LAB ASSIGNMENT – 3. 2

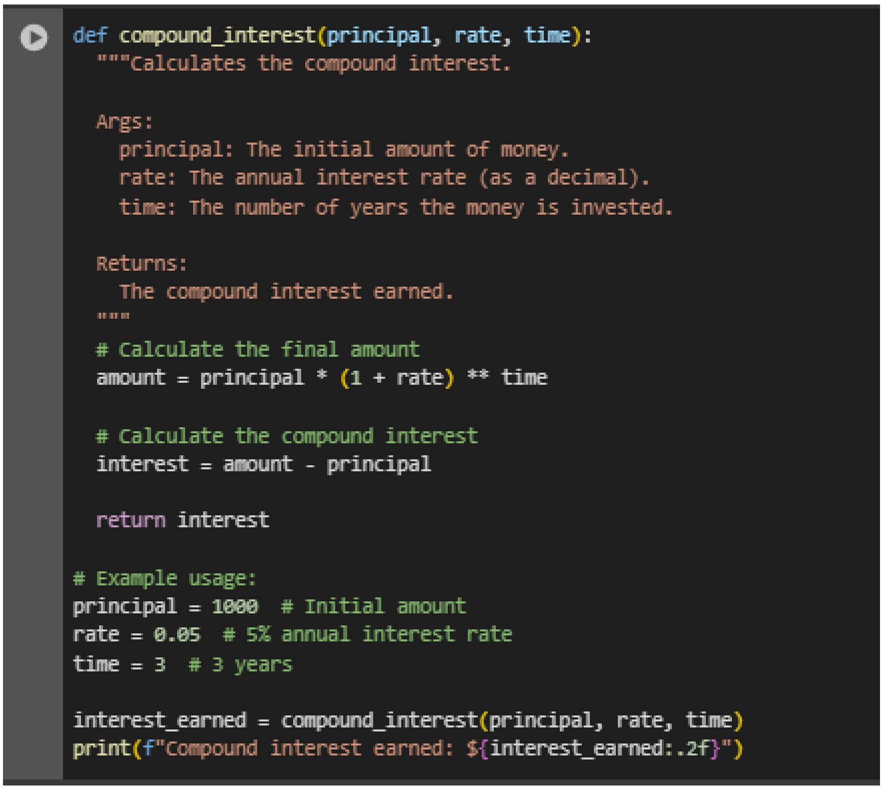
NAME : M.Shiva

HALL.NO : 2403A52377

BATCH.NO : AI 14

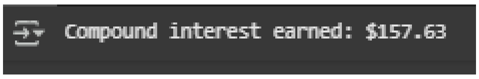
ASSIGNMENT : 3.2

PROMPT 01 :



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

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

  Returns:
    The compound interest earned.
  """
  # Calculate the final amount
  amount = principal * (1 + rate) ** time

  # Calculate the compound interest
  interest = amount - principal

  return interest

# Example usage:
principal = 1000  # Initial amount
rate = 0.05  # 5% annual interest rate
time = 3  # 3 years

interest_earned = compound_interest(principal, rate, time)
print(f"Compound interest earned: ${interest_earned:.2f}")
```

OUTPUT :



