Name –Agulla Aishwarya

Roll – 2403A54128

Batch – 04 (DS)

Task Description#1

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

PROMPT : Write a function in python code which takes input from user to calculate compound interest

```
[ ]  Start coding or generate with AI.

37s
def calculate_compound_interest():
    """
    Calculates compound interest based on user input.
    """
    principal = float(input("Enter the principal amount: "))
    rate = float(input("Enter the annual interest rate (as a decimal): "))
    time = float(input("Enter the number of years: "))
    compounding_periods = int(input("Enter the number of times interest is compounded per year: "))

    amount = principal * (1 + rate / compounding_periods)**(compounding_periods * time)
    compound_interest = amount - principal

    print(f"The final amount after compound interest is: {amount:.2f}")
    print(f"The compound interest earned is: {compound_interest:.2f}")

# Call the function to get user input and calculate the interest
calculate_compound_interest()

Enter the principal amount: 100000
Enter the annual interest rate (as a decimal): 2
Enter the number of years: 4
Enter the number of times interest is compounded per year: 1
The final amount after compound interest is: 8100000.00
The compound interest earned is: 8000000.00
```

EXPLAINATION:

This Python code defines a function called `calculate_compound_interest`. Here's a breakdown of what it does:

1. **Function Definition:** `def calculate_compound_interest():` defines a function named `calculate_compound_interest`.
2. **Docstring:** The triple-quoted string inside the function is a docstring, explaining that the function calculates compound interest based on user input.
3. **Get User Input:** The lines using `input()` prompt the user to enter the principal amount, annual interest rate (as a decimal), number of years, and the number of times interest is compounded per year. `float()` and `int()` are used to convert the input strings into numerical types (floating-point numbers and integers, respectively).
4. **Calculate Amount:** `amount = principal * (1 + rate / compounding_periods)**(compounding_periods * time)` calculates the final amount after compound interest using the compound interest formula.
5. **Calculate Compound Interest:** `compound_interest = amount - principal` calculates the actual amount of interest earned by subtracting the initial principal from the final amount.
6. **Print Results:** The `print()` statements display the final amount and the compound interest earned, formatted to two decimal places using f-strings (`:.2f`).
7. **Function Call:** `calculate_compound_interest()` calls the function to execute the code and perform the calculation.

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. Learn more

Task Assignment 2:

Task Description#2

- Do math stuff, then refine it to: # Write a function to calculate average, median, and mode of a list of numbers.

PROMPT: write a function in python to calculate average, median, mode of a list of numbers

CODE:

```
[1]  from collections import Counter

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

       Args:
         numbers: A list of numbers.

       Returns:
         A dictionary containing the average, median, and mode of the list.
       """
       if not numbers:
         return {"average": None, "median": None, "mode": None}

       # Calculate average
       average = sum(numbers) / len(numbers)

       # Calculate median
       sorted_numbers = sorted(numbers)
       n = len(sorted_numbers)
       if n % 2 == 0:
         median = (sorted_numbers[n // 2 - 1] + sorted_numbers[n // 2]) / 2
       else:
         median = sorted_numbers[n // 2]

       # Calculate mode
       count = Counter(numbers)
       max_freq = max(count.values())
       mode = [num for num, freq in count.items() if freq == max_freq]
       # If all numbers appear the same number of times, there is no unique mode
       if len(mode) == len(numbers):
           mode = None
       elif len(mode) == 1:
           mode = mode[0]
```

```python
# Example usage
my_list = [1, 2, 3, 4, 5, 5, 6, 6, 6, 7]
stats = calculate_stats(my_list)
print(stats)

my_list_2 = [1, 2, 3, 4, 5]
stats_2 = calculate_stats(my_list_2)
print(stats_2)

my_list_3 = [1, 1, 2, 2, 3, 3]
stats_3 = calculate_stats(my_list_3)
print(stats_3)
```

OUTPUT:

```
{'average': 4.5, 'median': 5.0, 'mode': 6}
{'average': 3.0, 'median': 3, 'mode': None}
{'average': 2.0, 'median': 2.0, 'mode': [1, 2, 3]}
```

EXPLANATION:

The selected ccde defines a Python function called calculate_stats that takes a list of numbers as input and returns a dictionary containing the average, median, and mode of those numbers.

