Bank Management System

By Sanyam Sharma

Introduction

The **Bank Management System** is a Python-based project designed to simulate basic banking operations in a simple and user-friendly manner. This system allows users to perform essential functions such as creating an account, depositing money, withdrawing funds, checking balance, and managing account details. By using core Python concepts like conditional statements, loops, and data handling, the project demonstrates how real-world banking activities can be modeled through programming. The system ensures accuracy, security of transactions, and efficient data processing, making it a practical example of applying Python to automate everyday financial tasks.

Code:

```python
# Bank Account Management System

print("Welcome to Simple Bank Account System (Extended Beginner Version)")

# Data storage:
accounts = []   # list of account dictionaries: each account is {'acc_no':int, 'name':str, 'type':str, 'balance':int, 'txs':list}
next_acc_no = 1001   # starting account number

# Main loop
running = True
while running:
    print("\n----- MAIN MENU -----")
    print("1. Create New Account")
    print("2. Select Existing Account (to deposit/withdraw/check/tx history)")
    print("3. List All Accounts (numbers & names)")
    print("4. Exit")
    choice = input("Enter your choice (1-4): ").strip()

    # 1) Create new account
    if choice == "1":
        name = input("Enter account holder name: ").strip()
        if name == "":
            print("Name cannot be empty. Cancelled.")
        else:
            acc_type = ""
            while acc_type not in ("savings", "current"):
                acc_type = input("Enter account type (savings/current): ").strip().lower()
                if acc_type not in ("savings", "current"):
                    print("Please type exactly 'savings' or 'current'.")
```

```
        # create account record
        acc = {
            'acc_no': next_acc_no,
            'name': name,
            'type': acc_type,
            'balance': 0,    # integer balance
            'txs': []        # transaction history as list of strings
        }
        accounts.append(acc)
        print("Account created successfully!")
        print("Account holder:", name)
        print("Account number:", next_acc_no)
        next_acc_no = next_acc_no + 1

# 2) Select existing account to operate
elif choice == "2":
    if len(accounts) == 0:
        print("No accounts exist yet. Create an account first.")
    else:
        # show account numbers to help user
        print("Existing accounts:")
        for a in accounts:
            print(f"  {a['acc_no']}  -  {a['name']} ({a['type']})  balance: {a['balance']}")
        acc_input = input("Enter account number to select: ").strip()
        if not acc_input.isdigit():
            print("Invalid account number. Use digits only.")
```

```
        print("Invalid account number. Use digits only.")
    else:
        acc_no = int(acc_input)
        found = False
        # find the account
        for a in accounts:
            if a['acc_no'] == acc_no:
                found = True
                current = a
                break
        if not found:
            print("Account not found.")
        else:
            # account selected, open account menu
            in_account = True
            while in_account:
                print("\n--- Account Menu ---")
                print("Account:", current['acc_no'], "-", current['name'])
                print("1. Deposit")
                print("2. Withdraw")
                print("3. Check Balance")
                print("4. View Transaction History")
                print("5. Back to Main Menu")
                sub = input("Choose (1-5): ").strip()

                # Deposit
                if sub == "1":
                    amt_str = input("Enter amount to deposit (whole number): ").strip()
                    if not amt_str.isdigit():
                        print("Enter a valid positive number (no decimals).")
```

```
            else:
                amt = int(amt_str)
                if amt <= 0:
                    print("Amount must be positive.")
                else:
                    current['balance'] = current['balance'] + amt
                    # record transaction as readable string
                    now = __import__('datetime').datetime.now().strftime("%Y-%m-%d %H:%M:%S")
                    tx_str = f"{now} | deposit | +{amt} | balance {current['balance']}"
                    current['txs'].append(tx_str)
                    print(f"Deposited {amt}. New balance: {current['balance']}")

        # Withdraw
        elif sub == "2":
            amt_str = input("Enter amount to withdraw (whole number): ").strip()
            if not amt_str.isdigit():
                print("Enter a valid positive number (no decimals).")
            else:
                amt = int(amt_str)
                if amt <= 0:
                    print("Amount must be positive.")
                elif amt > current['balance']:
                    print("Insufficient funds! Current balance:", current['balance'])
```

```
            else:
                current['balance'] = current['balance'] - amt
                now = __import__('datetime').datetime.now().strftime("%Y-%m-%d %H:%M:%S")
                tx_str = f"{now} | withdrawal | -{amt} | balance {current['balance']}"
                current['txs'].append(tx_str)
                print(f"Withdrew {amt}. New balance: {current['balance']}")

    # Check balance
    elif sub == "3":
        print("Account:", current['acc_no'], "-", current['name'])
        print("Type:", current['type'])
        print("Balance:", current['balance'])

    # View transaction history
    elif sub == "4":
        if len(current['txs']) == 0:
            print("No transactions yet for this account.")
        else:
            print(f"--- Transactions for account {current['acc_no']} ---")
            for t in current['txs']:
                print(t)

    # Back
    elif sub == "5":
        in_account = False
```

```
                    else:
                        print("Invalid option in account menu. Choose 1-5.")

    # 3) List all accounts
    elif choice == "3":
        if len(accounts) == 0:
            print("No accounts created yet.")
        else:
            print("--- All accounts ---")
            for a in accounts:
                print(f"Account {a['acc_no']} | {a['name']} | {a['type']} | Balance: {a['balance']}")

    # 4) Exit
    elif choice == "4":
        print("Thank you visit again. Goodbye!")
        running = False

    # invalid main menu choice
    else:
        print("Invalid choice. Please enter a number between 1 and 4.")
```