```
Compound interest earned: $157.63
```

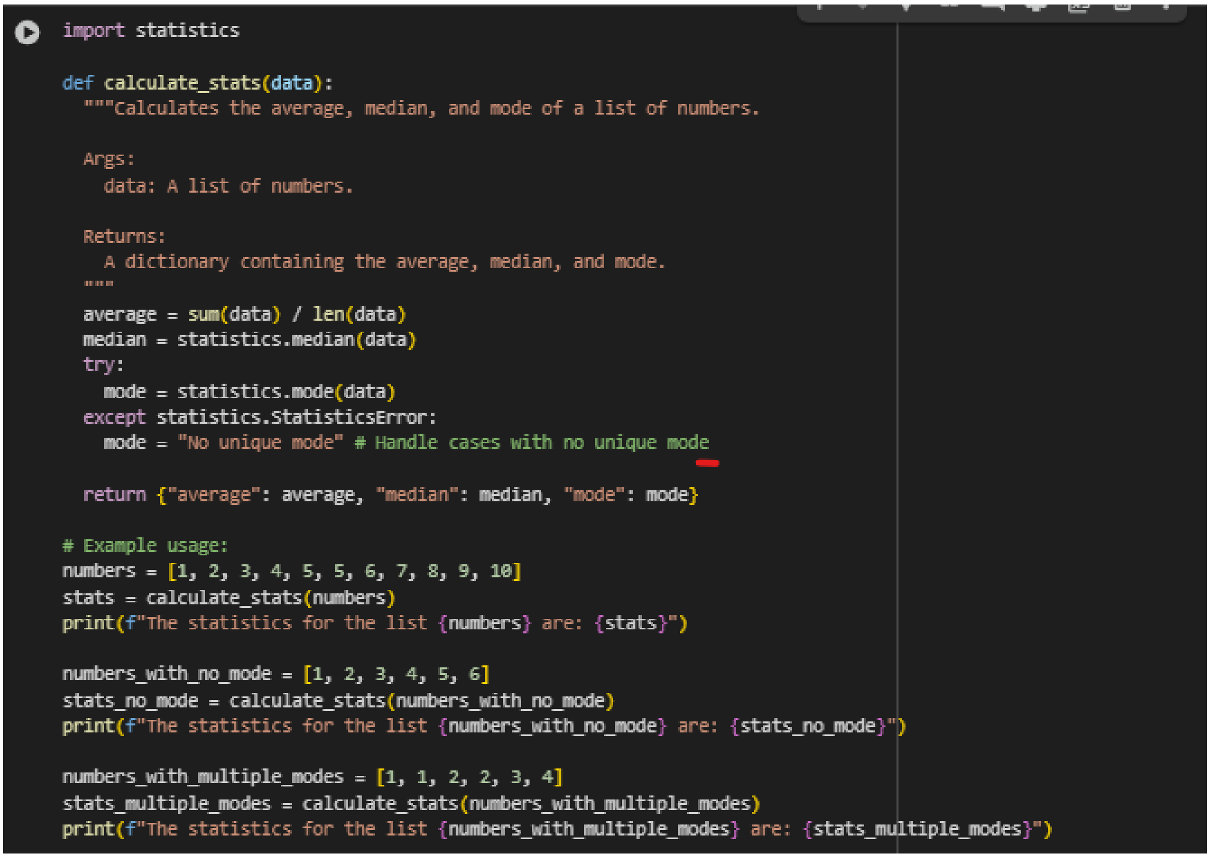
CODE EXPLANATION :

- def compound_interest(principal, rate, time):: This line defines a function named compound_interest that takes three arguments: principal, rate, and time.
- amount = principal * (1 + rate) ** time: Inside the function, this line calculates the final amount after applying compound interest. It uses the formula: Amount = Principal * (1 + Rate)^Time.
- interest = amount - principal: This line calculates the compound interest earned by subtracting the initial principal from the calculated amount.
- return interest: The function then returns the calculated interest.
- principal = 1000, rate = 0.05, time = 3: These lines set the values for the principal amount, interest rate (as a decimal), and time in years for the example usage.
- interest_earned = compound_interest(principal, rate, time): This line calls the compound_interest function with the specified values and stores the returned interest in the interest_earned variable.
- print(f"Compound interest earned: ${interest_earned:.2f}"): This line prints the calculated compound interest, formatted to two decimal places.

PROMPT 02 :

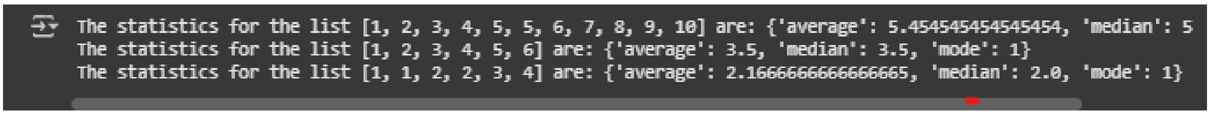
I NEED A PYHTON CODE , FUNCTION TO CALCULATE AVERAGE , MEDIAN AND MODE OF A LIST OF NUMBERS

CODE :



```python
import statistics

def calculate_stats(data):
  """Calculates the average, median, and mode of a list of numbers.

