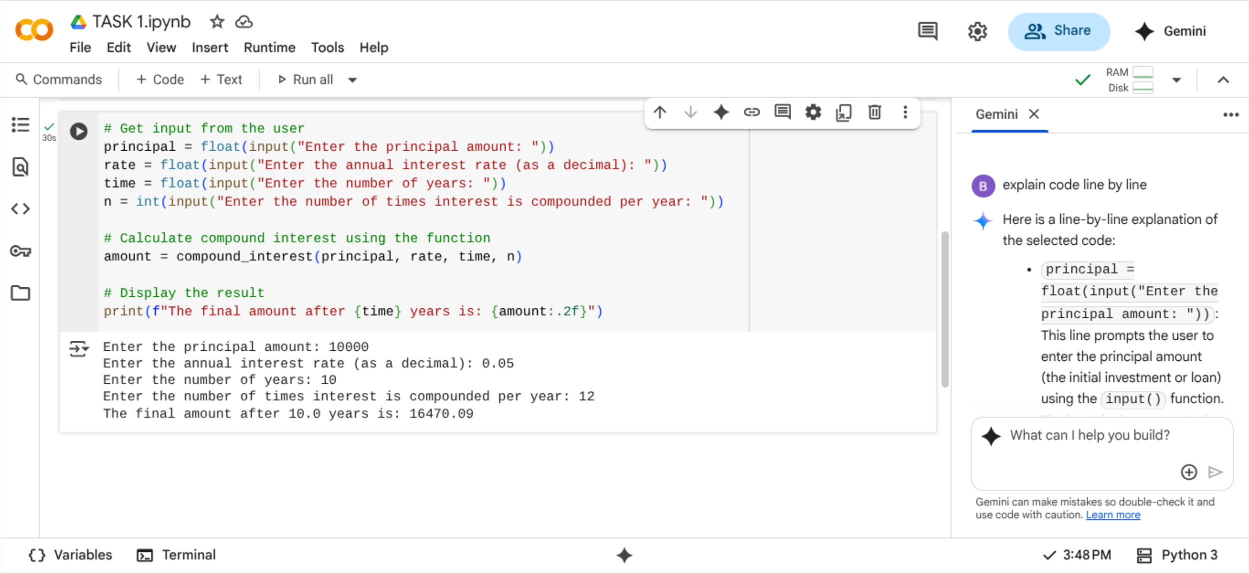
ASSIGNMENT-3.2

TASK-1:

Write a program in python to calculate compound interest using functions

Code and Output:



```python
# 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}")
```

```
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
```

Gemini

explain code line by line

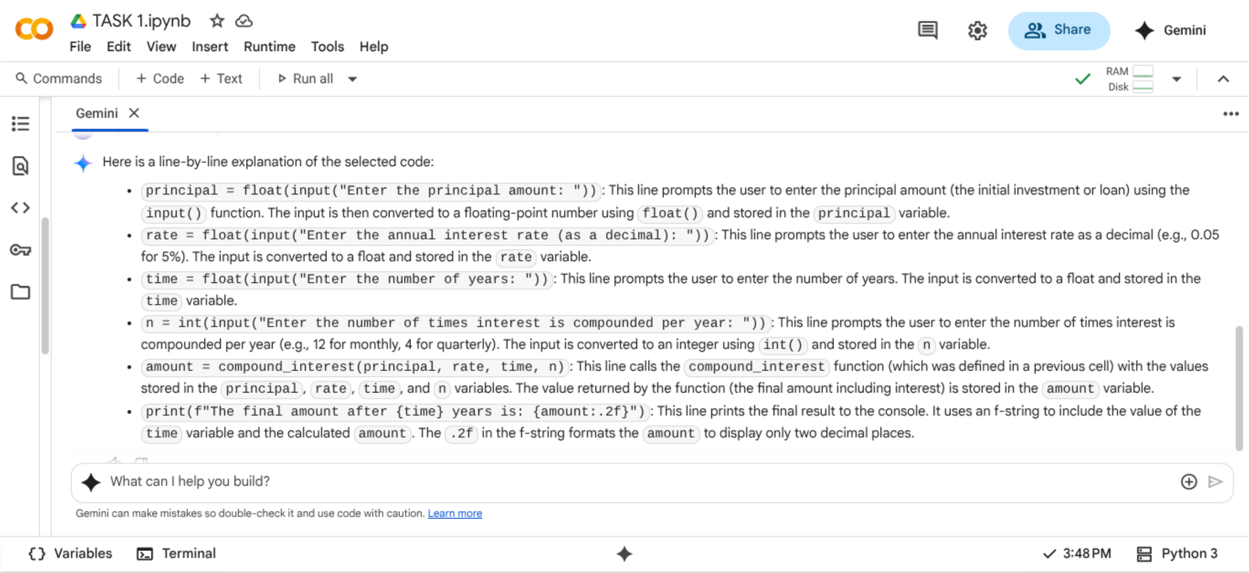
Here is a line-by-line explanation of the selected code:

- `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.

