LAB ASSIGNMENT: 03

TASK:01

PROMPT:

Write a python program to create a function to calculate compound interest, starting with only the function name and give the total amount and interest amount and the amount after the interest added to it with inputs.

CODE:

C: > Users > sahit > New folder (2) > 2403a54070 > interest.py > calculate_compound_interest

```python
#Ask the AI to write a function to calculate compound interest, starting with only the function

def calculate_compound_interest(principal, rate, time, n):
    """
    Calculate compound interest.

    Args:
        principal (float): The initial amount of money.
        rate (float): Annual interest rate (in percent).
        time (float): Time in years.
        n (int): Number of times interest applied per year.

    Returns:
        dict: {
            'total_amount': total amount after interest,
            'interest_amount': interest earned,
            'amount_after_interest': total amount after interest
        }
    """
    # Convert rate from percent to decimal
    r = rate / 100
    # Compound interest formula
    total_amount = principal * (1 + r / n) ** (n * time)
    interest_amount = total_amount - principal
    return {
        'total_amount': total_amount,
        'interest_amount': interest_amount,
        'amount_after_interest': total_amount
    }

# Example usage:
principal = float(input("Enter principal amount: "))
rate = float(input("Enter annual interest rate (in %): "))
```

```python
    }

# Example usage:
principal = float(input("Enter principal amount: "))
rate = float(input("Enter annual interest rate (in %): "))
time = float(input("Enter time in years: "))
n = int(input("Enter number of times interest applied per year: "))

result = calculate_compound_interest(principal, rate, time, n)
print(f"Total Amount: {result['total_amount']:.2f}")
print(f"Interest Amount: {result['interest_amount']:.2f}")
# Print the amount after interest is added (same as total amount)
print(f"Amount After Interest: {result['amount_after_interest']:.2f}")
```

OUTPUT:

```
on\Python313\python.exe "c:/Users/sahit/New folder (2)/2403a54070/interest.py"
Enter principal amount: 10000
Enter annual interest rate (in %): 5
Enter time in years: 8
Enter number of times interest applied per year: 6
Total Amount: 14893.54
Interest Amount: 4893.54
Amount After Interest: 14893.54
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code>
```

EXPLANATION:

This Python function calculates compound interest by taking in four inputs: the initial principal amount, the annual interest rate (as a percentage), the time in years, and how many times the interest is compounded per year. It first converts the interest rate into decimal form, then applies the compound interest formula to compute the total amount after interest is applied over time. The interest earned is found by subtracting the original principal from the total amount. Finally, the function returns a dictionary containing the total amount, the interest earned, and the amount after interest—making it easy to access and display each result clearly.

TASK:2

PROMPT:

Write a python program to calculate the average, median, and mode of the given list of numbers.

CODE:

```python

# Program to calculate the average, median, and mode of a given list of numbers

import statistics

def calculate_stats(numbers):
    average = sum(numbers) / len(numbers)
    median = statistics.median(numbers)
    try:
        mode = statistics.mode(numbers)
    except statistics.StatisticsError:
        mode = "No unique mode"
    return average, median, mode

# Example usage:
numbers = [float(x) for x in input("Enter numbers separated by spaces: ").split()]
average, median, mode = calculate_stats(numbers)
print(f"Average: {average}")
print(f"Median: {median}")
print(f"Mode: {mode}")
```

OUTPUT:

```
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code> & C:\Users\sahit\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/sahit/New folder (2)/2403a54070/average.py"
Enter numbers separated by spaces: 2 4 5 4 6
Average: 4.2
Median: 4.0
Mode: 4.0
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code>
```

Ln 6, Col 30 Spaces: 4

EXPLANATION:

