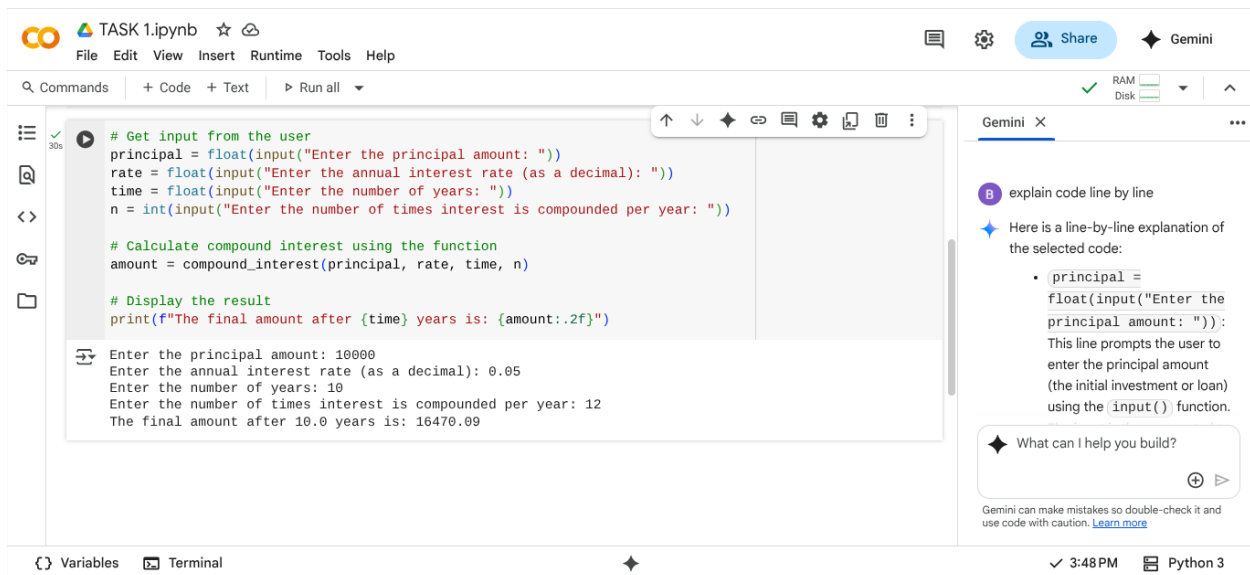


# ASSIGNMENT-3.2

## TASK-1:

Write a program in python to calculate compound interest using functions

## Code and Output:



The screenshot shows a Jupyter Notebook titled "TASK 1.ipynb" with a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar. The code cell contains the following Python code:

```
# Get input from the user
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: "))
n = int(input("Enter the number of times interest is compounded per year: "))

# Calculate compound interest using the function
amount = compound_interest(principal, rate, time, n)

# Display the result
print(f"The final amount after {time} years is: {amount:.2f}")
```

The output cell shows the execution results:

```
Enter the principal amount: 10000
Enter the annual interest rate (as a decimal): 0.05
Enter the number of years: 10
Enter the number of times interest is compounded per year: 12
The final amount after 10.0 years is: 16470.09
```

On the right side, the Gemini chat interface is visible, showing a prompt "explain code line by line" and a response explaining the first line of code: "Here is a line-by-line explanation of the selected code: • principal = float(input('Enter the principal amount: ')): This line prompts the user to enter the principal amount (the initial investment or loan) using the input() function."

## Explanation:

The screenshot shows a Jupyter Notebook titled "TASK 1.ipynb" with a Gemini AI chat window open. The chat window displays a line-by-line explanation of a Python code snippet that calculates compound interest. The code snippet is as follows:

```
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: "))
n = int(input("Enter the number of times interest is compounded per year: "))
amount = compound_interest(principal, rate, time, n)
print(f"The final amount after {time} years is: {amount:.2f}")
```

The Gemini AI response provides a detailed explanation of each line of code, including the purpose of the input function, the conversion of input to float or integer, the use of the compound\_interest function, and the formatting of the output using an f-string.

