

ASSIGNMENT - 3.1

NAME:M.ABHISHEK

ROOL NO:2303A51762

BATCH:11

Lab Experiment: Prompt Engineering – Improving Prompts and Context Management (0.5 marks)

Objective

To explore how prompt design and context influence AI-generated outputs and to learn techniques to improve AI responses.

Tools Required

- GitHub Copilot / Google Gemini / ChatGPT
- VS Code / Google Colab
- Internet access

Procedure

1. Select a simple task: "*Write a Python function to check if a number is prime.*"
2. Use different prompting strategies to generate the solution:
 - a) Zero-Shot – no examples.
 - b) One-Shot – one example provided.
 - c) Few-Shot – multiple examples provided.
 - d) Context-Managed – detailed prompt with constraints and instructions.
3. Record AI responses and refine prompts to improve code quality.
4. Request AI to optimize the logic for efficiency.
5. Compare results and document improvements.

1. Sample Prompts

- Zero-Shot:

Write a Python function to check if a number is prime.

PROMPT: #Write a Python function to check if a number is prime.

The screenshot shows a code editor window with two tabs: 'Lab1.py' and 'lab3.py'. The 'lab3.py' tab is active and contains the following Python code:

```
lab3.py > ...
1  #Write a Python function to check if a number is prime.
2  def is_prime(n):
3      if n <= 1:
4          return False
5      for i in range(2, int(n**0.5) + 1):
6          if n % i == 0:
7              return False
8      return True
9 #Example usage
10 if __name__ == "__main__":
11     test_number = 29
12     if is_prime(test_number):
13         print(f"{test_number} is a prime number.")
14     else:
15         print(f"{test_number} is not a prime number.")
```

OUTPUT:

```
PS C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding> c:& cd 'c:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding'; & 'C:\Users\CHAITRA REDDY\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\CHAITRA REDDY\.vscode\extensions\ms-python.python\2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '57384' '--' 'c:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding\lab3.py'
29 is a prime number.
PS C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding> []
```

CODE:

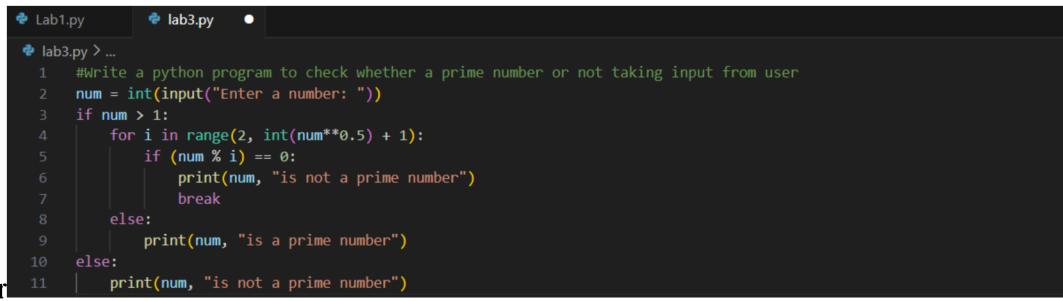
```
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

#Example usage
if __name__ == "__main__":
    test_number = 29
    if is_prime(test_number):
        print(f"{test_number} is a prime number.")
    else:
        print(f"{test_number} is not a prime number.")
```

- One-Shot:

Example: Input: 5 → Output: Prime. Now, write a function to check if a number is prime.

PROMPT: #Write a python program to check whether a prime number or not taking input from



```
# Write a python program to check whether a prime number or not taking input from user
num = int(input("Enter a number: "))
if num > 1:
    for i in range(2, int(num**0.5) + 1):
        if (num % i) == 0:
            print(num, "is not a prime number")
            break
    else:
        print(num, "is a prime number")
else:
    print(num, "is not a prime number")
```

OUTPUT:

```
PS C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding> cd "C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding"; & "C:\Users\CHAITRA REDDY\AppData\Local\Microsoft\WindowsApps\python3.11.exe" "C:\Users\CHAITRA REDDY\.vscode\extensions\ms-python.debugpy\2025.18.0\win32_x64\bundled\libs\debugpy\launcher" "63981" ... 'C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding\lab3.py'
Enter a number: 5
5 is a prime number
```

CODE:

```
if num > 1:
    for i in range(2, int(num**0.5) + 1):
        if (num % i) == 0:
            print(num, "is not a prime number")
            break
    else:
        print(num, "is a prime number")
else:
    print(num, "is not a prime number")
```

- Few-Shot:

Example 1: Input: 7 → Output: Prime

Example 2: Input: 10 → Output: Not Prime

Example 3: Input: 2 → Output: Prime

Generate the function accordingly.

