GROUP 1 WEEK 1 ASSESMENT REPORT

PROJECT REPORT: ATM MACHINE SIMULATION

INTRODUCTION

This project employs the Python programming language to simulate an ATM by integrating a continuous control loop with modular functions. The aim is to demonstrate how a primary loop can coordinate with specific functions to manage banking operations such as logging in, viewing balances, and performing transfers.

METHODOLOGY

Initialization: The program begins by prompting the user to enter a PIN, a step that restricts access to the program loop and its banking functions.

Verifying the PIN: Logic is used to confirm the PIN is 4 digits before the system enters the primary loop or executes any functions.

Security Check: If the PIN is invalid, the program terminates immediately to prevent unauthorized access to the system's loop and sensitive functions.

The Primary Loop: The software employs a while loop to maintain operation, repeatedly displaying the menu so users can choose different functions until they exit.

KEY FEATURES

Modular Code: The code is structured into smaller units known as functions, such as deposit() and withdraw(), which are called repeatedly within the execution loop.

Global Variables: Global variables like balance are utilized within these functions, ensuring the loop always displays the current state of the account.

Verifying Numeric Input: Prior to calculations, the functions employ .isdigit() to validate input, ensuring the primary loop continues smoothly without crashing due to non-numeric errors.

Logic Checks: Logic checks within the withdrawal functions confirm the user possesses sufficient funds, maintaining data integrity throughout the execution of the program loop.

CONCLUSION

This project is a simple example of how Python works, successfully demonstrating how a logical loop and defined functions can build a working ATM system.

Algorithm: ATM Machine Simulation

1. **Start**
2. **Card and PIN input**
    a. Display “Card inserted”.
    b. Ask user to enter PIN.
    c. If PIN length is not 4
        i. Display “Invalid pin”.
        ii. **Stop**.
3. **Initialize balance**
    a. Set `balance = 1000`.
4. **Repeat the following steps (main loop):**
    a. Display ATM menu:
        1. Check Balance
        2. Deposit Money
        3. Withdraw Money
        4. Exit
    b. Read user choice.
5. **If choice = 1 (Check Balance)**
    a. Call `check_balance()` which prints “Your balance is: balance”.
    b. Go back to step 4.
6. **Else if choice = 2 (Deposit Money)**
    a. Ask user to enter deposit amount as text.
    b. If amount contains only digits (`isdigit` is true)
        i. Convert to integer.
        ii. Add amount to balance.
        iii. Display “Amount deposited successfully: amount”.
        iv. Display “Current balance is: balance”.
    c. Else
    Display "invalid amount ", number only .
    Go back to step 4.

7. **Else if choice = 3 (Withdraw Money)**
    a. Ask user to enter withdrawal amount as text.

    b. If amount contains only digits

        i. Convert to integer.
        ii. If amount ≤ balance
            1. Subtract amount from balance.
            2. Display “Amount withdrawn successfully: amount”.
            3. Display “Current balance is: balance”.
        iii. Else
            1. Display “Insufficient balance.”

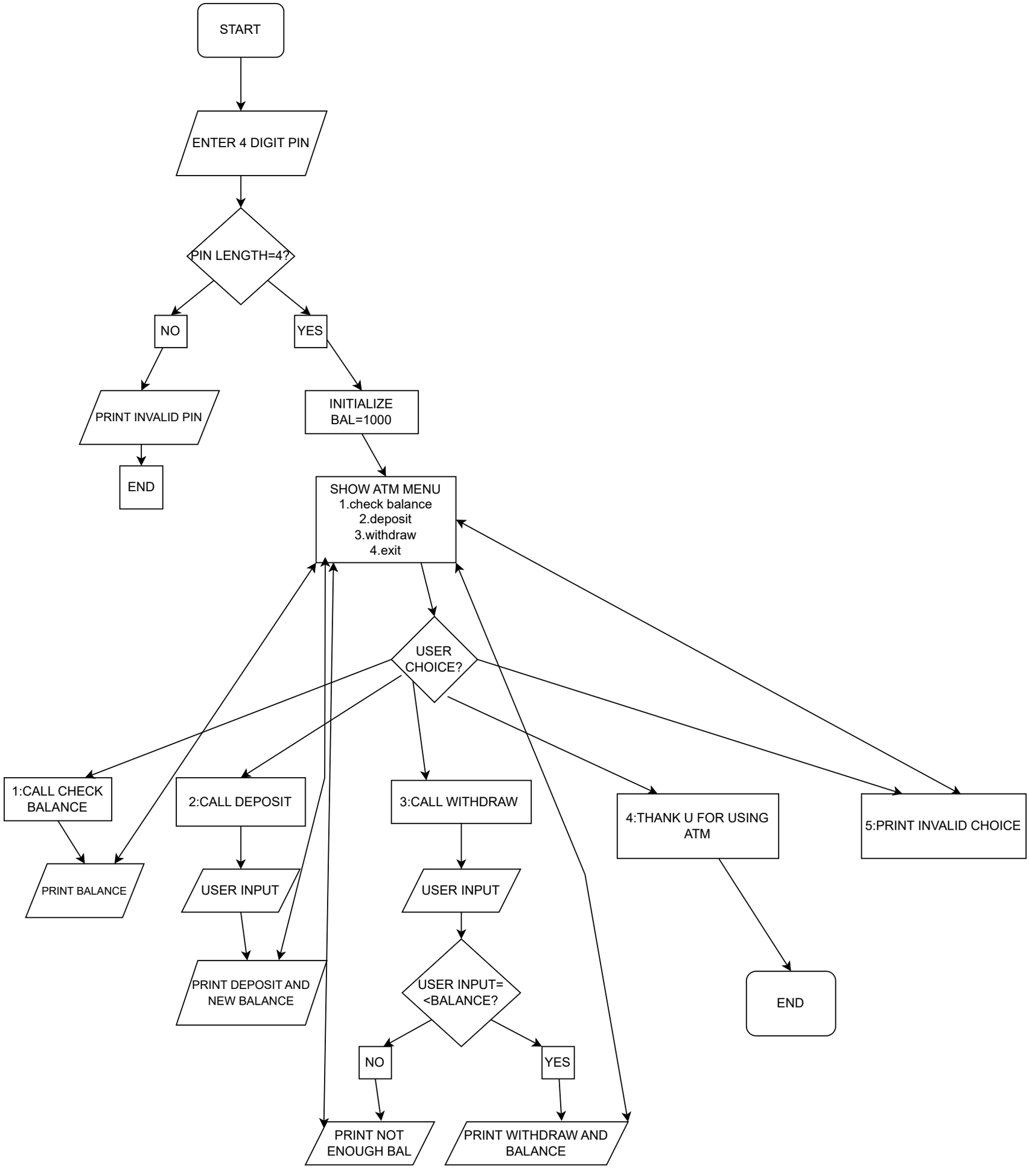
    c. Else
        i. Display “Invalid amount. Numbers only.”