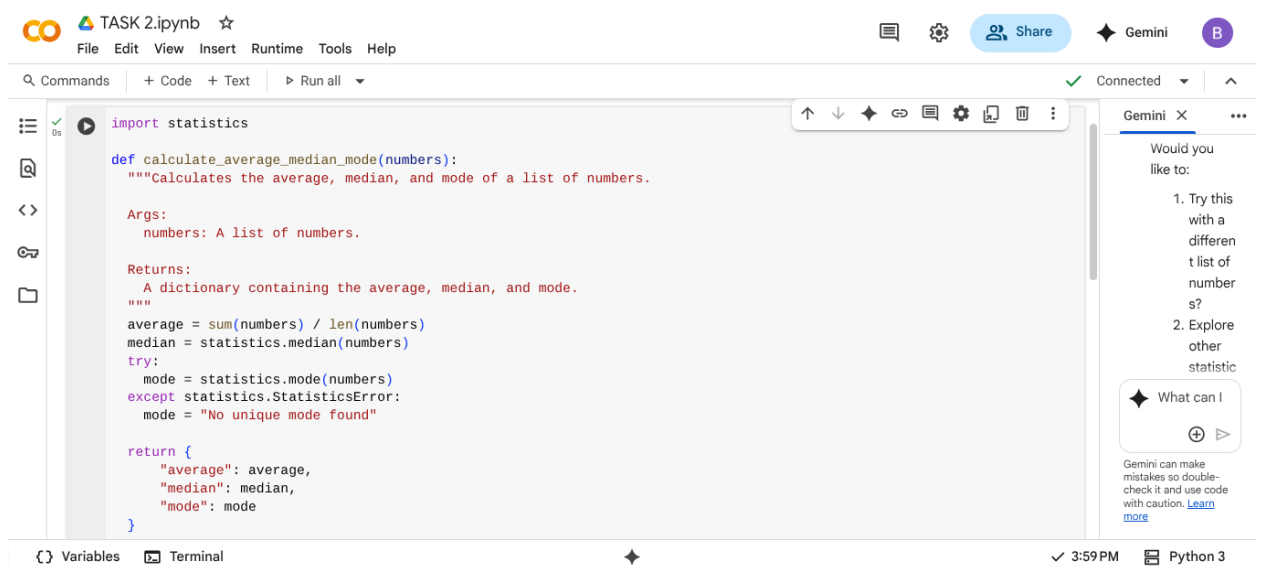
Below the explanation, there is a prompt: "What can I help you build?" and a link to "Learn more".

The bottom of the interface shows the "Variables" and "Terminal" tabs, along with the system time "3:48 PM" and the Python version "Python 3".