  Args:
    data: A list of numbers.

  Returns:
    A dictionary containing the average, median, and mode.
  """
  average = sum(data) / len(data)
  median = statistics.median(data)
  try:
    mode = statistics.mode(data)
  except statistics.StatisticsError:
    mode = "No unique mode" # Handle cases with no unique mode

  return {"average": average, "median": median, "mode": mode}

# Example usage:
numbers = [1, 2, 3, 4, 5, 5, 6, 7, 8, 9, 10]
stats = calculate_stats(numbers)
print(f"The statistics for the list {numbers} are: {stats}")

numbers_with_no_mode = [1, 2, 3, 4, 5, 6]
stats_no_mode = calculate_stats(numbers_with_no_mode)
print(f"The statistics for the list {numbers_with_no_mode} are: {stats_no_mode}")

numbers_with_multiple_modes = [1, 1, 2, 2, 3, 4]
stats_multiple_modes = calculate_stats(numbers_with_multiple_modes)
print(f"The statistics for the list {numbers_with_multiple_modes} are: {stats_multiple_modes}")
```

OUTPUT :



```
The statistics for the list [1, 2, 3, 4, 5, 5, 6, 7, 8, 9, 10] are: {'average': 5.454545454545454, 'median': 5
The statistics for the list [1, 2, 3, 4, 5, 6] are: {'average': 3.5, 'median': 3.5, 'mode': 1}
The statistics for the list [1, 1, 2, 2, 3, 4] are: {'average': 2.1666666666666665, 'median': 2.0, 'mode': 1}
```

CODE EXPLANATION :

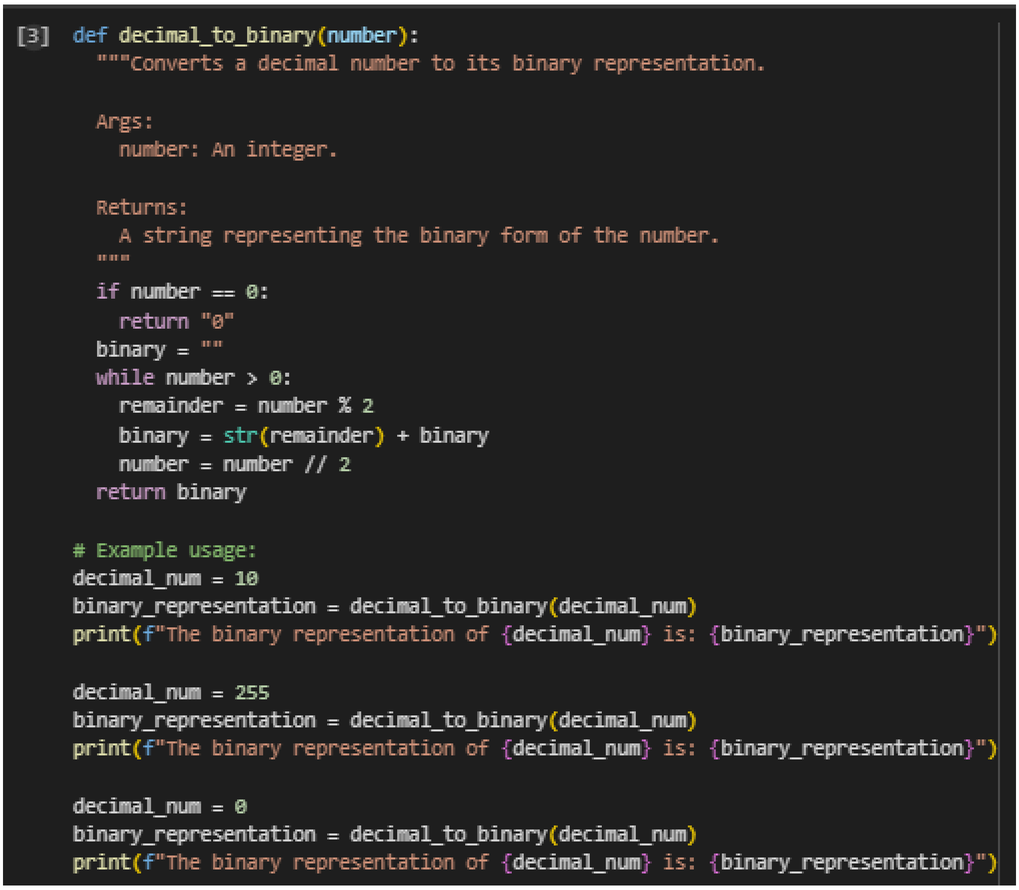
- import statistics: This line imports the statistics module, which provides functions for mathematical statistics.
- def calculate_stats(data):: This line defines a function named calculate_stats that takes one argument: data, which is expected to be a list of numbers.
- average = sum(data) / len(data): This line calculates the average (mean) of the numbers in the data list by dividing the sum of the elements by the number of elements.
- median = statistics.median(data): This line uses the median() function from the statistics module to calculate the median of the numbers in the data list.

