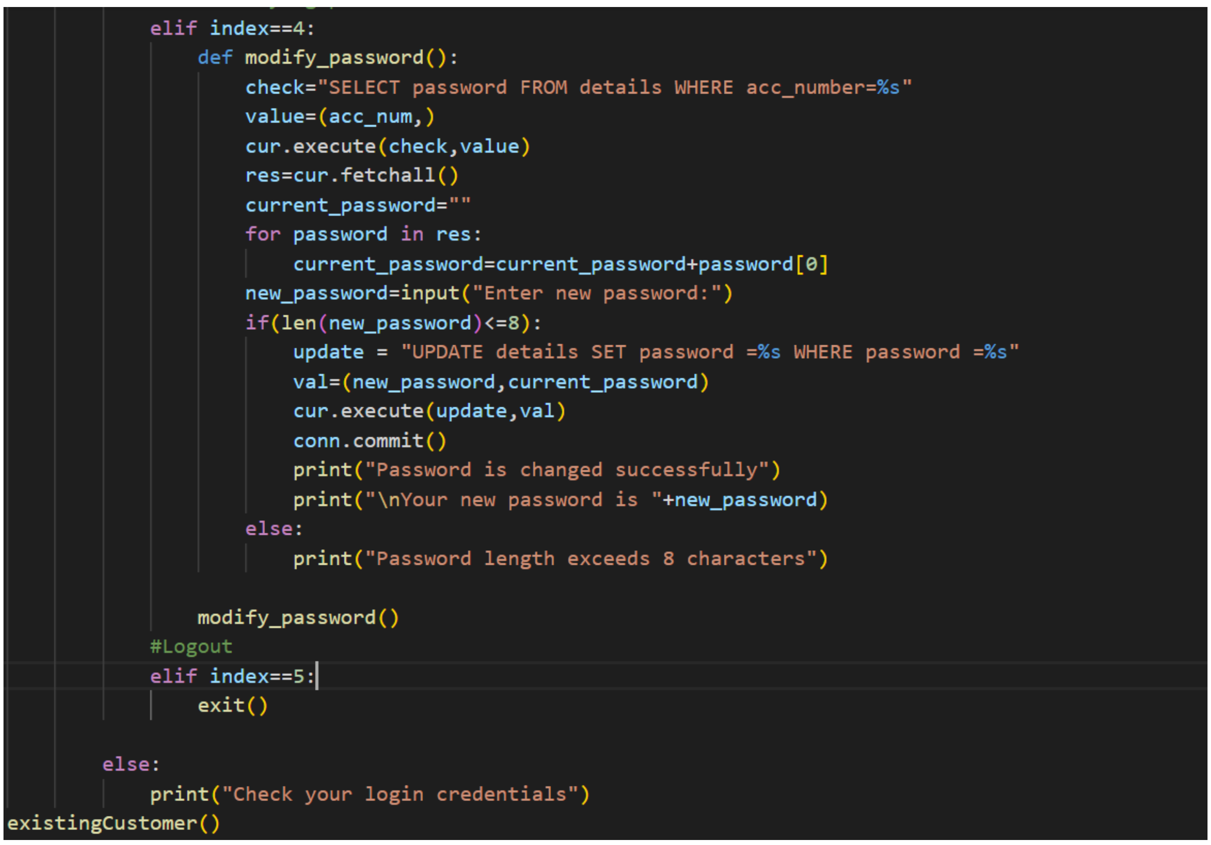
Software Re-Engineering is the examination and alteration of a system to reconstitute it in a new form. The principle of Re-Engineering when applied to the software development process is called software re-engineering. It affects positively software cost, quality, service to the customer, and speed of delivery. In Software Re-engineering, we are improving the software to make it more efficient and effective.



```
Thanks for creating your account.
Your account number is 641392875
Your password is 28126918

C:\Users\Shravya U\Documents\sem 5\SE\code>python atm.py
11/09/22                                                        23:14:51
                         WELCOME TO ATM SIMULATOR
                             1.Create Account
                             2.Existing Customer
2
11/09/22                                                        23:14:51
Enter your account number:641392875
Enter your password:28126918
                        WELCOME shravya TO ATM SIMULATOR
                             1.Account Statement
                             2.Withdraw
                             3.Deposit
                             4.Change Password
                             5.Logout
1
11/09/22                                                        23:14:51
Customer Name -shravya
Available Balance -1000
Account -Current
Account Number -641392875
```

