**ASSIGNMENT-3**

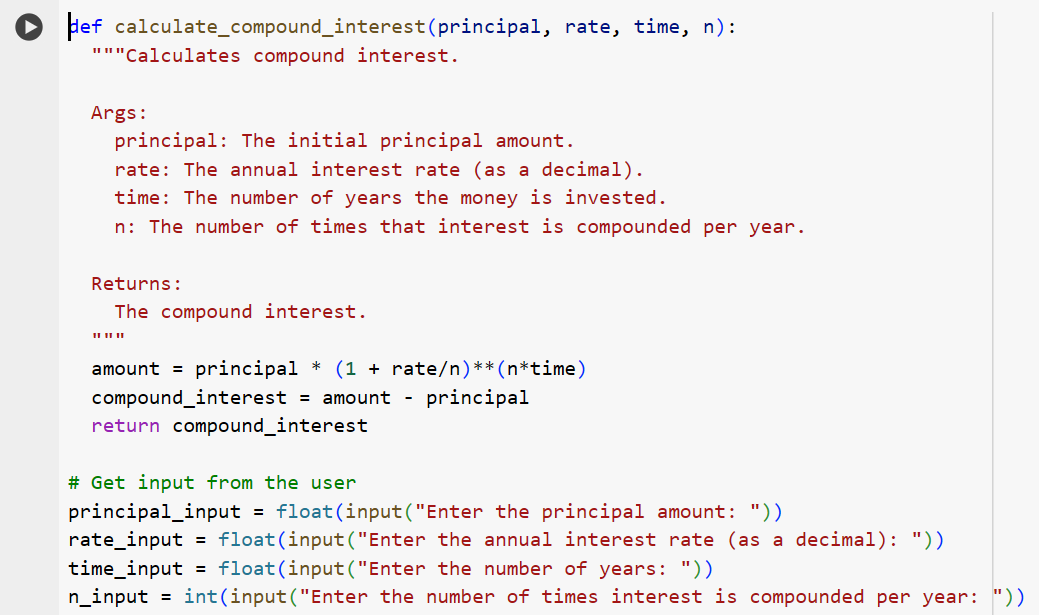
**Name:** V. Vivek vardhan

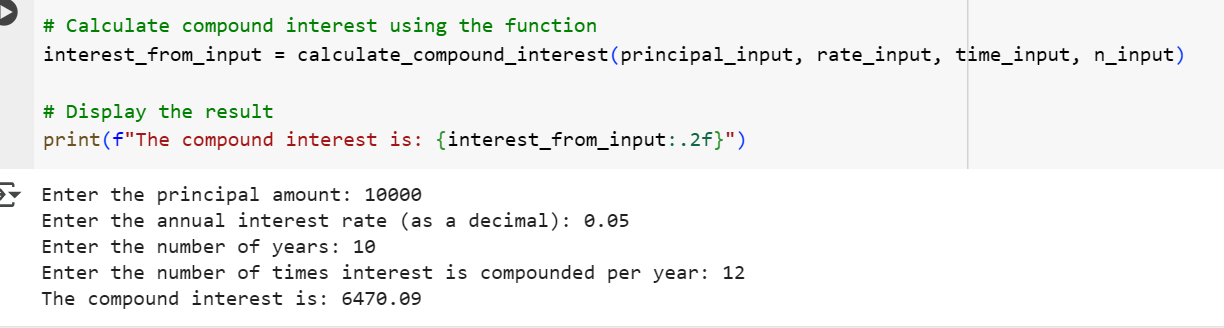
**HT NO:**2403A52097

**BATCH:**24BTCAIAIB05

**TASK-1:** Write a program in python to calculate compound interest using functions**.**

**CODE AND OUTPUT:**

****

****

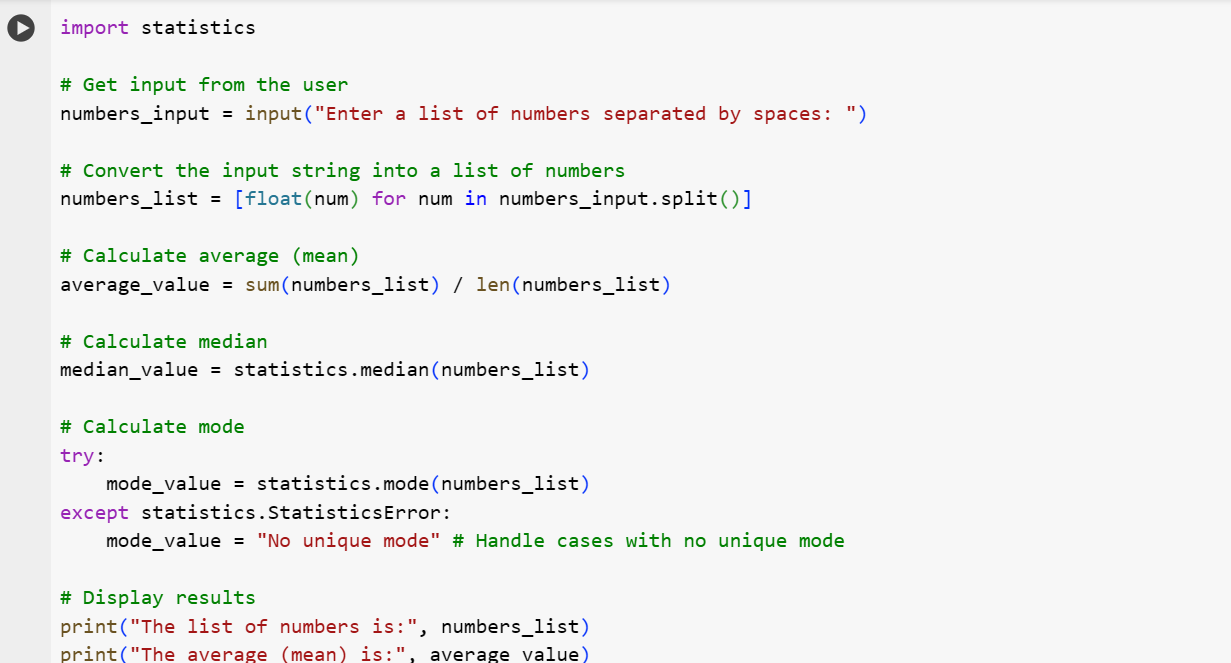
**EXPLANATION:**

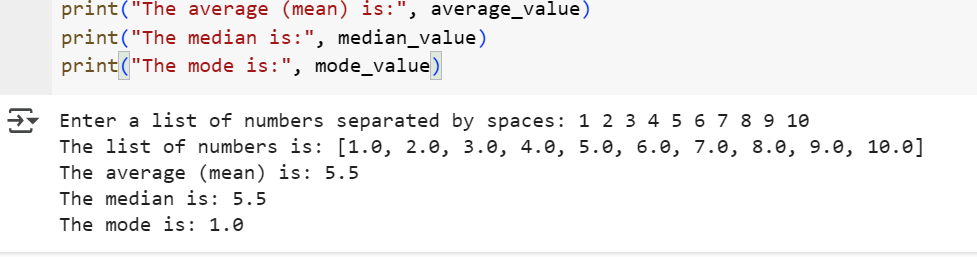
1. **calculate\_compound\_interest(principal, rate, time, n) function:**
   * This function takes four arguments: principal (initial amount), rate (annual interest rate as a decimal), time (number of years), and n (number of times interest is compounded per year).
   * It calculates the future value of the investment using the compound interest formula: amount = principal \* (1 + rate/n)\*\*(n\*time).
   * It then calculates the compound interest by subtracting the initial principal from the future value: compound\_interest = amount - principal.
   * Finally, it returns the calculated compound\_interest.
2. **Getting user input:**
   * The code prompts the user to enter the principal amount, annual interest rate, number of years, and the number of times interest is compounded per year using the input() function.
   * float() is used to convert the principal, rate, and time inputs to floating-point numbers, and int() is used to convert the compounding frequency (n) to an integer.
3. **Calculating and displaying the result:**
   * The calculate\_compound\_interest function is called with the user-provided inputs to get the compound interest.
   * The result is stored in the interest\_from\_input variable.
   * Finally, the calculated compound interest is printed to the console using an f-string, formatted to two decimal places (:.2f).

