LAB ASSIGNMENT-3.2

NAME : S.NADHIYA

ROLL NO : 2403A510C6

BATCH : 05

BRANCH : CSE

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
ProgramName:B. Tech		Assignment Type: Lab	AcademicYear:2025-2026
CourseCoordinatorName		Venkataramana Veeramsetty	
Instructor(s)Name		1. Dr. Mohammed Ali Shaik 2. Dr. T Sampath Kumar 3. Mr. S Naresh Kumar 4. Dr. V. Rajesh 5. Dr. Brij Kishore 6. Dr Pramoda Patro 7. Dr. Venkataramana 8. Dr. Ravi Chander 9. Dr. Jagjeeth Singh	
CourseCode	24CS002PC215	CourseTitle	AI Assisted Coding
Year/Sem	II/I	Regulation	R24
Date and Day of Assignment	Week2-Tuesday	Time(s)	
Duration	2 Hours	Applicableto Batches	24CSBTB01 To 24CSBTB39
AssignmentNumber:3.2(Present assignment number)/24(Total number of assignments)			

SCREEN SHOTS:

TASK-1

write a python code to calculate compound interest starting with only the function name. Then add a docstring, then input-output example

```
Total Bill Amount: $63.25
```

```python
def calculate_compound_interest(principal, rate, time):
    """Calculates compound interest.

    Args:
        principal (float): The initial amount of money.
        rate (float): The annual interest rate (as a decimal).
        time (float): The number of years the money is invested or borrowed for.

    Returns:
        float: The compound interest.
    """
    # Compound interest formula: A = P(1 + r/n)^(nt)
    # Where:
    # A = the future value of the investment/loan, including interest
    # P = the principal investment amount (the initial deposit or loan amount)
    # r = the annual interest rate (as a decimal)
    # n = the number of times that interest is compounded per year
    # t = the number of years the money is invested or borrowed for

    # For simplicity, assuming interest is compounded annually (n=1)
    amount = principal * (1 + rate)**time
    compound_interest = amount - principal
    return compound_interest

# Input-output example
principal_amount = 1000
annual_rate = 0.05  # 5%
years = 10

interest = calculate_compound_interest(principal_amount, annual_rate, years)
print(f"Principal: ${principal_amount}")
print(f"Annual Rate: {annual_rate*100}%")
print(f"Time (years): {years}")
print(f"Compound Interest: ${interest:.2f}")
```

```
Principal: $1000
Annual Rate: 5.0%
Time (years): 10
Compound Interest: $628.89
```

EXPLANATION:

The code defines a function calculate mobile_ bill to compute mobile data charges, including base rates, additional services, and tax.

Dictionaries are used to store the costs for different plan types and additional services.

The function calculates data charges, value-added service charges, subtotal, tax, and the total bill amount.

The code then prompts the user for data usage, plan type, and additional services.

Finally, it calculates the bill using the function and displays the detailed breakdown of the charges.

TASK-2

write a pythode to calculate average ,median,mode of a numbers in a list using functions

```
from collections import Counter

def calculate_average(numbers):
    """Calculates the average of a list of numbers."""
    if not numbers:
        return 0
    return sum(numbers) / len(numbers)

def calculate_median(numbers):
    """Calculates the median of a list of numbers."""
    if not numbers:
        return None
    sorted_numbers = sorted(numbers)
    n = len(sorted_numbers)
    mid_index = n // 2
    if n % 2 == 0:
        return (sorted_numbers[mid_index - 1] + sorted_numbers[mid_index]) / 2
    else:
        return sorted_numbers[mid_index]

def calculate_mode(numbers):
    """Calculates the mode of a list of numbers."""
    if not numbers:
        return None
    count = Counter(numbers)
    max_count = max(count.values())
    modes = [num for num, freq in count.items() if freq == max_count]
    return modes

# Example usage:
my_list = [1, 2, 3, 4, 5, 5, 6, 6, 6, 7]

average = calculate_average(my_list)
```

Variables Terminal

```
    mid_index = n // 2
    if n % 2 == 0:
        return (sorted_numbers[mid_index - 1] + sorted_numbers[mid_index]) / 2
    else:
        return sorted_numbers[mid_index]

def calculate_mode(numbers):
    """Calculates the mode of a list of numbers."""
    if not numbers:
        return None
    count = Counter(numbers)
    max_count = max(count.values())
    modes = [num for num, freq in count.items() if freq == max_count]
    return modes

# Example usage:
my_list = [1, 2, 3, 4, 5, 5, 6, 6, 6, 7]

average = calculate_average(my_list)
median = calculate_median(my_list)
mode = calculate_mode(my_list)

print(f"List: {my_list}")
print(f"Average: {average}")
print(f"Median: {median}")
print(f"Mode: {mode}")

my_list_empty = []
average_empty = calculate_average(my_list_empty)
median_empty = calculate_median(my_list_empty)
mode_empty = calculate_mode(my_list_empty)

print(f"\nList: {my_list_empty}")
print(f"Average: {average_empty}")
print(f"Median: {median_empty}")
print(f"Mode: {mode_empty}")
```

```
print(f"Average: {average_empty}")
print(f"Median: {median_empty}")
print(f"Mode: {mode_empty}")
```

