documentation module

Math-library and GUI documentation

`documentation.Divide(n, p)` [source]

Divide two numbers and return the result.

Parameters:

n: float

The numerator.

p: float

The denominator.

Returns:

x: int or float

If p is zero, returns the string “error”. Otherwise, returns the result of dividing n by p.

Description:

This function divides the first input (numerator) by the second input (denominator) and returns the result.

`documentation.Factorial(n)` [source]

Calculates the factorial of a given integer.

Parameters:

n : int

The integer to calculate the factorial of.

Returns:

Union[int, str]: If the input is a positive integer, the function returns the factorial of that integer. If the input is 0, the function returns 1. If the input is not a positive integer, the function returns “error”.

Description:

This function calculates the factorial of a given positive integer by multiplying that integer by the factorial of the next smallest positive integer until it reaches 1.

`documentation.Minus(n, p)` [source]

Subtracts p from n and returns the result.

Parameters:

n: Union[int, float]

The number to be subtracted from.

p: Union[int, float]

The number to subtract from n.

Returns:

Union[int, float]: The result of subtracting p from n.

Description:

This function subtracts the number p from n and returns the result.

`documentation.Multiply(n, p)` [source]

Multiply two numbers and return the result as a string.

Parameters:

n: Union[int, float]

The first number to be multiplied.

p: Union[int, float]

The second number to be multiplied.

Returns:

x: int or float

A representation of the product of the two numbers.

Description:

This method multiplies two numbers.

`documentation.Plus`(*n*, *p*) [source]

Add two numbers

Parameters:

n: float

The first number to add.

p: float

The second number to add.

Returns:

float

The sum of the two numbers

Description:

This function adds the two input numbers, converts the result to a string.

`documentation.Power`(*n*, *p*) [source]

Calculates the value of the first argument raised to the power of the second argument.

Parameters:

n : int or float

The base number.

p : int or float

The exponent number.

Returns:

str:

The result of the exponentiation in scientific notation

Description:

This function calculates the value of the first argument raised to the power of the second argument.

`documentation.SquareRoot`(*n*) [source]

Returns the square root of a given number.

Parameters:

n : int or float

The number to calculate the square root of.

Returns:

int or float or str:

The square root of the given number if it’s non-negative, or ‘error’ if it’s negative.

Description:

This function takes a number as input and returns its square root. If the input is negative, it returns ‘error’.

`documentation.add_history`(*self, math_sign: str*) [source]

Adds the current entry and the math sign to the history widget.

Parameters:

math_sign : str

The math symbol to add to the history.

Returns:

None

Description:

This method adds the current entry and the math sign to the history widget, clearing the entry afterwards. If the history widget is empty or the last operation performed was an equality operation, it creates a new history string using the current entry and the math symbol. It replaces the dot with a comma for localization. Finally, it sets the text of the history widget to the new string and sets the entry text to “0”.

`documentation.add_number`*(self, Button_text: str)* [source]

Add a number to the UI entry widget.

Parameters:

Button_text : str

A string representing the number to be added to the UI entry widget.

Returns:

None

Description:

This method adds a number to the current text in the UI entry widget. If the current text in the UI entry widget is “0”, the method replaces it with the *Button_text*. Otherwise, the method appends the *Button_text* to the current text in the UI entry widget. After modifying the text in the UI entry widget, the method calls the *adjust_entry_font_size* method to adjust the font size of the text if necessary.

`documentation.add_point`*(self)* [source]

Add a decimal point to the UI entry widget.

Parameters:

None

Returns:

None

Description:

This method adds a decimal point (" , ") to the current text in the UI entry widget if one is not already present. If a decimal point is already present, the method does nothing. After modifying the text in the UI entry widget, the method calls the *adjust_entry_font_size* method to adjust the font size of the text if necessary.

`documentation.adjust_entry_font_size`(*self*) [source]

Adjust the font size of the entry field to fit the text.

Parameters:

None

Returns:

None

Description:

This method iteratively reduces the font size of the entry field until the width of the text is less than the width of the entry field minus 20 pixels. It then iteratively increases the font size of the entry field until the width of the text is greater than the width of the entry field minus 20 pixels, or until the default font size is reached.

`documentation.backspace`(*self*) [source]

Delete the last character from the UI entry widget.

