ASSSIGNMENT 9.3

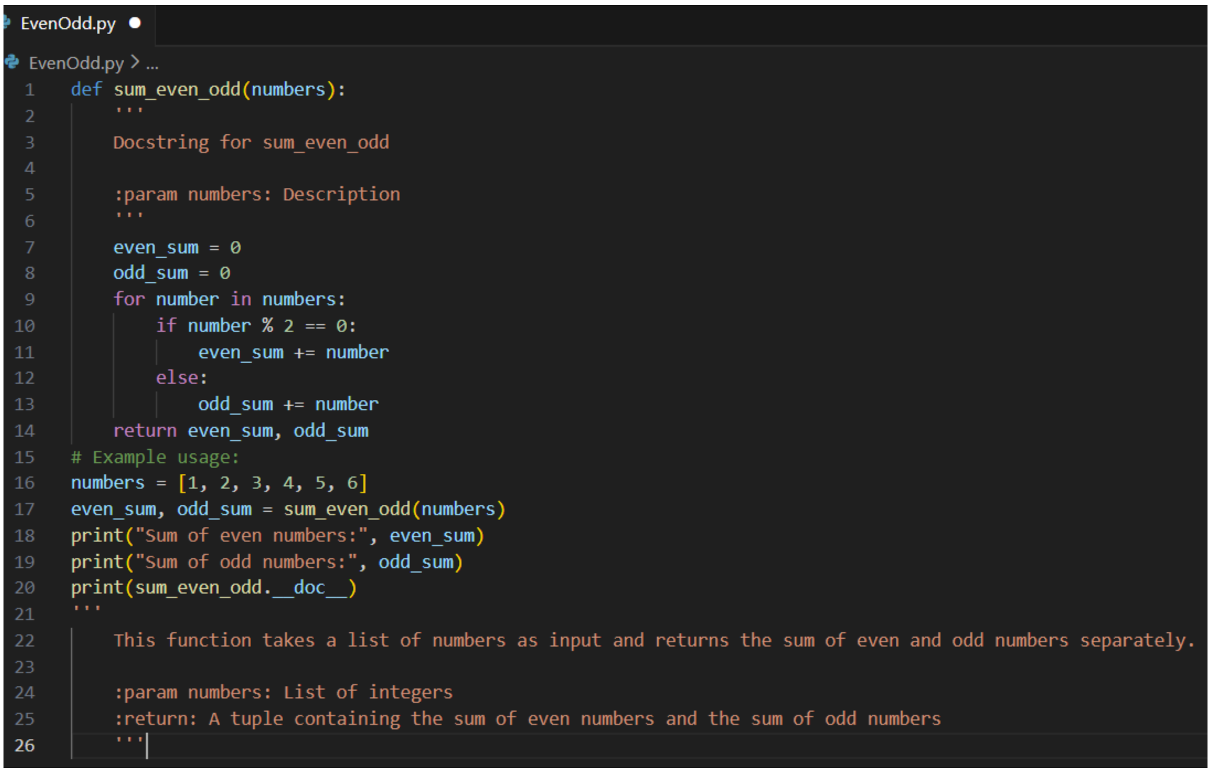
D.Charmi

2303A51314

Batch – 05

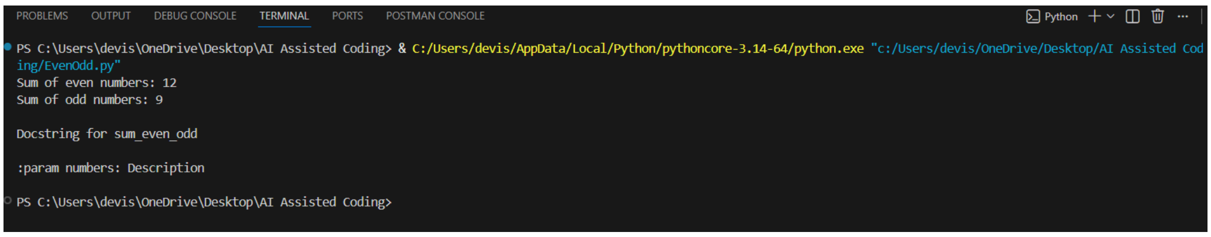
TASK – 01 :

Code :



```python
def sum_even_odd(numbers):
    '''
    Docstring for sum_even_odd

    :param numbers: Description
    '''
    even_sum = 0
    odd_sum = 0
    for number in numbers:
        if number % 2 == 0:
            even_sum += number
        else:
            odd_sum += number
    return even_sum, odd_sum
# Example usage:
numbers = [1, 2, 3, 4, 5, 6]
even_sum, odd_sum = sum_even_odd(numbers)
print("Sum of even numbers:", even_sum)
print("Sum of odd numbers:", odd_sum)
print(sum_even_odd.__doc__)
'''
    This function takes a list of numbers as input and returns the sum of even and odd numbers separately.

    :param numbers: List of integers
    :return: A tuple containing the sum of even numbers and the sum of odd numbers
    '''
```

Output :



```
PS C:\Users\devis\OneDrive\Desktop\AI Assisted Coding> & C:/Users/devis/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/devis/OneDrive/Desktop/AI Assisted Cod
ing/EvenOdd.py"
Sum of even numbers: 12
Sum of odd numbers: 9

Docstring for sum_even_odd

:param numbers: Description

PS C:\Users\devis\OneDrive\Desktop\AI Assisted Coding>
```

