

Assignment-III

NAME : R.Shivani

ROLL NO. : 2403A52411

BATCH:15

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.

-

One-Shot:

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

```
Prime_number.py X
1 # Implementations from different prompting strategies
2
3 # Zero-Shot
4 def is_prime_zero_shot(n):
5     if n <= 1:
6         return False
7     for i in range(2, n):
8         if n % i == 0:
9             return False
10    return True
11
12 # One-Shot
13 def is_prime_one_shot(n):
14     if n <= 1:
15         return "Not Prime"
16     for i in range(2, n):
17         if n % i == 0:
18             return "Not Prime"
19     return "Prime"
20
21 # Few-Shot
22 def is_prime_few_shot(n):
23     if n <= 1:
24         return "Not Prime"
25     if n == 2:
26         return "Prime"
27     for i in range(2, int(n ** 0.5) + 1):
28         if n % i == 0:
29             return "Not Prime"
30    return "Prime"
```

PROBLEMS TERMINAL ... powershell + ...

```
PS D:\BTECH\AI Assisted Coding\LABS ASSIGNMENTS\Lab 3> & C:\Users\sakr\AppData\Local\Programs\Python\Python313\python.exe "d:/BTECH/AI Assisted Coding/LAB ASSIGNMENTS/Lab 3/Prime_number.py"
```

Number	Zero-Shot	One-Shot	Few-Shot	Context-Managed
2	True	Prime	Prime	True
5	True	Prime	Prime	True
5	True	Prime	Prime	True
5	True	Prime	Prime	True
17	True	Prime	Prime	True
19	True	Prime	Prime	True
5	True	Prime	Prime	True
False	Not Prime	Not Prime	Not Prime	False
17	True	Prime	Prime	True
False	Not Prime	Not Prime	Not Prime	False
19	True	Prime	Prime	True
True	Prime	Prime	Prime	True
20	False	Not Prime	Not Prime	False
5	True	Prime	Prime	True
5	True	Prime	Prime	True
5	True	Prime	Prime	True
5	True	Prime	Prime	True
5	True	Prime	Prime	True
17	True	Prime	Prime	True
19	True	Prime	Prime	True
5	True	Prime	Prime	True
5	True	Prime	Prime	True
17	True	Prime	Prime	True
19	True	Prime	Prime	True
20	False	Not Prime	Not Prime	False
97	True	Prime	Prime	True

Task: Mobile Data Usage Billing Application

Objective:

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

Instructions

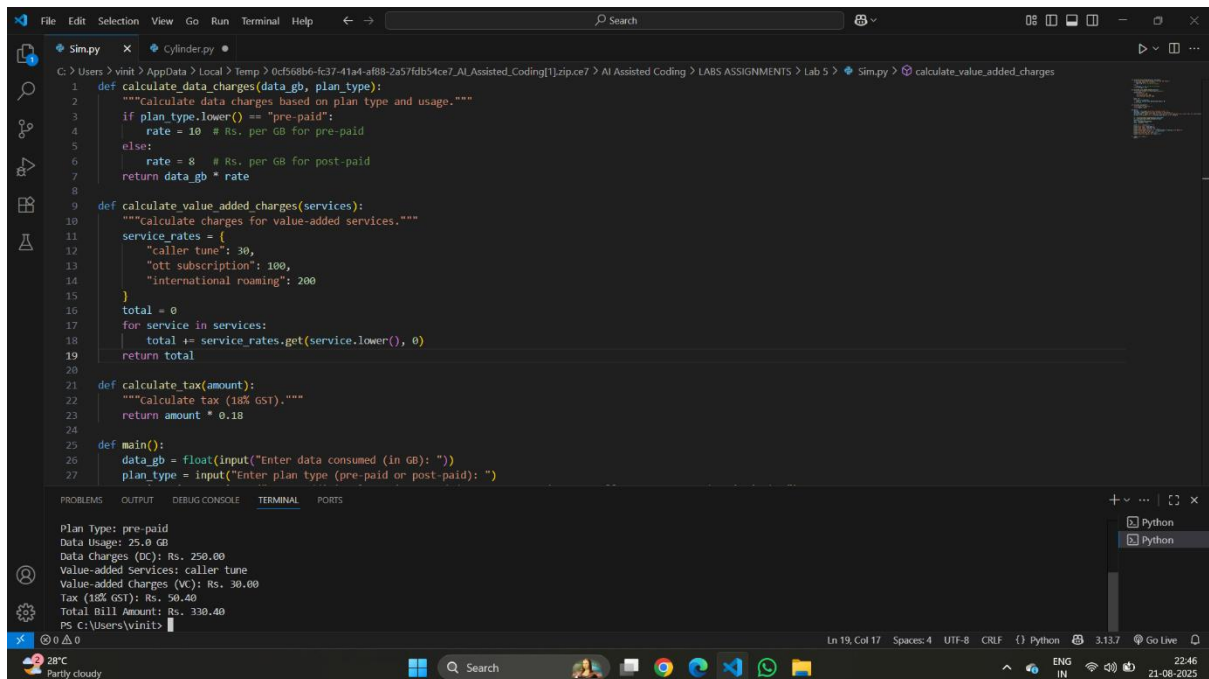
1. Use GitHub Copilot or Google Gemini to assist in writing the program.
2. Read the following inputs from the user:
 - Data Consumed (in GB)
 - Plan Type (Prepaid / Postpaid)
 - Additional Services Used (e.g., caller tune, OTT subscription, etc.)
3. Implement billing logic to calculate:
 - DC (Data Charges) – charges based on data consumption
 - VC (Value-added Charges) – charges for additional services
 - Tax – applicable tax on the total bill
4. Display an itemized bill showing:
 - Plan Type
 - Data Usage and Charges
 - Value-added Services and Charges
 - Tax
 - Total Bill Amount