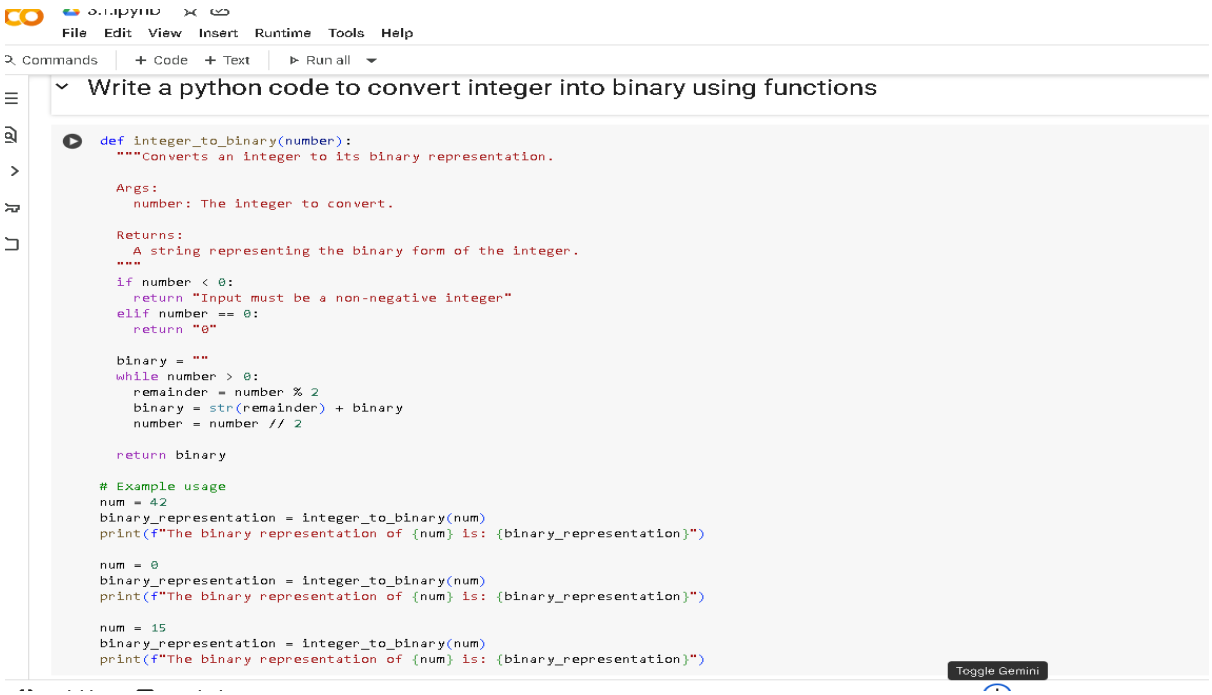
```
List: [1, 2, 3, 4, 5, 5, 6, 6, 6, 7]
Average: 4.5
Median: 5.0
Mode: [6]

List: []
Average: 0
Median: None
Mode: None
```

EXPLANATION :

This code defines three functions: calculate average, calculate median, and calculate mode for a list of numbers. It uses the Counter from the collections module to help find the mode. The code includes examples demonstrating how to use these functions with both a list of numbers and an empty list, and prints the calculated average, median, and mode for each case.

TASK -03



```
The binary representation of 42 is: 101010
The binary representation of 0 is: 0
The binary representation of 15 is: 1111
```

EXPLANATION:

1. The code defines a function integer_to_binary to convert a non-negative integer into its binary representation.
2. It handles base cases for negative numbers and zero.
3. For positive integers, it repeatedly takes the remainder when divided by 2 and builds the binary string.
4. The process continues until the number becomes zero.
5. The function returns the resulting binary string, and the code includes examples demonstrating its usage.

TASK – 04

write a python code to create an interface for hostel to generate bill based on custmer requirements

```python
# 1. Design the menu
rooms = {
    "single": 50,
    "double": 80,
    "suite": 150
}

services = {
    "laundry": 10,
    "wifi": 5,
    "breakfast": 15
}

print("Available Rooms:")
for room, price in rooms.items():
    print(f"- {room.title()}: ${price} per night")

print("\nAvailable Services:")
for service, price in services.items():
    print(f"- {service.title()}: ${price}")
```

```
        print(f"- {service.title()}: ${price}")
```

```
Available Rooms:
- Single: $50 per night
- Double: $80 per night
- Suite: $150 per night

Available Services:
- Laundry: $10
- Wifi: $5
- Breakfast: $15
```

EXPLANATION:

1. The code defines two dictionaries, rooms and services, to store the available room types and services with their respective prices.
2. It then prints a header "Available Rooms:".
3. A loop iterates through the rooms dictionary and prints each room type and its price per night.
4. Another header "Available Services:" is printed.
5. A second loop iterates through the services dictionary and prints each service and its price.

TASK – 05

Write a python code to convert temperature using functions

```
def celsius_to_fahrenheit(celsius):
  """Converts Celsius to Fahrenheit."""
  return (celsius * 9/5) + 32

def fahrenheit_to_celsius(fahrenheit):
  """Converts Fahrenheit to Celsius."""
  return (fahrenheit - 32) * 5/9

# Example usage
celsius_temp = 25
fahrenheit_temp = celsius_to_fahrenheit(celsius_temp)
print(f"{celsius_temp}°C is equal to {fahrenheit_temp:.2f}°F")

fahrenheit_temp = 77
celsius_temp = fahrenheit_to_celsius(fahrenheit_temp)
print(f"{fahrenheit_temp}°F is equal to {celsius_temp:.2f}°C")
```

```
25°C is equal to 77.00°F
77°F is equal to 25.00°C
```

EXPLANATION:

This code defines two functions to convert between Celsius and Fahrenheit temperatures. It then demonstrates their use with example temperature values and prints the conversion results.