Sara Aurora Rusi Ruiz 15/12/2024 Element 1

1406423  Cyber Security (Threat Detection)

**Introduction to Programming**

Cash Application Program (Code Report)

**Contents Page**

**Introduction to Programming Code Report**

**Element 1 Report**

1. Code Description and Key Functionalities…………………………………………..2

2. Code Structure…………………………………………………………...…………….2

3. Key Functionalities………………………………………………………………...…..3

4. Main Menu Options……………………………………………………………………4

5. Balance Enquiry.……………………………………………………………………….4

6. Deposit…………………...………………………………………..……………………4

7. Withdrawal…………………………………………………………………………..….5

8. Session Termination………………….…………………………………………..……6

9. Implementation Details and Optimisations………………………………………….7

10. PIN Validation Algorithm……………………………………………………………..8

11. Menu Navigation……………………………………………………………………...8

12. Dynamic Balance Updates…………………………………………………………..8

13. Error Handling…………………………………………………………………….…..8

14. User Instructions……………………………………………………..……………….8

15. Development Challenges…………………………………………………………….9

16. Program Flow/Loops .………………………………………………………………..9

17. Error handling Challenges…………………………………………………………...10

18. Input Validation………………………………………………………………………..10

19. Problem-Solving Implementations……………………………………………….....10

20. Project Success……………………………………………………………………....11

21. Code Strengths…………………………………………………………………….....11

22. Code Weakness……………………………………………………………………....11

23. Future Enhancements……………………………………………………….…..…...11

24. Reflexion…………………………………………………………………………….....12

25. Bibliography…………………………………………………………………………....12

**Link to GitHub Repository:** https://github.com/sararusi1030/introprog

# Introduction to Programming Code Report

## Element 1 Report

## Code Description and Key Functionalities

The code provided is a cash machine simulation designed to let users interact with a banking system. After validating their PIN, users can withdraw and deposit funds, check their balance, or exit the program. The code uses input validation, conditional statements, and looping structures to achieve functionality.

## Code Structure

The code is modularised into functions and comprises six components: Initialisation and PIN validation, Main Menu Modular Function, Withdrawal and Deposit Functions, and Session Termination. These components are related in sequence but conditionally.

A computer screen shot of a white background

Description automatically generated

The menu functionality has been isolated into a separate function to ensure the code is cleaner and easier to read. Users must successfully validate their PIN before obtaining access to more features, and specific conditions must be met to proceed with deposits and withdrawals.

# Key Functionalities

## Initialisation and PIN validation

This function ensures only authorised users access the system. The code uses a while loop providing users 3 attempts to introduce the correct PIN, once the PIN is validated the user can access the “Main Menu”.

The try-except block ensures the program does not crash if the user enters a non-numeric value. If the three attempts are invalid the program proceeds to exit with the message “Invalid pin. Please enter a numeric value”.

A screen shot of a computer

Description automatically generated

## Flowchart for PIN Validation:

Start -> Input PIN

|

Correct? -- Yes --> Main Menu

|

Attempts Left? -- No --> Exit

|

Yes --> Display Error, Retry

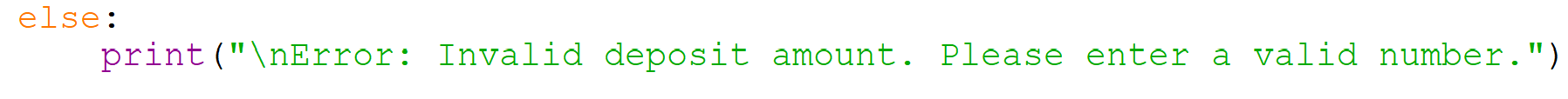
## Main Menu Options

After PIN validation, the program calls the display\_menu function from menu.py to display the main menu. This function gives the user four options: balance, deposit, withdrawal, and cancel.

A close-up of red text

Description automatically generated

The input is validated to ensure the option entered is numeric and within the rang 1-4. If the input is invalid, the program shows the message "Invalid option. Please enter a number."



## Balance and Inquiry

The purpose of this function is to display the user’s updated balance. When the user selects option 1 from the menu, the program automatically prints the value of the balance variable.

A close-up of a white background

Description automatically generated

## Deposit

Allows users to add funds to their accounts when selecting the second option from the menu. Users are prompted to enter a deposit amount validating input to ensure numeric value is greater than zero. If valid, the deposit is added to the balance variable, and an updated balance is displayed.