Here's a breakdown of the code:

1. **from collections import Counter**: This line imports the Counter class from the collections module. The Counter class is used to easily count the occurrences of each item in a list, which is helpful for finding the mode.
2. **def calculate_stats(numbers):**: This defines the function calculate_stats that accepts one argument, numbers, which is expected to be a list.
3. **Docstring**: The triple-quoted string explains what the function does, its arguments (Args), and what it returns (Returns).
4. **if not numbers:**: This checks if the input list numbers is empty. If it is, the function returns a dictionary with None for average, median, and mode, as these statistics cannot be calculated for an empty list.
5. **Calculate average**:
    - average = sum(numbers) / len(numbers): This calculates the average by summing all the numbers in the list using sum() and dividing by the total number of elements in the list using len().

6. **Calculate median**:
    - sorted_numbers = sorted(numbers): The list is sorted in ascending order because the median is the middle value of a sorted dataset.
    - n = len(sorted_numbers): The number of elements in the sorted list is stored in n.
    - if n % 2 == 0:: This checks if the number of elements is even. If it is, the median is the average of the two middle elements.
    - median = (sorted_numbers[n // 2 - 1] + sorted_numbers[n // 2]) / 2: This calculates the average of the two middle elements. n // 2 - 1 gives the index of the first middle element, and n // 2 gives the index of the second middle element (remembering that list indices start at 0).
    - else:: If the number of elements is odd, the median is the single middle element.
    - median = sorted_numbers[n // 2]: This gets the middle element using integer division n // 2.
7. **Calculate mode**:
    - count = Counter(numbers): A Counter object is created, which counts the occurrences of each number in the list.
    - max_freq = max(count.values()): This finds the highest frequency (the maximum number of times any number appears) in the list.
    - mode = [num for num, freq in count.items() if freq == max_freq]: This creates a list called mode containing all the numbers that have the max_freq. There can be multiple modes if several numbers appear with the same highest frequency.
    - if len(mode) == len(numbers): mode = None: This checks if all numbers in the list appear the same number of times (meaning every number is a mode). In this case, there is no unique mode, so mode is set to None.
    - elif len(mode) == 1: mode = mode[0]: If there is only one number with the maximum frequency, that number is the unique mode, and mode is set to that single number.
8. **return {"average": average, "median": median, "mode": mode}**: The function returns a dictionary containing the calculated average, median, and mode.

The example usage at the end of the code demonstrates how to call the function with different lists and print the results.

Task Description#3

- Provide multiple examples of input-output to the AI for convert_to_binary(num) function. Observe how AI uses few-shot prompting to generalize.

PROMPT: write a function in python to convert numbers into binary by taking input from user and display it

CODE:

```
pinary():

cimal number to its binary representation based on user input.


 = int(input("Enter a decimal number: "))
num < 0:
ease enter a non-negative integer.")


= bin(decimal_num)
urns a string like '0b101', so we slice from index 2 to remove '0b'
 binary representation of {decimal_num} is: {binary_num[2:]}")

rror:
lid input. Please enter an integer.")

tion to get user input and perform the conversion
ry()
```

OUTPUT:

```
Enter a decimal number: 100
The binary representation of 100 is: 1100100
```

EXPLANATION:

The selected code defines a Python function called decimal_to_binary that prompts the user to enter a decimal number and then displays its binary representation.

Here's a breakdown of the code:

1. **def decimal_to_binary():**: This defines the function decimal_to_binary which takes no arguments.
2. **Docstring**: The triple-quoted string explains what the function does.
3. **try...except ValueError:**: This block is used for error handling. It attempts to execute the code within the try block. If a ValueError occurs (which happens if the user enters something that cannot be converted to an integer), the code within the except block is executed.

4. **decimal_num = int(input("Enter a decimal number: "))**: This line prompts the user to enter a decimal number using the input() function. The text inside the parentheses is displayed to the user as a prompt. The input received from the user is initially a string, so int() is used to convert it to an integer. If the input is not a valid integer, a ValueError will be raised, and the except block will be triggered.
5. **if decimal_num < 0:**: This checks if the entered decimal number is negative. The function is designed to work with non-negative integers for binary conversion.
6. **print("Please enter a non-negative integer.")**: If the number is negative, this message is printed to the user.
7. **return**: This exits the function if a negative number is entered.
8. **binary_num = bin(decimal_num)**: This is the core of the conversion. The built-in Python function bin() takes an integer as input and returns a string representing its binary form. The string starts with "0b" to indicate that it's a binary number (e.g., bin(10) returns '0b1010').
9. **print(f"The binary representation of {decimal_num} is: {binary_num[2:]}")**: This line prints the result.
    - An f-string is used to easily embed the values of variables within the output string.
    - {decimal_num} inserts the original decimal number entered by the user.
    - {binary_num[2:]} inserts the binary string. binary_num[2:] uses string slicing to remove the first two characters ("0b") from the string returned by bin(), providing a clean binary representation.
10. **except ValueError:**: If the int(input(...)) line failed because the user entered non-integer input, this block is executed.
11. **print("Invalid input. Please enter an integer.")**: This message is printed to inform the user that their input was invalid.
12. **decimal_to_binary()**: This line calls the function to start the prccess.