    d. Go back to step 4.

8. **Else if choice = 4 (Exit)**
    a. Display “Thank you for using ATM.”
    b. **Stop**.

9. **Else (any other choice)**
    a. Display “Invalid choice. Please try again.”
    b. Go back to step 4.

FLOWCHART


START
ENTER 4 DIGIT PIN
PIN LENGTH=4?
NO
YES
PRINT INVALID PIN
END
INITIALIZE
BAL=1000
SHOW ATM MENU
1.check balance
2.deposit
3.withdraw
4.exit
USER
CHOICE?
1:CALL CHECK
BALANCE
PRINT BALANCE
2:CALL DEPOSIT
USER INPUT
PRINT DEPOSIT AND
NEW BALANCE
3:CALL WITHDRAW
USER INPUT
USER INPUT=
<BALANCE?
NO
YES
PRINT NOT
ENOUGH BAL
PRINT WITHDRAW AND
BALANCE
4:THANK U FOR USING
ATM
END
5:PRINT INVALID CHOICE

CODE EXPLANATION

1. Card insertion and PIN verification

- `print(" : Card inserted :")`
  Shows a message that the user has inserted the card.
- `pin = input(" enter your pin:")`
  Asks the user to enter the ATM PIN as input.
- `if len(pin) != 4:`
  Checks whether the entered PIN has exactly 4 characters (like a standard 4-digit PIN).
- `print(" Invalid pin")` and `exit()`
  If the PIN is not 4 digits, it shows an error message and stops the program immediately.

2. Initial balance setup

- `balance = 1000`
  Sets the starting account balance to 1000 units (for example, 1000 rupees) for the simulation.

`def check_balance():`
Defines a function whose job is to show the current balance.

`print("Your balance is:", balance)`
When this function is called, it prints the current value stored in `balance` on the screen.

3.Deposit module

- `def deposit():`
  Defines a function to handle deposit operations.
- `global balance`
  Tells Python that this function will use and modify the global variable `balance` instead of creating a local one.
- `amount = input("Enter amount to deposit: ")`
  Takes the deposit amount from the user as input (initially as text)
- In the improved version from your document, first `amount.isdigit()` is checked, then `amount = int(amount)` is done, and finally `balance = balance + amount` to update the balance safely.
- `print("Amount deposited successfully.")` and `print("Current balance is:", balance)`
  Confirms that money is deposited and shows the updated balance.

4. Withdrawal module

- `def withdraw():`
  Defines a function to handle withdrawal operations.

- `global balance`
  Again uses the global `balance` variable to update the main account balance.

- `amount = int(input("Enter amount to withdraw: "))`
  Takes withdrawal amount from the user and converts it to an integer for calculation.

- `if amount <= balance:`
  Checks if the account has sufficient balance for the requested withdrawal.
- `balance = balance - amount`
  Deducts the withdrawal amount from the current balance if there is enough money.
- `print("Please collect your cash.", amount)` and `print("Current balance is:", balance)`
  Informs the user to take the cash and displays the updated balance.

- `else: print("Insufficient balance.")`
  If requested amount is more than balance, it shows an error message without changing the balance.

5. ATM menu loop (main control module)

- `while True:`
  Creates an infinite loop so that the ATM menu keeps showing again and again until the user chooses to exit.
- The menu `print` statements show the 4 options: check balance, deposit, withdraw, and exit.
- `choice = input("Enter your choice: ")`
  Reads the user's menu selection as text.

6.Decision part(final module)

- `if choice == '1': check_balance()`
  If user selects 1, the program calls `check_balance()` to display the current balance.
- `elif choice == '2': deposit()`
  If user selects 2, the program calls `deposit()` to add money to the account.
- `elif choice == '3': withdraw()`
  If user selects 3, the program calls `withdraw()` to take money out of the account.
- `elif choice == '4':` then print exit message and `break`
  If user selects 4, the program thanks the user and breaks out of the `while True` loop, ending the ATM session.
- `else: print("Invalid choice. Please try again.")`
  If the user enters any other option, an error message is displayed and the menu is shown again.

GROUP MEMBERS CONTRIBUTIONS

Team divided ATM code into 6 modules with each member contributing one, then assigned matching report sections.

Pavitra Gaddam & Konijeti Yamini - Algorithm
Abhishek Suman & Ritikesh Kumar - Flowchart
Lokesh Prabhu & Himaja Sudha - Code explanation

-----Thank you-----