

AI ASSISTED CODING

LAB-3.5

Name: J.Varshini

H.No:2303A51758

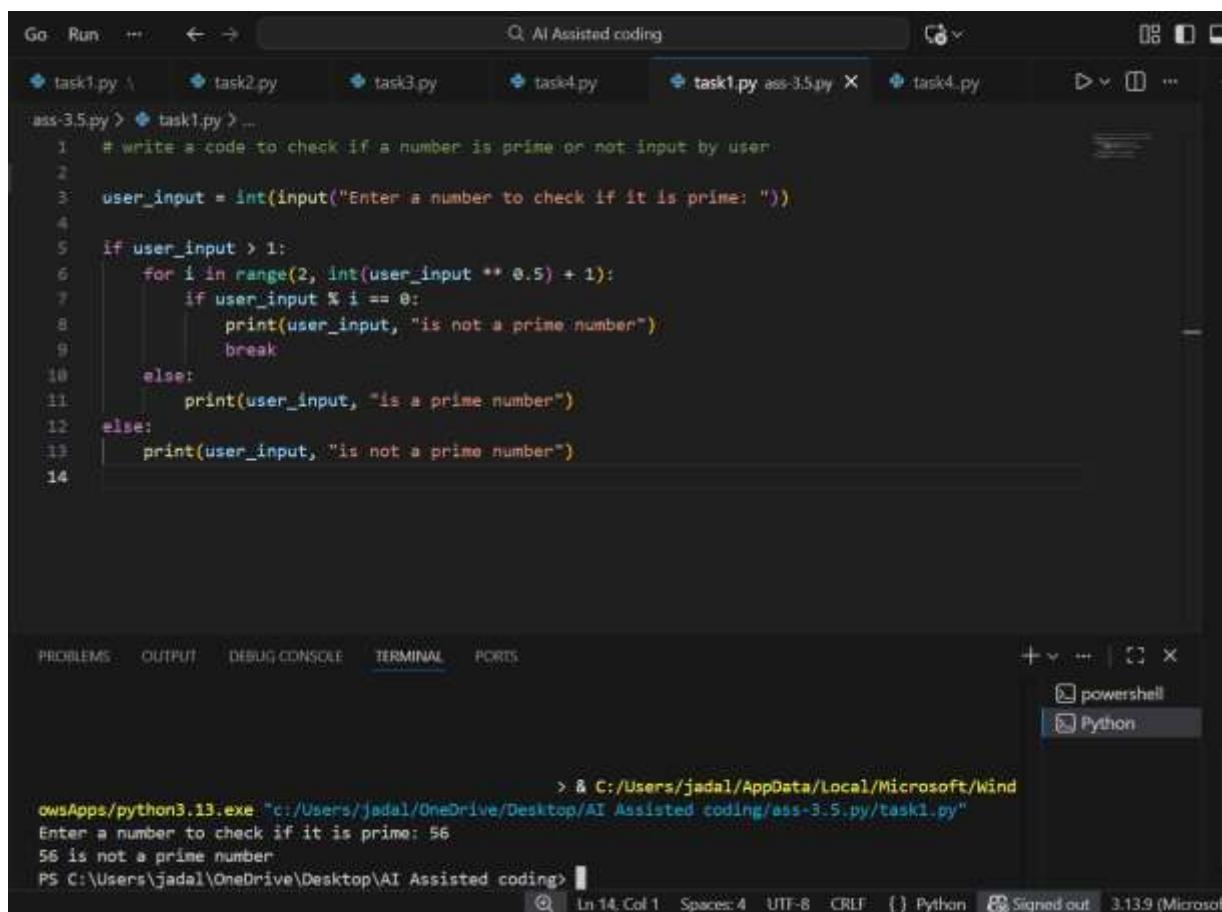
Batch:11

1. Select a simple task: "Write a Python function to check if a number is prime."

Prompt 1:

Write a code to check if a number is prime or not input by user.