PROMPT: #Write a python program to check whether a prime number or not taking input from user 3 numbers.

```
Lab1.py | lab3.py > ... || ⏪ ⏴ ⏵ ⏹ ⏺ ⏻
```

```
1 #Write a python program to check whether a prime number or not taking input from user 3 numbers.
2 def is_prime(num):
3     if num <= 1:
4         return False
5     for i in range(2, int(num**0.5) + 1):
6         if num % i == 0:
7             return False
8     return True
9 #Example usage
10 if __name__ == "__main__":
11     numbers = []
12     for _ in range(3):
13         number = int(input("Enter a number: "))
14         numbers.append(number)
15     for number in numbers:
16         if is_prime(number):
17             print(f"{number} is a prime number.")
18         else:
19             print(f"{number} is not a prime number.")
```

OUTPUT:

```
AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\CHAITRA REDDY\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55432' '--' 'C:\Users\CHAITRA REDDY\Desktop\AI Assistant coding\lab3.py'
Enter a number: 7
Enter a number: 10
Enter a number: 2
7 is a prime number.
10 is not a prime number.
2 is a prime number.
PS C:\Users\CHAITRA REDDY\Desktop\AI Assistant coding>
```

CODE:

```
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

#Example usage

if __name__ == "__main__":

    numbers = []

    for _ in range(3):

        number = int(input("Enter a number: "))


```

```

numbers.append(number)

for number in numbers:

    if is_prime(number):

        print(f'{number} is a prime number.')

    else:

        print(f'{number} is not a prime number.')

```

- Context-Managed (With Optimization)

PROMPT:

```

#Iam a experienced Python programmer.Write a Python program to check whether a
given number is prime.The program must read the number using the input() function.
#The program must read the number using the input() function.

#Convert the input to an integer.

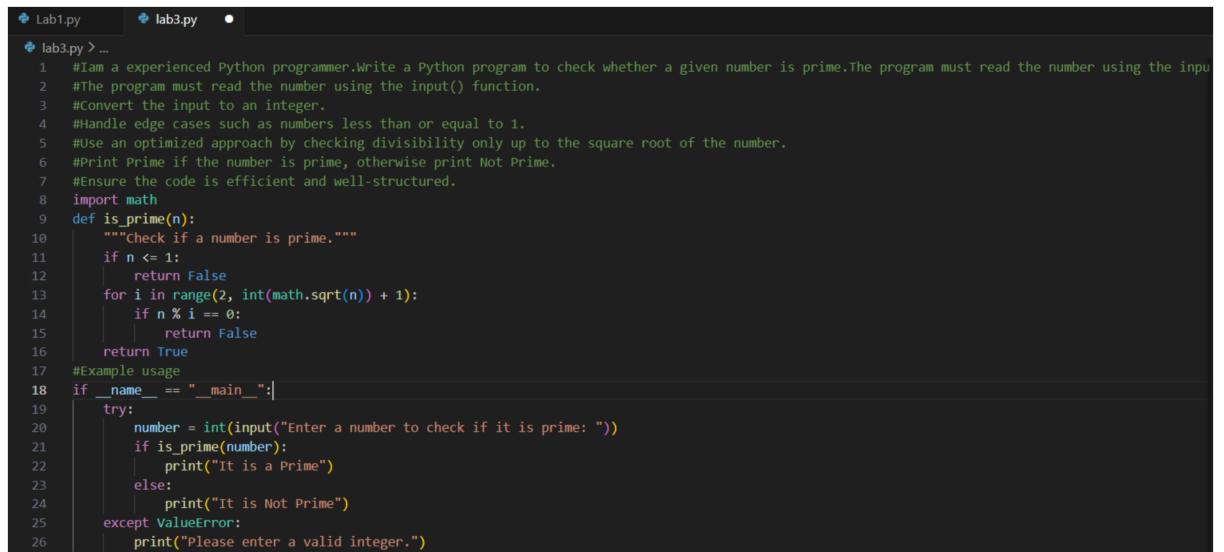
#Handle edge cases such as numbers less than or equal to 1.

#Use an optimized approach by checking divisibility only up to the square root of the
number.

#print Prime if the number is prime, otherwise print Not Prime.

#Ensure the code is efficient and well-structured.

