Lucenext Calculator

Generated by Doxygen 1.9.1

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 src Namespace Reference

Namespaces

- calculator
- gui
- math_lib
- stddev
- test_math_lib

4.2 src.calculator Namespace Reference

Classes

- class Parser

  Class to parse the tokenized input.

Functions

- def tokenize (expr)

  Function to tokenize the input expression.
- def eval_node (node, ns)

  Function to evaluate the AST node.
- def build_safe_ns (last_ans, base=10)

  Function to build a safe namespace for the calculator.
- def evaluate (expr, base=10)

  Function to evaluate the expression.

4.2.1 Function Documentation

4.2.1.1 build_safe_ns()

```
def src.calculator.build_safe_ns (
              last_ans,
              base = 10 )
```

Function to build a safe namespace for the calculator.

Parameters

last_ans	last answer used in the calculator
base	base for number conversion

Returns

dictionary of functions and constants

4.2.1.2 eval_node()

```
def src.calculator.eval_node (
              node,
              ns )
```

Function to evaluate the AST node.

Parameters

node	AST node
ns	namespace for functions and constants

Returns

evaluated value of the node

4.2.1.3 evaluate()

```
def src.calculator.evaluate (
              expr,
              base = 10 )
```

Function to evaluate the expression.

Parameters

expr	input expression
base	base for number conversion (2, 8, or 10)

Returns

evaluated result as a string

4.2.1.4 tokenize()

```
def src.calculator.tokenize (
              expr )
```

Function to tokenize the input expression.

Parameters

expr	input expression

Returns

generator of tokens

4.3 src.gui Namespace Reference

Classes

- class CalculatorGUI

Functions

- def resource_path (relative_path)
- def main ()

  Main entry point launching the calculator GUI.

4.3.1 Function Documentation

4.3.1.1 main()

```
def src.gui.main ( )
```

Main entry point launching the calculator GUI.

Initializes the Tk root window, sets minimum and maximum sizes, and starts the main event loop.

Returns

None

4.3.1.2 resource_path()

```
def src.gui.resource_path (
              relative_path )
```

4.4 src.math_lib Namespace Reference

Functions

- def add (a, b)

  Function to add two numbers.
- def sub (a, b)

  Function to subtract second number from first.
- def mul (a, b)

  Function to multiply two numbers.
- def div (a, b)

  Function to divide first number by second.
- def fact (n)

  Function to calculate factorial of a number.
- def compute_e (precision=20)

  Function to compute the value of Euler's number.
- def arctan (x, precision=1e-17)

  Function to calculate arcus tangens of x.
- def pi (precision=1e-17)

  Function to compute the value of pi.
- def square (a)

  Function to calculate the square of a number.
- def power (a, b)

  Function to calculate the power of a number.
- def sqrt (a)

  Function to compute the square root of a number.
- def nthroot (a, n)

  Function to compute the nth root of a number.
- def ln (a, precision=1e-20)

  Function to calculate the natural logarithm of a number.
- def log (a, b)

  Function to calculate the logarithm of a number with base b.
- def abs (a)

  Function to calculate the absolute value of a number.
- def _snap_to_integer (val, precision)

  Function to round result into more readable format.
- def sin (x, precision=1e-17)

  Function to calculate the sine of an angle in degrees.
- def cos (x, precision=1e-17)

  Function to calculate the cosine of an angle in degrees.
- def tg (x, precision=1e-10)

  Function to calculate the tangent of an angle in degrees.
- def cotg (x, precision=1e-10)

  Function to calculate the cotangent of an angle in degrees.
- def sum (numbers)

  Function to calculate the sum of a list of numbers.

4.4.1 Function Documentation

4.4.1.1 _snap_to_integer()

```
def src.math_lib._snap_to_integer (
              val,
              precision )  [private]
```

Function to round result into more readable format.

Parameters

val	number
precision	number of decimal places

Returns

rounded value

Used in some goniometric functions

4.4.1.2 abs()

```
def src.math_lib.abs (
              a )
```

Function to calculate the absolute value of a number.

Parameters

a	number

Returns

absolute value of a

4.4.1.3 add()

```
def src.math_lib.add (
              a,
              b )
```

Function to add two numbers.

Parameters

a	first number
b	second number

Returns

sum

4.4.1.4 arctan()

```
def src.math_lib.arctan (
              x,
              precision = 1e-17 )
```

Function to calculate arcus tangens of x.

Parameters

x	number
precision	number of decimal places

Returns

arctan of x

Computation using the Taylor series expansion.