Given Code and Output:



The screenshot shows a code editor in VS Code with a dark theme. The file 'task1.py' is open, containing the following Python code:

```
ass-3.5.py > task1.py > ...
1  # write a code to check if a number is prime or not input by user
2
3  user_input = int(input("Enter a number to check if it is prime: "))
4
5  if user_input > 1:
6      for i in range(2, int(user_input ** 0.5) + 1):
7          if user_input % i == 0:
8              print(user_input, "is not a prime number")
9              break
10         else:
11             print(user_input, "is a prime number")
12     else:
13         print(user_input, "is not a prime number")
14
```

The terminal below shows the execution of the code and its output:

```
> & C:/Users/jadal/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/jadal/OneDrive/Desktop/AI Assisted coding/ass-3.5.py/task1.py"
Enter a number to check if it is prime: 56
56 is not a prime number
PS C:\Users\jadal\OneDrive\Desktop\AI Assisted coding>
```

Prompt 2:

Write a code to check if a number is prime or not input by user using functions.

Given Code and Output:

The screenshot shows a code editor with a Python script named task2.py. The code defines a function is_prime that checks if a given number is prime by testing divisibility from 2 to the square root of the number. It then prompts the user for input and prints whether the entered number is prime or not. The terminal below shows the script being run and a prime number 67 being checked.

```
task2.py > task2.py > ...
1 def is_prime(num):
2     if num <= 1:
3         return False
4     for i in range(2, int(num ** 0.5) + 1):
5         if num % i == 0:
6             return False
7     return True
8
9 user_input = int(input("Enter a number to check if it is prime: "))
10
11 if is_prime(user_input):
12     print(f"{user_input} is a prime number")
13 else:
14     print(f"{user_input} is not a prime number")
15
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

WindowsApps/python3.13.exe "c:/Users/jadal/OneDrive/Desktop/AI Assisted coding/ass-3.5.py/task2.py"
PS C:\Users\jadal> & C:/Users/jadal/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/jadal/OneDrive/Desktop/AI Assisted coding/ass-3.5.py/task2.py"
Enter a number to check if it is prime: 67
67 is a prime number
PS C:\Users\jadal\OneDrive\Desktop\AI Assisted coding>

Prompt 3:

Give an optimized code to check if a number is prime Given

Code and Output:

The screenshot shows a code editor with a Python script named task3.py. This script uses a more efficient prime-checking algorithm. It first handles small numbers (0, 1, 2, 3). For larger numbers, it checks divisibility only up to the square root of n, skipping even numbers after 2. It uses a step of 6 in a while loop to skip multiples of 3. The terminal shows the script being run and a prime number 17 being checked.

```
ass-3.5.py > task3.py > ...
1 # Give an optimized code to check if a number is prime.
2 def is_prime(n):
3     if n <= 1:
4         return False
5     if n <= 3:
6         return True
7     if n % 2 == 0 or n % 3 == 0:
8         return False
9     i = 5
10    while i * i <= n:
11        if n % i == 0 or n % (i + 2) == 0:
12            return False
13        i += 6
14    return True
15
16 number = int(input("Enter a number to check if it is prime: "))
17 if is_prime(number):
18     print(f"{number} is a prime number.")
19 else:
20     print(f"{number} is not a prime number.")
21
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

/WindowsApps/python3.13.exe "c:/Users/jadal/OneDrive/Desktop/AI Assisted coding/ass-3.5.py/task3.py"
Enter a number to check if it is prime: 17
17 is a prime number.
PS C:\Users\jadal\OneDrive\Desktop\AI Assisted coding>

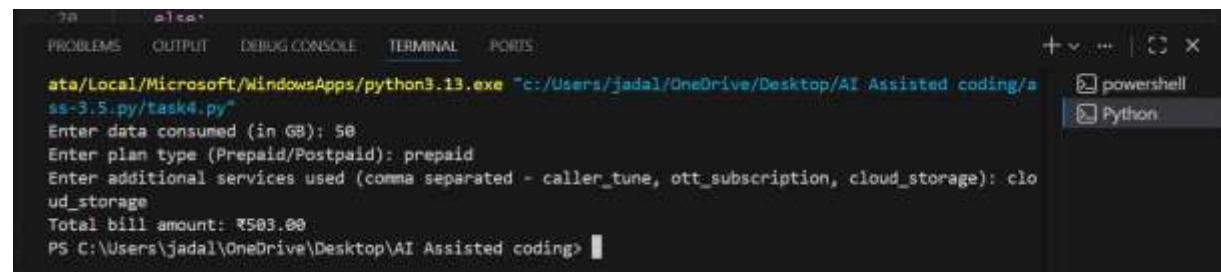
Task: Mobile Data Usage Billing Application (1.0 Marks)