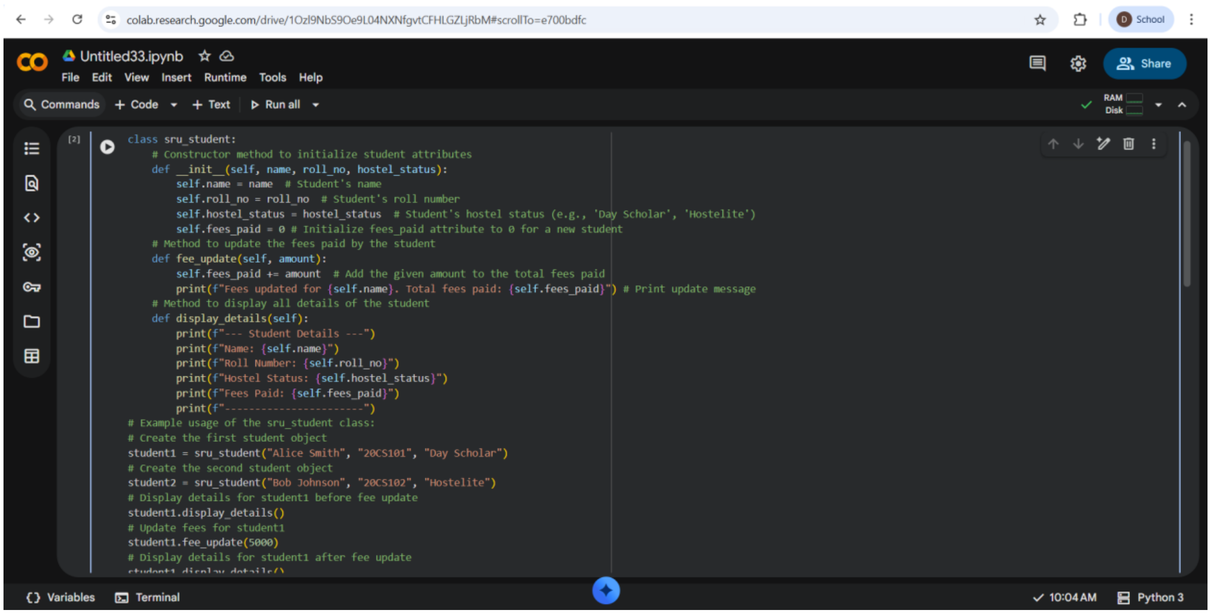
Explanation :

- The manual docstring is too brief and does not clearly explain the function.
- It fails to describe the return values, making it incomplete.
- The AI-generated docstring clearly explains the purpose of the function.

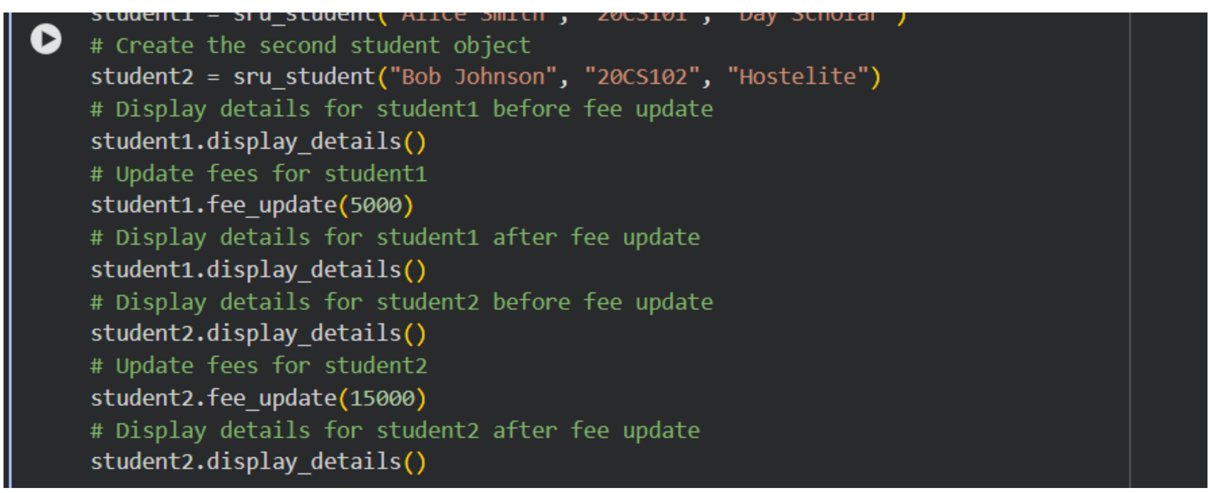
- It properly describes the input parameter and output.
- Overall, the AI-generated docstring is clearer, more correct, and more complete than the manual one.

TASK – 02 :

Code:

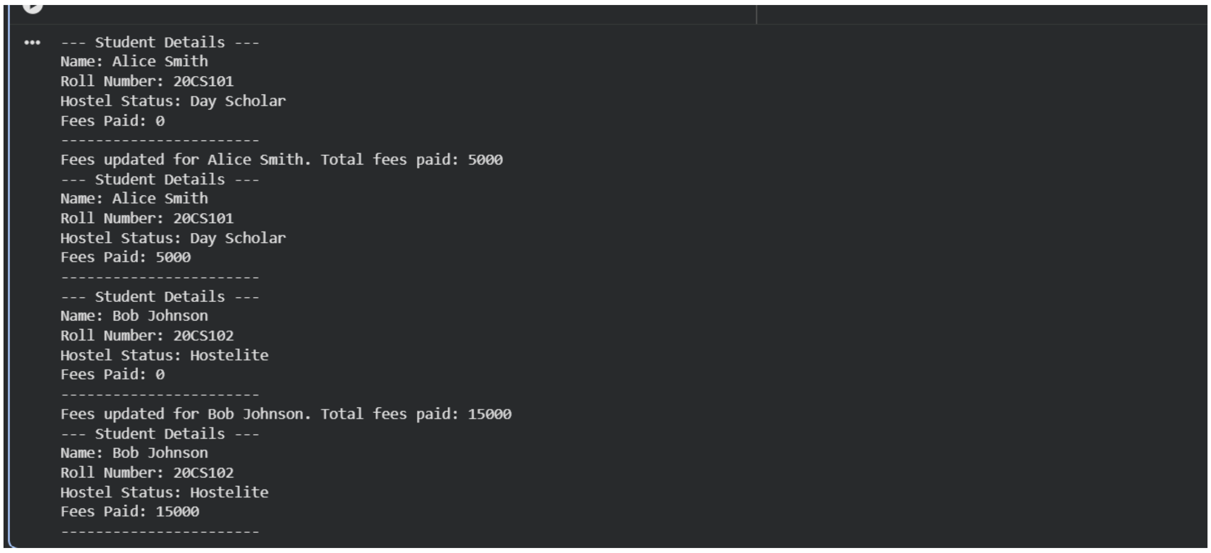


```python
class sru_student:
    # Constructor method to initialize student attributes
    def __init__(self, name, roll_no, hostel_status):
        self.name = name  # Student's name
        self.roll_no = roll_no  # Student's roll number
        self.hostel_status = hostel_status  # Student's hostel status (e.g., 'Day Scholar', 'Hostelite')
        self.fees_paid = 0 # Initialize fees_paid attribute to 0 for a new student
    # Method to update the fees paid by the student
    def fee_update(self, amount):
        self.fees_paid += amount  # Add the given amount to the total fees paid
        print(f"Fees updated for {self.name}. Total fees paid: {self.fees_paid}") # Print update message
    # Method to display all details of the student
    def display_details(self):
        print(f"--- Student Details ---")
        print(f"Name: {self.name}")
        print(f"Roll Number: {self.roll_no}")
        print(f"Hostel Status: {self.hostel_status}")
        print(f"Fees Paid: {self.fees_paid}")
        print(f"-----------------------")
# Example usage of the sru_student class:
# Create the first student object
student1 = sru_student("Alice Smith", "20CS101", "Day Scholar")
# Create the second student object
student2 = sru_student("Bob Johnson", "20CS102", "Hostelite")
# Display details for student1 before fee update
student1.display_details()
# Update fees for student1
student1.fee_update(5000)
# Display details for student1 after fee update
```



```python
# Create the second student object
student2 = sru_student("Bob Johnson", "20CS102", "Hostelite")
# Display details for student1 before fee update
student1.display_details()
# Update fees for student1
student1.fee_update(5000)
# Display details for student1 after fee update
student1.display_details()
# Display details for student2 before fee update
student2.display_details()
# Update fees for student2
student2.fee_update(15000)
# Display details for student2 after fee update
student2.display_details()
```

