# Lab Assignment - 3.5

**Prompt Engineering – Improving Prompts and Context Management**

**Ch.Sushanth(2303A51750)**
**Batch-11**

**Task-1: "Write a Python function to check if a number is prime."**
**Prompts used:**
#Write a python code for checking a prime number.

#Optimize the logic for better performance

**Code:**

lab2.py > ...

```python
# Write a Python code for checking a prime number.
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            return False
    return True

input_number = int(input("Enter a number to check if it is prime: "))
if is_prime(input_number):
    print(f"{input_number} is a prime number.")
else:
    print(f"{input_number} is not a prime number.")

```

**#2(optimised):**

```python
#Optimize the logic for better performance.
def is_prime_optimized(num):
    if num <= 1:
        return False
    if num <= 3:
        return True
    if num % 2 == 0 or num % 3 == 0:
        return False

    i = 5
    while i * i <= num:
        if num % i == 0 or num % (i + 2) == 0:
            return False
        i += 6
    return True

input_number = int(input("Enter a number to check if it is prime (optimized): "))
if is_prime_optimized(input_number):
    print(f"{input_number} is a prime number.")
else:
    print(f"{input_number} is not a prime number.")

```

**Output:**

```
/usr/bin/python3 "/Users/sushanth/Downloads/College/Ai Coding/lab2.py"
sushanth@Sushanth-2 Ai Coding % /usr/bin/python3 "/Users/sushanth/Downloads/College/Ai Coding/lab2.py"
Enter a number to check if it is prime: 12
12 is not a prime number.
sushanth@Sushanth-2 Ai Coding % /usr/bin/python3 "/Users/sushanth/Downloads/College/Ai Coding/lab2.py"
Enter a number to check if it is prime: 12
12 is not a prime number.
```

```
Enter a number to check if it is prime (optimized): 12
12 is not a prime number.
```

**<u>Comparision of both results:</u> This function checks whether a number is divisible by any value from 2 up to the square root of the number. And This optimized version removes unnecessary checks by - Skipping even numbers , Skipping multiples of 3 , This reduces the total number of iterations.**

**Task-2: Mobile Data Usage Billing Application**

**Prompts used:**

# Use Python programming and AI-assisted coding tools to create an application that simulates mobile data billing for a telecom service provider.
# Generate python code such that user should have Data Consumed (in GB), Plan Type (Prepaid / Postpaid),Additional Services Used (e.g., caller tune, OTT subscription, etc.)
#Implement billing logic to calculate: DC (Data Charges) – charges based on data consumption, VC (Value-added Charges) – charges for additional services, Tax – applicable tax on the total bill.
#Display an itemized bill showing: Plan Type, Data Usage and Charges, Value-added Services and Charges, Tax, Total Bill Amount

**Code:**

lab2_1.py > MobileDataPlanItemized

```python
class MobileDataPlanItemized:
    def __init__(self, plan_name, data_limit_gb, cost_per_gb, service_cost, tax_rate):
        self.plan_name = plan_name
        self.data_limit_gb = data_limit_gb
        self.cost_per_gb = cost_per_gb
        self.service_cost = service_cost
        self.tax_rate = tax_rate

    def calculate_bill(self, data_used_gb):
        if data_used_gb <= self.data_limit_gb:
            data_charges = data_used_gb * self.cost_per_gb
        else:
            extra_data = data_used_gb - self.data_limit_gb
            data_charges = (
                self.data_limit_gb * self.cost_per_gb
                + extra_data * self.cost_per_gb * 1.5
            )

        total_before_tax = data_charges + self.service_cost
        tax_amount = total_before_tax * self.tax_rate
        total_bill = total_before_tax + tax_amount

        return data_charges, self.service_cost, tax_amount, total_bill
```

```
    def display_itemized_bill(self, data_used_gb, plan_type, value_added_services):
        data_charges, service_charges, tax_amount, total_bill = self.calculate_bill(data_used_gb)

        print("\n--- Itemized Mobile Data Bill ---")
        print(f"Plan Type: {plan_type}")
        print(f"Plan Name: {self.plan_name}")
        print(f"Data Usage: {data_used_gb} GB")
        print(f"Data Charges: Rs.{data_charges:.2f}")
        print(f"Value-added Services: {', '.join(value_added_services) if value_added_services else 'None'}")
        print(f"Value-added Services Charges: Rs.{service_charges:.2f}")
        print(f"Tax: Rs.{tax_amount:.2f}")
        print(f"Total Bill Amount: Rs.{total_bill:.2f}")


# Define available plans
basic_plan = MobileDataPlanItemized("Basic", 5, 10, 50, 0.18)
premium_plan = MobileDataPlanItemized("Premium", 20, 8, 100, 0.18)

# User input
data_used = float(input("Enter data used in GB: "))
plan_type = input("Select plan type (Prepaid/Postpaid): ")
selected_plan = input("Select plan (Basic/Premium): ")
```

```
# Value-added services
services = []
add_services = input("Did you use any value-added services? (yes/no): ").strip().lower()

if add_services == "yes":
    while True:
        service = input("Enter service name (or press Enter to finish): ").strip()
        if service:
            services.append(service)
        else:
            break

# Bill calculation and display
if selected_plan.lower() == "basic":
    basic_plan.display_itemized_bill(data_used, plan_type, services)
elif selected_plan.lower() == "premium":
    premium_plan.display_itemized_bill(data_used, plan_type, services)
else:
    print("Invalid plan selected.")
```