4.4.1.5 compute_e()

```
def src.math_lib.compute_e (
              precision = 20 )
```

Function to compute the value of Euler's number.

Parameters

precision	number of decimal places

Computation using the Taylor series expansion.

Returns

value of euler's number

4.4.1.6 cos()

```
def src.math_lib.cos (
              x,
              precision = 1e-17 )
```

Function to calculate the cosine of an angle in degrees.

Parameters

x	angle in degrees
precision	number of decimal places

Returns

cosine of x

Computation using the Taylor series expansion.

4.4.1.7 cotg()

```
def src.math_lib.cotg (
              x,
              precision = 1e-10 )
```

Function to calculate the cotangent of an angle in degrees.

Parameters

x	angle in degrees
precision	number of decimal places

Returns

cotangent of x

Computation using the sine and cosine functions.

4.4.1.8 div()

```
def src.math_lib.div (
              a,
              b )
```

Function to divide first number by second.

Parameters

a	first number
b	second number

Returns

quotient

If b is 0, returns "Error".

4.4.1.9 fact()

```
def src.math_lib.fact (
              n )
```

Function to calculate factorial of a number.

Parameters

n	number

Returns

factorial of n

If n is negative, returns "Error".

4.4.1.10 ln()

```
def src.math_lib.ln (
              a,
              precision = 1e-20 )
```

Function to calculate the natural logarithm of a number.

Parameters

a	number
precision	number of decimal places

Returns

natural logarithm of a

Computation using the Taylor series expansion.

If a is less than or equal to 0, returns "Error".

4.4.1.11 log()

```
def src.math_lib.log (
              a,
              b )
```

Function to calculate the logarithm of a number with base b.

Parameters

a	number
b	base

Returns

logarithm of a with base b

4.4.1.12 mul()

```
def src.math_lib.mul (
              a,
              b )
```

Function to multiply two numbers.

Parameters

a	first number
b	second number

Returns

product

4.4.1.13 nthroot()

```
def nthroot (
              a,
              n )
```

Function to compute the nth root of a number.

Parameters

a	number
n	root

Returns

nth root of a

If a is negative and n is even, returns "Error".

4.4.1.14 pi()

```
def src.math_lib.pi (
              precision = 1e-17 )
```

Function to compute the value of pi.

Parameters

precision	number of decimal places

Returns

value of pi

Computation using the Machin-like formula.

4.4.1.15 power()

```
def src.math_lib.power (
              a,
              b )
```

Function to calculate the power of a number.

Parameters

a	base
b	exponent

Returns

a raised to the power of b

4.4.1.16 sin()

```
def src.math_lib.sin (
              x,
              precision = 1e-17 )
```

Function to calculate the sine of an angle in degrees.

Parameters

x	angle in degrees
precision	number of decimal places

Returns

sine of x

Computation using the Taylor series expansion.

4.4.1.17 sqrt()

```
def src.math_lib.sqrt (
              a )
```

Function to compute the square root of a number.

Parameters

a	number

Returns

square root of a

If a is negative, returns "Error".

4.4.1.18 square()

```
def src.math_lib.square (
              a )
```

Function to calculate the square of a number.

Parameters

a	number

Returns

square of a

4.4.1.19 sub()

```
def src.math_lib.sub (
              a,
              b )
```

Function to subtract second number from first.

Parameters

a	first number
b	second number

Returns

difference

4.4.1.20 sum()

```
def src.math_lib.sum (
              numbers )
```

Function to calculate the sum of a list of numbers.

Parameters

numbers	list of numbers

Returns

sum of numbers

4.4.1.21 tg()

```
def src.math_lib.tg (
              x,
              precision = 1e-10 )
```

Function to calculate the tangent of an angle in degrees.

Parameters

x	angle in degrees
precision	number of decimal places

Returns

tangent of x

Computation using the sine and cosine functions.

4.5 src.stddev Namespace Reference

Functions

- def load_data ()

  Function to load data from standard input.
- def calculate_stddev (data)

  Function to calculate the standard deviation of a list of numbers.

Variables

- def data = load_data()

  list for storing input
- def stddev = calculate_stddev(data)

4.5.1 Function Documentation

4.5.1.1 calculate_stddev()

```
def src.stddev.calculate_stddev (
              data )
```

Function to calculate the standard deviation of a list of numbers.

Parameters

data	list of numbers

Returns

standard deviation of data

4.5.1.2 load_data()

```
def src.stddev.load_data ( )
```

Function to load data from standard input.