```



```

Lab1.py  lab3.py

lab3.py > ...
1  #Iam a experienced Python programmer.Write a Python program to check whether a given number is prime.The program must read the number using the input() function.
2  #The program must read the number using the input() function.
3  #Convert the input to an integer.
4  #Handle edge cases such as numbers less than or equal to 1.
5  #Use an optimized approach by checking divisibility only up to the square root of the number.
6  #Print Prime if the number is prime, otherwise print Not Prime.
7  #Ensure the code is efficient and well-structured.
8  import math
9  def is_prime(n):
10     """Check if a number is prime."""
11     if n <= 1:
12         return False
13     for i in range(2, int(math.sqrt(n)) + 1):
14         if n % i == 0:
15             return False
16     return True
17  #Example usage
18  if __name__ == "__main__":
19      try:
20          number = int(input("Enter a number to check if it is prime: "))
21          if is_prime(number):
22              print("It is a Prime")
23          else:
24              print("It is Not Prime")
25      except ValueError:
26          print("Please enter a valid integer.")

```

OUTPUT:

```

r' '55043' '--' 'C:\Users\CHAITRA REDDY\Desktop\AI Assistant coding\lab3.py'
Enter a number to check if it is prime: 7
r' '55043' '--' 'C:\Users\CHAITRA REDDY\Desktop\AI Assistant coding\lab3.py'
Enter a number to check if it is prime: 7
Prime
PS C:\Users\CHAITRA REDDY\Desktop\AI Assistant coding> []

```

CODE:

```
import math

def is_prime(n):
    """Check if a number is prime."""
    if n <= 1:
        return False
    for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
            return False
    return True

#Example usage
if __name__ == "__main__":
    try:
        number = int(input("Enter a number to check if it is prime: "))
        if is_prime(number):
            print("It is a Prime")
        else:
            print("It is Not Prime")
    except ValueError:
        print("Please enter a valid integer.")
```

Task: Mobile Data Usage Billing Application (1.0 Marks)**Objective:**

Use Python programming and AI-assisted coding tools to create an application that simulates mobile data billing for a telecom service provider.

Instructions

1. Use GitHub Copilot or Google Gemini to assist in writing the program.
2. Read the following inputs from the user:
 - o Data Consumed (in GB)
 - o Plan Type (Prepaid / Postpaid)
 - o Additional Services Used (e.g., caller tune, OTT subscription, etc.)

PROMPT: #Write a Python program to simulate a mobile data usage billing system for a telecom provider.Data Consumed (in GB),Plan Type (Prepaid / Postpaid),Additional Services Used (e.g., caller tune, OTT subscription, etc.)

```
#Write a Python program to simulate a mobile data usage billing system for a telecom provider.Data Consumed (in GB),Plan Type (Prepaid / Postpaid),/
def calculate_bill(data_consumed_gb, plan_type, additional_services):
    prepaid_rate_per_gb = 10
    postpaid_rate_per_gb = 8
    service_costs = {
        'caller_tune': 5,
        'ott_subscription': 15,
        'cloud_storage': 10
    }

    if plan_type.lower() == 'prepaid':
        base_cost = data_consumed_gb * prepaid_rate_per_gb
    elif plan_type.lower() == 'postpaid':
        base_cost = data_consumed_gb * postpaid_rate_per_gb
    else:
        raise ValueError("Invalid plan type. Choose 'Prepaid' or 'Postpaid'.") 

    additional_cost = sum(service_costs[service] for service in additional_services if service in service_costs)

    total_bill = base_cost + additional_cost
    return total_bill

# Example usage
data_used = input("Enter data consumed (in GB): ")
plan = input("Enter plan type (Prepaid/Postpaid): ")
services = input("Enter additional services used (comma separated, e.g., caller_tune,ott_subscription): ").split(',')
services = [service.strip() for service in services]
print("Total Bill Amount: ${}", calculate_bill(float(data_used), plan, services))
```

OUTPUT:

```
r' '58689' '--' 'C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding\lab3.py'
Enter data consumed (in GB): 23
Enter plan type (Prepaid/Postpaid): Prepaid
Enter additional services used (comma separated, e.g., caller_tune,ott_subscription): caller_tune,No
Total Bill Amount: $ 235.0
PS C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding> []
```