Output :



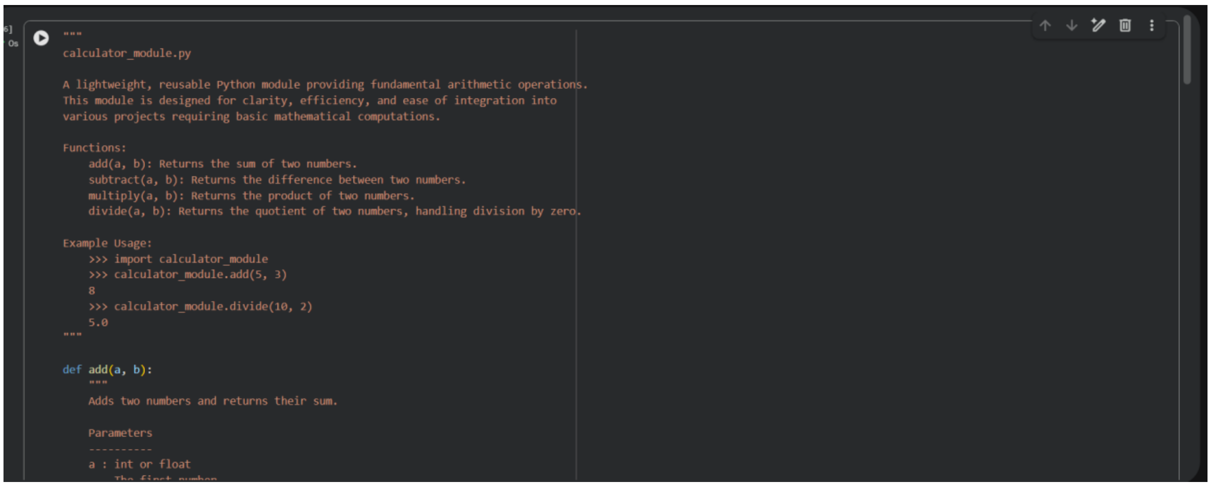
```
--- Student Details ---
Name: Alice Smith
Roll Number: 20CS101
Hostel Status: Day Scholar
Fees Paid: 0
-----------------------
Fees updated for Alice Smith. Total fees paid: 5000
--- Student Details ---
Name: Alice Smith
Roll Number: 20CS101
Hostel Status: Day Scholar
Fees Paid: 5000
-----------------------
--- Student Details ---
Name: Bob Johnson
Roll Number: 20CS102
Hostel Status: Hostelite
Fees Paid: 0
-----------------------
Fees updated for Bob Johnson. Total fees paid: 15000
--- Student Details ---
Name: Bob Johnson
Roll Number: 20CS102
Hostel Status: Hostelite
Fees Paid: 15000
-----------------------
```

Explanation :

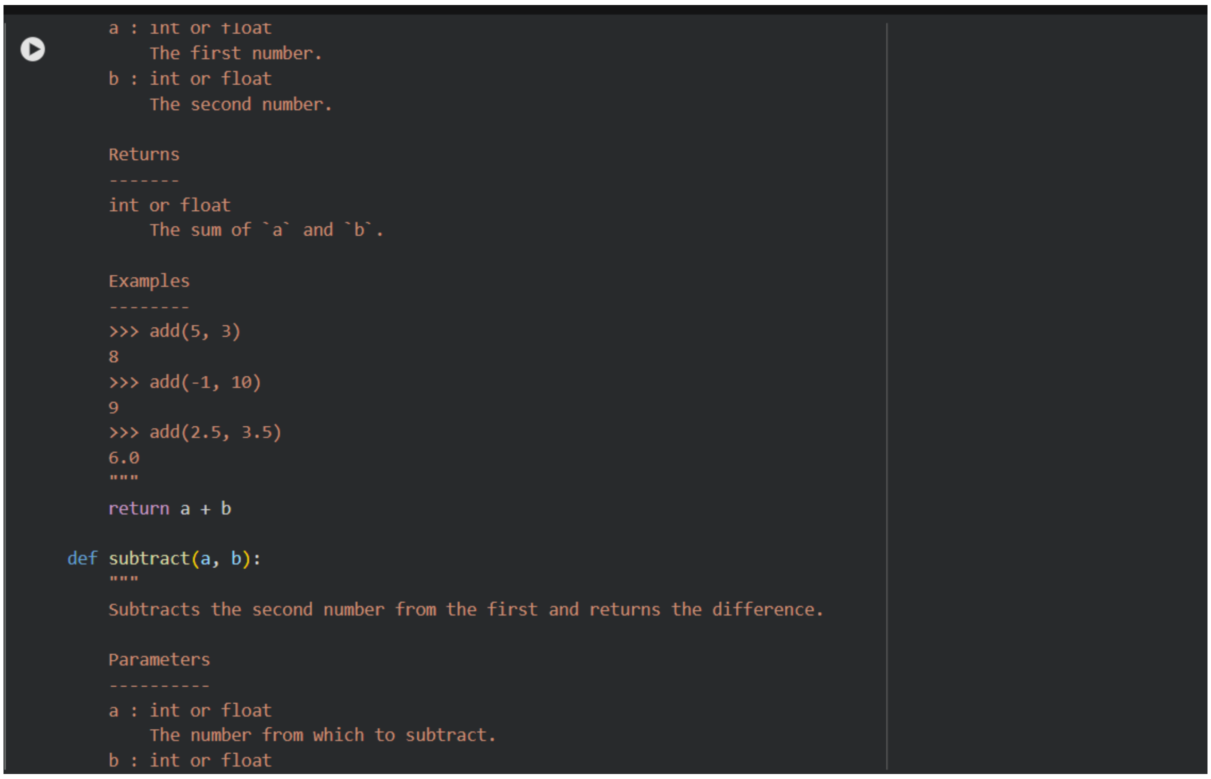
- Manual documentation is written by developers who understand the code deeply and can explain its purpose clearly.
- It gives better explanations, but it takes more time and effort to create.
- AI-generated documentation is created automatically and is quick and consistent.
- However, AI documentation may be too general and may not capture the real intention behind the code.
- Structured documentation explains what each function does, its inputs, and outputs in an organized way.
- This makes the code easier to read, maintain, and work on as a team.