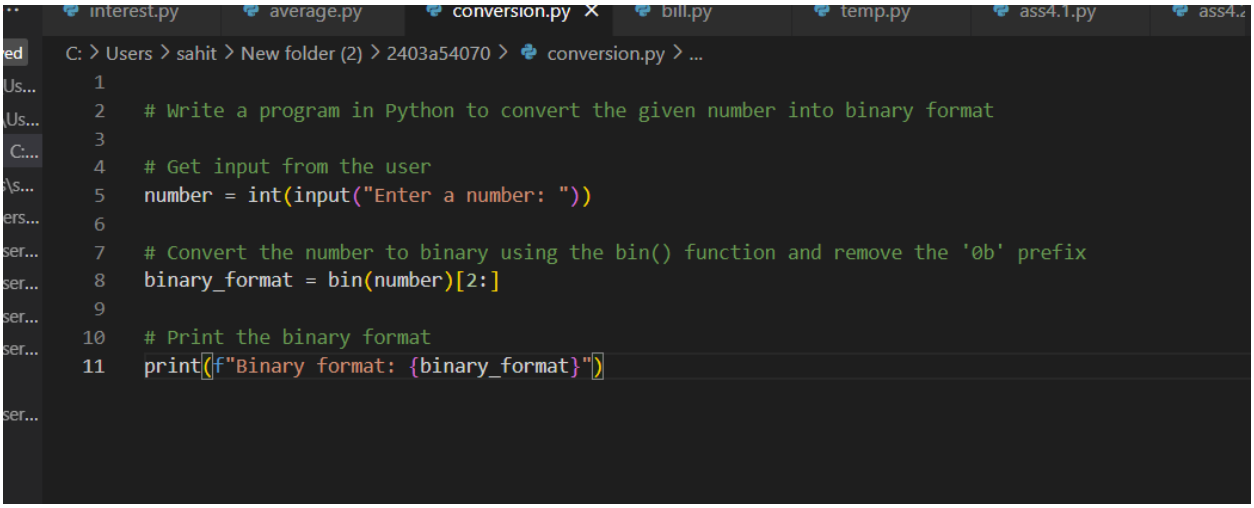
This Python program calculates three key statistical measures—average, median, and mode—from a list of numbers entered by the user. It first converts the input string into a list of floating-point numbers. The `calculate_stats` function then computes the average by summing all the numbers and dividing by how many there are. It uses Python's `statistics` module to find the median, which is the middle value when the list is sorted. For the mode, which is the most frequently occurring number, it uses a try-except block to handle cases where no unique mode exists (like when all numbers appear equally). The results are printed clearly, giving a quick snapshot of the dataset's central tendencies. Want to add standard deviation or visualize the data next?

TASK:03

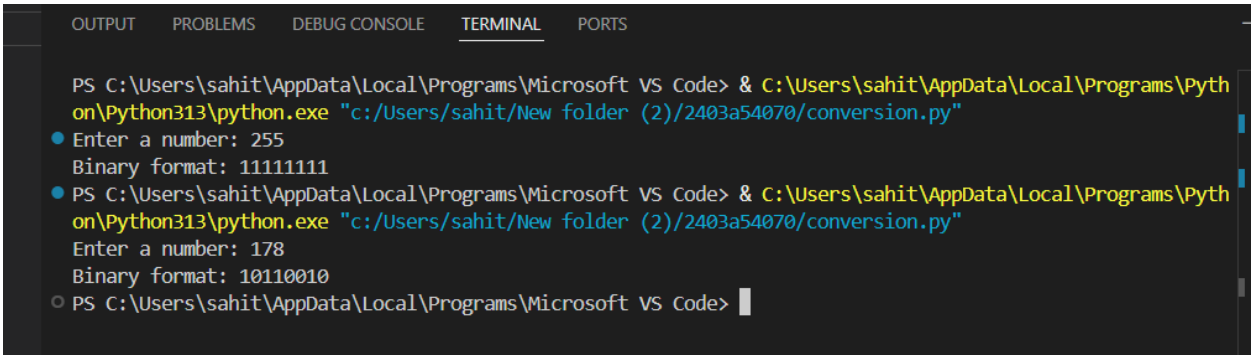
PROMPT:

Write a python program to convert the given number into binary format.

CODE:



OUTPUT:



EXPLANATION:

Nice and clean, Banda! This Python program takes a number from the user and converts it into its binary representation. Here's a quick breakdown:

- `input()` prompts the user to enter a number, which is then converted to an integer using `int()`.
- `bin(number)` converts the number to binary, but it includes a `'0b'` prefix to indicate it's binary.
- `[2:]` slices off the first two characters (`'0b'`) so you're left with just the binary digits.
- Finally, `print()` displays the result in a readable format.