This function reads lines from standard input, splits them into individual numbers, converts them to floats, and stores them in a list. It handles invalid input. Stops reading when EOF is reached.

Returns

list of floats

4.5.2 Variable Documentation

4.5.2.1 data

```
data = load_data()
```

list for storing input

4.5.2.2 stddev

```
def stddev = calculate_stddev(data)
```

4.6 src.test_math_lib Namespace Reference

Functions

- def test_add ()
  Test addition functionality from math_lib.
- def test_sub ()
  Test subtraction functionality.
- def test_mul ()
  Test multiplication functionality.
- def test_div ()
  Test division functionality.
- def test_fact ()
  Test factorial computation.
- def test_compute_e ()
  Test Euler's number approximation.
- def test_arctan_and_pi ()
  Test arctangent and pi approximation.
- def test_square_and_power ()
  Test squaring and exponentiation.
- def test_sqrt ()
  Test square root functionality.
- def test_nthroot ()
  Test n-th root computation.
- def test_ln_and_log ()
  Test natural logarithm and logarithm with custom base.
- def test_abs_sum ()
  Test absolute value and list summation.
- def test_sin_cos_tg_cotg ()
  Test trigonometric functions: sin, cos, tan, cotangent.

4.6.1 Function Documentation

4.6.1.1 test_abs_sum()

```
def src.test_math_lib.test_abs_sum ( )
```

Test absolute value and list summation.

Tests abs for positive/negative numbers and sum over lists, including empty list

4.6.1.2 test_add()

```
def src.test_math_lib.test_add ( )
```

Test addition functionality from math_lib.

Verifies correct results for positive, negative, and floating-point additions

4.6.1.3 test_arctan_and_pi()

```
def src.test_math_lib.test_arctan_and_pi ( )
```

Test arctangent and pi approximation.

Validates arctan result and compares custom pi approximation with math.pi

4.6.1.4 test_compute_e()

```
def src.test_math_lib.test_compute_e ( )
```

Test Euler's number approximation.

Compares custom approximation with Python’s math.e constant

4.6.1.5 test_div()

```
def src.test_math_lib.test_div ( )
```

Test division functionality.

Tests normal division, floating-point division, and division by zero

4.6.1.6 test_fact()

```
def src.test_math_lib.test_fact ( )
```

Test factorial computation.

Checks correct calculation for 0, positive integers and handling of negative inputs

4.6.1.7 test_ln_and_log()

```
def src.test_math_lib.test_ln_and_log ( )
```

Test natural logarithm and logarithm with custom base.

Includes valid and invalid input handling for ln and log

4.6.1.8 test_mul()

```
def src.test_math_lib.test_mul ( )
```

Test multiplication functionality.

Includes tests with integers, negative numbers and floating-point values

4.6.1.9 test_nthroot()

```
def src.test_math_lib.test_nthroot ( )
```

Test n-th root computation.

Tests for positive/negative inputs, even/odd roots, and invalid inputs

4.6.1.10 test_sin_cos_tg_cotg()

```
def src.test_math_lib.test_sin_cos_tg_cotg ( )
```

Test trigonometric functions: sin, cos, tan, cotangent.

Validates results for known angles and handles undefined cases

4.6.1.11 test_sqrt()

```
def src.test_math_lib.test_sqrt ( )
```

Test square root functionality.

Checks for perfect square, irrational square root and negative number handling

4.6.1.12 test_square_and_power()

```
def src.test_math_lib.test_square_and_power ( )
```

Test squaring and exponentiation.

Verifies squaring a number and raising to various powers

4.6.1.13 test_sub()

```
def src.test_math_lib.test_sub ( )
```

Test subtraction functionality.

Validates subtraction for various positive and negative values including floats

Chapter 5

Class Documentation

5.1 CalculatorGUI Class Reference

Public Member Functions

- def __init__ (self, master)

  Constructor for the CalculatorGUI class.
- def show_help (self)

  Opens a help window with usage instructions.
- def on_keypress (self, event)

  Handles keyboard input mapping.
- def on_button (self, char)

  Processes button presses from UI or keyboard.
- def format_result (self, res)

  Formats numeric results according to selected base and type.
- def change_base (self)

  Enables/disables calculator buttons based on current base.

Public Attributes

- master
- after_equal
- expr_var
- result_var
- last_result
- base_var
- buttons

Private Member Functions

- def _get_mapping (self)

  Provides mapping from display symbols to evaluation tokens.