A user can be allowed to navigate to any provided option any number of times unless the user wants to logout. It positively affects the quality of the user's experience. It makes the user to withdraw or deposit without logging in again. The portion of the code that we would like to change:

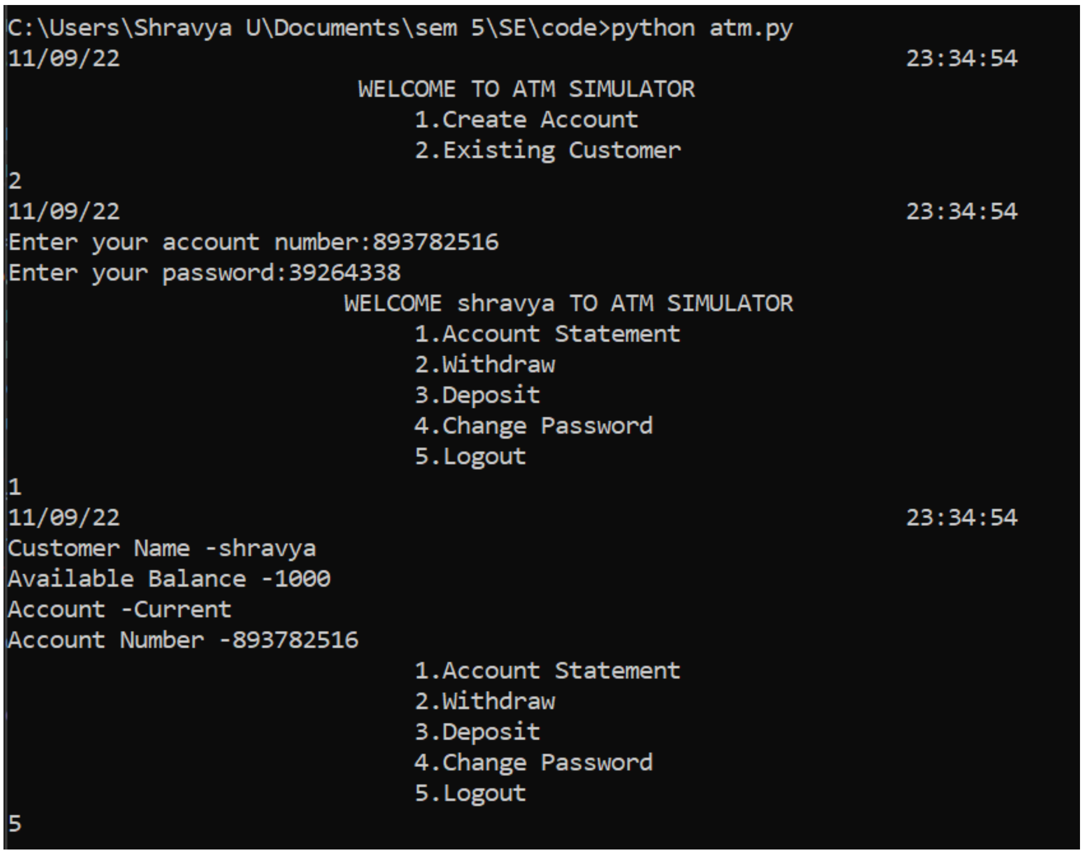


```python
            elif index==4:
                def modify_password():
                    check="SELECT password FROM details WHERE acc_number=%s"
                    value=(acc_num,)
                    cur.execute(check,value)
                    res=cur.fetchall()
                    current_password=""
                    for password in res:
                        current_password=current_password+password[0]
                    new_password=input("Enter new password:")
                    if(len(new_password)<=8):
                        update = "UPDATE details SET password =%s WHERE password =%s"
                        val=(new_password,current_password)
                        cur.execute(update,val)
                        conn.commit()
                        print("Password is changed successfully")
                        print("\nYour new password is "+new_password)
                    else:
                        print("Password length exceeds 8 characters")

                modify_password()
            #Logout
            elif index==5:
                exit()

        else:
            print("Check your login credentials")
existingCustomer()
```

All the options can be provided to the user after a transaction. The time complexity remains the same but the space complexity increases due to the introduction of a new feature to improve user experience.