TASK – 03:

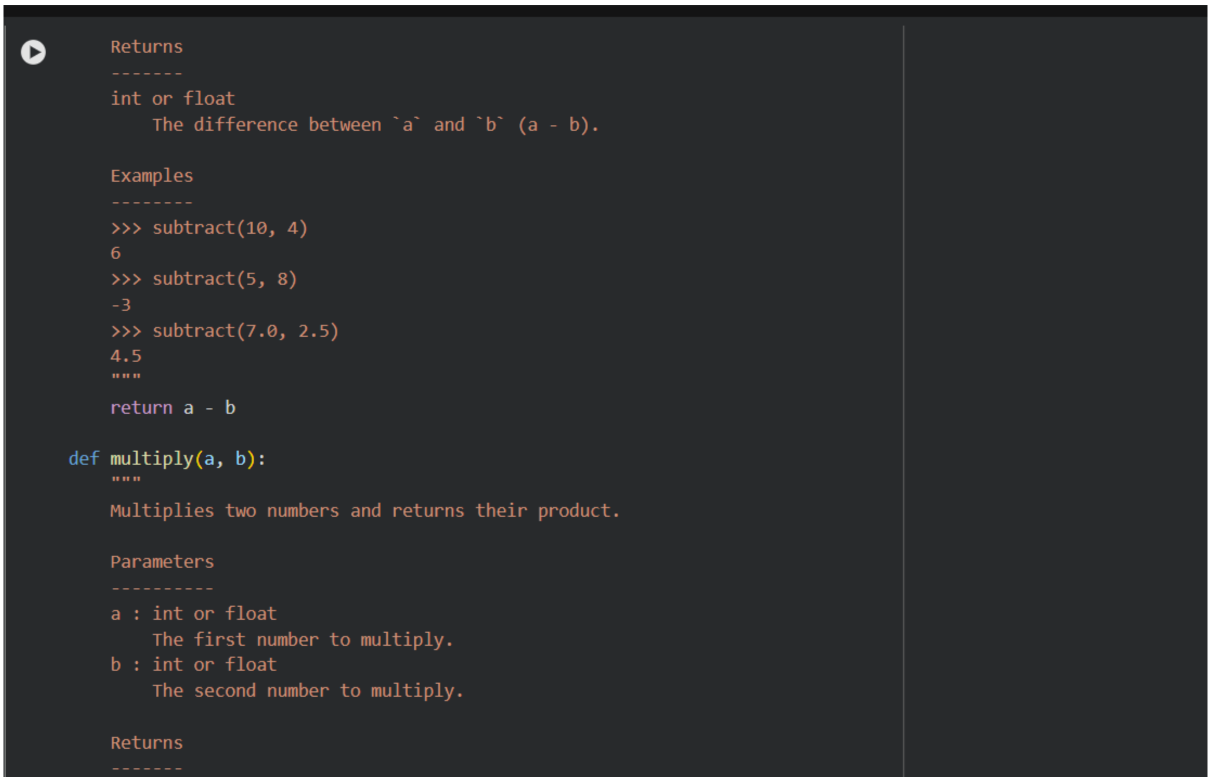
Code :



