

ASSIGNMENT-3.2

TASK-1:

Prompt:

#write a python program to calculate compound interest using function.

Code and Output:

```
ai assign-3.2.py > ...
1  #write a python program to calculate compound interest using functions.
2  """
3  Calculate the compound interest for a given principal amount, interest rate, time period, and compounding frequency.
4  Parameters:
5      principal (float): The initial amount of money.
6      rate (float): The annual interest rate (in percent).
7      time (float): The number of years the money is invested or borrowed for.
8      n (int): The number of times interest is compounded per year.
9  Returns:
10     float: The compound interest earned over the specified period.
11     """
12  #write a python program to calculate compound interest using functions.
13  def compound_interest(principal, rate, time, n):
14      # principal: initial amount
15      # rate: annual interest rate (in percent)
16      # time: number of years
17      # n: number of times interest applied per year
18      amount = principal * (1 + rate / (100 * n)) ** (n * time)
19      ci = amount - principal
20      return ci
21
22  # Input values
23  p = float(input("Enter principal amount: "))
24  r = float(input("Enter annual interest rate (%): "))
25  t = float(input("Enter time in years: "))
26  n = int(input("Enter number of times interest is compounded per year: "))
27
28  result = compound_interest(p, r, t, n)
29  print("Compound Interest:", result)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Enter principal amount: 20000
Enter annual interest rate (%): 10
Enter time in years: 5
Enter number of times interest is compounded per year: 10
Compound Interest: 12892.636436877656
```

Explanation:

```
"""
Calculate the compound interest for a given principal amount, interest rate, time period, and compounding frequency.
Parameters:
    principal (float): The initial amount of money.
    rate (float): The annual interest rate (in percent).
    time (float): The number of years the money is invested or borrowed for.
    n (int): The number of times interest is compounded per year.
Returns:
    float: The compound interest earned over the specified period.
"""
```

TASK-2:

Prompt:

#write a python program that calculates average, median, mode using functions.


Code and Output:

```
def calculate_average(numbers):  
    """Calculates the average of a list of numbers."""  
    if not numbers:  
        return 0 # Return 0 for an empty list  
    return sum(numbers) / len(numbers)  
  
def calculate_median(numbers):  
    """Calculates the median of a list of numbers."""  
    if not numbers:  
        return 0 # Return 0 for an empty list  
    sorted_numbers = sorted(numbers)  
    n = len(sorted_numbers)  
    mid_index = n // 2  
    if n % 2 == 0:  
        # Even number of elements, median is the average of the middle two  
        return (sorted_numbers[mid_index - 1] + sorted_numbers[mid_index]) / 2  
    else:  
        # Odd number of elements, median is the middle element  
        return sorted_numbers[mid_index]  
  
def calculate_mode(numbers):  
    """Calculates the mode(s) of a list of numbers."""  
    if not numbers:  
        return [] # Return an empty list for an empty list  
  
    from collections import Counter  
    counts = Counter(numbers)  
    max_count = max(counts.values())  
    modes = [number for number, count in counts.items() if count == max_count]  
    return modes
```

```
# Example usage  
my_list = [1, 2, 3, 4, 5, 5, 6, 7, 8, 8, 8]  
  
average = calculate_average(my_list)  
median = calculate_median(my_list)  
mode = calculate_mode(my_list)  
  
print(f"The list is: {my_list}")  
print(f"The average is: {average}")  
print(f"The median is: {median}")  
print(f"The mode(s) is/are: {mode}")
```

```
→ The list is: [1, 2, 3, 4, 5, 5, 6, 7, 8, 8, 8]  
The average is: 5.181818181818182  
The median is: 5  
The mode(s) is/are: [8]
```

Explanation:

 explain the code step by step

◆ Certainly! Here's a step-by-step explanation of the provided Python code:

1. `def calculate_average(numbers):`: This defines a function named `calculate_average` that takes one argument, `numbers`, which is expected to be a list of numbers.
2. `if not numbers:`: Inside the `calculate_average` function, this checks if the input list `numbers` is empty.
3. `return 0`: If the list is empty, the function returns `0` as the average.
4. `return sum(numbers) / len(numbers)`: If the list is not empty, this line calculates the sum of all numbers in the list using `sum()` and divides it by the number of elements in the list using `len()`, returning the calculated average.
5. `def calculate_median(numbers):`: This defines a function named `calculate_median` that also takes a list of `numbers` as input.
6. `if not numbers:`: Similar to the average function, this checks if the input list is empty.
7. `return 0`: If the list is empty, it returns `0` as the median.
8. `sorted_numbers = sorted(numbers)`: If the list is not empty, this line creates a new list called `sorted_numbers` by sorting the input `numbers` list in ascending order.
9. `n = len(sorted_numbers)`: This gets the number of elements in the sorted list and stores it in the variable `n`.
10. `mid_index = n // 2`: This calculates the middle index of the sorted list using integer division (`//`).
11. `if n % 2 == 0:`: This checks if the number of elements (`n`) is even.
12. `(sorted_numbers[mid_index - 1] + sorted_numbers[mid_index]) / 2`: If `n` is even, the median is the average of the two middle elements. This line calculates that average.
13. `else:`: This is executed if the number of elements (`n`) is odd.
14. `return sorted_numbers[mid_index]`: If `n` is odd, the median is the middle element. This line returns the element at the `mid_index`.
15. `def calculate_mode(numbers):`: This defines a function named `calculate_mode` that takes a list of `numbers` as input.

