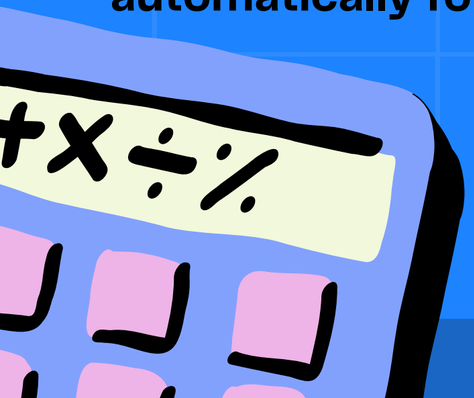
Bank Account Management System (USING PYTHON).

Introduction

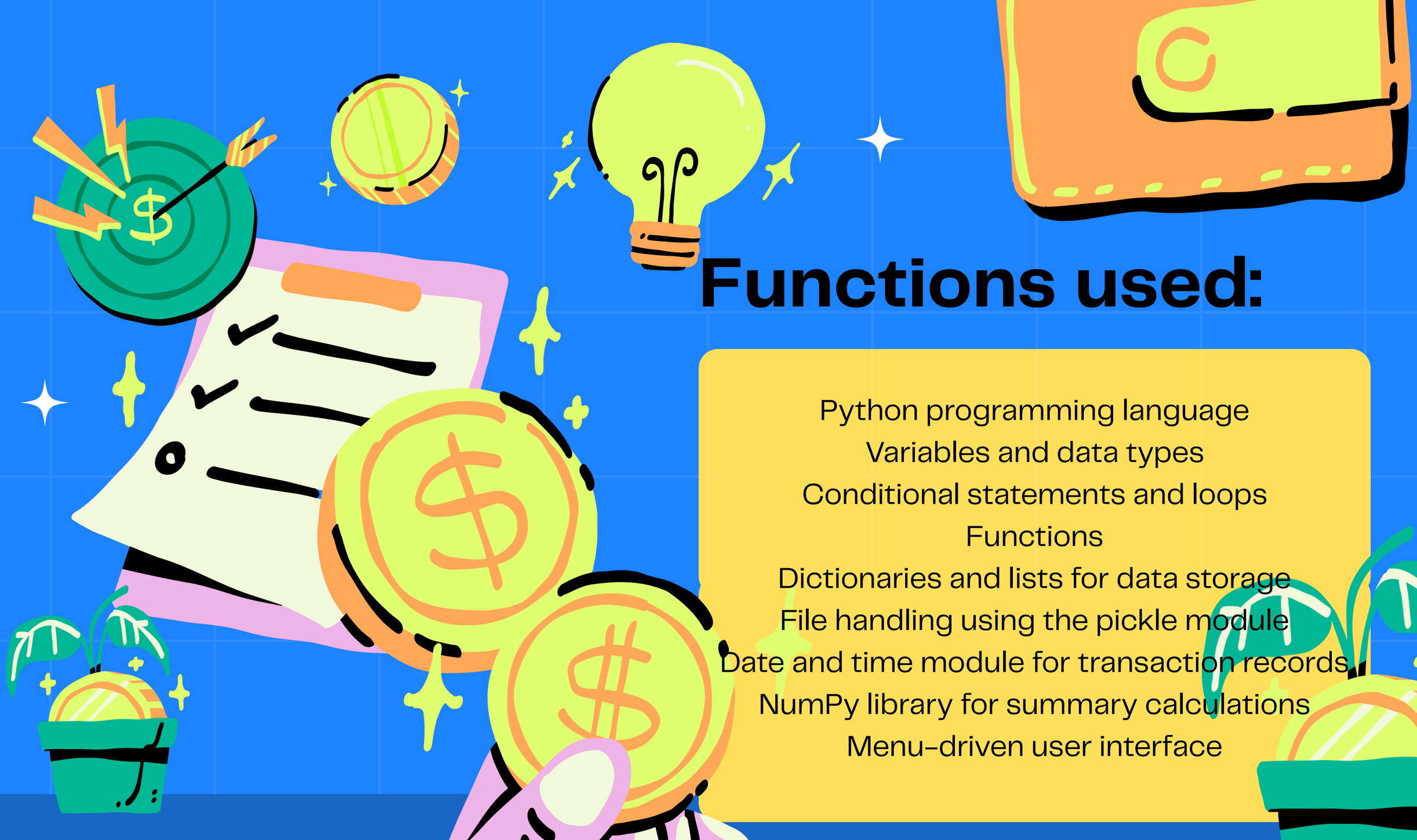
In this program, the user is required to create a bank account by entering basic details such as name, account type, and initial balance. After creating an account, the user can perform operations like deposit, withdrawal, and money transfer using a menu-driven system. All account details and transactions are saved automatically for future use.