- try...except statistics.StatisticsError:: This block attempts to calculate the mode using statistics.mode(data). The statistics.mode() function raises a StatisticsError if there is no unique mode (i.e., all values appear the same number of times or there are multiple values with the highest frequency).
- mode = statistics.mode(data): If a unique mode exists, this line calculates it using the mode() function.
- mode = "No unique mode": If a StatisticsError occurs (meaning no unique mode), this line sets the mode variable to the string "No unique mode".
- return {"average": average, "median": median, "mode": mode}: The function returns a dictionary containing the calculated average, median, and mode.
- Example Usage: The code then provides example lists (numbers, numbers_with_no_mode, numbers_with_multiple_modes) and calls the calculate_stats function with each list, printing the results.

PROMPT 03 :

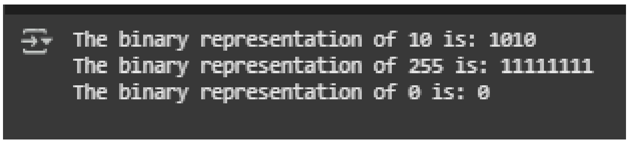
I NEED A PYHTON CODE , FUNCTION THAT CONVERTS THE NUMER INTO BINAY

CODE :



```python
def decimal_to_binary(number):
  """Converts a decimal number to its binary representation.

  Args:
    number: An integer.

  Returns:
    A string representing the binary form of the number.
  """
  if number == 0:
    return "0"
  binary = ""
  while number > 0:
    remainder = number % 2
    binary = str(remainder) + binary
    number = number // 2
  return binary

# Example usage:
decimal_num = 10
binary_representation = decimal_to_binary(decimal_num)
print(f"The binary representation of {decimal_num} is: {binary_representation}")

decimal_num = 255
binary_representation = decimal_to_binary(decimal_num)
print(f"The binary representation of {decimal_num} is: {binary_representation}")

decimal_num = 0
binary_representation = decimal_to_binary(decimal_num)
print(f"The binary representation of {decimal_num} is: {binary_representation}")
```

OUTPUT :