16. `if not numbers:`: Checks if the input list is empty.
17. `return []`: If the list is empty, it returns an empty list as there is no mode.
18. `from collections import Counter`: This imports the `Counter` class from the `collections` module, which is useful for counting hashable objects.
19. `counts = Counter(numbers)`: This creates a `Counter` object named `counts` that stores the frequency of each number in the input list.
20. `max_count = max(counts.values())`: This finds the highest frequency among all the numbers in the list and stores it in `max_count`.
21. `modes = [number for number, count in counts.items() if count == max_count]`: This uses a list comprehension to create a list called `modes` containing all the numbers that have a frequency equal to `max_count`. These are the modes of the list.
22. `return modes`: The function returns the list of modes.
23. `my_list = [1, 2, 3, 4, 5, 5, 6, 7, 8, 8]`: This line creates a list named `my_list` with some example numbers.
24. `average = calculate_average(my_list)`: This calls the `calculate_average` function with `my_list` and stores the result in the `average` variable.
25. `median = calculate_median(my_list)`: This calls the `calculate_median` function with `my_list` and stores the result in the `median` variable.
26. `mode = calculate_mode(my_list)`: This calls the `calculate_mode` function with `my_list` and stores the result in the `mode` variable.
27. `print(f"The list is: {my_list}")`: This prints the original list.
28. `print(f"The average is: {average}")`: This prints the calculated average.
29. `print(f"The median is: {median}")`: This prints the calculated median.
30. `print(f"The mode(s) is/are: {mode}")`: This prints the calculated mode(s).

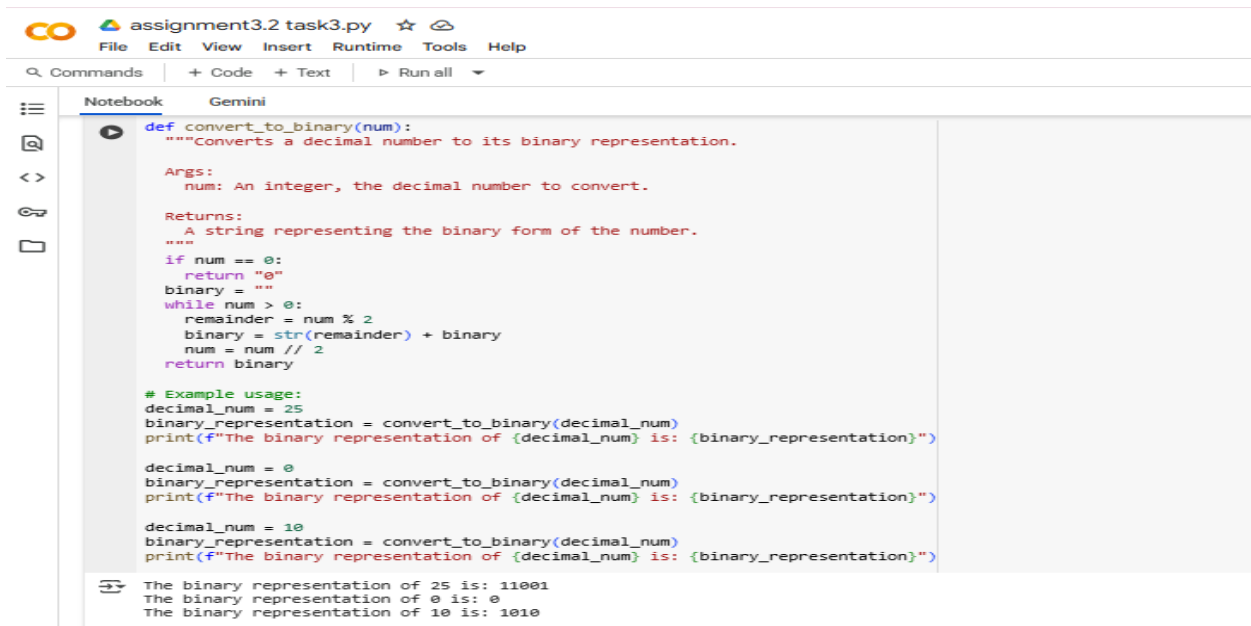
The output below the code cell shows the results of these print statements for the example list `my_list`.

TASK-3

Prompt:

#write a python program to convert number into binary using functions.