Bank Management

Objectives

The objective of this program is to develop a menu-driven Bank Account Management System using Python. The system aims to help users create and manage bank accounts, perform basic banking transactions, and store account and transaction data using file handling for future access.



Functions used:

Python programming language
Variables and data types
Conditional statements and loops
Functions
Dictionaries and lists for data storage
File handling using the pickle module
Date and time module for transaction records
NumPy library for summary calculations
Menu-driven user interface

In [1]:

```
import pickle
import os
from datetime import datetime
import numpy as np

DATA_FILE = "accounts_data.pkl"


# -------------------- Helper Functions --------------------

def load_data():
    if os.path.exists(DATA_FILE):
        with open(DATA_FILE, "rb") as f:
            return pickle.load(f)
    return {"accounts": {}, "transactions": []}


def save_data(data):
    with open(DATA_FILE, "wb") as f:
        pickle.dump(data, f)


def generate_account_number(accounts):
    return 100000 + len(accounts) + 1


# -------------------- Account Functions --------------------

def create_account(data):
    name = input("Enter account holder name: ").strip()
    acc_type = input("Enter account type (savings/current): ").strip().lower()

    try:
        balance = float(input("Enter initial balance: "))
        if balance < 0:
            print("Initial balance cannot be negative.")
            return
    except ValueError:
        print("Invalid amount.")
        return

    acc_no = generate_account_number(data["accounts"])

    data["accounts"][acc_no] = {
        "name": name,
        "type": acc_type,
        "balance": balance
    }

    data["transactions"].append({
        "account": acc_no,
        "type": "deposit",
        "amount": balance,
        "date": datetime.now()
    })

    print(f"\nAccount created successfully. Account Number: {acc_no}")


def view_account(data):
    try:
        acc_no = int(input("Enter account number: "))
        acc = data["accounts"].get(acc_no)

        if not acc:
            print("Account not found.")
            return

        print("\n--- Account Details ---")
        print("Name:", acc["name"])
        print("Account Number:", acc_no)
        print("Account Type:", acc["type"])
        print("Current Balance:", acc["balance"])

    except ValueError:
        print("Invalid account number.")


# -------------------- Transaction Functions --------------------

def deposit(data):
    try:
```

```
        acc_no = int(input("Enter account number: "))
        amount = float(input("Enter deposit amount: "))

        if amount <= 0:
            print("Amount must be positive.")
            return

        if acc_no not in data["accounts"]:
            print("Account does not exist.")
            return

        data["accounts"][acc_no]["balance"] += amount

        data["transactions"].append({
            "account": acc_no,
            "type": "deposit",
            "amount": amount,
            "date": datetime.now()
        })

        print("Deposit successful.")

    except ValueError:
        print("Invalid input.")


def withdraw(data):
    try:
        acc_no = int(input("Enter account number: "))
        amount = float(input("Enter withdrawal amount: "))

        if amount <= 0:
            print("Amount must be positive.")
            return

        if acc_no not in data["accounts"]:
            print("Account does not exist.")
            return

        if amount > data["accounts"][acc_no]["balance"]:
            print("Insufficient balance.")
            return

        data["accounts"][acc_no]["balance"] -= amount

        data["transactions"].append({
            "account": acc_no,
            "type": "withdrawal",
            "amount": amount,
            "date": datetime.now()
        })

        print("Withdrawal successful.")

    except ValueError:
        print("Invalid input.")


def transfer(data):
    try:
        from_acc = int(input("From account number: "))
        to_acc = int(input("To account number: "))
        amount = float(input("Enter transfer amount: "))

        if amount <= 0:
            print("Amount must be positive.")
            return