What can I help you build?

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

Explanation:



Here is a line-by-line explanation of the selected code:

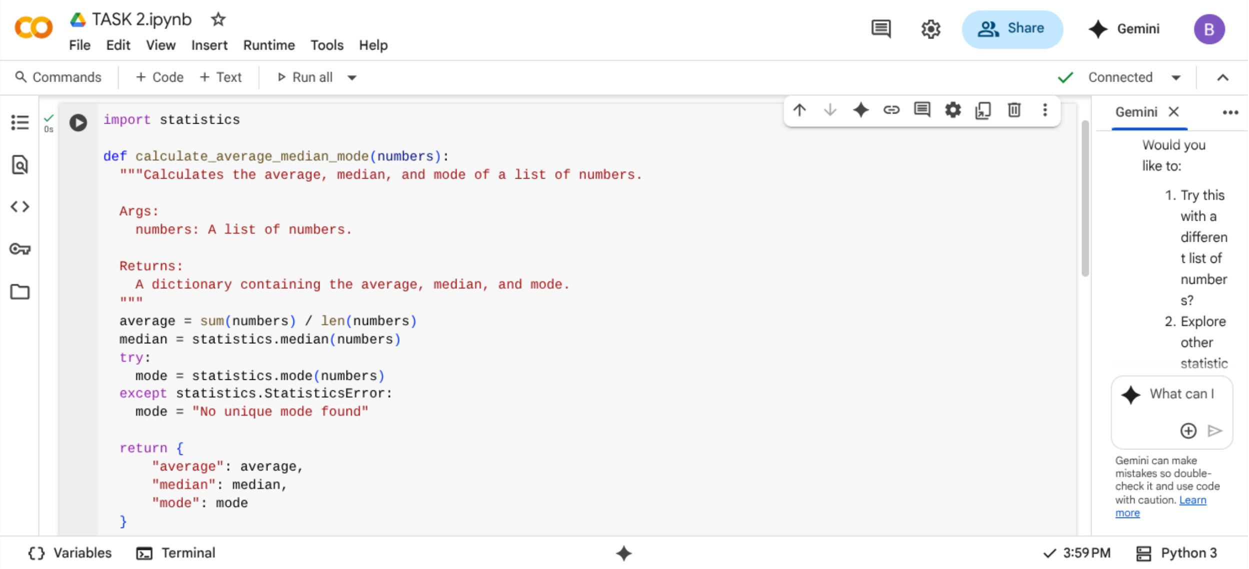
- `principal = float(input("Enter the principal amount: "))`: This line prompts the user to enter the principal amount (the initial investment or loan) using the `input()` function. The input is then converted to a floating-point number using `float()` and stored in the `principal` variable.
- `rate = float(input("Enter the annual interest rate (as a decimal): "))`: This line prompts the user to enter the annual interest rate as a decimal (e.g., 0.05 for 5%). The input is converted to a float and stored in the `rate` variable.
- `time = float(input("Enter the number of years: "))`: This line prompts the user to enter the number of years. The input is converted to a float and stored in the `time` variable.
- `n = int(input("Enter the number of times interest is compounded per year: "))`: This line prompts the user to enter the number of times interest is compounded per year (e.g., 12 for monthly, 4 for quarterly). The input is converted to an integer using `int()` and stored in the `n` variable.
- `amount = compound_interest(principal, rate, time, n)`: This line calls the `compound_interest` function (which was defined in a previous cell) with the values stored in the `principal`, `rate`, `time`, and `n` variables. The value returned by the function (the final amount including interest) is stored in the `amount` variable.
- `print(f"The final amount after {time} years is: {amount:.2f}")`: This line prints the final result to the console. It uses an f-string to include the value of the `time` variable and the calculated `amount`. The `.2f` in the f-string formats the `amount` to display only two decimal places.