Prompt 1:

Write a code for mobile data usage billing application that takes the input of o Data Consumed (in GB), Plan Type (Prepaid / Postpaid), Additional Services Used (e.g., caller tune, OTT subscription, etc.) and calculate the total bill amount Code:

```
lab3.py > ...
1  # Write a code for mobile data usage billing application that takes the input of o Data Consumed
2  def calculate_bill(data_consumed_gb, plan_type, additional_services):
3      # Define base rates and additional service costs
4      base_rate_prepaid = 10 # per GB
5      base_rate_postpaid = 8 # per GB
6      additional_service_costs = {
7          'caller_tune': 2,
8          'ott_subscription': 5,
9          'cloud_storage': 3
10     }
11
12     # Calculate base bill based on plan type
13     if plan_type.lower() == 'prepaid':
14         base_bill = data_consumed_gb * base_rate_prepaid
15     elif plan_type.lower() == 'postpaid':
16         base_bill = data_consumed_gb * base_rate_postpaid
17     else:
18         raise ValueError("Invalid plan type. Please choose 'Prepaid' or 'Postpaid'.")
19
20     # Calculate additional services cost
21     additional_cost = sum(additional_service_costs[service] for service in additional_services if
22
23         # Total bill amount
24         total_bill = base_bill + additional_cost
25         return total_bill
26
27     # Example usage
28     data_consumed = float(input("Enter data consumed (in GB): "))
29     plan_type = input("Enter plan type (Prepaid/Postpaid): ")
30     additional_services_input = input("Enter additional services used (comma separated, e.g., caller_t
31     additional_services = [service.strip() for service in additional_services_input.split(',') if serv
32     total_amount = calculate_bill(data_consumed, plan_type, additional_services)
33     print(f"Total bill amount: ${total_amount:.2f}")
```

Output:



```
78  alca:
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v ... C3 X
ata:Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/jadal/OneDrive/Desktop/AI Assisted coding/ass-3.5.py/task4.py"
Enter data consumed (in GB): 50
Enter plan type (Prepaid/Postpaid): prepaid
Enter additional services used (comma separated - caller_tune, ott_subscription, cloud_storage): caller_tune, ott_subscription, cloud_storage
Total bill amount: $503.00
PS C:\Users\jadal\OneDrive\Desktop\AI Assisted coding>
```

Prompt 2:

improve the code by adding Data Charges (based on data usage), Value-added service charges, Tax on total amount, Display an itemized bill.