**TASK-2:**

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

**CODE AND OUTPUT:**

****

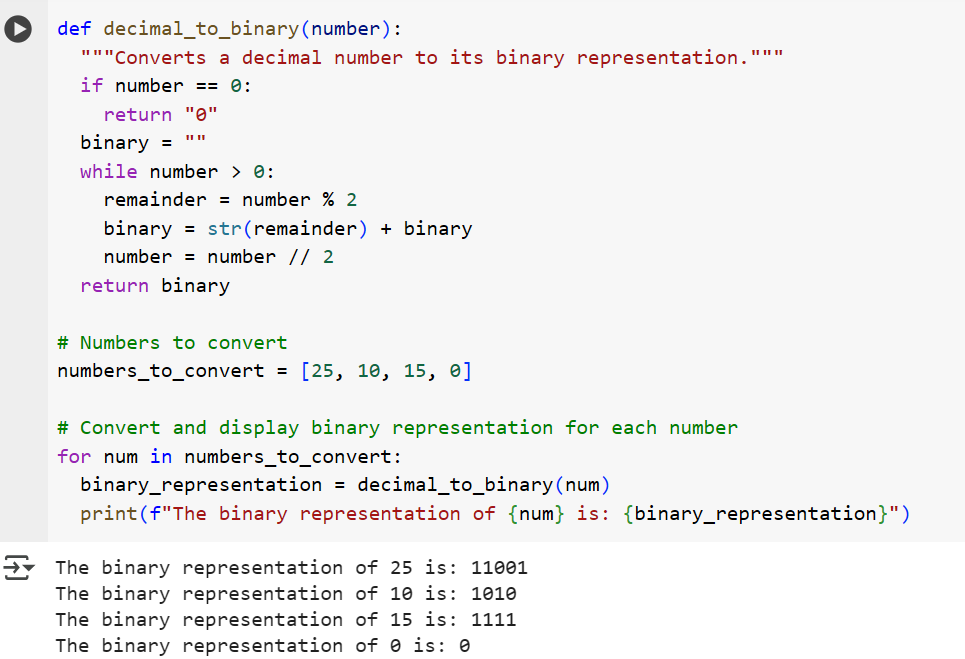
****

**EXPLANATION:**

1. **import statistics**: This line imports the statistics module, which provides functions for calculating common statistical measures like median and mode.
2. **numbers\_input = input(...)**: This prompts the user to enter a list of numbers separated by spaces and stores the input as a string in the numbers\_input variable.
3. **numbers\_list = [float(num) for num in numbers\_input.split()]**: This line does two things:
   * numbers\_input.split(): This splits the input string into a list of individual strings, using spaces as the delimiter.
   * [float(num) for num in ... ]: This is a list comprehension that iterates through each string in the split list (num) and converts it to a floating-point number using float(). These floating-point numbers are then collected into a new list called numbers\_list.
4. **average\_value = sum(numbers\_list) / len(numbers\_list)**: This calculates the average (mean) by summing all the numbers in numbers\_list using sum() and dividing by the total count of numbers using len().
5. **median\_value = statistics.median(numbers\_list)**: This uses the median() function from the imported statistics module to calculate the median of the numbers in numbers\_list. The median is the middle value in a sorted list of numbers.
6. **try...except statistics.StatisticsError**: This block calculates the mode:
   * **mode\_value = statistics.mode(numbers\_list)**: This uses the mode() function from the statistics module to calculate the mode, which is the most frequent number in the list.
   * **except statistics.StatisticsError:**: This handles the case where there is no unique mode (e.g., if all numbers appear the same number of times, or if there are multiple numbers with the same highest frequency).
   * **mode\_value = "No unique mode"**: If a StatisticsError occurs, the mode\_value is set to the string "No unique mode".
7. **print(...) statements**: These lines display the original list of numbers and the calculated average, median, and mode.