Code and Output:



```
def convert_to_binary(num):
    """Converts a decimal number to its binary representation.

    Args:
        num: An integer, the decimal number to convert.

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

# Example usage:
decimal_num = 25
binary_representation = convert_to_binary(decimal_num)
print(f"The binary representation of {decimal_num} is: {binary_representation}")

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

decimal_num = 10
binary_representation = convert_to_binary(decimal_num)
print(f"The binary representation of {decimal_num} is: {binary_representation}")
```

The binary representation of 25 is: 11001
The binary representation of 0 is: 0
The binary representation of 10 is: 1010

Explanation:

explain the code step by step

Sure, let's break down the code step by step:

- def convert_to_binary(num):** : This line defines a function named `convert_to_binary` that takes one argument, `num`. This function is designed to convert the decimal number passed as `num` into its binary equivalent.
- if num == 0:** : This is a conditional statement that checks if the input number `num` is equal to 0. The binary representation of 0 is simply "0", so this handles that specific case directly.
- return "0"** : If `num` is 0, the function immediately returns the string "0".
- binary = ""** : This line initializes an empty string variable called `binary`. This string will be used to build the binary representation of the number. The binary digits will be added to this string from right to left as they are calculated.
- while num > 0:** : This is a `while` loop that continues as long as the value of `num` is greater than 0. The loop iteratively calculates the binary digits.
- remainder = num % 2:** : Inside the loop, this line calculates the remainder when `num` is divided by 2. In binary conversion, the remainder of the division by 2 is the next binary digit (either 0 or 1), starting from the rightmost digit.
- binary = str(remainder) + binary** : This line takes the `remainder` (which is either 0 or 1), converts it to a string using `str()`, and then prepends it to the `binary` string. This builds the binary representation in reverse order, adding the least significant bit first.
- num = num // 2** : This line updates the value of `num` by performing integer division by 2. This effectively shifts the focus to the next bit in the binary conversion.
- return binary** : Once the `while` loop finishes (meaning `num` has become 0), the function returns the `binary` string, which now holds the complete binary representation of the original decimal number.

The rest of the code demonstrates how to use the function with different decimal numbers and prints the results.

TASK-4

Prompt:

#write a python program for hotel billing using items, quantity, bill using functions.

Code :

```
ai assign 3.2(4).py X
ai assign 3.2(4).py > calculate_bill
1 #write a python program for hotel bill using items,quantity,bill using function.
2 # Sample menu with prices
3 MENU = {
4     "Idli": 30,
5     "Dosa": 40,
6     "Vada": 25,
7     "Puri": 35,
8     "Tea": 15
9 }
10 def calculate_bill(order):
11     total = 0
12     print("\n----- Bill Details -----")
13     for item, qty in order.items():
14         if item in MENU:
15             item_total = MENU[item] * qty
16             print(f"{item}: {qty} x ₹{MENU[item]} = ₹{item_total}")
17             total += item_total
18     print(f"Total Bill: ₹{total}")
19     return total
20 # Input order from user
21 order = {}
22 print("Menu:")
23 for item, price in MENU.items():
24     print(f"{item}: ₹{price}")
25 print("\nEnter quantity for each item (0 if not ordered):")
26 for item in MENU:
27     qty = int(input(f"{item}: "))
28     if qty > 0:
29         order[item] = qty
30 calculate_bill(order)
```

Output:

```
Menu:
Idli: ₹30
Dosa: ₹40
Vada: ₹25
Puri: ₹35
Tea: ₹15

Enter quantity for each item (0 if not ordered):
Idli: 3
Dosa: 2
Vada: 1
Puri: 5
Tea: 2

----- Bill Details -----
Idli: 3 x ₹30 = ₹90
Dosa: 2 x ₹40 = ₹80
Vada: 1 x ₹25 = ₹25
Puri: 5 x ₹35 = ₹175
Tea: 2 x ₹15 = ₹30
Total Bill: ₹400
```

Explanation:

```
"""
This program simulates a hotel billing system using a predefined menu of items and their prices.
It allows the user to input the quantity of each item ordered, calculates the total bill, and displays a detailed bill summary.
Functions:
    calculate_bill(order):
        Calculates and prints the bill details for the given order.
        Args:
            order (dict): Dictionary with item names as keys and quantities as values.
        Returns:
            int: The total bill amount.
Variables:
    MENU (dict): Dictionary containing menu items as keys and their prices as values.
Usage:
    The user is prompted to enter the quantity for each menu item.
    The program then calculates and displays the bill details and total amount.
"""
```

TASK-5:

Prompt:

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:

```
[7] def c_to_f(celsius):  
    """Converts Celsius to Fahrenheit."""  
    return (celsius * 9/5) + 32  
  
    # Example usage:  
    celsius_temp = 25  
    fahrenheit_temp = c_to_f(celsius_temp)  
    print(f"{celsius_temp}°C is equal to {fahrenheit_temp}°F")  
  
    celsius_temp_2 = 0  
    print(f"{celsius_temp_2}°C is equal to {c_to_f(celsius_temp_2)}°F")
```

Output:

```
⇒ 25°C is equal to 77.0°F  
   0°C is equal to 32.0°F
```

Explanation:

This code defines a simple function `c_to_f`. It takes a temperature in Celsius as input. The formula $(\text{celsius} * 9/5) + 32$ is used for the conversion. This formula multiplies the Celsius temperature by 9/5 and adds 32. The function then returns the calculated Fahrenheit temperature. The code also includes example usage. It calls the function with 25°C and 0°C. Finally, it prints the original Celsius temperature and the converted Fahrenheit temperature. This provides a clear demonstration of the function's usage and output.