Calc Manual

Last updated, 2025-02-08

Table of Contents

I. Introduction 3
I.1. Install 3
I.2. Contributors 3
II. Usage 4
II.1. