ASSIGNMENT-3.1

Lab Experiment: Prompt Engineering – Improving Prompts  
and Context Management

---------------------------------------

**NAME: PRADEEP GUPTHA**

**HALL-TICKET: 2403A510C7**

**BATCH NO: 05**

Task 1:   
● Mobile Data Usage Billing Application:  
Use Python programming and AI-assisted coding tools to create an  
application that simulates mobile data billing for a telecom service  
provider.

Prompt: Create an application that simulates mobile data billing for a telecom service  
provider. Make the program realistic by taking reference to the actual data provider plans.

CODE**:**

# Telecom Billing Simulator

# Assumptions (easily editable below):

# - Prepaid: tiered data pricing

# - Postpaid: first 20 GB included, then flat per-GB rate

# - Value-added services: fixed monthly fees

# - Tax: 18%

from dataclasses import dataclass

from typing import List, Tuple

TAX\_RATE = 0.18  # 18% GST (edit as needed)

# Pricing configuration (edit to match your scenario)

PRICING = {

    "prepaid": {

        # slabs: (upper\_limit\_inclusive, rate\_per\_gb)

        # applies from 0 upward; each slab covers its range

        "slabs": [(2, 15.0), (10, 10.0), (float("inf"), 8.0)],

        "included\_gb": 0.0,   # no free data

        "plan\_note": "Prepaid tiered data pricing",

    },

    "postpaid": {

        # postpaid: first 20 GB free, then flat rate

        "included\_gb": 20.0,

        "post\_included\_rate": 8.0,  # ₹ per GB beyond included

        "plan\_note": "Postpaid: first 20 GB included, then ₹8/GB",

    },

}

# Value-added services (edit or extend as needed)

SERVICES = {

    "caller tune": 30.0,

    "ott basic": 149.0,

    "international roaming": 499.0,

    "data rollover": 59.0,

}

@dataclass

class BillBreakdown:

    plan\_type: str

    data\_gb: float

    data\_lines: List[Tuple[str, float]]  # (description, amount)

    dc\_total: float

    vc\_lines: List[Tuple[str, float]]    # (service name, price)

    vc\_total: float

    tax: float

    total: float

    notes: List[str]

def money(x: float) -> str:

    return f"₹{x:,.2f}"

def calc\_data\_charges(plan\_type: str, data\_gb: float) -> Tuple[List[Tuple[str, float]], float, List[str]]:

    pt = plan\_type.lower()

    lines = []

    notes = []

    if pt == "prepaid":

        cfg = PRICING["prepaid"]

        remaining = max(0.0, data\_gb - cfg["included\_gb"])

        start = 0.0

        total = 0.0

        for upper, rate in cfg["slabs"]:

            slab\_width = min(remaining, upper - start)

            if slab\_width > 0:

                amount = slab\_width \* rate

                lines.append((f"{slab\_width:.2f} GB @ {money(rate)}/GB", amount))

                total += amount

                remaining -= slab\_width

                start = upper

            if remaining <= 1e-9:

                break

        notes.append(cfg["plan\_note"])

        return lines, total, notes

    elif pt == "postpaid":

        cfg = PRICING["postpaid"]

        free\_gb = min(data\_gb, cfg["included\_gb"])

        billable = max(0.0, data\_gb - cfg["included\_gb"])

        total = billable \* cfg["post\_included\_rate"]

        if free\_gb > 0:

            lines.append((f"{free\_gb:.2f} GB included", 0.0))

        if billable > 0:

            lines.append((f"{billable:.2f} GB @ {money(cfg['post\_included\_rate'])}/GB", total))

        notes.append(cfg["plan\_note"])

        return lines, total, notes

    else:

        raise ValueError("Invalid plan type. Use Prepaid or Postpaid.")

def calc\_service\_charges(selected\_services: List[str]) -> Tuple[List[Tuple[str, float]], float, List[str]]:

    lines = []

    total = 0.0

    notes = []

    for s in selected\_services:

        key = s.strip().lower()

        if not key:

            continue

        if key in SERVICES:

            price = SERVICES[key]

            lines.append((s.strip(), price))

            total += price

        else:

            # Unknown services are ignored but noted

            notes.append(f"Ignored unknown service: '{s.strip()}'")

    return lines, total, notes

def generate\_bill():

    # --- Inputs ---

    try:

        data\_gb = float(input("Enter Data Consumed (in GB): ").strip())

        plan\_type = input("Enter Plan Type (Prepaid/Postpaid): ").strip()

        services\_in = input("Additional Services (comma-separated, e.g., Caller Tune, OTT Basic): ").strip()

    except Exception as e:

        print("Input error:", e)

        return

    if plan\_type.lower() not in ("prepaid", "postpaid"):

        print("Invalid plan type. Please enter 'Prepaid' or 'Postpaid'.")

        return

    selected\_services = [s for s in services\_in.split(",")] if services\_in else []

    # --- Calculations ---

    data\_lines, dc\_total, notes\_dc = calc\_data\_charges(plan\_type, data\_gb)

    vc\_lines, vc\_total, notes\_vc = calc\_service\_charges(selected\_services)

    subtotal = dc\_total + vc\_total

    tax = subtotal \* TAX\_RATE

    total = subtotal + tax

    breakdown = BillBreakdown(

        plan\_type=plan\_type.title(),

        data\_gb=data\_gb,

        data\_lines=data\_lines,

        dc\_total=dc\_total,

        vc\_lines=vc\_lines,

        vc\_total=vc\_total,

        tax=tax,

        total=total,

        notes=[\*notes\_dc, \*notes\_vc, f"Tax rate: {TAX\_RATE\*100:.0f}%"],

    )

    # --- Output: Itemized Bill ---

    print("\n" + "="\*58)

    print("                TELECOM ITEMIZED BILL")

    print("="\*58)

    print(f"Plan Type: {breakdown.plan\_type}")

    print(f"Data Used: {breakdown.data\_gb:.2f} GB")

    print("-"\*58)

    print("Data Charges (DC):")

    if breakdown.data\_lines:

        for desc, amount in breakdown.data\_lines:

            print(f"  - {desc:<35} {money(amount):>12}")

    else:

        print("  - No data charges")

    print(f"{'Subtotal DC':<40}{money(breakdown.dc\_total):>18}")

    print("\nValue-added Services (VC):")

    if breakdown.vc\_lines:

        for svc, price in breakdown.vc\_lines:

            print(f"  - {svc:<35} {money(price):>12}")

    else:

        print("  - None")

    print(f"{'Subtotal VC':<40}{money(breakdown.vc\_total):>18}")

    print("-"\*58)

    print(f"{'Subtotal (DC + VC)':<40}{money(breakdown.dc\_total + breakdown.vc\_total):>18}")

    print(f"{'Tax':<40}{money(breakdown.tax):>18}")

    print("="\*58)

    print(f"{'TOTAL PAYABLE':<40}{money(breakdown.total):>18}")

    print("="\*58)

    if breakdown.notes:

        print("\nNotes:")

        for n in breakdown.notes:

            print(f" - {n}")

if \_\_name\_\_ == "\_\_main\_\_":

    generate\_bill()

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

EXPLANATION:

The program simulates mobile data billing for a telecom provider. It takes three inputs from the user:

* **Data consumed (in GB)**
* **Plan type** (Prepaid or Postpaid)
* **Additional services used** (e.g., caller tune, OTT subscription)

**Billing logic:**

* Calculates **Data Charges (DC)** based on plan:
  + **Prepaid** uses tiered per-GB rates.
  + **Postpaid** includes free data (20 GB), then charges per GB.
* Adds **Value-added Charges (VC)** for extra services (fixed rates).
* Applies **tax** (e.g., 18%) on the subtotal.

**Output:**

* Displays an **itemized bill** showing plan details, data usage and charges, service charges, tax, and total payable amount.
* Notes about included benefits or ignored inputs are also displayed.

--------------------------------------------------------------------------------------------------------------

Task 2: Develop an LPG Billing System (1.0 Marks)  
Objective:  
Apply your Python programming skills and utilize AI-assisted coding tools to build an application that calculates the LPG bill based on specified customer inputs and billing parameters.

Prompt: Help me build an application that calculates the LPG bill based on specified customer inputs and billing parameters.

CODE:

import random

# Price list for LPG cylinders

LPG\_PRICES = {

    "domestic 14.2 kg": 905.00,

    "domestic 5 kg": 335.50,

    "commercial 19 kg": 1886.50,

    "commercial 47.5 kg": 4712.00,

}

def calculate\_lpg\_bill(cylinder\_type, quantity, subsidy, delivery\_charge):

    # Get the price per cylinder

    cylinder\_type\_lower = cylinder\_type.strip().lower()

    if cylinder\_type\_lower not in LPG\_PRICES:

        raise ValueError("Invalid cylinder type entered.")

    price\_per\_cylinder = LPG\_PRICES[cylinder\_type\_lower]

    # Base amount before subsidy and delivery

    base\_amount = price\_per\_cylinder \* quantity

    # Subsidy applies only to domestic cylinders

    subsidy\_amount = subsidy if "domestic" in cylinder\_type\_lower else 0.0

    # Final bill calculation

    total\_amount = base\_amount - subsidy\_amount + delivery\_charge

    # Prepare itemized details

    bill\_details = {

        "Cylinder Type": cylinder\_type.title(),

        "Number of Cylinders": quantity,

        "Price per Cylinder": price\_per\_cylinder,

        "Base Amount": base\_amount,

        "Subsidy": subsidy\_amount,

        "Delivery Charges": delivery\_charge,

        "Total Bill": total\_amount

    }

    return bill\_details

def main():

    # User inputs

    cylinder\_type = input("Enter Cylinder Type (Domestic 14.2 kg / Domestic 5 kg / Commercial 19 kg / Commercial 47.5 kg): ")

    quantity = int(input("Enter Number of Cylinders Booked: "))

    subsidy = float(input("Enter Subsidy Amount (if applicable, else 0): "))

    # Delivery charge (random between 10 and 50 or could be fixed)

    delivery\_charge = random.randint(10, 50)

    # Calculate bill

    try:

        bill = calculate\_lpg\_bill(cylinder\_type, quantity, subsidy, delivery\_charge)

        # Display itemized bill

        print("\n====== LPG BILL ======")

        print(f"Cylinder Type       : {bill['Cylinder Type']}")

        print(f"Number of Cylinders : {bill['Number of Cylinders']}")

        print(f"Price per Cylinder  : ₹{bill['Price per Cylinder']:.2f}")

        print(f"Base Amount         : ₹{bill['Base Amount']:.2f}")

        print(f"Subsidy             : -₹{bill['Subsidy']:.2f}")

        print(f"Delivery Charges    : ₹{bill['Delivery Charges']:.2f}")

        print("-----------------------------")

        print(f"Total Bill Amount   : ₹{bill['Total Bill']:.2f}")

        print("=============================\n")

    except ValueError as e:

        print("Error:", e)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

output:

A computer screen shot of a black screen

AI-generated content may be incorrect.

EXPLANATION:

The program calculates an LPG bill based on customer inputs. It takes:

* **Cylinder type** (Domestic 14.2 kg / Domestic 5 kg / Commercial 19 kg / Commercial 47.5 kg)
* **Number of cylinders booked**
* **Subsidy amount** (applies only to domestic cylinders)

**Billing logic:**

* Uses a fixed price list for each cylinder type.
* Calculates **base amount** = price per cylinder × quantity.
* Subtracts **subsidy** for domestic bookings.
* Adds **delivery charges** (random between ₹10–₹50).

**Output:**

* Displays an **itemized bill** including cylinder type, quantity, price per cylinder, base amount, subsidy, delivery charges, and the **total bill amount**.