Code:

```
# Improve the code by adding Data Charges (based on data usage), Value-added service charges, Tax on total amount, Display an itemized bill.
def calculate_bill(data_consumed_gb, plan_type, additional_services):
    # Define base rates and additional service costs
    base_rate_prepaid = 10 # per GB
    base_rate_postpaid = 8 # per GB
    additional_service_costs = [
        'caller_tune': 2,
        'ott_subscription': 5,
        'cloud_storage': 3
    ]
    tax_rate = 0.1 # 10% tax

    # Calculate base bill based on plan type
    if plan_type.lower() == 'prepaid':
        data_charges = data_consumed_gb * base_rate_prepaid
    elif plan_type.lower() == 'postpaid':
        data_charges = data_consumed_gb * base_rate_postpaid
    else:
        raise ValueError("Invalid plan type. Please choose 'Prepaid' or 'Postpaid'.") 

    # Calculate additional services cost
    additional_cost = sum(additional_service_costs.get(service) for service in additional_services if service in additional_service_costs)

    # Calculate subtotal
    subtotal = data_charges + additional_cost

    # Calculate tax
    tax_amount = subtotal * tax_rate

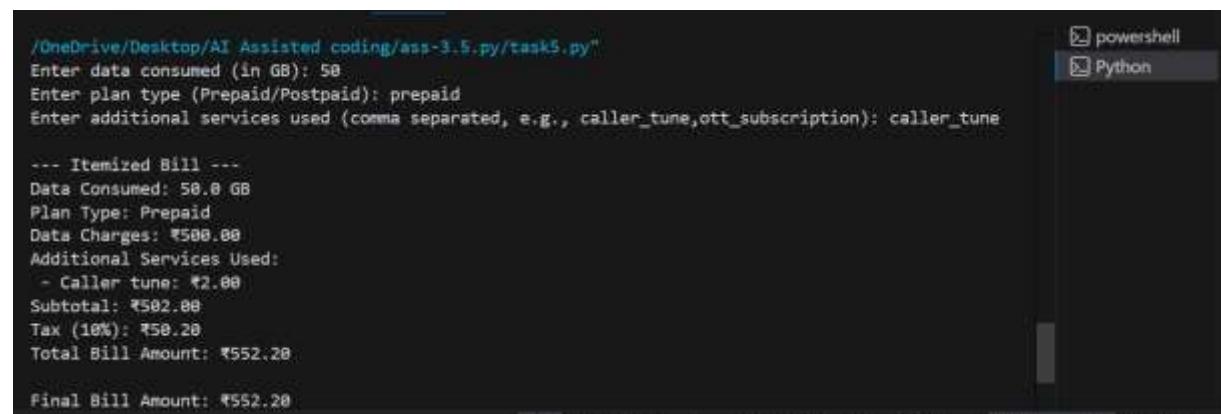
    # Total bill amount
    total_bill = subtotal + tax_amount

    # Display itemized bill
    print("\n--- Itemized Bill ---")
    print(f"Data Consumed: {data_consumed_gb} GB")
    print(f"Plan Type: {plan_type.capitalize()}")
    print(f"Data Charges: ${data_charges:.2f}")
    print("Additional Services Used:")
    for service in additional_services:
        if service in additional_service_costs:
            print(f" - {service.replace('_', ' ').capitalize(): ${additional_service_costs[service]:.2f}}")

    print(f"Subtotal: ${subtotal:.2f}")
    print(f"Tax (10%): ${tax_amount:.2f}")
    print(f"Total Bill Amount: ${total_bill:.2f}")

    return total_bill
# Example usage
data_consumed = float(input("Enter data consumed (in GB): "))
plan_type = input("Enter plan type (Prepaid/Postpaid): ")
additional_services_input = input("Enter additional services used (comma separated, e.g., caller_tune,ott_subscription): ")
additional_services = [service.strip() for service in additional_services_input.split(',') if service.strip()]
total_amount = calculate_bill(data_consumed, plan_type, additional_services)
print(f"\nTotal bill amount: ${total_amount:.2f}")
```

Output:



```
/OneDrive/Desktop/AI Assisted coding/ass-3.5.py/task5.py"
Enter data consumed (in GB): 50
Enter plan type (Prepaid/Postpaid): prepaid
Enter additional services used (comma separated, e.g., caller_tune,ott_subscription): caller_tune

--- Itemized Bill ---
Data Consumed: 50.0 GB
Plan Type: Prepaid
Data Charges: $500.00
Additional Services Used:
 - Caller tune: $2.00
Subtotal: $502.00
Tax (10%): $50.20
Total Bill Amount: $552.20

Final Bill Amount: $552.20
```

Prompt 3:

