BANK ACCOUNT MANAGEMENT SYSTEM

By:- K. K. Sahani

Python Bank Account Management System

Larana, Inc.

Objective

Create a Python program that simulates a bank account management system. The system should allow users to create and manage bank accounts, perform transactions, and retrieve relevant details. The project should incorporate key Python concepts, such as variables, data structures, functions, file handling, and modules.

Requirements

1. Account Management

Users should be able to create new bank accounts. Each account must have the following attributes:

– Account holder's name
– Account number (auto-generated)
– Account type (e.g., savings or current)
– Initial balance

Allow users to retrieve account details, including:

– Account holder's name
– Account number
– Account type
– Current balance

Larana, Inc.

2. Transactions

- Implement the following transactions:
  – Deposit: Users can deposit money into their accounts.
  – Withdrawal: Users can withdraw money, ensuring that the withdrawal does not exceed the current balance.
  – Transfer: Implement a function to transfer funds between two accounts.
- Ensure that transactions update the account balance accordingly

3. File Handling

- Store account details and transaction history in files
- Implement functionality to read and write account data to a file (using the pickle library).
- Load account data from the file at the start of the program and save updates to the file when the program terminates.
- Ensure the program can append new transactions to the existing file without overwriting previous data.

4. Reports

- Allow users to view transaction history for a specific account.
- Show details like:
  – Date
  – Type of transaction (deposit, withdrawal, transfer)
  – Amount

Larana, Inc.

- Generate summary statistics for each account, such as:
  - Total deposits
  - Total withdrawals
  - Average transaction amounts (using NumPy functions)

5. User Interaction

- Create an interactive menu to perform the following operations:
  - Open a new account
  - View account details
  - Perform transactions (deposit, withdraw, transfer)
  - View transaction history
  - Exit the program
- Use conditional statements and loops to ensure smooth navigation between different menu options.

Larana, Inc.

6. Error Handling

- Implement input validation (e.g., ensure deposit and withdrawal amounts are positive).
- Handle potential errors, such as:
  - Attempting to withdraw more money than available in the account.
  - Transferring funds between non-existent accounts.

7. Bonus (Optional)

- Implement login functionality for multiple users, requiring a username and password to access each account.
- Use advanced Python features, such as lambda functions, for quick calculations or specific operations within the program.

Larana, Inc.

Bank Account Managnment System

[6]:

```python
import random
import pickle
import os
from datetime import datetime

# File paths for storing data

ACCOUNTS_FILE = "accounts.pkl"             # File for user account data
TRANSACTIONS_FILE = "transactions.pkl"     # File for transaction history data

# Initialize or load existing account data
try:
     # Check if accounts file exists, and load it
    if os.path.exists(ACCOUNTS_FILE):
        with open(ACCOUNTS_FILE, "rb") as f:
            users_db = pickle.load(f)  # Load accounts data into `users_db`
    else:
        users_db = {}  # If file doesn't exist, initialize empty dictionary
except (AttributeError, pickle.UnpicklingError):
    # Handle corrupt or incompatible file
    print("Corrupt or incompatible accounts data. Starting fresh.")
    users_db = {}
```

Larana, Inc.

```python
try:
    if os.path.exists(TRANSACTIONS_FILE):
        with open(TRANSACTIONS_FILE, "rb") as f:
            transaction_history = pickle.load(f)  # Load transaction history
        # Ensure transaction_history is a dictionary
        if not isinstance(transaction_history, dict):
            print("Corrupt transactions data. Starting fresh.")
            transaction_history = {}
    else:
        transaction_history = {}  # If file doesn't exist, initialize empty dictionary
except (AttributeError, pickle.UnpicklingError):
    # Handle corrupt or incompatible file
    print("Corrupt or incompatible transactions data. Starting fresh.")
    transaction_history = {}

# Dictionary for storing login credentials (username-password mapping)
credentials = {}

# Welcome message
print("-" * 8, "Welcome to WsCube Bank", "-" * 8)

def save_data():
    """
    Save account data and transaction history to files.
    Ensures persistence of user data and transactions.
    """
    with open(ACCOUNTS_FILE, "wb") as f:
        pickle.dump(users_db, f)  # Save `users_db` to file
    with open(TRANSACTIONS_FILE, "wb") as f:
        pickle.dump(transaction_history, f)  # Save `transaction_history` to file
```

