import time

print("Hello! I am your virtual assistant.")

print("You can ask me to do simple tasks like greeting, telling the time, or saying goodbye.")

while True:

user\_input = input("You: ").strip().lower()

if user\_input == "hello":

print("Assistant: Hello! How can I assist you today?")

elif user\_input == "time":

current\_time = time.strftime("%H:%M:%S")

print(f"Assistant: The current time is {current\_time}.")

elif user\_input == "bye":

print("Assistant: Goodbye! Have a great day!")

break

else:

print("Assistant: Sorry, I don't understand that command.")

2.

# Hospital Billing System

# Input from user

Print(“Welcome to the Hospital Billing System”)

Patient\_name = input(“Enter patient’s name: “)

Patient\_age = int(input(“Enter patient’s age: “))

Patient\_id = int(input(“Enter patient’s ID: “))

Room\_charge\_per\_day = float(input(“Enter room charge per day: “))

Days\_stayed = int(input(“Enter number of days stayed: “))

Doctor\_fee = float(input(“Enter doctor’s fee: “))

Medication\_cost = float(input(“Enter medication cost: “))

Other\_charges = float(input(“Enter other charges: “))

# Calculating total bill

Room\_charge\_total = room\_charge\_per\_day \* days\_stayed

Total\_bill = room\_charge\_total + doctor\_fee + medication\_cost + other\_charges

# Displaying the hospital bill

Print(“\n--- Hospital Bill ---“)

Print(f”Patient Name: {patient\_name}”)

Print(f”Patient Age: {patient\_age}”)

Print(f”Patient ID: {patient\_id}”)

Print(f”Room Charge: ₹{room\_charge\_per\_day} x {days\_stayed} days = ₹{room\_charge\_total}”)

Print(f”Doctor’s Fee: ₹{doctor\_fee}”)

Print(f”Medication Cost: ₹{medication\_cost}”)

Print(f”Other Charges: ₹{other\_charges}”)

Print(f”Total Bill: ₹{total\_bill}\n”)

Print(“Thank you for using the Hospital Billing System.”)

3.

# Stationary Shop Billing System

Print(“Welcome to the Stationary Shop Billing System!”)

Print(“Please enter item details. Type ‘done’ when you’re finished.\n”)

Total\_amount = 0 # Keeps track of the total bill

Item\_count = 1 # Counter for item numbering

# Loop to take multiple items

While True:

Item\_name = input(f”Enter the name of item {item\_count} (or ‘done’ to finish): “).strip()

If item\_name.lower() == “done”:

Break # Exit the loop if user finishes input

# Get price and quantity

Try:

Price = float(input(f”Enter the price of {item\_name}: “))

Quantity = int(input(f”Enter the quantity of {item\_name}: “))

If price < 0 or quantity < 0:

Print(“Price and quantity must be non-negative. Please try again.\n”)

Continue # Ask for input again if invalid

# Calculate cost for this item

Item\_total = price \* quantity

Total\_amount += item\_total

# Display item-wise cost

Print(f”{item\_name}: ₹{price} x {quantity} = ₹{item\_total}\n”)

Item\_count += 1 # Increment item count

Except ValueError:

Print(“Invalid input. Please enter a valid number for price and quantity.\n”)

# Print the final bill

Print(“\n--- Stationary Shop Bill ---“)

Print(f”Total Amount: ₹{total\_amount:.2f}”)

Print(“Thank you for shopping with us!”)

4.

# Welcome message

Print(“Welcome to the Theme Park Pricing System!\n”)

Total\_cost = 0 # Variable to store the total cost

Ticket\_count = 1 # Counter for ticket numbering

# Loop to collect ticket details

While True:

Ticket\_type = input(f”Enter the type of ticket {ticket\_count} (e.g., Adult, Child, Senior) or type ‘done’ to finish: “).strip()

If ticket\_type.lower() == “done”:

Break # Exit loop when user is done

Try:

# Get ticket price and quantity

Ticket\_price = float(input(f”Enter the price of {ticket\_type} ticket: “))

Ticket\_quantity = int(input(f”Enter the quantity of {ticket\_type} tickets: “))

If ticket\_price < 0 or ticket\_quantity < 0:

Print(“Price and quantity must be positive values. Please try again.\n”)

Continue # Ask for input again if values are invalid

# Calculate total for this ticket type

Ticket\_total = ticket\_price \* ticket\_quantity

Total\_cost += ticket\_total # Add to total cost

# Display cost breakdown

Print(f”{ticket\_type} Ticket: ₹{ticket\_price} x {ticket\_quantity} = ₹{ticket\_total}\n”)

Ticket\_count += 1 # Move to the next ticket type

Except ValueError:

Print(“Invalid input! Please enter a valid number for price and quantity.\n”)

# Print final bill

Print(“\n--- Theme Park Total Cost ---“)

Print(f”Total Cost: ₹{total\_cost:.2f}\n”)

Print(“Thank you for using the Theme Park Pricing System!”)

5.