5.1.1 Constructor & Destructor Documentation

5.1.1.1 __init__()

```
def __init__ (
              self,
              master )
```

Constructor for the CalculatorGUI class.

Parameters

master	The Tkinter root window.

Initializes window properties, state variables, layouts, buttons, display labels, and keyboard bindings.

5.1.2 Member Function Documentation

5.1.2.1 _get_mapping()

```
def _get_mapping (
              self )  [private]
```

Provides mapping from display symbols to evaluation tokens.

Returns a dictionary mapping button labels (e.g., '×','sin') to the corresponding expression strings for the evaluator.

Returns

A dict mapping UI labels to evaluator tokens.

5.1.2.2 change_base()

```
def change_base (
              self )
```

Enables/disables calculator buttons based on current base.

Activates only valid digit buttons for binary/octal, always enables operators and control keys, and resets display.

Returns

None

5.1.2.3 format_result()

```
def format_result (
              self,
              res )
```

Formats numeric results according to selected base and type.

Parameters

res	The result to format (int or float).

Converts ints to binary/octal/decimal strings, floats to fixed precision (10 decimals, trimmed).

Returns

A string representation of the result.

5.1.2.4 on_button()

```
def on_button (
              self,
              char )
```

Processes button presses from UI or keyboard.

Parameters

char	The label of the button pressed.

Handles insertion of characters/functions, evaluation on '=', clear/backspace, ANS insertion, and state resets.

Returns

None

5.1.2.5 on_keypress()

```
def on_keypress (
              self,
              event )
```

Handles keyboard input mapping.

Parameters

event	The Tkinter keyboard event.

Maps keys (digits, operators, Enter, Backspace, Delete) to corresponding button actions, validating against current base.

Returns

None

5.1.2.6 show_help()

```
def show_help (
              self )
```

Opens a help window with usage instructions.

Creates a scrollable Toplevel window listing all operations, parameters, and notes.

Returns

None

5.1.3 Member Data Documentation

5.1.3.1 after_equal

```
after_equal
```

5.1.3.2 base_var

```
base_var
```

5.1.3.3 buttons

```
buttons
```

5.1.3.4 expr_var

```
expr_var
```

5.1.3.5 last_result

```
last_result
```

5.1.3.6 master

```
master
```

5.1.3.7 result_var

```
result_var
```

The documentation for this class was generated from the following file:

- gui.py

5.2 Parser Class Reference

Class to parse the tokenized input.

Public Member Functions

- def __init__ (self, tokens, last_ans=0)
  Constructor for the Parser class.
- def current (self)
  Function to get the current token.
- def advance (self)
  Function to advance to the next token.
- def parse (self)
  Function to parse the expression.
- def parse_expression (self)
  Function to parse the expression.
- def parse_term (self)
  Function to parse the term.
- def parse_power (self)
  Function to parse power.
- def parse_factor (self)
  Function to parse the factor.

Public Attributes

- tokens
- pos
- last_ans

5.2.1 Detailed Description

Class to parse the tokenized input.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 __init__()

```
def __init__ (
              self,
              tokens,
              last_ans = 0 )
```

Constructor for the Parser class.

Parameters

tokens	list of tokens
last_ans	last answer used in the calculator

5.2.3 Member Function Documentation

5.2.3.1 advance()

```
def advance (
              self )
```

Function to advance to the next token.

Increments the position of the current token.

5.2.3.2 current()

```
def current (
              self )
```

Function to get the current token.

Returns

current token

5.2.3.3 parse()

```
def parse (
              self )
```

Function to parse the expression.

Parses the entire expression and returns the abstract syntax tree (AST).

Returns

AST node representing the expression

5.2.3.4 parse_expression()

```
def parse_expression (
              self )
```

Function to parse the expression.

Handles addition and subtraction.

Returns

AST node representing the expression

5.2.3.5 parse_factor()

```
def parse_factor (
              self )
```

Function to parse the factor.

Handles parentheses, numbers, identifiers, and unary minus.

Returns

AST node representing the factor

5.2.3.6 parse_power()

```
def parse_power (
              self )
```

Function to parse power.

Handles exponentiation.

Returns

AST node representing power

5.2.3.7 parse_term()

```
def parse_term (
              self )
```

Function to parse the term.

Handles multiplication and division.