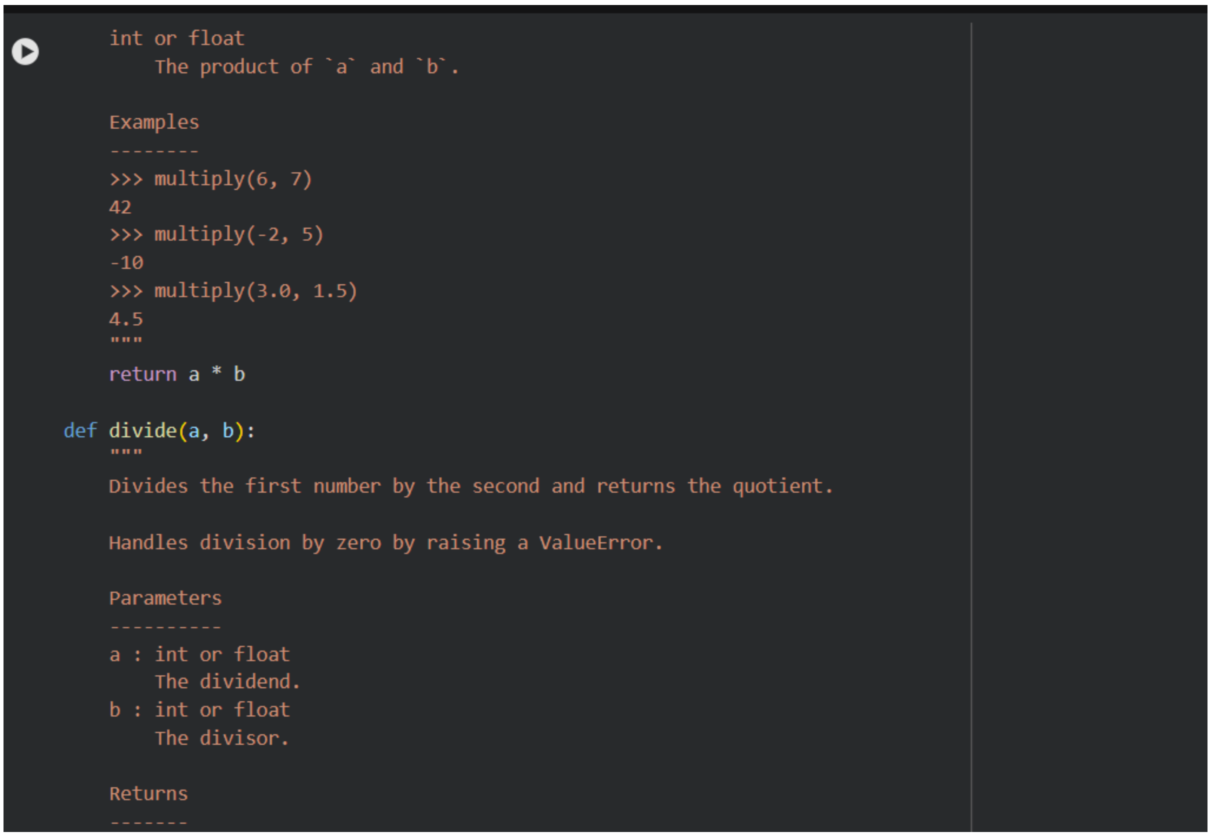
```
"""
calculator_module.py

A lightweight, reusable Python module providing fundamental arithmetic operations.
This module is designed for clarity, efficiency, and ease of integration into
various projects requiring basic mathematical computations.

Functions:
    add(a, b): Returns the sum of two numbers.
    subtract(a, b): Returns the difference between two numbers.
    multiply(a, b): Returns the product of two numbers.
    divide(a, b): Returns the quotient of two numbers, handling division by zero.

Example Usage:
    >>> import calculator_module
    >>> calculator_module.add(5, 3)
    8
    >>> calculator_module.divide(10, 2)
    5.0
"""

def add(a, b):
    """
    Adds two numbers and returns their sum.

    Parameters
    ----------
    a : int or float
```



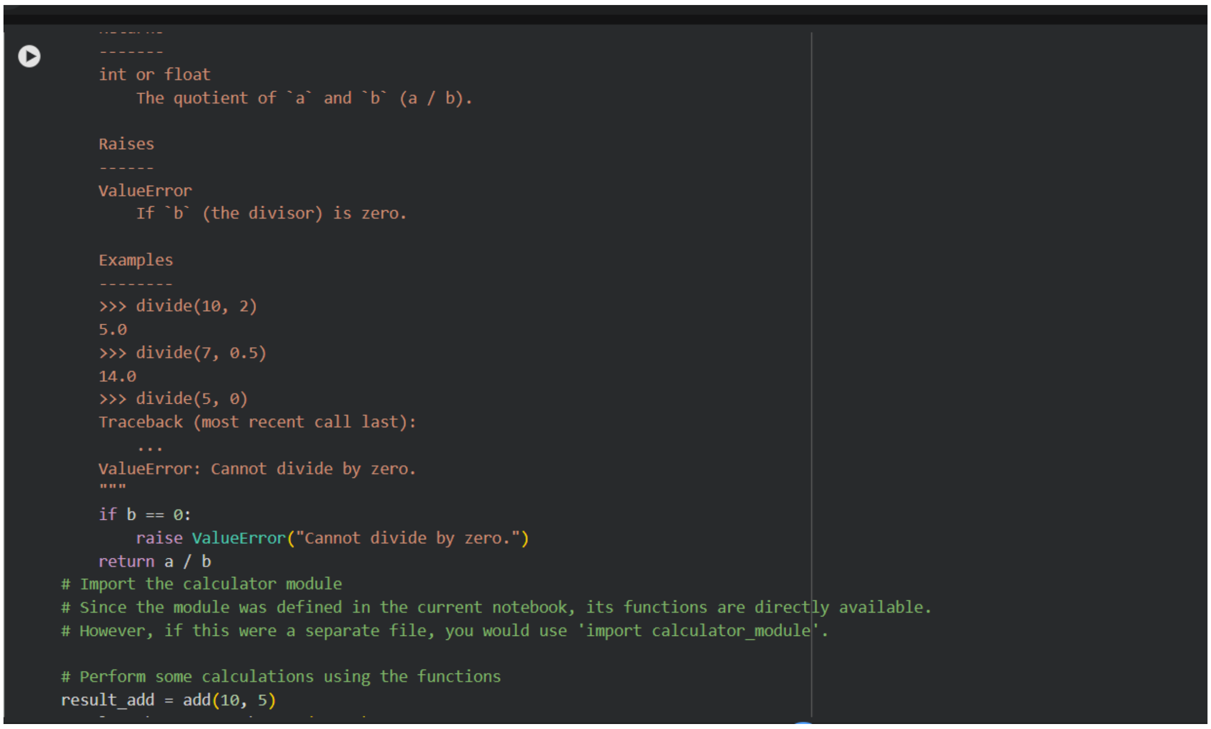
```
    a : int or float
        The first number.
    b : int or float
        The second number.

    Returns
    -------
    int or float
        The sum of `a` and `b`.

    Examples
    --------
    >>> add(5, 3)
    8
    >>> add(-1, 10)
    9
    >>> add(2.5, 3.5)
    6.0
    """
    return a + b

def subtract(a, b):
    """
    Subtracts the second number from the first and returns the difference.

    Parameters
    ----------
    a : int or float
        The number from which to subtract.
    b : int or float
```