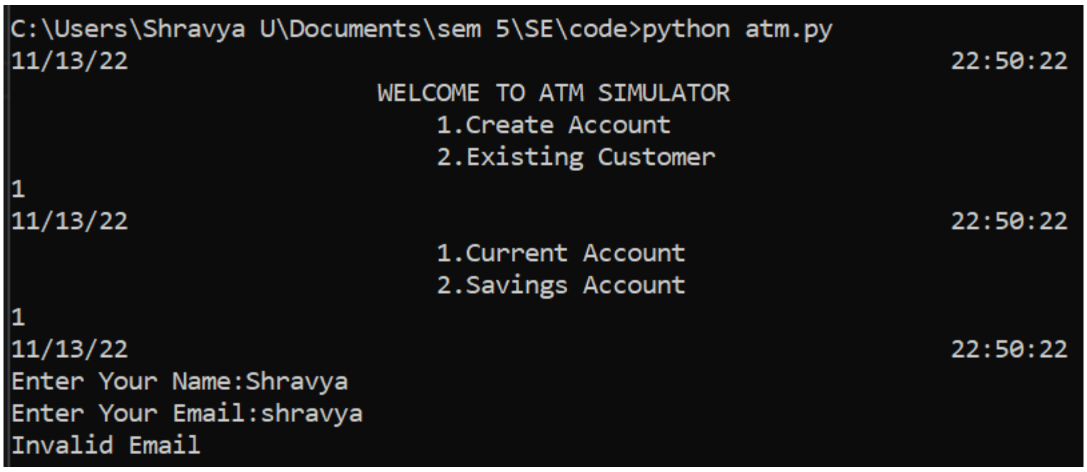
```
C:\Users\Shravya U\Documents\sem 5\SE\code>python atm.py
11/09/22                                                        23:34:54
                         WELCOME TO ATM SIMULATOR
                             1.Create Account
                             2.Existing Customer
2
11/09/22                                                        23:34:54
Enter your account number:893782516
Enter your password:39264338
                        WELCOME shravya TO ATM SIMULATOR
                             1.Account Statement
                             2.Withdraw
                             3.Deposit
                             4.Change Password
                             5.Logout
1
11/09/22                                                        23:34:54
Customer Name -shravya
Available Balance -1000
Account -Current
Account Number -893782516
                             1.Account Statement
                             2.Withdraw
                             3.Deposit
                             4.Change Password
                             5.Logout
5
```

The user is now allowed to navigate the all the options available even after performing a transaction without logging in again.

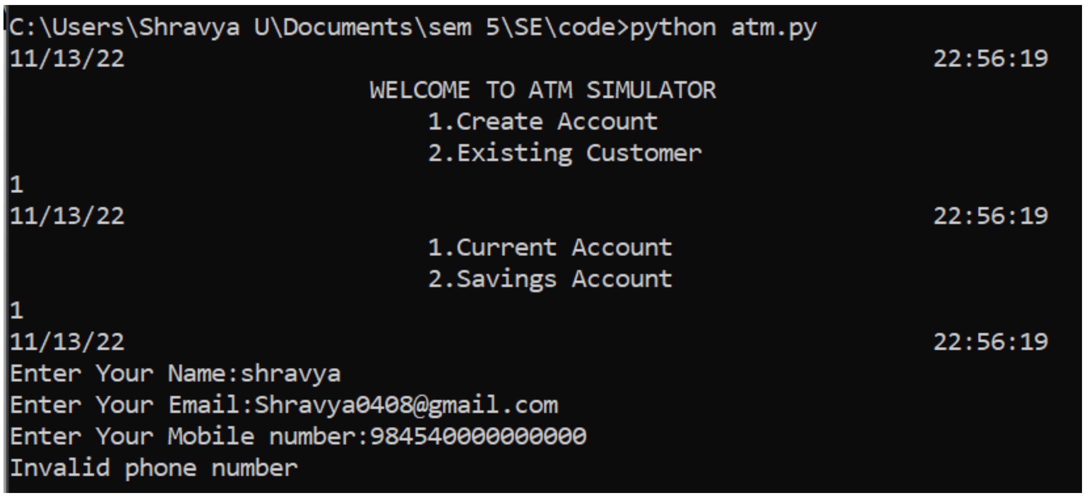
As a client, the product meets all the user requirements. We have done user acceptance testing. User acceptance testing is used to determine whether the product is working for the user correctly. Specific requirements which are quite often used by the customers are primarily picked for the testing purpose. This is also termed as End-User Testing. The product is tested against the SRS provided by the product manager. No bug is found in the new product.