Returns

AST node representing the term

5.2.4 Member Data Documentation

5.2.4.1 last_ans

```
last_ans
```

5.2.4.2 pos

```
pos
```

5.2.4.3 tokens

```
tokens
```

The documentation for this class was generated from the following file:

- calculator.py

Chapter 6

File Documentation

6.1 __init__.py File Reference

Namespaces

- src

6.2 calculator.py File Reference

Calculator that evaluates mathematical expressions.

Classes

- class Parser

  Class to parse the tokenized input.

Namespaces

- src.calculator

Functions

- def tokenize (expr)

  Function to tokenize the input expression.
- def eval_node (node, ns)

  Function to evaluate the AST node.
- def build_safe_ns (last_ans, base=10)

  Function to build a safe namespace for the calculator.
- def evaluate (expr, base=10)

  Function to evaluate the expression.

6.2.1 Detailed Description

Calculator that evaluates mathematical expressions.

Date

2025-04-29

6.3 gui.py File Reference

GUI for the calculator.

Classes

- class CalculatorGUI

Namespaces

- src.gui

Functions

- def resource_path (relative_path)
- def main ()

  Main entry point launching the calculator GUI.

6.3.1 Detailed Description

GUI for the calculator.

This script creates a graphical user interface for a calculator using the Tkinter library.

Date

2025-04-29

6.4 math_lib.py File Reference

Library for mathematical operations.

Namespaces

- src.math_lib

Functions

- def add (a, b)

  Function to add two numbers.
- def sub (a, b)

  Function to subtract second number from first.
- def mul (a, b)

  Function to multiply two numbers.
- def div (a, b)

  Function to divide first number by second.
- def fact (n)

  Function to calculate factorial of a number.
- def compute_e (precision=20)

  Function to compute the value of Euler's number.
- def arctan (x, precision=1e-17)

  Function to calculate arcus tangens of x.
- def pi (precision=1e-17)

  Function to compute the value of pi.
- def square (a)

  Function to calculate the square of a number.
- def power (a, b)

  Function to calculate the power of a number.
- def sqrt (a)

  Function to compute the square root of a number.
- def nthroot (a, n)

  Function to compute the nth root of a number.
- def ln (a, precision=1e-20)

  Function to calculate the natural logarithm of a number.
- def log (a, b)

  Function to calculate the logarithm of a number with base b.
- def abs (a)

  Function to calculate the absolute value of a number.
- def _snap_to_integer (val, precision)

  Function to round result into more readable format.
- def sin (x, precision=1e-17)

  Function to calculate the sine of an angle in degrees.
- def cos (x, precision=1e-17)

  Function to calculate the cosine of an angle in degrees.
- def tg (x, precision=1e-10)

  Function to calculate the tangent of an angle in degrees.
- def cotg (x, precision=1e-10)

  Function to calculate the cotangent of an angle in degrees.
- def sum (numbers)

  Function to calculate the sum of a list of numbers.

6.4.1 Detailed Description

Library for mathematical operations.

Date

2025-04-29

6.5 stddev.py File Reference

Calculates the standard deviation of a series of numbers provided via standard input (stdin).

Namespaces

- src.stddev

Functions

- def load_data ()

  Function to load data from standard input.
- def calculate_stddev (data)

  Function to calculate the standard deviation of a list of numbers.

Variables

- def data = load_data()

  list for storing input
- def stddev = calculate_stddev(data)

6.5.1 Detailed Description

Calculates the standard deviation of a series of numbers provided via standard input (stdin).

It processes the input, calculates the mean, sum of squares, and the standard deviation. The result is printed to stdout. The script expects input in the form of whitespace-separated numbers (multiple lines allowed).

Date

2025-04-28

6.6 test_math_lib.py File Reference

Namespaces

- src.test_math_lib

Functions

- def test_add ()

  Test addition functionality from math_lib.
- def test_sub ()

  Test subtraction functionality.
- def test_mul ()

  Test multiplication functionality.
- def test_div ()

  Test division functionality.
- def test_fact ()

  Test factorial computation.
- def test_compute_e ()

  Test Euler's number approximation.
- def test_arctan_and_pi ()

  Test arctangent and pi approximation.
- def test_square_and_power ()

  Test squaring and exponentiation.
- def test_sqrt ()

  Test square root functionality.
- def test_nthroot ()

  Test n-th root computation.
- def test_ln_and_log ()

  Test natural logarithm and logarithm with custom base.
- def test_abs_sum ()

  Test absolute value and list summation.
- def test_sin_cos_tg_cotg ()

  Test trigonometric functions: sin, cos, tan, cotangent.

Index