What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
3:48 PM
Python 3

TASK-2:

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

Code:



```
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
  }
```

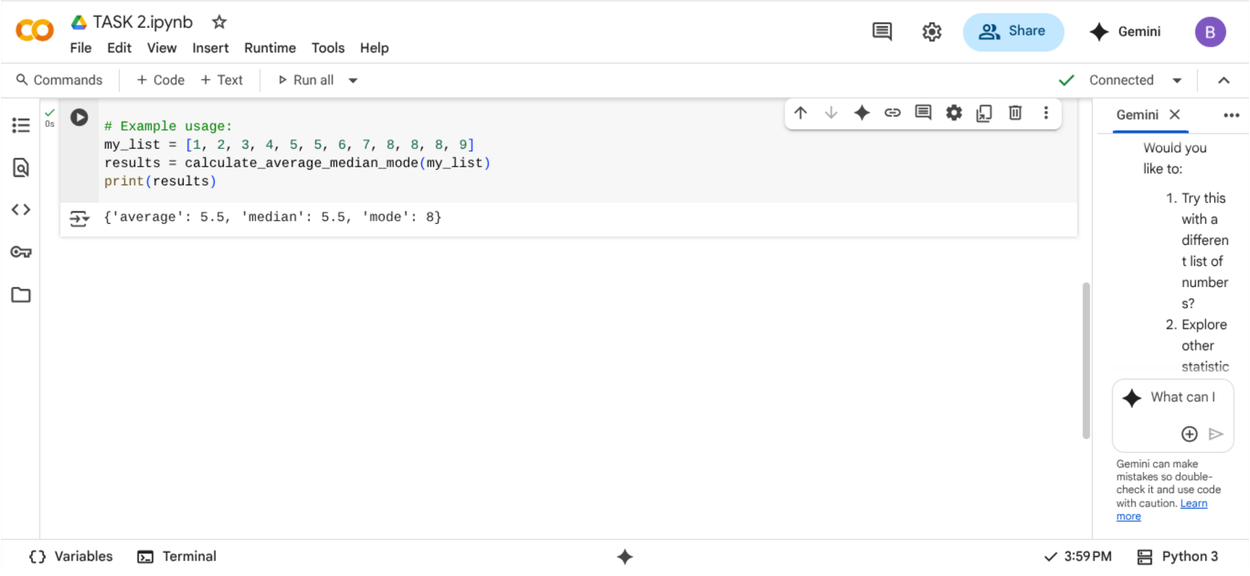
Would you like to:

1. Try this with a different list of numbers?
2. Explore other statistic

What can I

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

OUTPUT:



```
# 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}
```

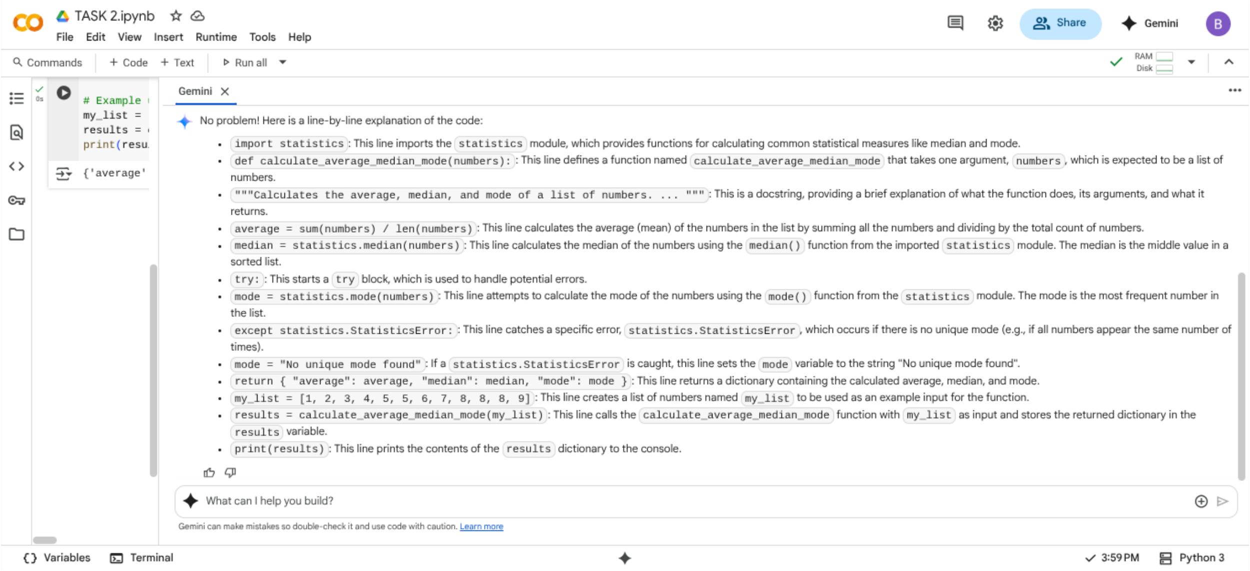
Would you like to:

1. Try this with a different list of numbers?
2. Explore other statistic

What can I

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

EXPLANATION:



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.

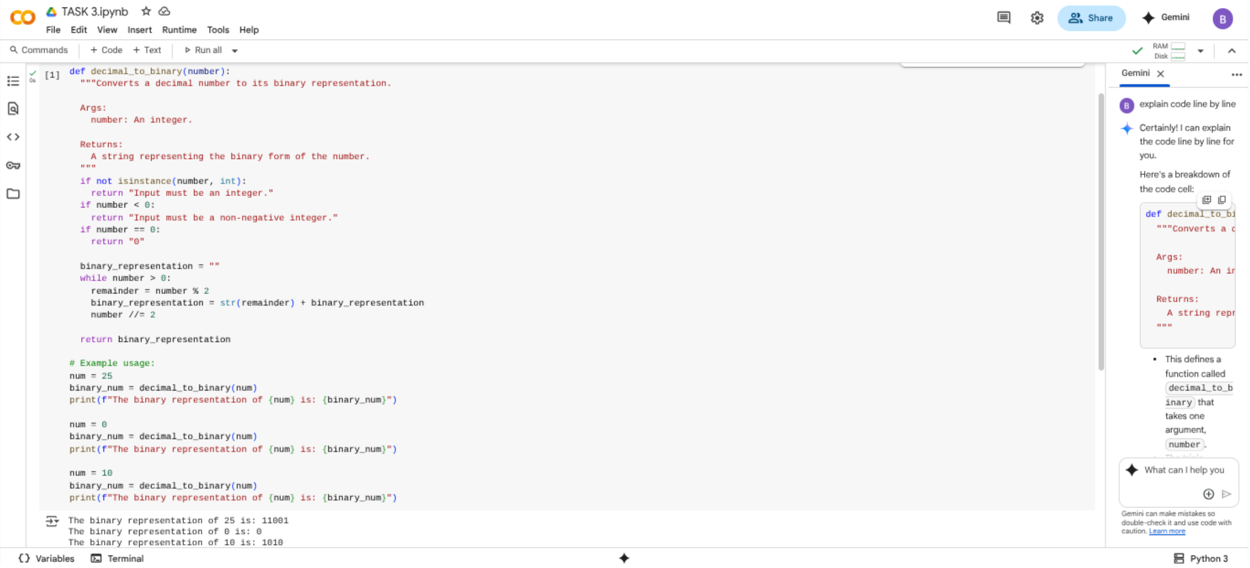
What can I help you build?