Requirements

- Students must refer to their actual mobile bill for charge structure (data cost, service fees, taxes) to make the program realistic.
- AI assistance (Copilot/Gemini) must be used to generate and refine the initial code.

Deliverables

- AI prompts used for code generation.
- AI-generated Python code and any optimized version.



```
1 def calculate_data_charges(data_gb, plan_type):
2     """calculate data charges based on plan type and usage."""
3     if plan_type.lower() == "pre-paid":
4         rate = 10 # Rs. per GB for pre-paid
5     else:
6         rate = 8 # Rs. per GB for post-paid
7     return data_gb * rate
8
9 def calculate_value_added_charges(services):
10    """calculate charges for value-added services."""
11    service_rates = {
12        "caller tune": 30,
13        "ott subscription": 100,
14        "international roaming": 200
15    }
16    total = 0
17    for service in services:
18        total += service_rates.get(service.lower(), 0)
19    return total
20
21 def calculate_tax(amount):
22    """calculate tax (18% GST)."""
23    return amount * 0.18
24
25 def main():
26    data_gb = float(input("Enter data consumed (in GB): "))
27    plan_type = input("Enter plan type (pre-paid or post-paid): ")
28
29 if __name__ == "__main__":
30    main()
```

Plan Type: pre-paid
Data Usage: 25.0 GB
Data Charges (DC): Rs. 250.00
Value-added Services: caller tune
Value-added Charges (VC): Rs. 30.00
Tax (18% GST): Rs. 50.40
Total Bill Amount: Rs. 330.40
PS C:\Users\win10>

Task: Develop an LPG Billing System

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:
 - Cylinder Type (Domestic 14.2 kg / Domestic 5 kg / Commercial 19 kg / Commercial 47.5 kg)
 - Number of Cylinders Booked
 - Subsidy Amount (applicable only for domestic cylinders)
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)

4. Implement the billing formula:

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

5. Calculate and display an itemized bill including:

- Cylinder Type
- Number of Cylinders
- Base Amount
- Subsidy
- Delivery Charges
- Total Bill Amount

```

1 def get_cylinder_price(cylinder_type):
2     """Return price per cylinder based on type."""
3     prices = {
4         "domestic 14.2kg": 995.00,
5         "domestic 5kg": 335.50,
6         "commercial 19kg": 1886.50,
7         "commercial 47.5kg": 4712.00
8     }
9     return prices.get(cylinder_type.lower(), 0)
10
11 def main():
12     cylinder_type = input("Enter cylinder type (Domestic 14.2kg / Domestic 5kg / Commercial 19kg / Commercial 47.5kg): ")
13     quantity = int(input("Enter number of cylinders booked: "))
14     delivery_charges = float(input("Enter delivery charges (10 to 50 Rs): "))
15
16     price_per_cylinder = get_cylinder_price(cylinder_type)
17     base_amount = price_per_cylinder * quantity
18     subsidy = 0
19     if "domestic" in cylinder_type.lower():
20         subsidy = float(input("Enter subsidy amount (Applicable only for domestic cylinders): "))
21
22     total_bill = base_amount + subsidy + delivery_charges
23
24     print("\n--- LPG Itemized Bill ---")
25     print(f"Cylinder Type      : {cylinder_type}")
26     print(f"Number of cylinders: {quantity}")
27     print(f"Base Amount         : Rs. {base_amount:.2f}")

```

```

--- LPG Itemized Bill ---
Cylinder Type      : Commercial 19kg
Number of cylinders: 5
Base Amount         : Rs. 9432.50
Subsidy             : Rs. 0.00
Delivery Charges    : Rs. 50.00
Total Bill Amount   : Rs. 9482.50

```