CODE:

```
def calculate_bill(data_consumed_gb, plan_type, additional_services):
    prepaid_rate_per_gb = 10
    postpaid_rate_per_gb = 8
    service_costs = {
        'caller_tune': 5,
        'ott_subscription': 15,
        'cloud_storage': 10
    }

    if plan_type.lower() == 'prepaid':
        base_cost = data_consumed_gb * prepaid_rate_per_gb
    elif plan_type.lower() == 'postpaid':
        base_cost = data_consumed_gb * postpaid_rate_per_gb
    else:
        raise ValueError("Invalid plan type. Choose 'Prepaid' or 'Postpaid'.")
```

```

additional_cost = sum(service_costs[service] for service in additional_services if service in
service_costs)

total_bill = base_cost + additional_cost
return total_bill

# Example usage
data_used = input("Enter data consumed (in GB): ")
plan = input("Enter plan type (Prepaid/Postpaid): ")
services = input("Enter additional services used (comma separated, e.g.,
caller_tune,ott_subscription): ").split(',')
services = [service.strip() for service in services]
print("Total Bill Amount: $", calculate_bill(float(data_used), plan, services))

```

3. Implement billing logic to calculate:

- o DC (Data Charges) – charges based on data consumption
- o VC (Value-added Charges) – charges for additional services
- o Tax – applicable tax on the total bill

PROMPT: #Improve the previous mobile billing Python program by making the charge structure realistic.DC (Data Charges) – charges based on data consumption,VC (Value-added Charges) – charges for additional services,Tax – applicable tax on the total bill

```

lab3.py > ...
lab3.py > ...
1  #Improve the previous mobile billing Python program by making the charge structure realistic.DC (Data Charges) – charges based
2  def calculate_bill_with_tax(data_consumed_gb, plan_type, additional_services):
3      prepaid_rate_per_gb = 10
4      postpaid_rate_per_gb = 8
5      service_costs = {
6          'caller_tune': 5,
7          'ott_subscription': 15,
8          'cloud_storage': 10
9      }
10     tax_rate = 0.18
11
12     if plan_type.lower() == 'prepaid':
13         base_cost = data_consumed_gb * prepaid_rate_per_gb
14     elif plan_type.lower() == 'postpaid':
15         base_cost = data_consumed_gb * postpaid_rate_per_gb
16     else:
17         raise ValueError("Invalid plan type. Choose 'Prepaid' or 'Postpaid'.")
18
19     additional_cost = sum(service_costs[service] for service in additional_services if service in service_costs)
20
21     subtotal = base_cost + additional_cost
22     tax = subtotal * tax_rate
23     total_bill = subtotal + tax
24     return total_bill
25
26     # Example usage
27     data_used = input("Enter data consumed (in GB): ")
28     plan = input("Enter plan type (Prepaid/Postpaid): ")
29     services = input("Enter additional services used (comma separated, e.g., caller_tune,ott_subscription): ").split(',')
30     services = [service.strip() for service in services]
31     print("Total Bill Amount: $", calculate_bill_with_tax(float(data_used), plan, services))

```

OUTPUT:

```
● ps C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding> cd 'c:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding'; & C:\Users\CHAITRA REDDY\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\CHAITRA REDDY\vscode\extensions\ms-python.debugpy-2.5.18.0-win32-x64\bundled\libs\debugpy\launcher' '65492' '--' 'C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding\lab3.py'
Enter data consumed (in GB): 20
Enter plan type (Prepaid/Postpaid): Prepaid
Enter additional services used (comma separated, e.g., caller_tune,ott_subscription): caller_tune,No
Total Bill Amount: $ 241.9
```

CODE:

```
def calculate_bill_with_tax(data_consumed_gb, plan_type, additional_services):
    prepaid_rate_per_gb = 10
    postpaid_rate_per_gb = 8
    service_costs = {
        'caller_tune': 5,
        'ott_subscription': 15,
        'cloud_storage': 10
    }
    tax_rate = 0.18

    if plan_type.lower() == 'prepaid':
        base_cost = data_consumed_gb * prepaid_rate_per_gb
    elif plan_type.lower() == 'postpaid':
        base_cost = data_consumed_gb * postpaid_rate_per_gb
    else:
        raise ValueError("Invalid plan type. Choose 'Prepaid' or 'Postpaid'.")  
  
    additional_cost = sum(service_costs[service] for service in additional_services if service in service_costs)  
  
    subtotal = base_cost + additional_cost
    tax = subtotal * tax_rate
    total_bill = subtotal + tax
    return total_bill  
  
# Example usage
data_used = input("Enter data consumed (in GB): ")
plan = input("Enter plan type (Prepaid/Postpaid): ")
services = input("Enter additional services used (comma separated, e.g., caller_tune,ott_subscription): ").split(',')
services = [service.strip() for service in services]
print("Total Bill Amount: $", calculate_bill_with_tax(float(data_used), plan, services))
```