Make the above code more optimized and structured

Code:

```
#Make the above code more optimized and structured
def calculate_data_charges(data_consumed_gb, plan_type):
    base_rate_prepaid = 10 # per GB
    base_rate_postpaid = 8 # per GB

    if plan_type.lower() == 'prepaid':
        return data_consumed_gb * base_rate_prepaid
    elif plan_type.lower() == 'postpaid':
        return data_consumed_gb * base_rate_postpaid
    else:
        raise ValueError("Invalid plan type. Please choose 'Prepaid' or 'Postpaid'.")
def calculate_additional_cost(additional_services, additional_service_costs):
    return sum(additional_service_costs[service] for service in additional_services if service in additional_service_costs)
def calculate_tax(subtotal, tax_rate):
    return subtotal * tax_rate
def calculate_bill(data_consumed_gb, plan_type, additional_services):
    additional_service_costs = []
    'caller_tune': 2,
    'ott_subscription': 5,
    'cloud_storage': 3
    ]
    tax_rate = 0.1 # 10% tax

    data_charges = calculate_data_charges(data_consumed_gb, plan_type)
    additional_cost = calculate_additional_cost(additional_services, additional_service_costs)
    subtotal = data_charges + additional_cost
    tax_amount = calculate_tax(subtotal, tax_rate)
    total_bill = subtotal + tax_amount

    # Display itemized bill
    print("\n--- Itemized Bill ---")
    print(f"Data Consumed: {data_consumed_gb} GB")
    print(f"Plan Type: {plan_type.capitalize()}")
    print(f"Data Charges: ${data_charges:.2f}")
    print("Additional Services Used:")
    for service in additional_services:
        if service in additional_service_costs:
            print(f" - {service.replace('_', ' ').capitalize(): ${additional_service_costs[service]:.2f}}")
    print(f"Subtotal: ${subtotal:.2f}")
    print(f"Tax (10%): ${tax_amount:.2f}")
    print(f"Total Bill Amount: ${total_bill:.2f}")

    return total_bill
# Example usage
data_consumed = float(input("Enter data consumed (in GB): "))
plan_type = input("Enter plan type (Prepaid/Postpaid): ")
additional_services_input = input("Enter additional services used (comma separated, e.g., caller_tune,ott_subscription): ")
additional_services = [service.strip() for service in additional_services_input.split(',') if service.strip()]
total_amount = calculate_bill(data_consumed, plan_type, additional_services)
print(f"\nTotal bill amount: ${total_amount:.2f}")
```

Output:

```

Enter data consumed (in GB): 50
Enter plan type (Prepaid/Postpaid): prepaid
Enter additional services used (comma separated, e.g., caller_tune,ott_subscription): caller_tune, cloud_storage

--- Itemized Bill ---
Data Consumed: 50.0 GB
Plan Type: Prepaid
Data Charges: $500.00
Additional Services Used:
  - Caller tune: $2.00
  - Cloud storage: $3.00
Subtotal: $505.00
Tax (10%): $50.50
Total Bill Amount: $555.50

Total bill amount: $555.50

```

Task: Develop an LPG Billing System (1.0 Marks)

Prompt 1:

#Write a code to calculate the LPG bill based on cylinder type, number of cylinders booked, subsidy amount, and delivery charges input by user.

Code:

```

lpg_bill.py / ...
1  #Write a code to calculate the LPG bill based on cylinder type, number of cylinders booked, subsidy amount, and delivery charges
2  def calculate_lpg_bill(cylinder_type, num_cylinders, subsidy_amount, delivery_charges):
3      # Define base prices for different cylinder types
4      base_prices = {
5          'domestic': 500,
6          'commercial': 1000,
7          'industrial': 1500
8      }
9
10     # Get the base price for the selected cylinder type
11     if cylinder_type not in base_prices:
12         raise ValueError("Invalid cylinder type. Choose from 'domestic', 'commercial', or 'industrial'.")
13
14     base_price = base_prices[cylinder_type]
15
16     # Calculate total cost before subsidy and delivery charges
17     total_cost = base_price * num_cylinders
18
19     # Apply subsidy
20     total_cost -= subsidy_amount
21
22     # Add delivery charges
23     total_cost += delivery_charges
24
25
26     return total_cost
27
28 # Example usage
29 cylinder_type = input("Enter cylinder type (domestic/commercial/industrial): ").strip().lower()
30 num_cylinders = int(input("Enter number of cylinders booked: "))
31 subsidy_amount = float(input("Enter subsidy amount: "))
32 delivery_charges = float(input("Enter delivery charges: "))
33 total_bill = calculate_lpg_bill(cylinder_type, num_cylinders, subsidy_amount, delivery_charges)
34 print(f"The total LPG bill is: {total_bill}")

```

Output:

```
ktop/AIAC/lab3.py
Enter cylinder type (domestic/commercial/industrial): domestic
Enter number of cylinders booked: 3
Enter subsidy amount: 905
Enter delivery charges: 60
The total LPG bill is: 655.0
```

Explanation:

- In all the tasks at first I asked a basic prompt without any requirements, then the response that was given was very basic.
- After specifying what I wanted in the next prompts the response that was given was more optimised and structured.