Calculation Docs

None

None

None

Table of contents

1. Welcome to MkDocs

For full documentation visit mkdocs.org.

1.1 Commands

- `mkdocs new [dir-name]` - Create a new project.
- `mkdocs serve` - Start the live-reloading docs server.
- `mkdocs build` - Build the documentation site.
- `mkdocs -h` - Print help message and exit.

1.2 Project layout

```
mkdocs.yml    # The configuration file.
docs/
    index.md  # The documentation homepage.
    ...       # Other markdown pages, images and other files.
```

2. Reference

Compute and return the sum of two numbers.

Examples:

```
>>> add(4.0, 2.0)
6.0
>>> add(4, 2)
6.0
```

Parameters:

Name	Type	Description	Default
`a`	`float`	A number representing the first addend in the addition.	*required*
`b`	`float`	A number representing the second addend in the addition.	*required*

Returns:

Name	Type	Description
`num`	`float`	A number representing the arithmetic sum of `a` and `b`.

Source code in `calculations.py`

```
20  def add(a, b):
21      """Compute and return the sum of two numbers.
22
23      Examples:
24          >>> add(4.0, 2.0)
25          6.0
26          >>> add(4, 2)
27          6.0
28
29      Args:
30          a (float): A number representing the first addend in the addition.
31          b (float): A number representing the second addend in the addition.
32
33      Returns:
34          num (float): A number representing the arithmetic sum of `a` and `b`.
35      """
36      return float(a + b)
```

Compute and return the difference between two numbers.

Examples:

```
>>> subtract(6.0, 4.0)
2.0
>>> subtract(10, 3)
7.0
```

Parameters:

Name	Type	Description	Default
`a`	`float`	A number representing the minuend in the subtraction.	*required*
`b`	`float`	A number representing the subtrahend in the subtraction.	*required*

Returns:

Name	Type	Description
`num`	`float`	A number representing the arithmetic difference between `a` and `b`.

Source code in `calculations.py`

```
def subtract(a, b):
    """Compute and return the difference between two numbers.

    Examples:
        >>> subtract(6.0, 4.0)
        2.0
        >>> subtract(10, 3)
        7.0

    Args:
        a (float): A number representing the minuend in the subtraction.
        b (float): A number representing the subtrahend in the subtraction.

    Returns:
        num (float): A number representing the arithmetic difference between `a` and `b`.
    """
    return float(a - b)
```

Compute and return the product of two numbers.

Examples:

```
>>> multiply(3.0, 2.0)
6.0
>>> multiply(7, 5)
35.0
```

Parameters:

Name	Type	Description	Default
`a`	`float`	A number representing the multiplicand in the multiplication.	*required*
`b`	`float`	A number representing the multiplier in the multiplication.	*required*

Returns:

Name	Type	Description
`num`	`float`	A number representing the arithmetic product of `a` and `b`.

Source code in `calculations.py`

```
def multiply(a, b):
    """Compute and return the product of two numbers.

    Examples:
        >>> multiply(3.0, 2.0)
        6.0
        >>> multiply(7, 5)
        35.0

    Args:
        a (float): A number representing the multiplicand in the multiplication.
        b (float): A number representing the multiplier in the multiplication.

    Returns:
        num (float): A number representing the arithmetic product of `a` and `b`.
    """
    return float(a * b)
```

Compute and return the quotient of two numbers.

Examples:

```
>>> divide(8.0, 2.0)
4.0
>>> divide(9, 3)
3.0
>>> divide(10, 0)
Traceback (most recent call last):
    ...
ZeroDivisionError: division by zero
```

Parameters:

Name	Type	Description	Default
`a`	`float`	A number representing the dividend in the division.	*required*
`b`	`float`	A number representing the divisor in the division.	*required*

Raises:

Type	Description
`ZeroDivisionError`	If `b` is zero, indicating division by zero is not allowed.

Returns:

Name	Type	Description
`num`	`float`	A number representing the arithmetic quotient of `a` divided by `b`.

Source code in `calculations.py`

```
74  def divide(a, b):
75      """Compute and return the quotient of two numbers.
76
77      Examples:
78          >>> divide(8.0, 2.0)
79          4.0
80          >>> divide(9, 3)
81          3.0
82          >>> divide(10, 0)
83          Traceback (most recent call last):
84              ...
85          ZeroDivisionError: division by zero
86
87      Args:
88          a (float): A number representing the dividend in the division.
89          b (float): A number representing the divisor in the division.
90
91      Raises:
92          ZeroDivisionError: If `b` is zero, indicating division by zero is not allowed.
93
94      Returns:
95          num (float): A number representing the arithmetic quotient of `a` divided by `b`.
96      """
97      if b == 0:
98          raise ZeroDivisionError("division by zero")
99      return float(a / b)
```

A test class for the calculations module.

This class is designed to contain unit tests for the functions defined in the calculations module, including add, subtract, multiply, and divide. Each method within this class should test a specific function from the calculations module for correctness, handling of edge cases, and error conditions.

Source code in `calculations.py`

```
class TestCalculations:
    """
    A test class for the calculations module.

    This class is designed to contain unit tests for the functions defined in the calculations module,
    including add, subtract, multiply, and divide. Each method within this class should test a specific
    function from the calculations module for correctness, handling of edge cases, and error conditions.
    """
    def __init__(self):
        """
        Initializes the TestCalculations class.

        This constructor can be used to set up any prerequisites for the tests,
        such as initializing variables or setting up test data.
        """
        print("Initializing TestCalculations Class...")
```

2.1 `__init__()`

Initializes the TestCalculations class.

This constructor can be used to set up any prerequisites for the tests, such as initializing variables or setting up test data.

Source code in `calculations.py`

```
def __init__(self):
    """
    Initializes the TestCalculations class.

    This constructor can be used to set up any prerequisites for the tests,
    such as initializing variables or setting up test data.
    """
    print("Initializing TestCalculations Class...")
```