Parameters:

None

Returns:

None

Description:

This method removes the last character from the text in the UI entry widget, if it is not already empty. If the entry widget contains only a negative sign and no other digits, this method sets the widget value to zero.

Additionally, this method calls the *adjust_entry_font_size* method to adjust the font size of the entry widget text `documentation.binary_calculate`(*self*, *math_sign: str*) [source]

Binary calculation

Parameters:

math_sign : str

The math symbol to currently perform (from the binary type).

Returns:

the result of the operation

Description:

The function performs the operation requested from math_operation function also operates from some possible calculator states and returns the result of the operation

`documentation.change_buttons_color`(*self*, *css_color: str*) [source]

Changes the color of all calculator buttons to the specified CSS color.

Parameters:

css_color : str

Returns:

None

Description:

This method takes a CSS color as input and applies it to the background color and border of all calculator buttons. This can be used to give a visual indication that the buttons are disabled or to change the color theme of the calculator based on user preferences.

`documentation.change_sign`(*self*) [source]

Change the sign of the number in the UI entry widget.

Parameters:

None

Returns:

None

Description:

This method changes the sign of the number currently displayed in the UI entry widget by adding or removing a minus sign (“-“). If the number is currently positive, the method adds a minus sign to make it negative. If the number is currently negative, the method removes the minus sign to make it positive. If the number is currently zero, the method does nothing.

After changing the sign of the number, the method updates the text in the UI entry widget to reflect the new value, and calls the *adjust_entry_font_size* method to adjust the font size of the text if necessary.

documentation.clear_all(*self*) [source]

Clear the UI entry widget and history, and reset other UI elements.

Parameters:

None

Returns:

None

Description:

This method clears the current value from the UI entry widget and sets it to zero. It also clears any existing entries from the history list widget, and calls the *adjust_entry_font_size* method to adjust the font size of the entry widget text if necessary.

documentation.clear_entry(*self*) [source]

Clear the UI entry widget.

Parameters:

None

Returns:

None

Description:

This method clears the current value from the UI entry widget and sets it to zero. It also calls the *adjust_entry_font_size* method to adjust the font size of the entry widget text if necessary.

`documentation.clear_history_if_equality`(*self*) [source]

Clears the history if the last operation performed was an equality operation.

Parameters:

None

Returns:

None

Description:

This method checks if the last operation performed was an equality operation by calling the *get_history_sign* method. If the result of the method is =, then it clears the history widget.

`documentation.disable_buttons`(*self*) [source]

Disables or enables calculator buttons based on the input argument.

Parameters:

disable : bool True to disable the buttons, False to enable them.

Returns:

None

Description:

This method disables or enables calculator buttons based on the input argument. If disable is True, it disables all calculator buttons except the Sign, and changes their color to indicate that they are disabled. If disable is False, it enables all calculator buttons and returns their color to normal.

`documentation.get_entry_number`(*self*) [source]

Converts the current text in the calculator's entry field to a number.

Parameters:

None

Returns:

int or float:

> The number representation of the current text in the entry field. If the text contains a decimal point, it is returned as a float; otherwise, it is returned as an integer.

Description:

This method retrieves the text currently displayed in the calculator's entry field, strips any commas from it, replaces commas with decimal points (to handle different locales), and then returns the resulting number. If the number contains a decimal point, it is returned as a float; otherwise, it is returned as an integer.

`documentation.get_entry_text_width`(*self*) [source]

Calculates the width (in pixels) of the current text in the calculator's entry field.

Parameters:

None

Returns:

int:

> The width (in pixels) of the current text in the calculator's entry field.

Description:

This method uses the font metrics of the entry field's font to calculate the width (in pixels) of the current text in the calculator's entry field. It returns this width as an integer value.

`documentation.get_history_number`(*self*) [source]

Parses the first number from the history text in the calculator's display.

Returns:

Union[int, float, None]

> The parsed number as an integer or float, or None if no number was found.

Description:

This method retrieves the text from the calculator's history display, removes any commas or leading/trailing spaces, splits the text into words and attempts to convert the first word to a float or integer. If no number is found, None is returned instead.

`documentation.get_history_sign`(*self*) [source]

Get the last operator sign from the history field.