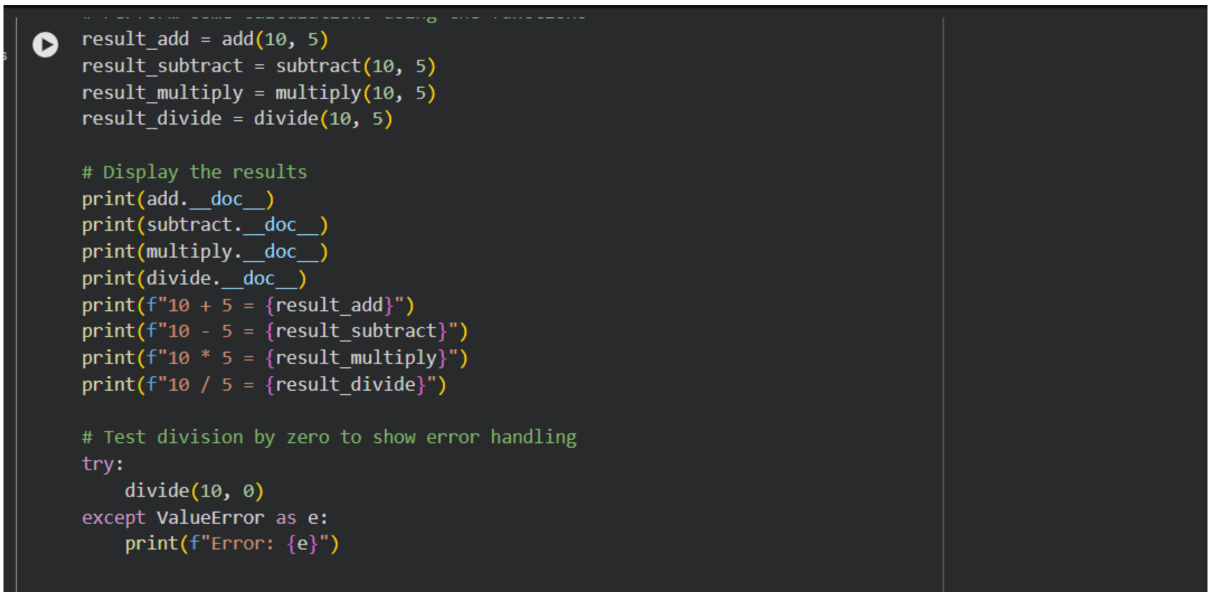
```
    Returns
    -------
    int or float
        The difference between `a` and `b` (a - b).

    Examples
    --------
    >>> subtract(10, 4)
    6
    >>> subtract(5, 8)
    -3
    >>> subtract(7.0, 2.5)
    4.5
    """
    return a - b

def multiply(a, b):
    """
    Multiplies two numbers and returns their product.

    Parameters
    ----------
    a : int or float
        The first number to multiply.
    b : int or float
        The second number to multiply.

    Returns
    -------
```



```
    int or float
        The product of `a` and `b`.

    Examples
    --------
    >>> multiply(6, 7)
    42
    >>> multiply(-2, 5)
    -10
    >>> multiply(3.0, 1.5)
    4.5
    """
    return a * b

def divide(a, b):
    """
    Divides the first number by the second and returns the quotient.

    Handles division by zero by raising a ValueError.

    Parameters
    ----------
    a : int or float
        The dividend.
    b : int or float
        The divisor.

    Returns
    -------
```

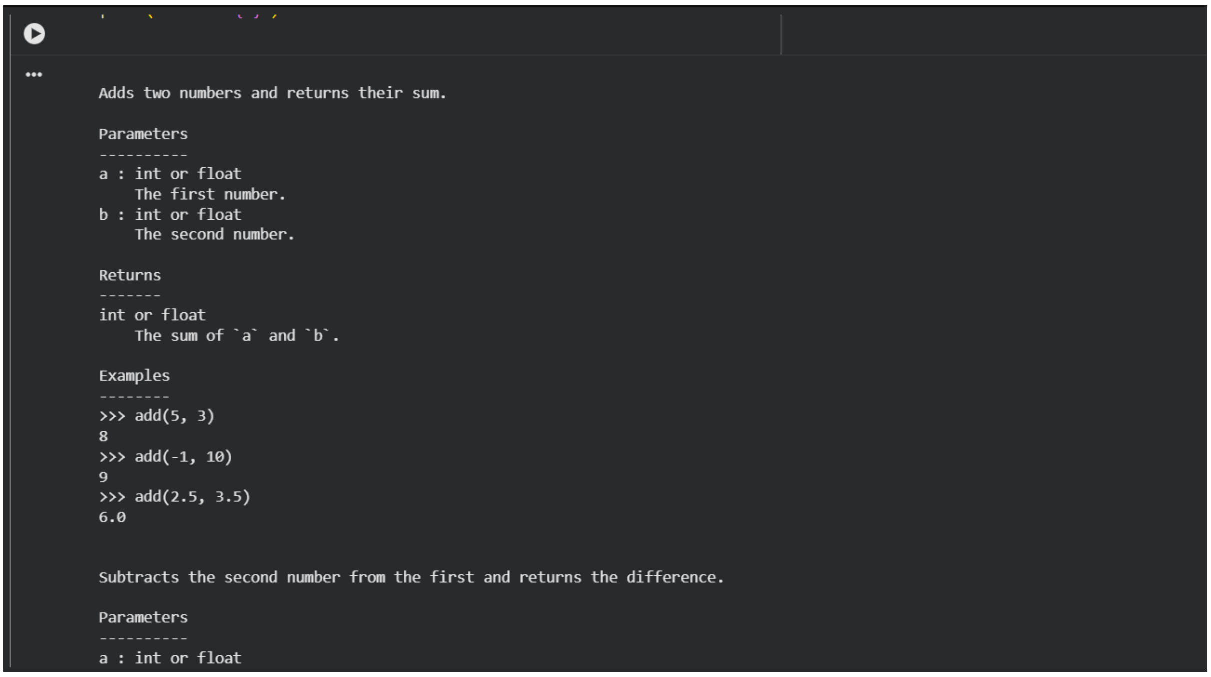


```
    -------
    int or float
        The quotient of `a` and `b` (a / b).

    Raises
    ------
    ValueError
        If `b` (the divisor) is zero.

    Examples
    --------
    >>> divide(10, 2)
    5.0
    >>> divide(7, 0.5)
    14.0
    >>> divide(5, 0)
    Traceback (most recent call last):
        ...
    ValueError: Cannot divide by zero.
    """
    if b == 0:
        raise ValueError("Cannot divide by zero.")
    return a / b
# Import the calculator module
# Since the module was defined in the current notebook, its functions are directly available.
# However, if this were a separate file, you would use 'import calculator_module'.

# Perform some calculations using the functions
result_add = add(10, 5)
```