In summary, the function is a simple tool for converting a non-negative decimal integer to its binary equivalent, including basic error handling for invalid or negative input.

Task Description#4

- Create an user interface for an hotel to generate bill based on customer requirements

PROMPT: write a python prcgram to Create an user interface for an hotel to generate bill based on customer requirements

CODE:

```python
customer_name = input("Enter customer name: ")
room_number = input("Enter room number: ")

ordered_items = []

while True:
  item_name = input("Enter item name (or type 'done' to finish): ")
  if item_name.lower() == 'done':
    break
  try:
    quantity = int(input(f"Enter quantity for {item_name}: "))
    ordered_items.append({"item": item_name, "quantity": quantity})
  except ValueError:
    print("Invalid quantity. Please enter a number.")

print("\nCustomer Details:")
print(f"Name: {customer_name}")
print(f"Room Number: {room_number}")
print("\nOrdered Items:")
for item in ordered_items:
  print(f"- {item['item']}: {item['quantity']}")
```

OUTPUT:

```
==============================
      THE PYTHON HOTEL
        GUEST BILL
==============================
Customer Name: Bandi
Room Number: 100
------------------------------
Ordered Items:
Burger x 2: $21.98
Fries x 3: $10.47
Soda x 1: $1.99
Steak x 1: $25.00
Extra Towels x 4: $8.00
Item 'Non-existent Item' (not found in menu)
------------------------------
Total: $67.44
==============================
Thank You for Your Stay!
```

EXPLANATION:

The selected code is responsible for formatting and printing the final guest bill based on the customer's details, ordered items, and the calculated total bill.

Here's a breakdown of the code:

1. print('\n" + "="*30): This prints a newline character (\n) for spacing, followed by a line of 30 equal signs (=) to create a visual border for the top of the bill.
2. print(" THE PYTHON HOTEL") and print(" GUEST BILL"): These lines print the hotel name and the title "GUEST BILL" centered within the defined width.
3. print("="*30): This prints another line of 30 equal signs to separate the header from the customer details.
4. print(f"Customer Name: {customer_name}"): This uses an f-string to print the "Customer Name:" label followed by the value of the customer_name variable.
5. print(f"Room Number: {room_number}"): Similarly, this prints the "Room Number:" label followed by the value of the room_number variable.
6. print("-" * 30): This prints a line of 30 hyphens (-) to separate the customer details from the ordered items.
7. print("Ordered Items:"): This prints the heading for the list of ordered items.
8. for item in ordered_items:: This starts a loop that iterates through each dictionary (item) in the ordered_items list.
9. item_name = item['item'] and quantity = item['quantity']: Inside the loop, these lines extract the item name and quantity from the current item dictionary.
10. if item_name in menu:: This checks if the current item_name exists as a key in the menu dictionary (meaning it's a valid menu item).
11. item_price = menu[item_name]: If the item is in the menu, this line retrieves its price from the menu dictionary.
12. item_cost = item_price * quantity: This calculates the total cost for the current item by multiplying its price by the ordered quantity.
13. print(f"{item_name} x {quantity}: ${item_cost:.2f}"): This prints the item details (name and quantity) and its calculated cost. :.2f formats the item_cost to two decimal places for currency.
14. else:: If the item_name is not found in the menu.
15. print(f"Item '{item_name}' (not found in menu)"): This prints a message indicating that the item was not found in the menu.
16. print("-" * 30): This prints another line of 30 hyphens to separate the ordered items from the total.
17. print(f"Total: ${total_bill:.2f}"): This prints the "Total:" label followed by the final calculated total_bill, formatted to two decimal places.
18. print("="*30): This prints a line of 30 equal signs for the bottom border of the bill.
19. print('Thank You for Your Stay!"): This prints a thank you message at the end of the bill.