# Welcome message

Print(“Welcome to the Cash Exchange System!\n”)

Total\_amount\_inr = 0 # Variable to store total exchanged amount in INR

Currency\_count = 1 # Counter for different currencies

# Loop to handle multiple currency exchanges

While True:

Currency\_type = input(f”Enter the type of currency {currency\_count} (e.g., USD, EUR, INR) or type ‘done’ to finish: “).strip()

If currency\_type.lower() == “done”:

Break # Exit loop when user is done

Try:

# Get exchange rate and amount

Exchange\_rate = float(input(f”Enter the exchange rate for {currency\_type} to INR: “))

Amount = float(input(f”Enter the amount of {currency\_type} to exchange: “))

If exchange\_rate < 0 or amount < 0:

Print(“Exchange rate and amount must be positive values. Please try again.\n”)

Continue # Ask for input again if values are invalid

# Convert amount to INR

Exchanged\_amount = amount \* exchange\_rate

Total\_amount\_inr += exchanged\_amount # Add to total exchanged amount

# Display exchange details

Print(f”{amount:.1f} {currency\_type} at exchange rate {exchange\_rate} = ₹{exchanged\_amount:.1f}\n”)

Currency\_count += 1 # Move to the next currency

Except ValueError:

Print(“Invalid input! Please enter a valid number for exchange rate and amount.\n”)

# Print final total amount in INR

Print(“\n--- Total Exchanged Amount ---“)

Print(f”Total Amount in INR: ₹{total\_amount\_inr:.1f}\n”)

Print(“Thank you for using the Cash Exchange System!”)

7.

# Welcome message

Print(“Welcome to the Top Users System!\n”)

Users = [] # List to store user data (name and score)

User\_count = 1 # Counter for users

# Loop to collect user data

While True:

Name = input(f”Enter the name of user {user\_count} (or type ‘done’ to finish): “).strip()

If name.lower() == “done”:

Break # Exit loop when user is done

Try:

Score = int(input(f”Enter the score of {name}: “))

If score < 0:

Print(“Score must be a positive integer. Please try again.\n”)

Continue # Ask for input again if score is invalid

Users.append((name, score)) # Store user data

User\_count += 1 # Increment user count

Except ValueError:

Print(“Invalid input! Please enter a valid integer for score.\n”)

# Sorting users by score in descending order

Users.sort(key=lambda x: x[1], reverse=True)

# Display sorted users

Print(“\n--- Top Users ---“)

For user in users:

Print(f”User: {user[0]}, Score: {user[1]}”)

Print(“\nThank you for using the Top Users System!”)

8.

From datetime import datetime

Print(“Welcome to the Date Format System!\n”)

Dates = [] # List to store formatted dates

Date\_count = 1 # Counter for user input

# Loop to take user inputs

While True:

Date\_input = input(f”Enter date {date\_count} (YYYY-MM-DD) or type ‘done’ to finish: “).strip()

If date\_input.lower() == “done”:

Break # Exit loop when user is done

Try:

# Convert string input to date object

Date\_obj = datetime.strptime(date\_input, “%Y-%m-%d”)

# Convert to required format (DD/MM/YYYY) and store

Formatted\_date = date\_obj.strftime(“%d/%m/%Y”)

Dates.append(formatted\_date)

Date\_count += 1 # Increment counter

Except ValueError:

Print(“Invalid date format! Please enter in YYYY-MM-DD format.\n”)

# Display formatted dates

Print(“\n--- Formatted Dates ---“)

For date in dates:

Print(date)

Print(“\nThank you for using the Date Format System!”)

9.

# Welcome message

Print(“Welcome to the Literary Journeys System!\n”)

Users\_books = {} # Dictionary to store users and their books

While True:

# Get user name

User = input(“Enter the name of user (or type ‘done’ to finish): “).strip()

If user.lower() == “done”:

Break # Exit loop if user is done

Books = [] # List to store books for the user

Book\_count = 1 # Book counter

While True:

# Get book title

Book = input(f”Enter the title of book {book\_count} for {user} (or type ‘done’ to finish): “).strip()

If book.lower() == “done”:

Break # Exit inner loop if no more books

Books.append(book) # Add book to list

Book\_count += 1 # Increment book counter

Users\_books[user] = books # Store books for the user

# Display the collected data

Print(“\n--- Users’ Literary Journeys ---\n”)

For user, books in users\_books.items():

Print(f”User: {user}”)

For book in books:

Print(f” – {book}”)

Print() # Blank line for spacing

Print(“Thank you for using the Literary Journeys System!”)

10.

# Welcome message

Print(“Welcome to the Active Borrowings System\n”)

# This list will store all borrowing records as dictionaries

Borrowings = []

# Continuously prompt for user info until ‘done’ is entered

While True:

User\_name = input(“Enter the name of user (or type ‘done’ to finish): “).strip()

If user\_name.lower() == “done”:

Break # Stop if the user types ‘done’

# Ask for book details

Book\_title = input(f”Enter the title of the book borrowed by {user\_name}: “).strip()

# Ask for borrowing date

Borrow\_date = input(f”Enter the borrowing date (YYYY-MM-DD) for {book\_title}: “).strip()

# Ask for return date or ‘not returned’

Return\_date = input(

F”Enter the return date (YYYY-MM-DD) for {book\_title} (or type ‘not returned’ if not returned yet): “

).strip()

# Store all details in a dictionary

Record = {

“user”: user\_name,

“book”: book\_title,

“borrow\_date”: borrow\_date,

“return\_date”: return\_date

}

# Append this record to our list of borrowings

Borrowings.append(record)

# Display all active borrowings

Print(“\n--- Active Borrowings ---\n”)

For entry in borrowings:

Print(f”User: {entry[‘user’]}”)

Print(f”Book: {entry[‘book’]}”)

Print(f”Borrow Date: {entry[‘borrow\_date’]}”)

Print(f”Return Date: {entry[‘return\_date’]}\n”)

Print(“Thank you for using the Active Borrowings System!”)