The new product will have the following features discussed during the acceptance testing:

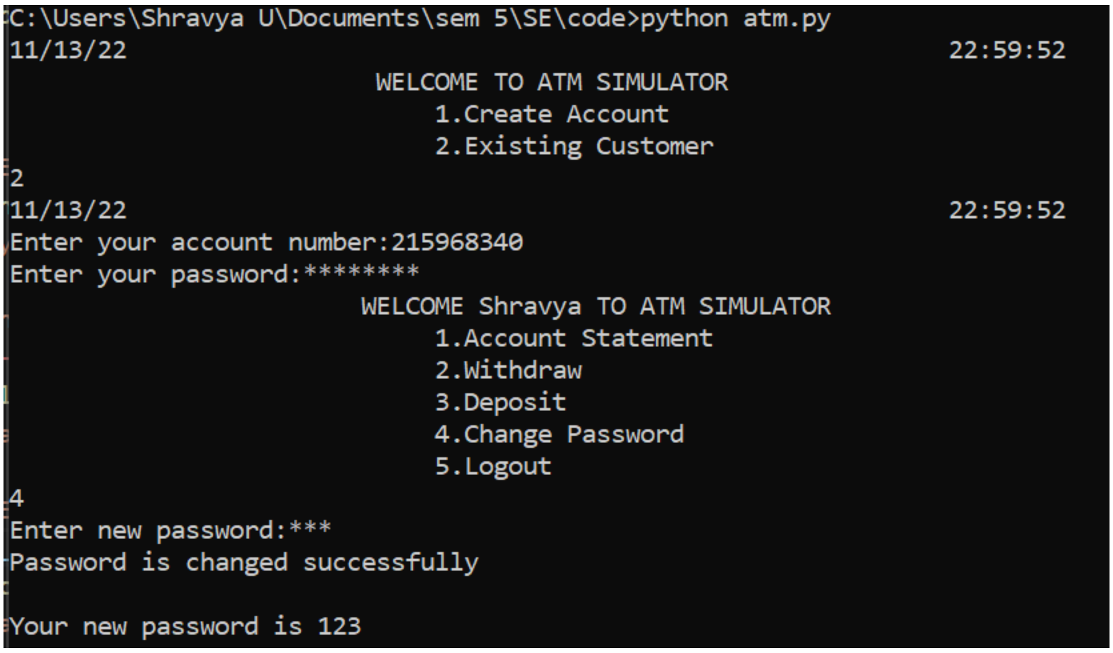
- Encrypted password
- Input validation



```
C:\Users\Shravya U\Documents\sem 5\SE\code>python atm.py
11/13/22                                                          22:50:22
                         WELCOME TO ATM SIMULATOR
                             1.Create Account
                             2.Existing Customer
1
11/13/22                                                          22:50:22
                             1.Current Account
                             2.Savings Account
1
11/13/22                                                          22:50:22
Enter Your Name:Shravya
Enter Your Email:shravya
Invalid Email
```



```
C:\Users\Shravya U\Documents\sem 5\SE\code>python atm.py
11/13/22                                                          22:56:19
                         WELCOME TO ATM SIMULATOR
                             1.Create Account
                             2.Existing Customer
1
11/13/22                                                          22:56:19
                             1.Current Account
                             2.Savings Account
1
11/13/22                                                          22:56:19
Enter Your Name:shravya
Enter Your Email:Shravya0408@gmail.com
Enter Your Mobile number:984540000000000
Invalid phone number
```



```
C:\Users\Shravya U\Documents\sem 5\SE\code>python atm.py
11/13/22                                                              22:59:52
                         WELCOME TO ATM SIMULATOR
                             1.Create Account
                             2.Existing Customer
2
11/13/22                                                              22:59:52
Enter your account number:215968340
Enter your password:********
                        WELCOME Shravya TO ATM SIMULATOR
                             1.Account Statement
                             2.Withdraw
                             3.Deposit
                             4.Change Password
                             5.Logout
4
Enter new password:***
Password is changed successfully

Your new password is 123
```