In essence, this code iterates through the list of ordered items, looks up their prices in the menu (and handles items not found), calculates the cost for each, and then neatly prints all this information along with the customer details and the final total, creating a formatted hotel bill.

Task Description#5

- Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions

PROMPT: write a function in python code to covert fahrenheit into celcius and celcius into fahrenheit and display it

CODE:

```python
def temperature_converter():
    """
    Converts temperature between Fahrenheit and Celsius based on user input.
    """
    print("Temperature Converter")
    print("1. Fahrenheit to Celsius")
    print("2. Celsius to Fahrenheit")

    choice = input("Enter your choice (1 or 2): ")

    try:
        if choice == '1':
            fahrenheit = float(input("Enter temperature in Fahrenheit: "))
            celsius = (fahrenheit - 32) * 5/9
            print(f"{fahrenheit}°F is equal to {celsius:.2f}°C")
        elif choice == '2':
            celsius = float(input("Enter temperature in Celsius: "))
            fahrenheit = (celsius * 9/5) + 32
            print(f"{celsius}°C is equal to {fahrenheit:.2f}°F")
        else:
            print("Invalid choice. Please enter 1 or 2.")
    except ValueError:
        print("Invalid input. Please enter a valid number for temperature.")

# Call the function to start the converter
temperature_converter()
```

OUTPUT:

```
Temperature Converter
1. Fahrenheit to Celsius
2. Celsius to Fahrenheit
Enter your choice (1 or 2): 1
Enter temperature in Fahrenheit: 100
100.0°F is equal to 37.78°C
```

EXPLANATION:

The selected code defines a Python function called temperature_converter that allows the user to convert temperatures between Fahrenheit and Celsius.

Here's a breakdown of the code:

1. **def temperature_converter():: This line defines the function temperature_converter. It doesn't take any arguments.**
2. **Docstring: The triple-quoted string explains what the function does.**
3. **print("Temperature Converter"): This prints a title for the converter.**
4. **print("1. Fahrenheit to Celsius") and print("2. Celsius to Fahrenheit"): These lines display the options for the user to choose from.**
5. **choice = input("Enter your choice (1 or 2): "): This line prompts the user to enter their choice (either '1' or '2') and stores their input in the choice variable.**
6. **try...except ValueError:: This block is used for error handling. It attempts to execute the code within the try block. If a ValueError occurs (which happens if the user enters something that cannot be converted to a floating-point number for the temperature), the code within the except block is executed.**
7. **if choice == '1':: This checks if the user's choice is '1'.**
8. **fahrenheit = float(input("Enter temperature in Fahrenheit: ")): If the choice is '1', this prompts the user to enter the temperature in Fahrenheit and converts the input string to a floating-point number using float().**
9. **celsius = (fahrenheit - 32) * 5/9: This line performs the conversion from Fahrenheit to Celsius using the standard formula.**
10. **print(f"{fahrenheit}°F is equal to {celsius:.2f}°C"): This prints the result of the conversion, showing the original Fahrenheit temperature and the calculated Celsius temperature formatted to two decimal places.**
11. **elif choice == '2':: This checks if the user's choice is '2'.**
12. **celsius = float(input("Enter temperature in Celsius: ")): If the choice is '2', this prompts the user to enter the temperature in Celsius and converts the input string to a floating-point number.**
13. **fahrenheit = (celsius * 9/5) + 32: This line performs the conversion from Celsius to Fahrenheit using the standard formula.**

14. print(f"{celsius}°C is equal to {fahrenheit:.2f}°F"): This prints the result of the conversion, showing the original Celsius temperature and the calculated Fahrenheit temperature formatted to two decimal places.
15. else:: If the user enters something other than '1' or '2' for the choice.
16. print("Invalid choice. Please enter 1 or 2."): This message is printed to inform the user that their choice was invalid.
17. except ValueError:: If the float(input(...)) line failed because the user entered non-numeric input for the temperature, this block is executed.
18. print("Invalid input. Please enter a valid number for temperature."): This message is printed to inform the user that their temperature input was invalid.
19. temperature_converter(): This line calls the function to start the temperature conversion process when the script is run.

In summary, this function provides a simple interactive tool for converting temperatures between Fahrenheit and Celsius, including basic input validation and error handling.