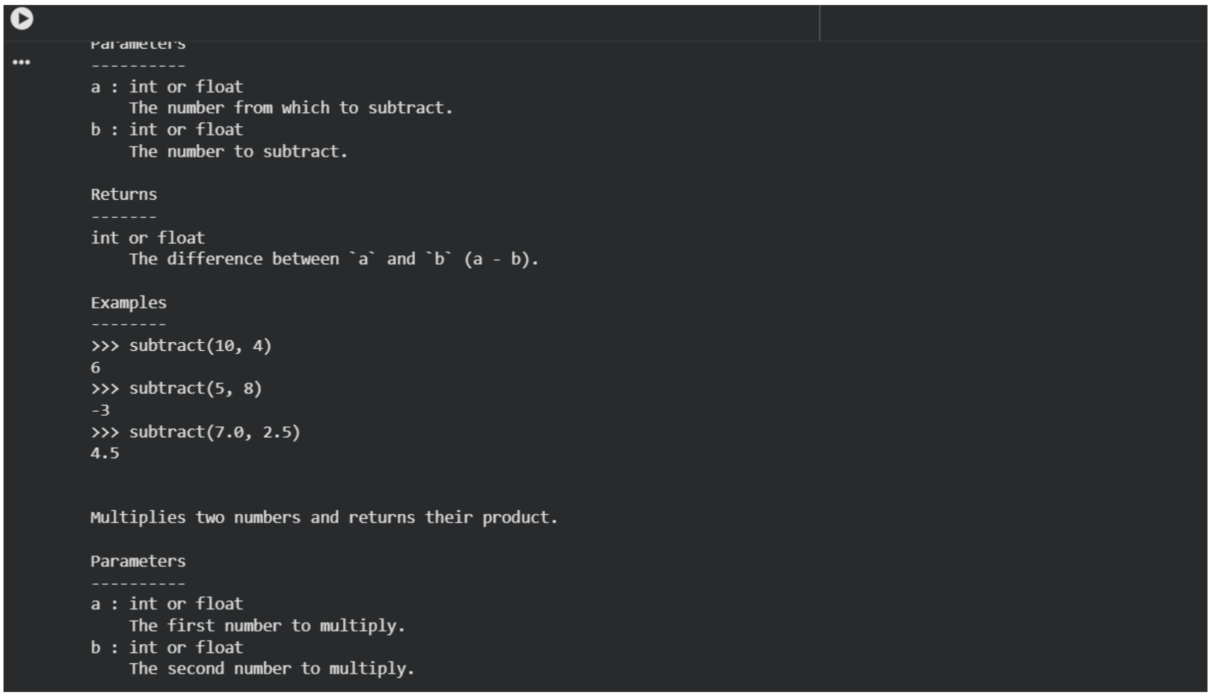


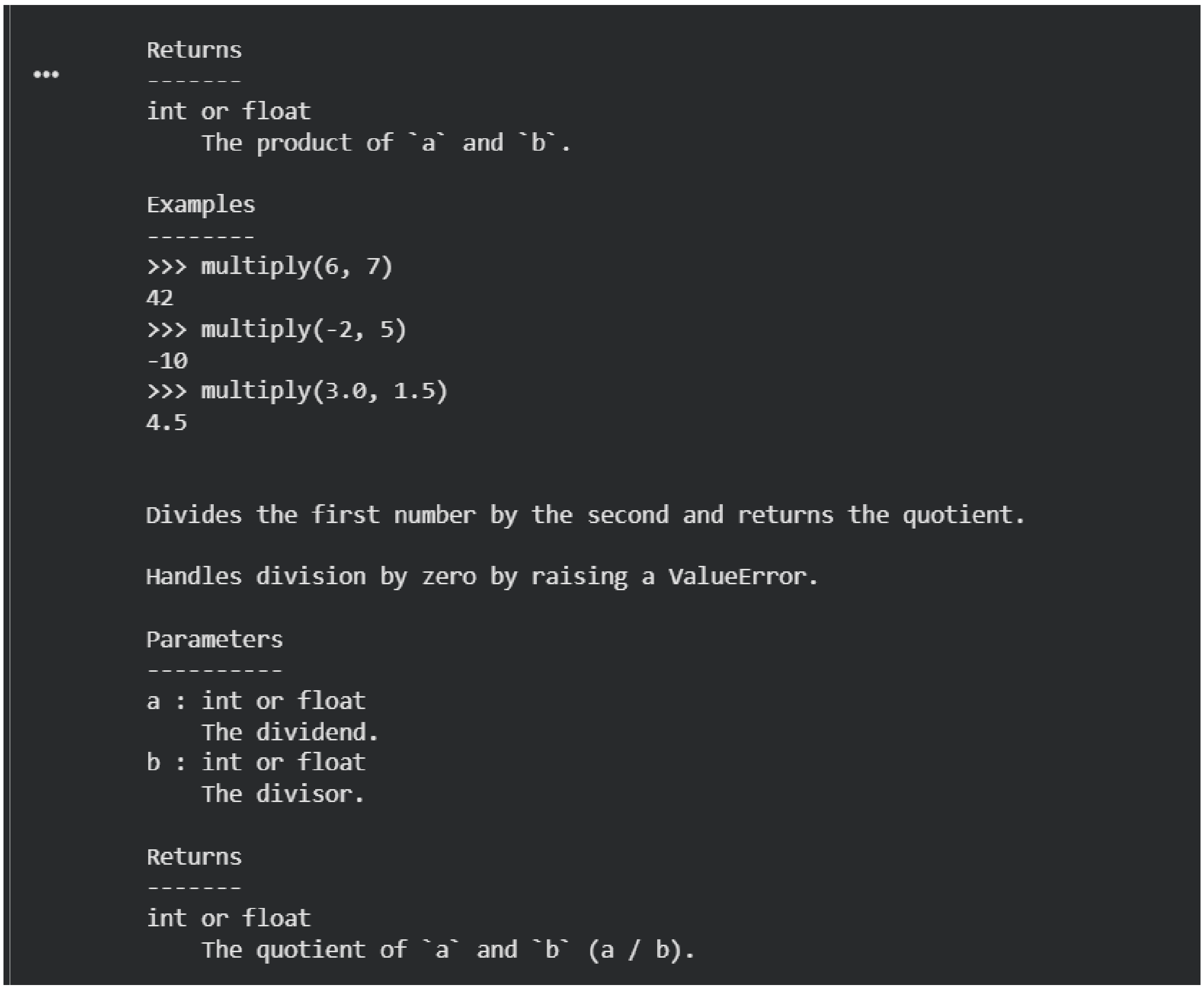
```python
result_add = add(10, 5)
result_subtract = subtract(10, 5)
result_multiply = multiply(10, 5)
result_divide = divide(10, 5)

# Display the results
print(add.__doc__)
print(subtract.__doc__)
print(multiply.__doc__)
print(divide.__doc__)
print(f"10 + 5 = {result_add}")
print(f"10 - 5 = {result_subtract}")
print(f"10 * 5 = {result_multiply}")
print(f"10 / 5 = {result_divide}")

# Test division by zero to show error handling
try:
    divide(10, 0)
except ValueError as e:
    print(f"Error: {e}")
```