```
The binary representation of 10 is: 1010
The binary representation of 255 is: 11111111
The binary representation of 0 is: 0
```

CODE EXPLANATION :

- def decimal_to_binary(number):: This line defines a function called decimal_to_binary that takes one argument, number, which is the decimal integer you want to convert.
- if number == 0:: This is a base case. If the input number is 0, the binary representation is simply "0", so the function immediately returns "0".
- binary = "": Initializes an empty string called binary. This string will be used to build the binary representation.
- while number > 0:: This starts a loop that continues as long as the value of number is greater than 0.

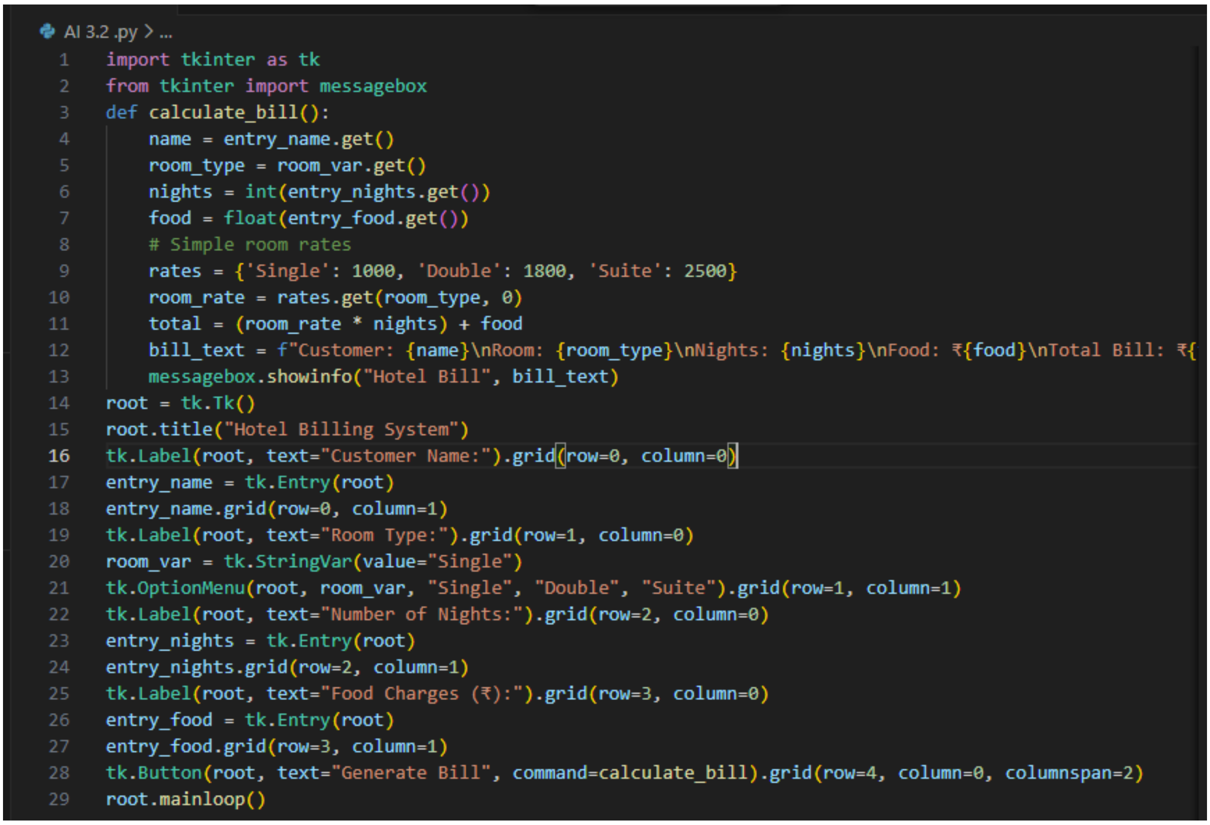
- remainder = number % 2: Inside the loop, this line calculates the remainder when number is divided by 2. In binary conversion, this remainder (either 0 or 1) is the next digit in the binary representation, starting from the rightmost digit.
- binary = str(remainder) + binary: This line converts the remainder (which is an integer) to a string and adds it to the *beginning* of the binary string. This is because we are calculating the binary digits from right to left.
- number = number // 2: This line updates the number by performing integer division by 2. This prepares for the next iteration of the loop to find the next binary digit.
- return binary: Once the while loop finishes (when number becomes 0), the function returns the binary string, which now contains the complete binary representation of the original decimal number.
- Example Usage: The code then shows how to use the function with different decimal numbers (10, 255, and 0) and prints the results.

PROMPT 04 :

I NEED A CODE IN PYTHON TO CREATE AN USER INTERFACE FOR AN HOTEL TO GENRATE BILL BASED ON CUSTOMER REQUIREMENTS , need just a simple code as a student

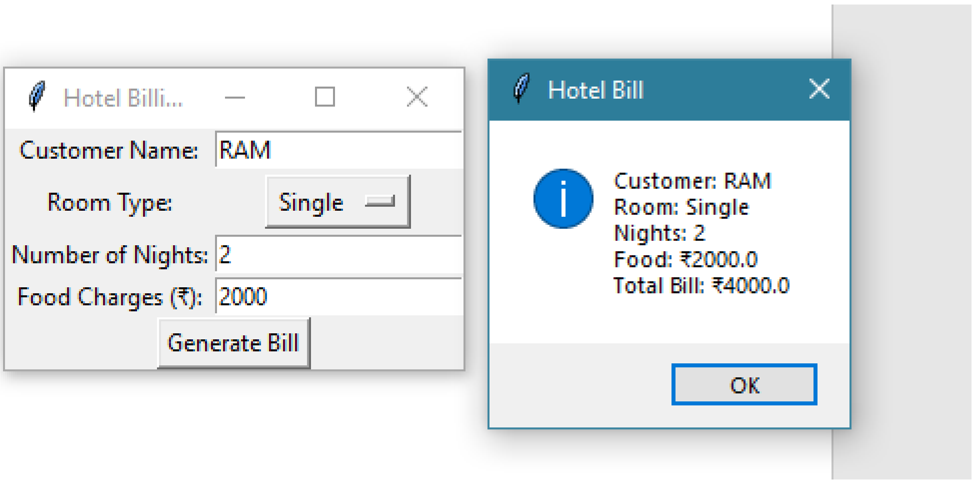
NOTE : the code is runned in vs code because , directly using libraries like tkinter for a graphical user interface is not supported in this Colab environment.

CODE :