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
```

explain code line by line

Certainly! I can explain the code line by line for you.

Here's a breakdown of the code cell:

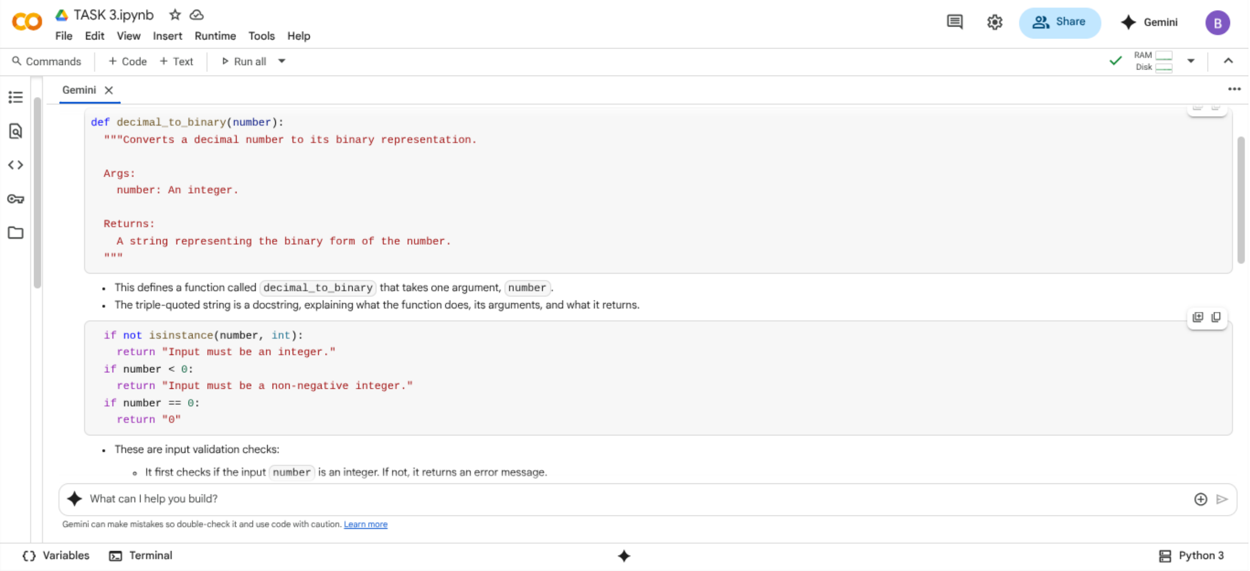
```
def decimal_to_bi
  """Converts a c

  Args:
    number: An ir

  Returns:
    A string repr
  """
```

- 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:



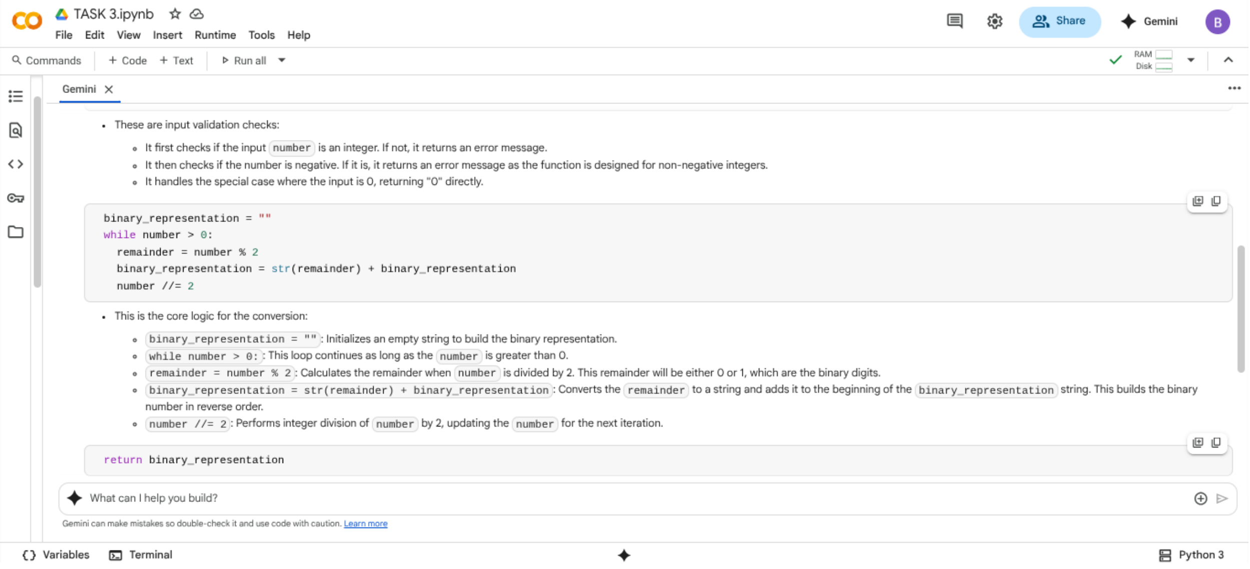
```
def decimal_to_binary(number):
  """Converts a decimal number to its binary representation.