TASK:4

PROMPT:

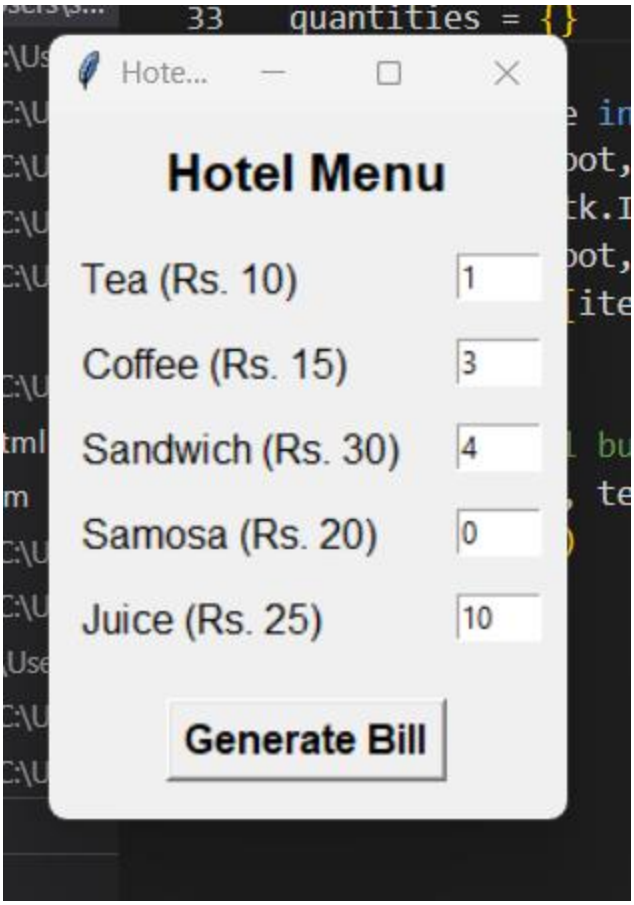
Write a python program to create an user interface for an hotel to generate bill on based on the hotel menu with some items like tea coffee with price beside it with the graphical user interface.

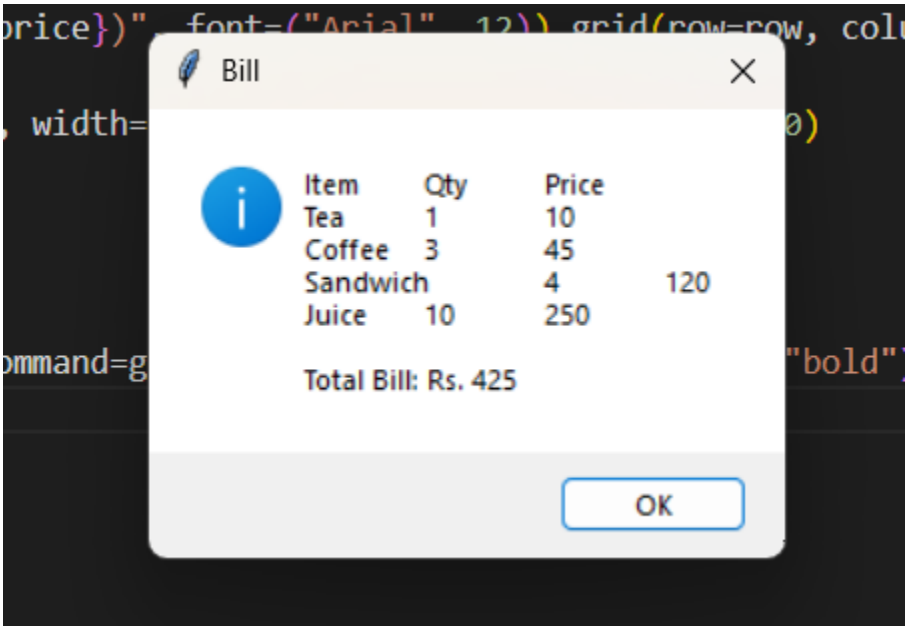
CODE:

```python
import tkinter as tk
from tkinter import messagebox

# Hotel menu with prices
menu = {
    "Tea": 10,
    "Coffee": 15,
    "Sandwich": 30,
    "Samosa": 20,
    "Juice": 25
}

def generate_bill():
    total = 0
    bill_details = "Item\tQty\tPrice\n"
    for item, var in quantities.items():
        qty = var.get()
        if qty > 0:
            price = menu[item] * qty
            bill_details += f"{item}\t{qty}\t{price}\n"
            total += price
    bill_details += f"\nTotal Bill: Rs. {total}"
    messagebox.showinfo("Bill", bill_details)

# Create main window
root = tk.Tk()
root.title("Hotel Bill Generator")

# Heading
tk.Label(root, text="Hotel Menu", font=("Arial", 16, "bold")).grid(row=0, column=0, columnspan=3
```

```python
# Heading
tk.Label(root, text="Hotel Menu", font=("Arial", 16, "bold")).grid(row=0, column=0, columnspan=3, pady=10)

# Menu display and quantity input
quantities = {}
row = 1
for item, price in menu.items():
    tk.Label(root, text=f"{item} (Rs. {price})", font=("Arial", 12)).grid(row=row, column=0, padx=10, pady=5, stic
    qty_var = tk.IntVar()
    tk.Entry(root, textvariable=qty_var, width=5).grid(row=row, column=1, padx=10)
    quantities[item] = qty_var
    row += 1

# Generate Bill button
tk.Button(root, text="Generate Bill", command=generate_bill, font=("Arial", 12, "bold")).grid(row=row, column=0, c
root.mainloop()
```