        if from_acc not in data["accounts"] or to_acc not in data["accounts"]:
            print("One or both accounts do not exist.")
            return

        if amount > data["accounts"][from_acc]["balance"]:
            print("Insufficient balance.")
            return

        data["accounts"][from_acc]["balance"] -= amount
        data["accounts"][to_acc]["balance"] += amount

        data["transactions"].append({
            "account": from_acc,
            "type": "transfer_out",
            "amount": amount,
```

```
            "date": datetime.now()
        })

        data["transactions"].append({
            "account": to_acc,
            "type": "transfer_in",
            "amount": amount,
            "date": datetime.now()
        })

        print("Transfer successful.")

    except ValueError:
        print("Invalid input.")


# -------------------- Reports --------------------

def transaction_history(data):
    try:
        acc_no = int(input("Enter account number: "))
        history = [t for t in data["transactions"] if t["account"] == acc_no]

        if not history:
            print("No transactions found.")
            return

        print("\n--- Transaction History ---")
        for t in history:
            print(f"{t['date']} | {t['type']} | {t['amount']}")

        amounts = np.array([t["amount"] for t in history])

        print("\n--- Summary ---")
        print("Total Transactions:", len(amounts))
        print("Total Amount:", np.sum(amounts))
        print("Average Transaction Amount:", np.mean(amounts))

    except ValueError:
        print("Invalid account number.")


# -------------------- Main Menu --------------------

def main():
    data = load_data()

    while True:
        print("\n--- Bank Account Management System ---")
        print("1. Open New Account")
        print("2. View Account Details")
        print("3. Deposit")
        print("4. Withdraw")
        print("5. Transfer")
        print("6. View Transaction History")
        print("7. Exit")

        choice = input("Choose an option: ")

        if choice == "1":
            create_account(data)
        elif choice == "2":
            view_account(data)
        elif choice == "3":
            deposit(data)
        elif choice == "4":
            withdraw(data)
        elif choice == "5":
            transfer(data)
        elif choice == "6":
            transaction_history(data)
        elif choice == "7":
            save_data(data)
            print("Data saved. Exiting program.")
            break
        else:
            print("Invalid choice. Try again.")


if __name__ == "__main__":
    main()
```

```
--- Bank Account Management System ---
1. Open New Account
2. View Account Details
3. Deposit
4. Withdraw
5. Transfer
6. View Transaction History
7. Exit
Account created successfully. Account Number: 100001

--- Bank Account Management System ---
1. Open New Account
2. View Account Details
3. Deposit
4. Withdraw
5. Transfer
6. View Transaction History
7. Exit
--- Account Details ---
Name: shiva kashyap
Account Number: 100001
Account Type: savings
Current Balance: 10000.0

--- Bank Account Management System ---
1. Open New Account
2. View Account Details
3. Deposit
4. Withdraw
5. Transfer
6. View Transaction History
7. Exit
Deposit successful.

--- Bank Account Management System ---
1. Open New Account
2. View Account Details
3. Deposit
4. Withdraw
5. Transfer
6. View Transaction History
7. Exit
Withdrawal successful.

--- Bank Account Management System ---
1. Open New Account
2. View Account Details
3. Deposit
4. Withdraw
5. Transfer
6. View Transaction History
7. Exit
One or both accounts do not exist.

--- Bank Account Management System ---
1. Open New Account
2. View Account Details
3. Deposit
4. Withdraw
5. Transfer
6. View Transaction History
7. Exit
--- Transaction History ---
2026-01-31 13:38:22.258018 | deposit | 10000.0
2026-01-31 13:39:12.065704 | deposit | 5000.0
2026-01-31 13:39:36.465756 | withdrawal | 10000.0

--- Summary ---
Total Transactions: 3
Total Amount: 25000.0
Average Transaction Amount: 8333.333333333334

--- Bank Account Management System ---
1. Open New Account
2. View Account Details
3. Deposit
4. Withdraw
5. Transfer
6. View Transaction History
7. Exit
Data saved. Exiting program.
```

In [ ]:


“Start now, and make
Bank management a part
of our daily lives”

Thank You