Larana, Inc.

```python
def open_account():
    """
    Create a new account for a user:
    - Collect user details
    - Generate unique account number, username, and password
    - Initialize balance and transaction history
    """
    # Collect user details
    fname = input("Enter first name: ")
    lname = input("Enter last name: ")
    pancard = input("Enter PAN card Number: ")

    # Check if account already exists for this PAN card
    if pancard in users_db:
        print("Your account already exists! Please try again.")
        return

    # Generate unique account number
    account_number = "11112222" + "".join(str(random.randint(0, 9)) for _ in range(4))
    # Ask for account type
    account_type = input("Which type of account would you like to open (current/savings): ").lower()
    # Initial deposit amount
    balance = int(input("How much amount do you wish to deposit? "))

    # Generate username and password
    username = fname + "".join(str(random.randint(0, 9)) for _ in range(2))
    password = "".join(str(random.randint(0, 9)) for _ in range(4))

    # Display the generated details to the user
    print(f"\nYour assigned username is: {username}")
    print(f"Your assigned password is: {password}")
    print(f"Account Number: {account_number}")
    print(f"Account Type: {account_type}")
    print(f"Your initial balance is Rs: {balance}\n")
```

```
    # Store credentials and account details
    credentials[username] = password
    users_db[pancard] = [fname, lname, account_number, account_type, balance]

    # Initialize empty transaction history for the new account
    transaction_history[account_number] = []
    save_data()  # Save data to file

def view_account_details():
    """
    View account details for a specific user.
    Requires username, password, and PAN card for authentication.
    """
    # Get authentication details
    username = input("Enter username: ")
    password = input("Enter password: ")
    pancard = input("Enter PAN card number: ")

    # Authenticate user
    if username in credentials and credentials[username] == password and pancard in users_db:
        account = users_db[pancard]  # Fetch account details
        print(f"\nAccount Details:")
        print(f"Account Number: {account[2]}")
        print(f"Account Holder's Name: {account[0]} {account[1]}")
        print(f"Account Type: {account[3]}")
        print(f"Total Balance: Rs {account[4]}\n")
    else:
        print("Invalid credentials or account not found. Please try again.")

def perform_transaction():
    """
    Perform banking transactions:
    - Deposit money
    - Withdraw money
    - Transfer money to another account
    Updates balance and records transaction history.
    """
```

```
"""
# Get authentication details
username = input("Enter username: ")
password = input("Enter password: ")
pancard = input("Enter PAN card number: ")

# Authenticate user
if username in credentials and credentials[username] == password and pancard in users_db:
    account = users_db[pancard]  # Fetch account details
    account_number = account[2]  # Get user's account number
    # Display transaction options
    print("\n1. Deposit\n2. Withdraw\n3. Transfer")
    choice = int(input("Enter your choice: "))

    if choice == 1:  # Deposit money
        amount = int(input("Enter amount to deposit: "))
        if amount > 0:
            account[4] += amount  # Update balance
            transaction_history[account_number].append((datetime.now(), "Deposit", amount))
            print(f"Rs {amount} deposited successfully!")
        else:
            print("Amount must be positive.")
    elif choice == 2:  # Withdraw money
        amount = int(input("Enter amount to withdraw: "))
        if 0 < amount <= account[4]:
            account[4] -= amount  # Deduct balance
            transaction_history[account_number].append((datetime.now(), "Withdrawal", amount))
            print(f"Rs {amount} withdrawn successfully!")
        else:
            print("Insufficient balance or invalid amount.")
    elif choice == 3:  # Transfer money
        target_account_number = input("Enter target account number: ")
        if target_account_number in transaction_history:
            amount = int(input("Enter amount to transfer: "))
            if 0 < amount <= account[4]:
                # Update recipient account balance
                for acc_details in users_db.values():
                    if acc_details[2] == target_account_number:
                        acc_details[4] += amount
```

```
                        break
                    account[4] -= amount  # Deduct sender's balance
                    # Record transactions for both accounts
                    transaction_history[account_number].append((datetime.now(), "Transfer", amount))
                    transaction_history[target_account_number].append((datetime.now(), "Transfer Received", amount))
                    print(f"Rs {amount} transferred successfully!")
                else:
                    print("Insufficient balance or invalid amount.")
            else:
                print("Target account not found.")
        else:
            print("Invalid choice.")
        save_data()  # Save updated data
    else:
        print("Invalid credentials or account not found. Please try again.")

def view_transaction_history():
    """
    View transaction history for a specific account.
    Displays all recorded transactions for the account number provided.
    """
    account_number = input("Enter account number: ")
    if account_number in transaction_history:
        print("\nTransaction History:")
        for entry in transaction_history[account_number]:
            print(f"{entry[0]} - {entry[1]} - Rs {entry[2]}")
    else:
        print("No transaction history found for this account.")

# Main menu loop
while True:
    print("""\nHow can I help you today?
1. Open an Account
2. View Account Details
3. Perform Transaction (Deposit/Withdraw/Transfer)
4. View Transaction History
5. Exit""")
    choice = int(input("Enter your choice (1-5): "))
    if choice == 1:
```

Larana, Inc.

```
# Main menu loop
while True:
    print("""\nHow can I help you today?
1. Open an Account
2. View Account Details
3. Perform Transaction (Deposit/Withdraw/Transfer)
4. View Transaction History
5. Exit""")
    choice = int(input("Enter your choice (1-5): "))
    if choice == 1:
        open_account()
    elif choice == 2:
        view_account_details()
    elif choice == 3:
        perform_transaction()
    elif choice == 4:
        view_transaction_history()
    elif choice == 5:
        print("Thank you for using WsCube Bank. Goodbye!")
        break
    else:
        print("Invalid choice. Please try again.")
```

Larana, Inc.

```
-------- Welcome to WsCube Bank --------

How can I help you today?
1. Open an Account
2. View Account Details
3. Perform Transaction (Deposit/Withdraw/Transfer)
4. View Transaction History
5. Exit
Enter your choice (1-5): 4
Enter account number: 1111
No transaction history found for this account.

How can I help you today?
1. Open an Account
2. View Account Details
3. Perform Transaction (Deposit/Withdraw/Transfer)
4. View Transaction History
5. Exit
Enter your choice (1-5): 2
Enter username: kk87
Enter password: 8109
Enter PAN card number: kks12345
Invalid credentials or account not found. Please try again.

How can I help you today?
1. Open an Account
2. View Account Details
3. Perform Transaction (Deposit/Withdraw/Transfer)
4. View Transaction History
5. Exit
Enter your choice (1-5): 5
Thank you for using WsCube Bank. Goodbye!
```

Thank You