## TASK-2:

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

# Code:



The screenshot shows a Jupyter Notebook titled "TASK 2.ipynb". The code defines a function `calculate_average_median_mode` that takes a list of numbers and returns a dictionary with the average, median, and mode. The function uses the `statistics` module. The mode calculation uses a try-except block to handle cases where there is no unique mode.

```
import statistics

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

    Args:
        numbers: A list of numbers.

    Returns:
        A dictionary containing the average, median, and mode.
    """
    average = sum(numbers) / len(numbers)
    median = statistics.median(numbers)
    try:
        mode = statistics.mode(numbers)
    except statistics.StatisticsError:
        mode = "No unique mode found"

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

On the right side, the Gemini chat interface is visible, showing a prompt "Would you like to:" and two suggestions: "1. Try this with a different list of numbers?" and "2. Explore other statistics". Below these is a "What can I" button and a note: "Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)".

# OUTPUT:

TASK 2.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

# Example usage:  
my\_list = [1, 2, 3, 4, 5, 5, 6, 7, 8, 8, 8, 9]  
results = calculate\_average\_median\_mode(my\_list)  
print(results)  
  
{'average': 5.5, 'median': 5.5, 'mode': 8}

Variables

Terminal

Share

Gemini

Connected

3:59 PM Python 3

Would you like to:  
  
1. Try this with a different list of numbers?  
2. Explore other statistics  
  
What can I  
  
Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

## EXPLANATION:

TASK 2.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

# Example  
my\_list = [1, 2, 3, 4, 5, 5, 6, 7, 8, 8, 8, 9]  
results = calculate\_average\_median\_mode(my\_list)  
print(results)  
  
{'average': 5.5, 'median': 5.5, 'mode': 8}

Variables

Terminal

Share

Gemini

Connected

3:59 PM Python 3

Gemini

No problem! Here is a line-by-line explanation of the code:

- `import statistics`: This line imports the `statistics` module, which provides functions for calculating common statistical measures like median and mode.
- `def calculate_average_median_mode(numbers):`: This line defines a function named `calculate_average_median_mode` that takes one argument, `numbers`, which is expected to be a list of numbers.
- `"""Calculates the average, median, and mode of a list of numbers. ... """`: This is a docstring, providing a brief explanation of what the function does, its arguments, and what it returns.
- `average = sum(numbers) / len(numbers)`: This line calculates the average (mean) of the numbers in the list by summing all the numbers and dividing by the total count of numbers.
- `median = statistics.median(numbers)`: This line calculates the median of the numbers using the `median()` function from the imported `statistics` module. The median is the middle value in a sorted list.
- `try:`: This starts a `try` block, which is used to handle potential errors.
- `mode = statistics.mode(numbers)`: This line attempts to calculate the mode of the numbers using the `mode()` function from the `statistics` module. The mode is the most frequent number in the list.
- `except statistics.StatisticsError:`: This line catches a specific error, `statistics.StatisticsError`, which occurs if there is no unique mode (e.g., if all numbers appear the same number of times).
- `mode = "No unique mode found"`: If a `statistics.StatisticsError` is caught, this line sets the `mode` variable to the string "No unique mode found".
- `return {"average": average, "median": median, "mode": mode}`: This line returns a dictionary containing the calculated average, median, and mode.
- `my_list = [1, 2, 3, 4, 5, 5, 6, 7, 8, 8, 8, 9]`: This line creates a list of numbers named `my_list` to be used as an example input for the function.
- `results = calculate_average_median_mode(my_list)`: This line calls the `calculate_average_median_mode` function with `my_list` as input and stores the returned dictionary in the `results` variable.
- `print(results)`: This line prints the contents of the `results` dictionary to the console.

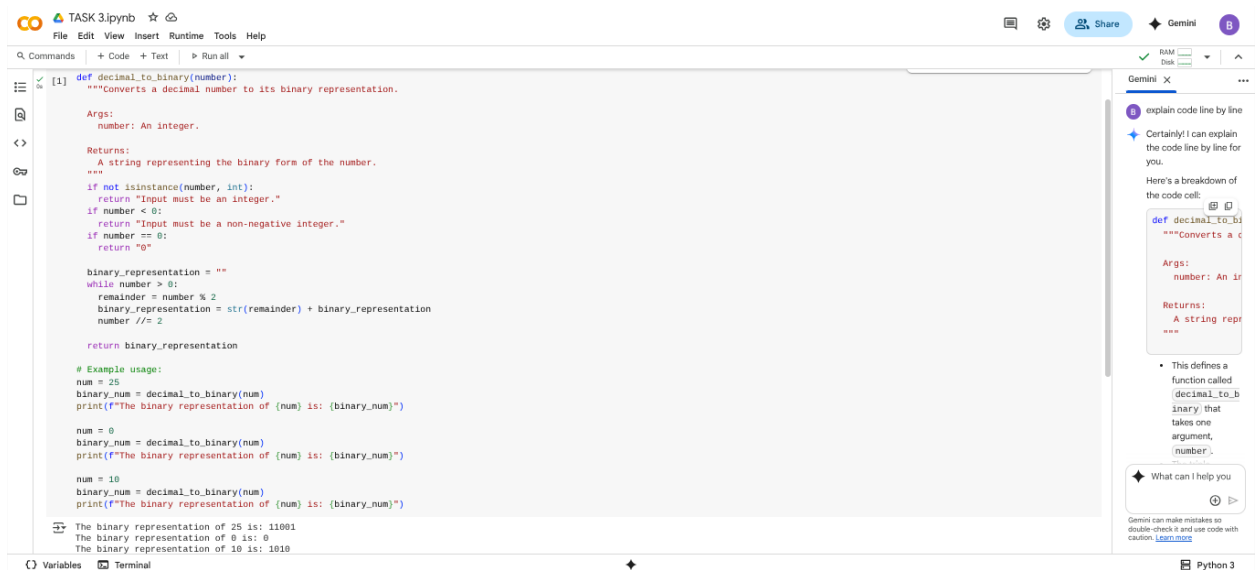
What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

## TASK-3:

Write a python program to convert numbers into binary using functions.

## Code and Output:



```
[1] 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 not isinstance(number, int):  
        return "Input must be an integer."  
    if number < 0:  
        return "Input must be a non-negative integer."  
    if number == 0:  
        return "0"  
  
    binary_representation = ""  
    while number > 0:  
        remainder = number % 2  
        binary_representation = str(remainder) + binary_representation  
        number //= 2  
  
    return binary_representation  
  
    # Example usage:  
    num = 25  
    binary_num = decimal_to_binary(num)  
    print(f"The binary representation of {num} is: {binary_num}")  
  
    num = 0  
    binary_num = decimal_to_binary(num)  
    print(f"The binary representation of {num} is: {binary_num}")  
  
    num = 10  
    binary_num = decimal_to_binary(num)  
    print(f"The binary representation of {num} is: {binary_num}")
```

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

**Gemini AI Explanation:**

- explain code line by line
- Certainly! I can explain the code line by line for you.
- Here's a breakdown of the code cell:

```
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 not isinstance(number, int):  
        return "Input must be an integer."  
    if number < 0:  
        return "Input must be a non-negative integer."  
    if number == 0:  
        return "0"  
  
    binary_representation = ""  
    while number > 0:  
        remainder = number % 2  
        binary_representation = str(remainder) + binary_representation  
        number //= 2  
  
    return binary_representation
```
- This defines a function called `decimal_to_binary` that takes one argument, `number`.

What can I help you

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

## EXPLANATION:

TASK 3.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Share Gemini

RAM Disk

Gemini

```
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.  
    """  
  
    • This defines a function called decimal_to_binary that takes one argument, number.  
    • The triple-quoted string is a docstring, explaining what the function does, its arguments, and what it returns.  
  
    if not isinstance(number, int):  
        return "Input must be an integer."  
    if number < 0:  
        return "Input must be a non-negative integer."  
    if number == 0:  
        return "0"  
  
    • These are input validation checks:  
        • It first checks if the input number is an integer. If not, it returns an error message.
```

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Variables Terminal Python 3

TASK 3.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Share Gemini

RAM Disk

Gemini

```
binary_representation = ""  
while number > 0:  
    remainder = number % 2  
    binary_representation = str(remainder) + binary_representation  
    number //= 2  
  
    • This is the core logic for the conversion:  
        • binary_representation = "": Initializes an empty string to build the binary representation.  
        • while number > 0: This loop continues as long as the number is greater than 0.  
        • remainder = number % 2: Calculates the remainder when number is divided by 2. This remainder will be either 0 or 1, which are the binary digits.  
        • binary_representation = str(remainder) + binary_representation: Converts the remainder to a string and adds it to the beginning of the binary_representation string. This builds the binary number in reverse order.  
        • number //= 2: Performs integer division of number by 2, updating the number for the next iteration.  
  
    return binary_representation
```

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Variables Terminal Python 3

```
number in reverse order.
• number //= 2: Performs integer division of number by 2, updating the number for the next iteration.

return binary_representation

• After the loop finishes (when number becomes 0), the function returns the built binary_representation string.

# Example usage:
num = 25
binary_num = decimal_to_binary(num)
print(f"The binary representation of {num} is: {binary_num}")

num = 0
binary_num = decimal_to_binary(num)
print(f"The binary representation of {num} is: {binary_num}")

num = 10
binary_num = decimal_to_binary(num)
print(f"The binary representation of {num} is: {binary_num}")