A screenshot of a computer program

Description automatically generated

## Logic Diagram:

Input Deposit Amount

|

Valid? -- No --> Error

|

Yes --> Positive? -- No --> Error

|

Yes --> Update Balance, Display Balance

## Withdrawal

Users can withdraw funds from their accounts by selecting option 3. The withdrawal amount is validated to ensure it is numeric, positive, and does not exceed the available balance.  
When all conditions are met, the withdrawal amount is subtracted from the balance variable, and both the withdrawal amount and updated balance are displayed. If the withdrawal amount exceeds the balance, an error message is displayed:

A screenshot of a computer

Description automatically generated

## Logic Diagram:

Input Withdrawal Amount

|

Valid? -- No --> Error

|

Yes --> Positive? -- No --> Error

|

Yes --> Sufficient Funds? -- No --> Error

|

Yes --> Update Balance, Display Balance

## Session Termination

Allows users to exit the program safely when option number 4 is selected, a thank you message is displayed, and the program exits the loop.

A close up of text

Description automatically generated

### The following Diagram shows the overall flow of the program:

Start -> PIN Validation

|

Valid? -- No --> Exit

|

Yes --> Display Main Menu

|

Option Selected?

|

1 --> Display Balance

|

2 --> Deposit Funds

|

3 --> Withdraw Funds

|

4 --> Exit Program

|

Invalid? --> Error, Retry

# Implementation Details and Optimisations

The design focuses on simplicity and robustness ensuring smooth functionality and the error handling and validation optimises the user experience. The code utilises and implements fundamental programming constructs such as conditional statements, functions, modular code, loops, and input validation. The following is a list of the key algorithms and strategies:

## PIN Validation Algorithm

A while loop manages the three PIN attempts to ensure secure authentication. The try-except block handles non-numeric input errors, preventing program crashes.

## Menu Navigation

The menu interface is modularized into the display\_menu() function, which continuously prompts users until they choose the exit option. Input is validated to ensure numeric entries between 1-4, optimizing usability and preventing invalid operations.

## Dynamic Balance Updates

The code ensures real-time updates to the balance variable during deposits and withdrawals. Input validation checks ensure that deposits and withdrawals are positive and lower than the available balance.

## Error Handling

User-friendly error messages guide users through invalid operations without terminating the session.

# User Instructions

## Follow the steps to use the BTE Bank Cash Machine Application:

1. Initialise the program: Enter your bank card into the cardholder.
2. Enter Your PIN: Input your 4-digit PIN when prompted (default PIN: 1234).

If PIN is incorrect, you will see: “Incorrect pin number. You have (x) attempts left.”

1. Access Menu: After a successful PIN entry, the main menu is displayed:

1 = Balance

2 = Deposit

3 = Withdraw

4 = Cancel

1. Operations Performed:

Option 1: View your account **balance.**

Option 2: **Deposi**t funds (e.g., enter 1500 to deposit £1500).

Option 3: **Withdrawal** funds (ensure sufficient balance).

Option 4: **Terminate** the session.

1. Exit Safely: Select 4 to finish and retrieve your bank card.

# Development Challenges

As a beginner in my first year of coding, developing the cash app provided for this assignment presented several challenges. The most significant are understanding program flow and code order, handling errors, and validating user input gracefully without causing the program to crash. Adding a modular function into the code was also a challenge. These challenges occurred due to my limited experience and overcoming them helped me understand the importance of structured logic, testing, and planning.

## Program Flow/Loops

At the start of the project, I struggled to design the structure of the program. Finding out how to repeatedly display the menu while allowing users to perform multiple actions was one of the initial obstacles. I overcame this by learning how to use while loops and modular functions. It also took me some time to understand loop termination and utilise the break statement.

## Error Handling Challenges

I encountered constant crashes when “users” entered non-numeric values, especially for the PIN and transaction amounts. This kept happening because the program attempted to process invalid inputs, which caused multiple runtime errors. I had to resolve this by incorporating “try-except” blocks to handle invalid input correctly. The process of testing this repeatedly showed me the importance of handling these cases.

## Input Validation

Another challenge was ensuring users entered the correct input amounts when processing deposits and withdrawals. In my first application design users could deposit negative values or withdraw a higher amount than their balance.