OUTPUT:





EXPLANATION:

This Python script uses the Tkinter library to create a simple GUI-based hotel bill generator. It defines a menu with item names and prices, then dynamically displays each item with an input field for quantity. When the "Generate Bill" button is clicked, the `generate_bill()` function calculates the total cost based on user input, formats the bill with itemized details, and displays it in a message box. The GUI includes a heading, item labels, entry fields for quantities, and a button to trigger bill generation. The use of `IntVar()` allows real-time tracking of input values, and the layout is managed using the grid system for clean alignment.

TASK:5

PROMPT:

Write a Python program that takes a temperature input from the user in Celsius and converts it to Fahrenheit and Kelvin.

CODE:

```python

def fahrenheit_to_celsius(f):
    return (f - 32) * 5 / 9

def celsius_to_fahrenheit(c):
    return (c * 9 / 5) + 32

print("Temperature Converter")
print("1. Fahrenheit to Celsius")
print("2. Celsius to Fahrenheit")
choice = input("Enter your choice (1 or 2): ")

if choice == "1":
    f = float(input("Enter temperature in Fahrenheit: "))
    c = fahrenheit_to_celsius(f)
    print(f"{f}°F is {c:.2f}°C")
elif choice == "2":
    c = float(input("Enter temperature in Celsius: "))
    f = celsius_to_fahrenheit(c)
    print(f"{c}°C is {f:.2f}°F")
else:
    print("Invalid choice.")
```

OUTPUT:

```
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code> & C:\Users\sahit\AppData\Local\Programs\
Python\Python313\python.exe "c:/Users/sahit/New folder (2)/2403a54070/temp.py"
Temperature Converter
1. Fahrenheit to Celsius
2. Celsius to Fahrenheit
Enter your choice (1 or 2): 2
Enter temperature in Celsius: 26
26.0°C is 78.80°F
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code>
```

EXPLANATION:

This Python program is a simple temperature converter that allows the user to convert between Fahrenheit and Celsius. It begins by defining two functions: `fahrenheit_to_celsius(f)` and `celsius_to_fahrenheit(c)`, which use standard formulas to perform the conversions. The program then displays a menu with two options and prompts the user to choose either Fahrenheit to Celsius or Celsius to Fahrenheit conversion. Based on the user's input (`choice`), it asks for the corresponding temperature value, performs the conversion using the appropriate function, and prints the result formatted to two decimal places. If the user enters an invalid option, the program gracefully handles it by displaying an "Invalid choice" message. The structure is clean and modular, making it easy to understand and extend—ideal for beginners learning about functions, user input, and conditional logic.