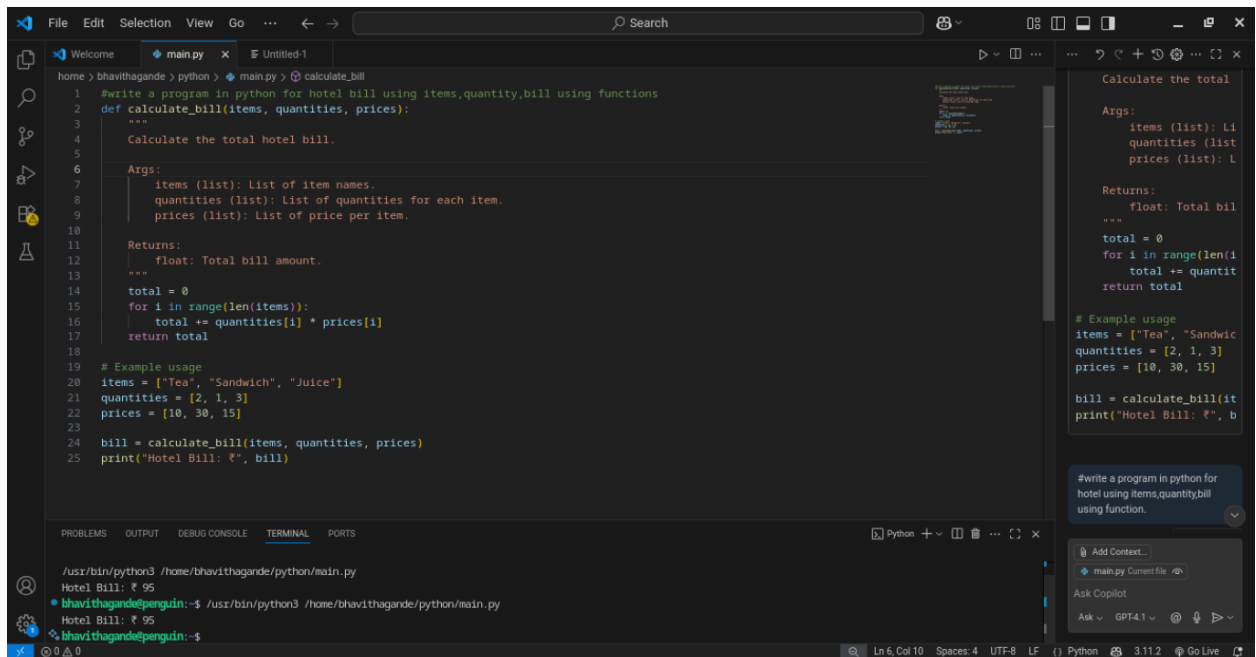
• These lines demonstrate how to use the decimal_to_binary function with different input values (25, 0, and 10) and print the results.
I hope this detailed explanation is helpful! Let me know if you have any more questions.

What can I help you build?
```

## TASK-4:

Write a python program for hotel bill using items, quantity and bill using functions

## Code and Output :



```
1 #write a program in python for hotel bill using items,quantity,bill using functions
2 def calculate_bill(items, quantities, prices):
3     """
4     Calculate the total hotel bill.
5
6     Args:
7         items (list): List of item names.
8         quantities (list): List of quantities for each item.
9         prices (list): List of price per item.
10
11     Returns:
12         float: Total bill amount.
13     """
14     total = 0
15     for i in range(len(items)):
16         total += quantities[i] * prices[i]
17     return total
18
19 # Example usage
20 items = ["Tea", "Sandwich", "Juice"]
21 quantities = [2, 1, 3]
22 prices = [10, 30, 15]
23
24 bill = calculate_bill(items, quantities, prices)
25 print("Hotel Bill: ₹", bill)
```

```
Python 3.11.2
/home/bhavithagande/python/main.py
Hotel Bill: ₹ 95
bhavithagandepenguin: /usr/bin/python3 /home/bhavithagande/python/main.py
Hotel Bill: ₹ 95
bhavithagandepenguin: $
```

## EXPLANATION:

This Python function `get_customer_requirements` simply provides a predefined set of hotel billing details.

- It returns a dictionary containing:
  - The chosen room type (e.g., "double").
  - The number of nights (e.g., 3).
  - A list of selected services (e.g., ["breakfast", "wifi"]).
  - Example prices for different room types.
  - Example prices for available services.

This is a simplified way to get customer information for the billing system without requiring user input.



## 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:

```
[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 `(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.