Parameters:

None

Returns:

Optional[str]

The last operator sign from the history field, or None if the history field is empty.

Description:

This method checks if the history field is not empty, removes any trailing commas from the text, splits the text by whitespace, and returns the last element of the resulting list. If the resulting list is empty, the method returns None.

`documentation.get_history_text_width`(*self*) [source]

Gets the width, in pixels, of the text in the calculator's history display.

Parameters:

None

Returns:

int: The width, in pixels, of the text in the calculator's history display.

Description:

This method uses the font metrics of the calculator's history display to calculate the bounding rectangle around the text in the display, and returns its width in pixels. This is useful for determining the amount of horizontal space needed to display the entire history entry without truncation or overflow.

`documentation.math_operation`*(self, math_sign: str)* [source]

Complete calculator math operation

Parameters:

math_sign : str

The math symbol to currently perform.

Returns:

None

Description:

The function performs the operation depending on entry and history status, using the symbol from the parameter. It chooses between the unary operand and binary operand and performs the calculation

`documentation.print_key_event`*(self, event)* [source]

Simulates a button click on the calculator UI based on the user's keyboard input.

Parameters:

event : str

A string representing the keyboard input event triggered by the user.

Returns:

None

Description:

This method maps the user's keyboard input to the corresponding button on the calculator's UI and simulates a button click event. If the input is a number or operator key, the corresponding button is clicked and the calculator's display is updated. If the input is a non-number key (e.g., Enter, Delete), the corresponding calculator button is clicked, triggering the appropriate action.

Examples:

```
>>> calc = Calculator()
>>> calc.print_key_event('1')
# Clicks the Button_1 on the calculator's UI and updates the display to show '1'
>>> calc.print_key_event('+')
# Clicks the Button_Plus on the calculator's UI and updates the display accordingly
```

documentation.register_keyboard_event(*self*) [source]

Register a function to be called when a key on the keyboard is pressed.

Parameters:

None

Returns:

None

Description:

This method uses the *on_press* method of the *keyboard* library to register a lambda function that takes a *keyboard* event as input and passes it to the *print_key_event* method of the current instance of the class.

documentation.remove_error(*self*) [source]

Removes error message from the calculator's display if present.

Parameters:

None

Returns:

None

Description:

This method checks if the current text in the calculator's display is an error message ('Undefined' or 'Zero Division Error'), and if so, it sets the maximum length of the text to the default entry maximum length, sets the text to '0', adjusts the font size, and enables all calculator buttons.

documentation.remove_trailing_zeroes(*number: str*) [source]

Remove trailing zeroes from a given number string.

Parameters:

number : str

A string representing the number to remove trailing zeroes from.

Returns:

str

A string representing the number with trailing zeroes removed.

Description:

This function takes a number string as input, converts it to a float, and then back to a string with trailing zeroes removed. If the input string represents an error (i.e. “error”), the function returns the input string.

`documentation.resizeEvent`(*self*, *event*) [source]

Adjusts the font size of the entry field to fit the available space when the widget is resized.

Parameters:

event: QResizeEvent

The resize event that triggered the function call.

Returns:

None

Description:

This method is called automatically by the Qt framework when the widget is resized. It adjusts the font size of the entry field to fit the available space while maintaining a minimum font size, to ensure that the text is always legible.

`documentation.return_button_color`(*self*, *button*) [source]

Change the color of the calculator buttons back to their default color.

Parameters:

None

Returns:

None

Description:

This method changes the style sheet of each button to its default style sheet, except for the comma and sign buttons which are set to the nums_style_sheet and sign_style_sheet, respectively.

`documentation.show_error`(*self*) [source]

Show an error message in the entry field and disable all buttons.

Parameters:

text: str

> The error message to display.

Returns:

None

Description:

This method sets the maximum length of the entry field to the length of the error message, sets the text of the entry field to the error message, adjusts the font size of the entry field to fit the text, and disables all buttons.

`documentation.unary_calculate`(*self*, *math_sign*) [source]

Unary calculation

Parameters:

math_sign : str

> The math symbol to currently perform (from the unary type).

Returns:

the result of the operation

Description:

The function performs the operation requested from math_operation function also operates from some possible calculator states and returns the result of the operation