```python
import tkinter as tk
from tkinter import messagebox
def calculate_bill():
    name = entry_name.get()
    room_type = room_var.get()
    nights = int(entry_nights.get())
    food = float(entry_food.get())
    # Simple room rates
    rates = {'Single': 1000, 'Double': 1800, 'Suite': 2500}
    room_rate = rates.get(room_type, 0)
    total = (room_rate * nights) + food
    bill_text = f"Customer: {name}\nRoom: {room_type}\nNights: {nights}\nFood: ₹{food}\nTotal Bill: ₹{
    messagebox.showinfo("Hotel Bill", bill_text)
root = tk.Tk()
root.title("Hotel Billing System")
tk.Label(root, text="Customer Name:").grid(row=0, column=0)
entry_name = tk.Entry(root)
entry_name.grid(row=0, column=1)
tk.Label(root, text="Room Type:").grid(row=1, column=0)
room_var = tk.StringVar(value="Single")
tk.OptionMenu(root, room_var, "Single", "Double", "Suite").grid(row=1, column=1)
tk.Label(root, text="Number of Nights:").grid(row=2, column=0)
entry_nights = tk.Entry(root)
entry_nights.grid(row=2, column=1)
tk.Label(root, text="Food Charges (₹):").grid(row=3, column=0)
entry_food = tk.Entry(root)
entry_food.grid(row=3, column=1)
tk.Button(root, text="Generate Bill", command=calculate_bill).grid(row=4, column=0, columnspan=2)
root.mainloop()
```

OUTPUT :



Customer Name: RAM
Room Type: Single
Number of Nights: 2
Food Charges (₹): 2000