# Output:

```
sushanth@Sushanth-2 Ai Coding % /usr/bin/python3 "/Users/sushanth/Downloads/College/Ai Coding/lab2_1.py"
Enter data used in GB: 4
Select plan type (Prepaid/Postpaid): prepaid
Select plan (Basic/Premium): basic
Did you use any value-added services? (yes/no): yes
Enter service name (or press Enter to finish):

--- Itemized Mobile Data Bill ---
Plan Type: prepaid
Plan Name: Basic
Data Usage: 4.0 GB
Data Charges: Rs.40.00
Value-added Services: None
Value-added Services Charges: Rs.50.00
Tax: Rs.16.20
Total Bill Amount: Rs.106.20
```

**Comparision of both results:**This Mobile Data Usage Billing Application helps users calculate their monthly mobile data bill in a simple and interactive way. The user selects their plan type (Prepaid or Postpaid) and chooses between a Basic or Premium plan, each with its own data limits and rates. The app asks for the amount of data used and whether any value-added services (like caller tunes or OTT subscriptions) were used. It then calculates the total bill, including extra charges for exceeding the data limit, service costs, and applicable taxes. Finally, it presents a clear, itemized bill that breaks down all charges, making it easy for users to understand exactly what they're paying for. This makes managing and reviewing mobile expenses straightforward and transparent.

## Task-3: Develop an LPG Billing System

## Prompts used:

#Develop a Python application and utilize AI-assisted coding tools to build an application
that calculates the LPG bill based on specified customer inputs and billing parameters.
#Generate python code such that user should have Customer Name, Customer ID,
Consumption (in kg), Connection Type (Domestic / Commercial)
# Refer to the given LPG Price List to determine the price per cylinder:
# Add delivery charge input and detailed billing class with display method
# Calculate per kg price from cylinder price

# Code:

```python
class MobileDataPlanItemized:
    def __init__(self, plan_name, data_limit_gb, cost_per_gb, service_cost, tax_rate):
        self.plan_name = plan_name
        self.data_limit_gb = data_limit_gb
        self.cost_per_gb = cost_per_gb
        self.service_cost = service_cost
        self.tax_rate = tax_rate

    def calculate_bill(self, data_used_gb):
        if data_used_gb <= self.data_limit_gb:
            data_charges = data_used_gb * self.cost_per_gb
        else:
            extra_data = data_used_gb - self.data_limit_gb
            data_charges = (
                self.data_limit_gb * self.cost_per_gb
                + extra_data * self.cost_per_gb * 1.5  # 50% surcharge
            )

        total_before_tax = data_charges + self.service_cost
        tax_amount = total_before_tax * self.tax_rate
        total_bill = total_before_tax + tax_amount

        return data_charges, self.service_cost, tax_amount, total_bill
```

```
    def display_itemized_bill(self, data_used_gb, plan_type, value_added_services):
        data_charges, service_charges, tax_amount, total_bill = self.calculate_bill(data_used_gb)

        print("\n--- Itemized Mobile Data Bill ---")
        print(f"Plan Type: {plan_type}")
        print(f"Plan Name: {self.plan_name}")
        print(f"Data Usage: {data_used_gb} GB")
        print(f"Data Charges: Rs.{data_charges:.2f}")
        print(f"Value-added Services: {', '.join(value_added_services) if value_added_services else 'None'}")
        print(f"Value-added Services Charges: Rs.{service_charges:.2f}")
        print(f"Tax: Rs.{tax_amount:.2f}")
        print(f"Total Bill Amount: Rs.{total_bill:.2f}")


# Define available plans
basic_plan = MobileDataPlanItemized("Basic", 5, 10, 50, 0.18)
premium_plan = MobileDataPlanItemized("Premium", 20, 8, 100, 0.18)

# User input
data_used = float(input("Enter data used in GB: "))
plan_type = input("Select plan type (Prepaid/Postpaid): ")
selected_plan = input("Select plan (Basic/Premium): ")
```

```
# Value-added services
services = []
add_services = input("Did you use any value-added services? (yes/no): ").strip().lower()
if add_services == "yes":
    while True:
        service = input("Enter service name (or press Enter to finish): ").strip()
        if service:
            services.append(service)
        else:
            break

# Bill calculation and display
if selected_plan.lower() == "basic":
    basic_plan.display_itemized_bill(data_used, plan_type, services)
elif selected_plan.lower() == "premium":
    premium_plan.display_itemized_bill(data_used, plan_type, services)
else:
    print("Invalid plan selected.")
```

# Output:

```
sushanth@Sushanth-2 Ai Coding % /usr/bin/python3 "/Users/sushanth/Downloads/College/Ai Coding/lab2_2.py"
Enter data used in GB: 4
Select plan type (Prepaid/Postpaid): postpaid
Select plan (Basic/Premium): basic
Did you use any value-added services? (yes/no): yes
Enter service name (or press Enter to finish):

--- Itemized Mobile Data Bill ---
Plan Type: postpaid
Plan Name: Basic
Data Usage: 4.0 GB
Data Charges: Rs.40.00
Value-added Services: None
Value-added Services Charges: Rs.50.00
Tax: Rs.16.20
Total Bill Amount: Rs.106.20
```

**Comparision of both results:**This LPG Gas Billing Application makes it easy for customers to calculate their monthly gas bill. Users enter their personal details, connection type (Domestic or Commercial), cylinder size, and the amount of gas consumed. The app automatically applies the correct price per cylinder, calculates any government subsidy, adds delivery charges, and computes the applicable tax. It then presents a clear, itemized bill showing all charges, including gross amount, subsidy, net amount, tax, and delivery fees. This helps users understand exactly what they're paying for and ensures transparency in their LPG billing. The process is straightforward, making it simple for anyone to review and manage their household or business gas expenses.