📂 Python Module-Based Math Operations Documentation

1. Project Overview

This project showcases how to structure a Python application using **modules** to separate functionality from the main execution logic. It consists of two files:

1. **`math_operations.py`**: A module containing the core arithmetic functions.
2. **`MathModuleMain.py`**: The main script that **imports** the functions from `math_operations.py` and executes the calculations.

This structure is a fundamental concept in software engineering, promoting **reusability** and **organization** of code.

---

2. File 1: `math_operations.py` (The Module)

This file defines the functions that perform the four basic arithmetic operations. It acts as a **library** of mathematical tools that can be reused by any other Python script.

2.1 Arithmetic Functions

Function Name	Parameters	Description
`add(a, b)`	a, b	Returns the **sum** of a and b.
`subtract(a, b)`	a, b	Returns the **difference** of a and b.
`multiply(a, b)`	a, b	Returns the **product** of a and b.
`divide(a, b)`	a, b	Returns the **quotient** of a and b, or an error message if b is zero.

2.2 Critical Implementation Detail: Division

The divide function incorporates basic **error handling** to manage the mathematical impossibility of dividing by zero.

```
Python

def divide(a, b):
    if b != 0:
        return a / b
    else:
        return "Division by zero is not allowed"
```

If the second number (b) is zero, it returns a descriptive error message instead of causing a runtime error.

---

3. File 2: MathModuleMain.py (The Main Script)

This file demonstrates how to **consume** the functionality provided by the module.

3.1 Importing the Module

The script begins with the import statement.

```
Python

import math_operations as mo
```

- **import math_operations**: This tells Python to load all functions and variables from the math_operations.py file.
- **as mo**: This is an **alias** that allows the script to refer to the module using the shorter name, **mo**. This improves code readability and simplifies calling the functions.

3.2 Execution Flow

1. **Define Variables**: Two numerical variables, num1 (10) and num2 (5), are initialized.
2.
```
Python

num1 = 10
num2 = 5
```
3.
4.
5. **Function Calls**: The script calls the functions using the module alias (mo) followed by the function name and the dot operator (e.g., mo.add).

- `mo.add(num1, num2)` calls the `add` function within the imported module.
- The results of these function calls are then printed directly to the console.

3.3 Expected Output

The script executes all four operations and prints the result of each calculation:

```
Addition:  15
Subtraction:  5
Multiplication:  50
Division:  2.0
```