```
Customer: RAM
Room: Single
Nights: 2
Food: ₹2000.0
Total Bill: ₹4000.0
```

CODE EXPLANATION :

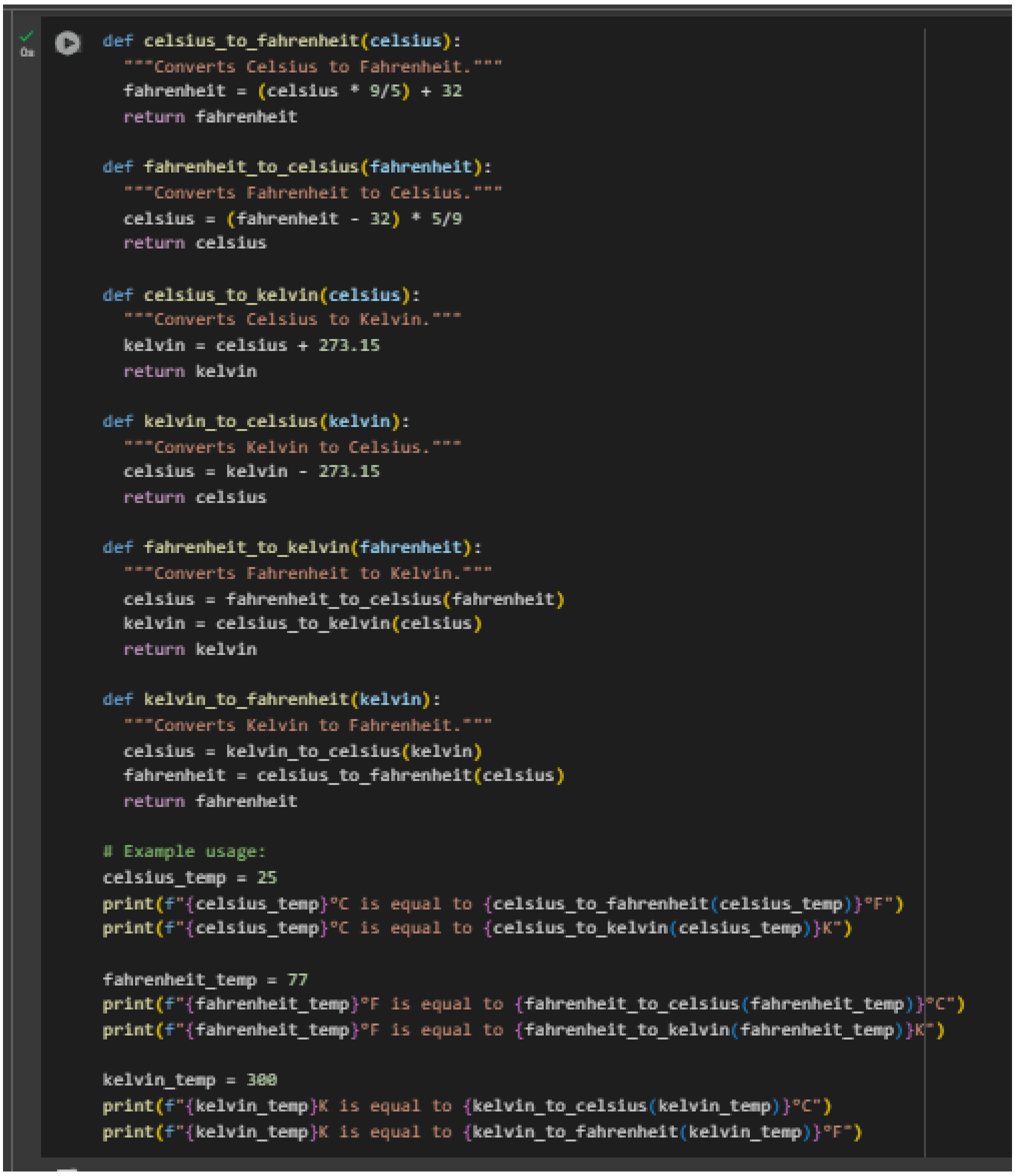
1. Imports tkinter modules:
   - Imports tkinter for GUI and messagebox for pop-up dialogs.

2. Defines calculate_bill function:
    - Gets user inputs (name, room type, nights, food charges).
    - Uses a dictionary for room rates.
    - Calculates total bill: (room rate × nights) + food charges.
    - Shows the bill in a pop-up message box.
3. Creates main window:
    - Sets the window title to "Hotel Billing System".
4. Adds input fields:
    - Customer name (Entry box).
    - Room type (Dropdown: Single, Double, Suite).
    - Number of nights (Entry box).
    - Food charges (Entry box).
5. Adds button:
    - "Generate Bill" button calls calculate_bill when clicked.
6. Runs the GUI loop:
    - root.mainloop() keeps the window open and responsive.

PROMPT 05 :

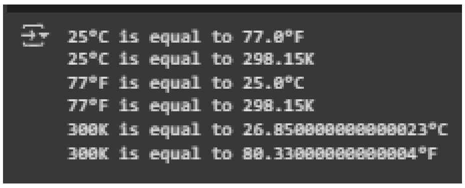
I NEED A PYTHON CODE THAT CONVERTS CELCIUS , FAREN HEIT , KELVIN INTO THE INSTRUCTUION GIVEN

CODE :



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

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

def celsius_to_kelvin(celsius):
  """Converts Celsius to Kelvin."""
  kelvin = celsius + 273.15
  return kelvin

def kelvin_to_celsius(kelvin):
  """Converts Kelvin to Celsius."""
  celsius = kelvin - 273.15
  return celsius

def fahrenheit_to_kelvin(fahrenheit):
  """Converts Fahrenheit to Kelvin."""
  celsius = fahrenheit_to_celsius(fahrenheit)
  kelvin = celsius_to_kelvin(celsius)
  return kelvin

def kelvin_to_fahrenheit(kelvin):
  """Converts Kelvin to Fahrenheit."""
  celsius = kelvin_to_celsius(kelvin)
  fahrenheit = celsius_to_fahrenheit(celsius)
  return fahrenheit

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

fahrenheit_temp = 77
print(f"{fahrenheit_temp}°F is equal to {fahrenheit_to_celsius(fahrenheit_temp)}°C")
print(f"{fahrenheit_temp}°F is equal to {fahrenheit_to_kelvin(fahrenheit_temp)}K")

kelvin_temp = 300
print(f"{kelvin_temp}K is equal to {kelvin_to_celsius(kelvin_temp)}°C")
print(f"{kelvin_temp}K is equal to {kelvin_to_fahrenheit(kelvin_temp)}°F")
```