4.Display an itemized bill showing:

- o Plan Type
- o Data Usage and Charges
- o Value-added Services and Charges
- o Tax
- o Total Bill Amount

PROMPT: #Modify the mobile billing program so that,All inputs are taken using the input() function,Plan type input is case-insensitive,Invalid inputs are handled gracefully,Output formatting resembles an actual mobile bill give code .

```
#Modify the mobile billing program so that,All inputs are taken using the input() function,Plan type input is case-insensitive,Invalid inputs are handled gracefully,Output
def calculate_bill_with_tax(data_consumed_gb, plan_type, additional_services):
    prepaid_rate_per_gb = 10
    postpaid_rate_per_gb = 8
    service_costs = {
        'caller_tune': 5,
        'ott_subscription': 15,
        'cloud_storage': 10
    }
    tax_rate = 0.18
    plan_type = plan_type.lower()
    if plan_type == 'prepaid':
        base_cost = data_consumed_gb * prepaid_rate_per_gb
    elif plan_type == 'postpaid':
        base_cost = data_consumed_gb * postpaid_rate_per_gb
    else:
        return "Invalid plan type. Choose 'Prepaid' or 'Postpaid'."
    additional_cost = sum(service_costs[service] for service in additional_services if service in service_costs)
    subtotal = base_cost + additional_cost
    tax = subtotal * tax_rate
    total_bill = subtotal + tax
    bill_format = f"""
    -----
    MOBILE BILL
    -----
    Data Consumed: {data_consumed_gb} GB
    Plan Type: {plan_type.capitalize()}
    Base Cost: ${base_cost:.2f}

    Additional Services:
    """
    for service in additional_services:
        if service in service_costs:
            bill_format += f" - {service}: ${service_costs[service]:.2f}\n"
    bill_format += f"""
    Subtotal: ${subtotal:.2f}
    Tax (18%): ${tax:.2f}
    -----
    Total Bill Amount: ${total_bill:.2f}
    -----
    """

    return bill_format
# Example usage
data_used = input("Enter data consumed (in GB): ")
plan = input("Enter plan type (Prepaid/Postpaid): ")
services = input("Enter additional services used (comma separated, e.g., caller_tune,ott_subscription): ").split(',')
services = [service.strip() for service in services]
print(calculate_bill_with_tax(float(data_used), plan, services))
```

OUTPUT:

```

PS C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding> c;; cd 'c:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding'; & 'c:\Users\CHAITRA REDDY\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\CHAITRA REDDY\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '52513' '--' 'c:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding\lab3.py'
Enter data consumed (in GB): 23
Enter plan type (Prepaid/Postpaid): Prepaid
Enter additional services used (comma separated, e.g., caller_tune,ott_subscription): caller_tune,No

-----
MOBILE BILL
-----
Data Consumed: 23.0 GB
Plan Type: Prepaid
Base Cost: $230.00

Additional Services:
- caller_tune: $5.00

Subtotal: $235.00
Tax (18%): $42.30
-----
Total Bill Amount: $277.30
-----
Total Bill Amount: $277.30
-----
```

CODE:

```

def calculate_bill_with_tax(data_consumed_gb, plan_type, additional_services):
    prepaid_rate_per_gb = 10
    postpaid_rate_per_gb = 8
    service_costs = {
        'caller_tune': 5,
        'ott_subscription': 15,
        'cloud_storage': 10
    }
    tax_rate = 0.18
    plan_type = plan_type.lower()
    if plan_type == 'prepaid':
        base_cost = data_consumed_gb * prepaid_rate_per_gb
    elif plan_type == 'postpaid':
        base_cost = data_consumed_gb * postpaid_rate_per_gb
    else:
        return "Invalid plan type. Choose 'Prepaid' or 'Postpaid'."
    additional_cost = sum(service_costs[service] for service in additional_services if service in service_costs)
    subtotal = base_cost + additional_cost
    tax = subtotal * tax_rate
    total_bill = subtotal + tax
    bill_format = f"""

-----
MOBILE BILL
-----

Data Consumed: {data_consumed_gb} GB
Plan Type: {plan_type.capitalize()}
Base Cost: ${base_cost:.2f}

Additional Services:
{"""}
```

```

for service in additional_services:
    if service in service_costs:
        bill_format += f"- {service}: ${service_costs[service]:.2f}\n"

bill_format += f"""
Subtotal: ${subtotal:.2f}
Tax (18%): ${tax:.2f}
-----
Total Bill Amount: ${total_bill:.2f}
-----
"""
return bill_format

```

Task: 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.