**TASK-3:** Write a python program to convert numbers into binary using functions.

**CODE AND EXPLANATION:**

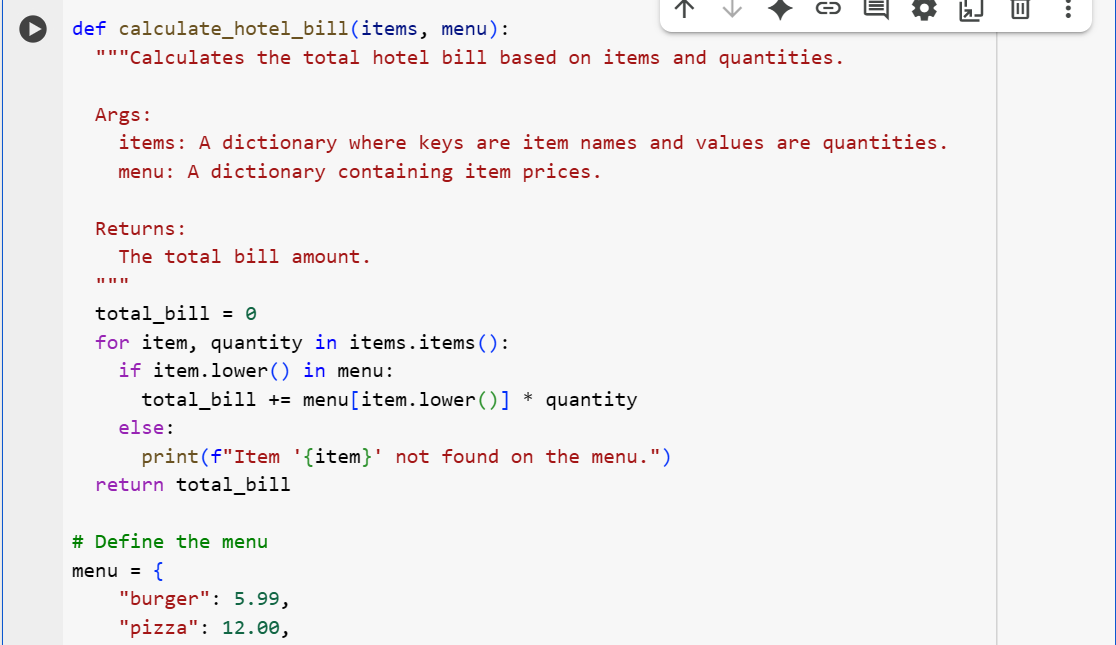
****

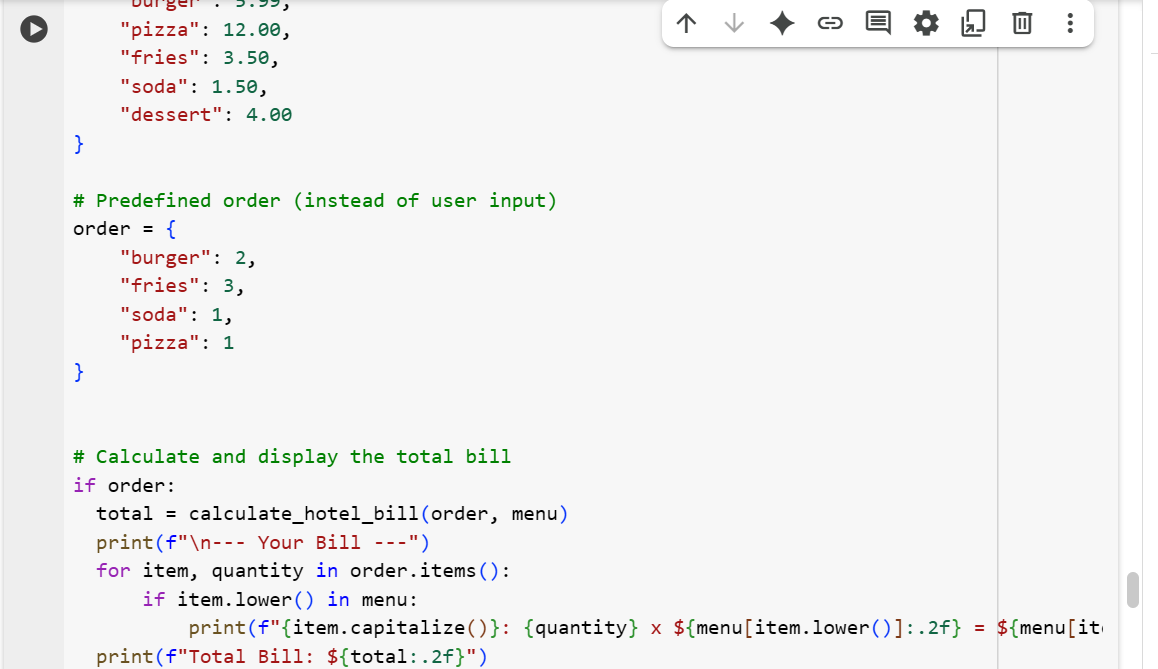
**EXPLANATION:**

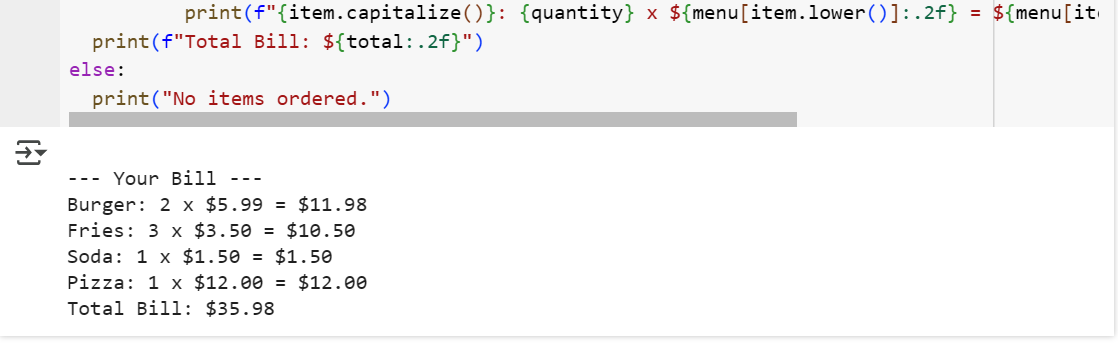
1. **def decimal\_to\_binary(number):**: This line defines a function named decimal\_to\_binary that takes one argument, number, which is the decimal integer you want to convert.
2. **"""Converts a decimal number to its binary representation."""**: This is a docstring that explains what the function does.
3. **if number == 0:**: This checks if the input number is 0. If it is, the binary representation is simply "0", so the function returns "0".
4. **binary = ""**: Initializes an empty string called binary which will store the binary digits as they are calculated.
5. **while number > 0:**: This loop continues as long as the number is greater than 0.
6. **remainder = number % 2**: This calculates the remainder when the number is divided by 2. This remainder will be either 0 or 1, which are the binary digits.
7. **binary = str(remainder) + binary**: This takes the remainder (converted to a string) and adds it to the beginning of the binary string. This is how the binary digits are collected in the correct order.
8. **number = number // 2**: This performs integer division of the number by 2, effectively moving to the next bit position in the binary conversion.
9. **return binary**: Once the while loop finishes (when number becomes 0), the function returns the accumulated binary string.
10. **numbers\_to\_convert = [25, 10, 15, 0]**: This line creates a list named numbers\_to\_convert containing the decimal numbers that will be converted to binary.
11. **for num in numbers\_to\_convert:**: This loop iterates through each number in the numbers\_to\_convert list.
12. **binary\_representation = decimal\_to\_binary(num)**: Inside the loop, for each num, the decimal\_to\_binary function is called to get its binary representation, which is stored in the binary\_representation variable.
13. **print(f"The binary representation of {num} is: {binary\_representation}")**: This line prints the original decimal number and its calculated binary representation using an f-string for formatted output.