Output:

```
Welcome to Simple Bank Account System (Extended Beginner Version)

----- MAIN MENU -----
1. Create New Account
2. Select Existing Account (to deposit/withdraw/check/tx history)
3. List All Accounts (numbers & names)
4. Exit
Enter your choice (1-4):  1
Enter account holder name:  sanyam
Enter account type (savings/current):  current
Account created successfully!
Account holder: sanyam
Account number: 1001

----- MAIN MENU -----
1. Create New Account
2. Select Existing Account (to deposit/withdraw/check/tx history)
3. List All Accounts (numbers & names)
4. Exit
Enter your choice (1-4):  2
Existing accounts:
  1001  -  sanyam (current)  balance: 0
Enter account number to select:  1001
```

```
--- Account Menu ---
Account: 1001 - sanyam
1. Deposit
2. Withdraw
3. Check Balance
4. View Transaction History
5. Back to Main Menu
Choose (1-5):  1
Enter amount to deposit (whole number):  1000000
Deposited 1000000. New balance: 1000000

--- Account Menu ---
Account: 1001 - sanyam
1. Deposit
2. Withdraw
3. Check Balance
4. View Transaction History
5. Back to Main Menu
Choose (1-5):  2
Enter amount to withdraw (whole number):  29445
Withdrew 29445. New balance: 970555

--- Account Menu ---
Account: 1001 - sanyam
1. Deposit
2. Withdraw
3. Check Balance
4. View Transaction History
5. Back to Main Menu
Choose (1-5):  3
Account: 1001 - sanyam
Type: current
Balance: 970555
```

```
--- Account Menu ---
Account: 1001 - sanyam
1. Deposit
2. Withdraw
3. Check Balance
4. View Transaction History
5. Back to Main Menu
Choose (1-5):  4
--- Transactions for account 1001 ---
2025-12-11 10:27:36 | deposit | +1000000 | balance 1000000
2025-12-11 10:27:44 | withdrawal | -29445 | balance 970555

--- Account Menu ---
Account: 1001 - sanyam
1. Deposit
2. Withdraw
3. Check Balance
4. View Transaction History
5. Back to Main Menu
Choose (1-5):  5

----- MAIN MENU -----
1. Create New Account
2. Select Existing Account (to deposit/withdraw/check/tx history)
3. List All Accounts (numbers & names)
4. Exit
Enter your choice (1-4):  3
--- All accounts ---
Account 1001 | sanyam | current | Balance: 970555

----- MAIN MENU -----
1. Create New Account
2. Select Existing Account (to deposit/withdraw/check/tx history)
3. List All Accounts (numbers & names)
4. Exit
Enter your choice (1-4):  4
Thank you visit again. Goodbye!
```

Conclusion:

The **Bank Management System** successfully demonstrates how Python can be used to build a reliable and efficient application for handling basic banking operations. Through features such as account creation, deposits, withdrawals, and balance inquiries, the project showcases the practical use of loops, conditional logic, and data storage techniques. It provides a simplified model of real-world banking processes while ensuring accuracy and ease of use. Overall, this project strengthens foundational Python skills and highlights how programming can be applied to automate and streamline financial tasks.

Thank You