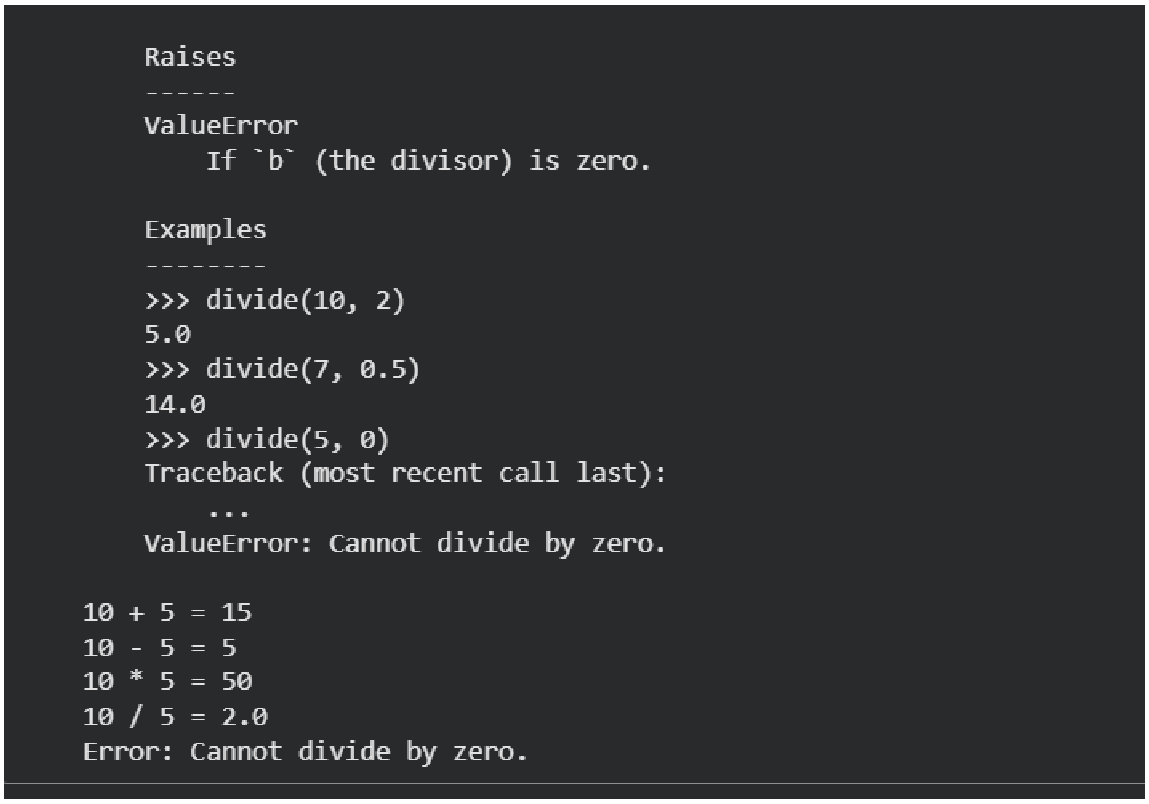
Output :



```
Adds two numbers and returns their sum.

Parameters
----------
a : int or float
    The first number.
b : int or float
    The second number.

Returns
-------
int or float
    The sum of `a` and `b`.

Examples
--------
>>> add(5, 3)
8
>>> add(-1, 10)
9
>>> add(2.5, 3.5)
6.0


Subtracts the second number from the first and returns the difference.

Parameters
----------
a : int or float
```


```
Parameters
----------
a : int or float
    The number from which to subtract.
b : int or float
    The number to subtract.

Returns
-------
int or float
    The difference between `a` and `b` (a - b).

Examples
--------
>>> subtract(10, 4)
6
>>> subtract(5, 8)
-3
>>> subtract(7.0, 2.5)
4.5


Multiplies two numbers and returns their product.

Parameters
----------
a : int or float
    The first number to multiply.
b : int or float
    The second number to multiply.
```


```
Returns
-------
int or float
    The product of `a` and `b`.

Examples
--------
>>> multiply(6, 7)
42
>>> multiply(-2, 5)
-10
>>> multiply(3.0, 1.5)
4.5


Divides the first number by the second and returns the quotient.

Handles division by zero by raising a ValueError.

Parameters
----------
a : int or float
    The dividend.
b : int or float
    The divisor.

Returns
-------
int or float
    The quotient of `a` and `b` (a / b).
```



```
    Raises
    ------
    ValueError
        If `b` (the divisor) is zero.

    Examples
    --------
    >>> divide(10, 2)
    5.0
    >>> divide(7, 0.5)
    14.0
    >>> divide(5, 0)
    Traceback (most recent call last):
        ...
    ValueError: Cannot divide by zero.

10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2.0
Error: Cannot divide by zero.
```

Explanation :

- Manual documentation is written with a full understanding of the code logic and design decisions, while AI-generated documentation is created automatically based on code patterns.
- Manual documentation follows strict documentation standards (such as NumPy style) more accurately, whereas AI documentation may not always fully follow these formats.
- AI-generated documentation is faster and saves time, but it can miss edge cases or exceptions.
- Manual documentation is usually more precise and reliable, especially for reusable or shared modules.
- AI documentation improves readability for beginners.