  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



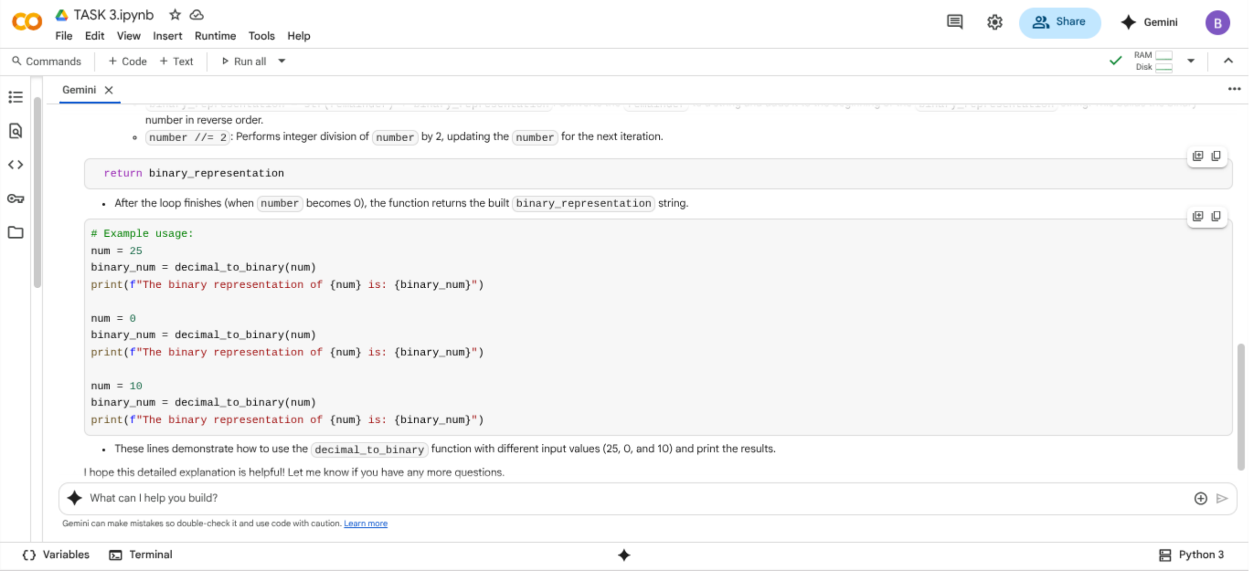
- These are input validation checks:
  - It first checks if the input `number` is an integer. If not, it returns an error message.
  - It then checks if the number is negative. If it is, it returns an error message as the function is designed for non-negative integers.
  - It handles the special case where the input is 0, returning "0" directly.

```
  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.

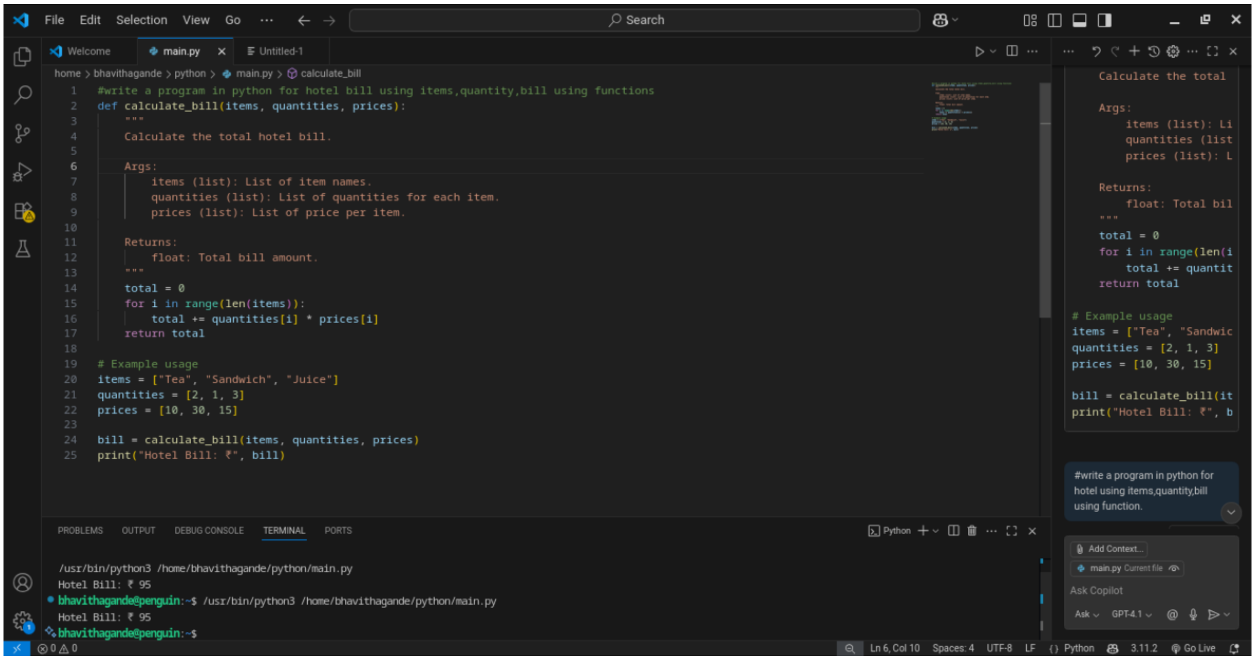
I hope this detailed explanation is helpful! Let me know if you have any more questions.

What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
Python 3

TASK-4:

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

Code and Output :


File Edit Selection View Go
Search
Welcome
main.py
Untitled-1
home > bhavithagande > python > main.py > calculate_bill
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
Add Context...
main.py Current file
Ask Copilot
Ask GPT-4.1
Ln 6, Col 10 Spaces: 4 UTF-8 LF {} Python 3.11.2 Go Live