**TASK-4:** Write a python program for hotel bill using items,quantity and bill using functions

**CODE AND OUTPUT:**

****

****

****

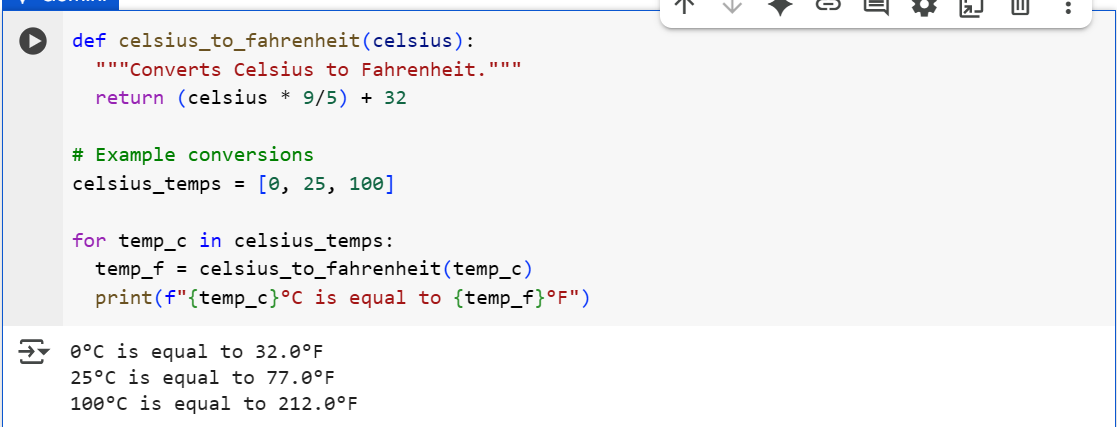
**EXPLANATION:**

1. **def calculate\_hotel\_bill(items, menu):**: This defines a function named calculate\_hotel\_bill that takes two arguments: items (a dictionary representing the order with item names as keys and quantities as values) and menu (a dictionary with item names as keys and prices as values).
2. **"""Calculates the total hotel bill based on items and quantities."""**: This is a docstring explaining the function's purpose.
3. **total\_bill = 0**: Initializes a variable total\_bill to 0 to keep track of the total cost.
4. **for item, quantity in items.items():**: This loop iterates through each item and its corresponding quantity in the items dictionary (the order).
5. **if item.lower() in menu:**: This checks if the item name (converted to lowercase) exists in the menu dictionary. This makes the item lookup case-insensitive.
6. **total\_bill += menu[item.lower()] \* quantity**: If the item is found in the menu, its price is retrieved from the menu dictionary, multiplied by the quantity, and added to the total\_bill.
7. **else: print(f"Item '{item}' not found on the menu.")**: If an item in the order is not found in the menu, a message is printed indicating this.
8. **return total\_bill**: The function returns the final calculated total\_bill.
9. **menu = {...}**: This defines the menu dictionary outside the function, making it accessible globally. It contains the prices for different food and drink items.
10. **order = {...}**: This defines the order dictionary, also outside the function. This dictionary represents a predefined customer order with item names and their quantities.
11. **if order:**: This checks if the order dictionary is not empty.
12. **total = calculate\_hotel\_bill(order, menu)**: If there are items in the order, the calculate\_hotel\_bill function is called with the order and menu dictionaries to get the total bill.
13. **print(f"\n--- Your Bill ---")**: Prints a header for the bill.
14. **for item, quantity in order.items(): ...**: This loop iterates through the items in the order to print an itemized breakdown of the bill.
15. **if item.lower() in menu:**: Again, this checks if the item is in the menu before trying to access its price.
16. **print(f"{item.capitalize()}: {quantity} x ${menu[item.lower()]:.2f} = ${menu[item.lower()] \* quantity:.2f}")**: This prints each item, its quantity, unit price, and the subtotal for that item, formatted to two decimal places. item.capitalize() is used to print the item name with the first letter capitalized.
17. **print(f"Total Bill: ${total:.2f}")**: Prints the final calculated total bill, formatted to two decimal places.
18. **else: print("No items ordered.")**: If the order dictionary is empty, this message is printed.

**TASK-5:**

Compare how different prompts affect the quality of code output, using a simple function (like temperature conversion) as an example. give me simpler and shorter code.

**CODE AND OUTPUT:**

****

**EXPLANATION:**

* **Basic Request (Prompt 1 & Cell b7e2bf61):** A simple prompt asking for a function results in a basic function without examples, documentation, or error handling.
* **Requesting Examples (Prompt 2 & Cell 2e4ed7ba):** Asking for examples leads to code that includes the function along with sample usage, making it easier to understand how to use the function.
* **Requesting Robustness and Documentation (Prompt 3 & Cell 38f85df3):** A prompt specifying the need for robustness and documentation yields a function with a docstring explaining its purpose, arguments, and potential errors, and includes basic input validation to handle invalid inputs.