1. Telco Customer Subscription Management System

Developing a basic Telecom Customer Subscription Management System as part of its digital transformation initiatives.

The objective of this application is to manage customer subscription details using core Python data structures and modular programming concepts.

Problem Statement:

As part of a telecom company's digital transformation initiative, you are required to develop a **Customer Subscription Management System**.
The application should store multiple customer subscription records, use Python data structures to manage the data, and allow filtering of customers based on city.

Apply user defined functions

Task No	Task Name	Activity Description	Expected Result
Task 1	Store Customer Records	Capture and store multiple customers	Customer list created
Task 2	Maintain Plan Lookup	Store plan details using dictionary	Plan info available
Task 3	Filter Customers by City	Display customers for a given city	Filtered customer list
Task 4	Function call	Modularize logic using functions	Structured program

Task 1: Store Customer Records

Capture details of multiple telecom customers.

Steps to Do

1. Create a function
2. Accept Customer Name
3. Accept City
4. Accept Plan Type (Prepaid / Postpaid)
5. Store each customer record as a **tuple**
6. Store all customer records in a **list**

Expected Result
Multiple customer records are stored successfully.

Task 2: Maintain Plan Lookup

Use a dictionary to store and retrieve plan information.

Steps to Do

1. Create a function
2. Create a dictionary for plan details
3. Store plan descriptions for Prepaid and Postpaid

4. Use dictionary lookup to fetch plan information

Expected Result
Correct plan details are retrieved using dictionary lookup.

Task 3: Filter Customers by City

Display customers belonging to a specific city.

Steps to Do

1. Create a function
2. Accept a city name from the user
3. Search customer records for matching city
4. Display matching customers along with plan details
5. Display a message if no customers are found

Expected Result
Filtered customer details are displayed correctly.

Task 4: Function call

Improve code structure using functions.

Steps to Do

1. Call a function to display customer details
2. Call a function for Plan lookup
3. Call a function to display filtered customers by city

Expected Result
Program is modular and easy to maintain.

Sample Input

Enter number of customers: 3

Customer 1:

Name : **Haruto**

City : Chennai

Plan : Prepaid

Customer 2:

Name : Ren

City : Bangalore

Plan : Postpaid

Customer 3:

Name : Takumi

City : Chennai

Plan : Postpaid

Enter city to filter: Chennai

Sample Output

Enter number of customers: 3

Customer 1:

Name : Haruto

City : Chennai

Plan : Prepaid

Customer 2:

Name : Ren

City : Bangalore

Plan : Postpaid

Customer 3:

Name : Takumi

City : Chennai

Plan : Postpaid

Enter city to filter: Chennai

Customers in Chennai:

Name : Haruto

Plan : Prepaid

Name : Karthik

Plan : Postpaid

---

2. Telecom Service Usage Tracker

Problem Statement

Developing a Telecom Service Usage Tracking System to monitor customer service consumption.

The objective of this application is to manage and analyze customer service usage using Python data structures.

Apply User defined functions

Task No	Task Name	Activity Description	Expected Result
Task 1	Store Usage Records	Capture service usage details	Usage list created
Task 2	Maintain Service Lookup	Store service descriptions	Service info available
Task 3	Filter Usage by Service	Display customers by service type	Filtered usage data
Task 4	Function Call	Modularize program logic	Clean structure

Design and develop a Python program that performs the following:

Task 1: Store Usage Records

Steps to Do

1. Create a function
2. Accept Customer Name
3. Accept City
4. Accept Service Type (Data / Voice / SMS)
5. Store each record as a tuple
6. Store all records in a list

Task 2: Maintain Service Lookup

Steps to Do

1. Create a function

2. Create a dictionary for service descriptions
3. Use dictionary lookup to retrieve service meaning

Task 3: Filter Usage by Service Type

Steps to Do

1. Create a function
2. Accept service type from the user
3. Display customers using the selected service
4. Display message if no records are found

Task 4: Use Functions

Steps to Do

1. Call a function to display service usage records
2. Call a function to display filtered usage details

Sample Input

Enter number of usage records: 3

Record 1:

Customer Name : Ravi

City : Chennai

Service Type : Data

Record 2:

Customer Name : Anjali

City : Bangalore

Service Type : Voice

Record 3:

Customer Name : Karthik

City : Chennai

Service Type : Data

Enter service type to filter: Data

Sample Output

Enter number of usage records: 3

Record 1:

Customer Name : Ravi

City : Chennai

Service Type : Data

Record 2:

Customer Name : Anjali

City : Bangalore

Service Type : Voice

Record 3:

Customer Name : Karthik

City : Chennai

Service Type : Data

Enter service type to filter: Data

Customers using Data Service:

Name : Ravi

City : Chennai

Name : Karthik

City : Chennai

---

3. Bank Account Transaction Analyzer

Problem Statement

Working with a banking client to develop a Bank Account Transaction Analyzer.

The objective of this application is to analyse customer transactions using Python collections and modular programming.

Apply User defined functions

Task No	Task Name	Activity Description	Expected Result
Task 1	Store Transaction Records	Capture multiple transactions	Transaction list created
Task 2	Maintain Transaction Lookup	Store transaction descriptions	Lookup enabled
Task 3	Filter Transactions	Display by transaction type	Filtered output
Task 4	Use Functions	Modularize transaction logic	Structured solution

Design and develop a Python program that performs the following:

Task 1: Store Transaction Records

Steps to Do

1. Create a function
2. Accept Account Holder Name
3. Accept Transaction Type (Deposit / Withdrawal)
4. Accept Amount
5. Store each transaction as a tuple
6. Store all transactions in a list

Task 2: Maintain Transaction Lookup

Steps to Do

1. Create a function
2. Create a dictionary for transaction descriptions
3. Use dictionary lookup to display description

Task 3: Filter Transactions by Type

Steps to Do

1. Create a function
2. Accept transaction type from the user
3. Display matching transactions
4. Display message if no records are found

Task 4: Use Functions

Steps to Do

1. Call a function to display transaction records
2. Call a function to display filtered transactions

Sample Input

Enter number of transactions: 3

Transaction 1:

Account Holder : Ravi

Transaction Type : Deposit

Amount : 10000

Transaction 2:

Account Holder : Ravi

Transaction Type : Withdrawal

Amount : 3000

Transaction 3:

Account Holder : Ravi

Transaction Type : Deposit

Amount : 5000

Enter transaction type to filter: Deposit

Sample Output

Enter number of transactions: 3

Transaction 1:

Account Holder : Ravi

Transaction Type : Deposit

Amount : 10000

Transaction 2:

Account Holder : Ravi

Transaction Type : Withdrawal

Amount : 3000

Transaction 3:

Account Holder : Ravi

Transaction Type : Deposit

Amount : 5000

Enter transaction type to filter: Deposit

Deposit Transactions:

Account Holder : Ravi

Amount : 10000

Account Holder : Ravi

Amount : 5000