```python
#write a program in python for hotel bill using items,quantity,bill using functions
def calculate_bill(items, quantities, prices):
    """
    Calculate the total hotel bill.

    Args:
        items (list): List of item names.
        quantities (list): List of quantities for each item.
        prices (list): List of price per item.

    Returns:
        float: Total bill amount.
    """
    total = 0
    for i in range(len(items)):
        total += quantities[i] * prices[i]
    return total

# Example usage
items = ["Tea", "Sandwich", "Juice"]
quantities = [2, 1, 3]
prices = [10, 30, 15]

bill = calculate_bill(items, quantities, prices)
print("Hotel Bill: ₹", bill)
```

```
/usr/bin/python3 /home/bhavithagande/python/main.py
Hotel Bill: ₹ 95
bhavithagande@penguin:~$ /usr/bin/python3 /home/bhavithagande/python/main.py
Hotel Bill: ₹ 95
bhavithagande@penguin:~$
```

```
    Calculate the total

    Args:
        items (list): Li
        quantities (list
        prices (list): L

    Returns:
        float: Total bil
    """
    total = 0
    for i in range(len(i
        total += quantit
    return total

# Example usage
items = ["Tea", "Sandwic
quantities = [2, 1, 3]
prices = [10, 30, 15]

bill = calculate_bill(it
print("Hotel Bill: ₹", b
```

#write a program in python for hotel using items,quantity,bill using function.

EXPLANATION:

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

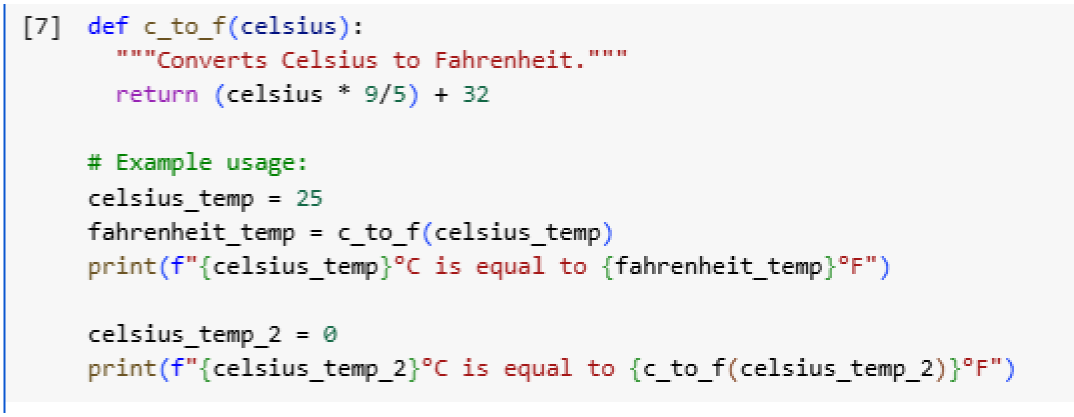
- It returns a dictionary containing:
    - The chosen room type (e.g., "double").
    - The number of nights (e.g., 3).
    - A list of selected services (e.g., ["breakfast", "wifi"]).
    - Example prices for different room types.
    - Example prices for available services.

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

TASK-5:

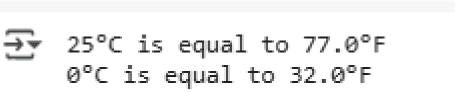
Compare how different prompts affect the quality of code output, using a simple function (like temperature conversion) as an example. give me simpler and shorter code.

Code:



```
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.