I had to add conditions to check the validity of inputs and provide clear error messages guiding the users after every error.

## Problem-Solving Implementations

Addressing development challenges was a key part of the application design process. Input Validation had to be implemented, for example, “isdigit()” takes care of checking that only numeric values are inserted as PINs.

The process of debugging showed me to test extensively and search for edge cases, such as entering excessive withdrawal amounts. I also learned the value of breaking problems into smaller steps by using loops and conditions effectively as well as prioritising user-friendly error messages to guide users through the program.

# Reflection and Future Improvements

## Project Success

Overall, the project was a success as it achieved the goal of simulating a basic cash machine. The program handles essential functions effectively such as pin validation, balance inquiries, deposits, and withdrawals while incorporating user-friendly error messages guiding the user and robust validation.

## Code Strengths

A) The code is modular, ensuring easier readability and maintenance.  
 B) Error handling prevents crashes/provides clear messages for invalid operations.  
 C) Real-time balance updates ensure an accurate reflection of transactions.

## Code Weaknesses

1. The program is limited to a single user and has no multi-account functionality.
2. Security could be compromised as the PIN is stored in plaintext non-encrypted.
3. Input validation could be further improved to handle decimal currency inputs.

## Future Enhancements

1. Multi-user support to allow multiple users with different PINs and account balances.
2. PIN Encryption: Implement encryption to improve security.
3. Transaction Log: Add an account history feature to store transaction logs.
4. Supporting decimal inputs for currency amounts expanding usability.

## Reflexion

Reflecting on this project, I have learned the importance of robust planning, testing, debugging, and breaking down complex problems into smaller manageable components. I have gained valuable insight into the importance of functions and modular code.

Initially, the code was more linear and difficult to read and maintain, by breaking it into smaller, reusable functions, such as display\_menu() and individual functions for balance checking, deposits, and withdrawals, I learned how functions help make the code more organized, readable, and scalable. Functions allow for the encapsulation of specific tasks, which makes it easier to update or fix isolated parts of the program without affecting the entire system.

These lessons learned will guide my future programming projects and efforts and gathering user feedback would be very important to guide optimisations for new future development.

# Bibliography

Python Software Foundation, 2024. *The Python Tutorial*. <https://docs.python.org/3/tutorial/index.html> [Accessed 29 November 2024].

Real Python, 2024. *Python Input and Output*. https://realpython.com/python-input-output/ [Accessed 29 November 2024].

W3Schools, 2024. *Python Tutorial*. https://www.w3schools.com/python/ [Accessed 30 November 2024].

GeeksforGeeks, 2024. *Python Program for ATM Machine*. https://www.geeksforgeeks.org/python-program-for-atm-machine/ [Accessed 01 December 2024].

TutorialsPoint, 2024. *Python Programming Language*. https://www.tutorialspoint.com/python/index.htm [Accessed 03 December 2024].

GitHub, 2024. *Simple ATM Machine Program in Python*. <https://github.com/minhaj-313/Simple-ATM-Machine-Program-in-Python> [Accessed 03 December 2024].

Stack Overflow, 2024. *Programming Q&A Platform*. <https://stackoverflow.com/> [Accessed 05 December 2024].

Programiz, 2024. *Python Programming Basics*. https://www.programiz.com/python-programming [Accessed 05 December 2024].

PythonForBeginners, 2024. *Python Basics for Beginners*.<https://www.pythonforbeginners.com/> [Accessed 09 December 2024].

Codecademy, 2024. *Learn Python 3*.https://www.codecademy.com/learn/learn-python-3 [Accessed 09 December 2024].

freeCodeCamp, 2024. *Python Tutorials*. https://www.freecodecamp.org/news/tag/python/ [Accessed 09 December 2024].

HackerRank, 2024. *10 Days of Python Programming Challenges*.https://www.hackerrank.com/domains/tutorials/10-days-of-python [Accessed 09 December 2024].

LearnPython.org, 2024. *Interactive Python Tutorials*.<https://www.learnpython.org/> [Accessed 10 December 2024].

JetBrains, 2024. *Programming in Python – Articles and Tips*. <https://blog.jetbrains.com/pycharm/> [Accessed 10 December 2024].

Python Examples, 2024. *Real-World Python Code Examples*. <https://pythonexamples.org/> [Accessed 10 December 2024].