Instructions

1. Use GitHub Copilot or Google Gemini to assist in writing and refining the program.
2. Read the following user inputs:
 - o Cylinder Type (Domestic 14.2 kg / Domestic 5 kg / Commercial 19 kg / Commercial 47.5 kg)
 - o Number of Cylinders Booked
 - o Subsidy Amount (applicable only for domestic cylinders)

PROMPT:# Generate a Python program for an LPG billing system that takes cylinder type, number of cylinders booked, and subsidy (only for domestic) as input, uses the given LPG price list and delivery charges, and calculates the total bill.

```

❶ lab3.py
❷ lab3.py > ...
1  #Generate a Python program for an LPG billing system that takes cylinder type, number of cylinders booked, and subsidy (only for domestic) as input
2  def calculate_lpg_bill(cylinder_type, num_cylinders, subsidy=0):
3
4      price_list = {
5          'domestic': 500, # Price per domestic cylinder
6          'commercial': 1000 # Price per commercial cylinder
7      }
8
9      delivery_charges = {
10         'domestic': 50, # Delivery charge for domestic cylinder
11         'commercial': 100 # Delivery charge for commercial cylinder
12     }
13
14     if cylinder_type not in price_list:
15         return "Invalid cylinder type. Please choose 'domestic' or 'commercial'."
16
17     base_price = price_list[cylinder_type] * num_cylinders
18
19     total_delivery_charge = delivery_charges[cylinder_type] * num_cylinders
20
21     total_bill = base_price + total_delivery_charge
22
23     if cylinder_type == 'domestic':
24         total_bill -= subsidy
25
26     return total_bill

```

OUTPUT:

```

PS C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding> c;; cd "c:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding"; & "C:\Users\CHAITRA REDDY\Local\Microsoft\WindowsApps\python3.13.exe" 'c:\Users\CHAITRA REDDY\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launc
hen' '65479' '---' 'C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding\lab3.py'
Enter cylinder type (domestic/commercial): domestic
Enter number of cylinders booked: 3
Enter subsidy amount (if any): 200
Total LPG Bill: 1450
PS C:\Users\CHAITRA REDDY\OneDrive\Desktop\AI Assistant coding>

```

CODE:

```
def calculate_lpg_bill(cylinder_type, num_cylinders, subsidy=0):
```

```

price_list = {
    'domestic': 500, # Price per domestic cylinder
    'commercial': 1000 # Price per commercial cylinder
}
```

```

delivery_charges = {
    'domestic': 50, # Delivery charge for domestic cylinder
    'commercial': 100 # Delivery charge for commercial cylinder
}
```

```

if cylinder_type not in price_list:
    return "Invalid cylinder type. Please choose 'domestic' or 'commercial'."
```

```
base_price = price_list[cylinder_type] * num_cylinders
```

```
total_delivery_charge = delivery_charges[cylinder_type] * num_cylinders
```

```

total_bill = base_price + total_delivery_charge

if cylinder_type == 'domestic':
    total_bill -= subsidy

return total_bill

# Example usage
cylinder_type = input("Enter cylinder type (domestic/commercial): ").strip().lower()
num_cylinders = int(input("Enter number of cylinders booked: "))
subsidy = 0
if cylinder_type == 'domestic':
    subsidy = int(input("Enter subsidy amount (if any): "))
total_bill = calculate_lpg_bill(cylinder_type, num_cylinders, subsidy)
print(f"Total LPG Bill: {total_bill}")

```

3. Refer to the given LPG Price List to determine the price per cylinder:

- Domestic LPG (14.2 kg) → ₹905.00
- Domestic LPG (5 kg) → ₹335.50
- Commercial LPG (19 kg) → ₹1,886.50
- Commercial LPG (47.5 kg) → ₹4,712.00
- Delivery Charges (₹10 to ₹50)

PROMPT:

4. Implement the billing formula:

Bill Amount = (Price per Cylinder × Quantity) - Subsidy (if applicable) + Delivery Charges

PROMPT:

5. Calculate and display an itemized bill including:

- Cylinder Type
- Number of Cylinders
- Base Amount

- Subsidy
- Delivery Charges
- Total Bill Amount

Deliverables

- A report containing:
 - AI prompts used to generate the program
 - AI-generated Python code
 - Line-by-line explanation of the code