

Project Documentation

Generated: 2025-12-04T10:37:25.945978 UTC

Index

hello_world

add_numbers

Calculator

Folder Summary

Overview

This codebase contains utilities for basic arithmetic operations and data processing. It includes simple functions

Architecture & Key Components

Based on the documentation files analyzed, the key components include:

`hello_world()`: A simple function that returns a greeting message

`add_numbers(a, b)`: A function that adds two numeric values

`Calculator` class: A class that provides multiplication operations with history tracking

Data Flow & Workflow

The components work together to process numerical data:

Simple operations can be performed directly with functions like `add_numbers()`

For more complex operations with history tracking, the `Calculator` class is used

Results are returned immediately, and operations are stored in the calculator's history

The history can be retrieved using the `get_history()` method

How to Get Started (For New Developers)

To use this codebase:

For simple operations, import and use the standalone functions like `add_numbers()`

For operations with history tracking, create an instance of the `Calculator` class

Call methods like `multiply()` on the calculator instance

Retrieve the operation history using `get_history()`

Limitations & Possible Improvements

Current limitations include:

Limited to basic arithmetic operations (addition and multiplication)

History tracking grows unbounded and may consume memory over time

No advanced error handling for edge cases

Possible improvements:

Add more arithmetic operations (subtraction, division, etc.)

Implement history size limits or persistence

Add comprehensive error handling and input validation

Create more detailed documentation with usage examples

Project Summary

Overview

This codebase contains utilities for basic arithmetic operations and data processing. It includes simple functions

Architecture & Key Components

Based on the documentation files analyzed, the key components include:

`hello_world()`: A simple function that returns a greeting message

`add_numbers(a, b)`: A function that adds two numeric values

`Calculator` class: A class that provides multiplication operations with history tracking

Data Flow & Workflow

The components work together to process numerical data:

Simple operations can be performed directly with functions like `add_numbers()`

For more complex operations with history tracking, the `Calculator` class is used

Results are returned immediately, and operations are stored in the calculator's history

The history can be retrieved using the `get_history()` method

How to Get Started (For New Developers)

To use this codebase:

For simple operations, import and use the standalone functions like `add_numbers()`

For operations with history tracking, create an instance of the `Calculator` class

Call methods like `multiply()` on the calculator instance

Retrieve the operation history using `get_history()`

Limitations & Possible Improvements

Current limitations include:

Limited to basic arithmetic operations (addition and multiplication)

History tracking grows unbounded and may consume memory over time

No advanced error handling for edge cases

Possible improvements:

Add more arithmetic operations (subtraction, division, etc.)

Implement history size limits or persistence

Add comprehensive error handling and input validation

Create more detailed documentation with usage examples

Add Two Numbers

ID: test1.py_add_numbers

Signature: add_numbers(a, b)

Summary: Adds two numeric values and returns the result.

Description

This function accepts two arguments, a and b, and returns their sum. It assumes both inputs support the + operator.

Example

```
result = add_numbers(2, 3)
```

```
print(result) # 5
```

Edge cases & Notes

Passing non-numeric types like strings or lists will raise a `TypeError`.

Very large integers may still work, but floating-point addition can introduce rounding errors.

Calculator Class

ID: test1.py_Calculator

Signature: Calculator()

Summary: A simple calculator class for basic arithmetic operations with history tracking.

Description

This class provides basic arithmetic operations, specifically multiplication, with built-in history tracking. When initialized, the history is empty.

Example

```
calc = Calculator()
result = calc.multiply(4, 5)
print(result) # 20
print(calc.get_history()) # ['4 * 5 = 20']
```

Edge cases & Notes

Multiplying very large numbers may lead to overflow in some systems.

Non-numeric inputs will raise a `TypeError` during multiplication.

History grows unbounded with each operation, which could consume significant memory over time.

Hello World Function

ID: test1.py_hello_world

Signature: hello_world()

Summary: Returns a simple greeting message.

Description

This function is a basic implementation that returns the classic "Hello, World!" greeting. It takes no parameters and

Example

```
greeting = hello_world()
```

```
print(greeting) # Output: Hello, World!
```

Edge cases & Notes

No edge cases - function always returns the same static string.

Function does not accept any parameters, so no type validation needed.