OUTPUT :



```
25°C is equal to 77.0°F
25°C is equal to 298.15K
77°F is equal to 25.0°C
77°F is equal to 298.15K
300K is equal to 26.850000000000023°C
300K is equal to 80.33000000000004°F
```

CODE EXPLANATION :

- **celsius_to_fahrenheit(celsius)**: Converts a temperature from Celsius to Fahrenheit using the formula (celsius * 9/5) + 32.
- **fahrenheit_to_celsius(fahrenheit)**: Converts a temperature from Fahrenheit to Celsius using the formula (fahrenheit - 32) * 5/9.
- **celsius_to_kelvin(celsius)**: Converts a temperature from Celsius to Kelvin by adding 273.15.
- **kelvin_to_celsius(kelvin)**: Converts a temperature from Kelvin to Celsius by subtracting 273.15.
- **fahrenheit_to_kelvin(fahrenheit)**: Converts a temperature from Fahrenheit to Kelvin by first converting Fahrenheit to Celsius and then Celsius to Kelvin.
- **kelvin_to_fahrenheit(kelvin)**: Converts a temperature from Kelvin to Fahrenheit by first converting Kelvin to Celsius and then Celsius to Fahrenheit.
- **Example Usage**: The code then demonstrates how to use these functions with example temperatures for each conversion type, printing the results.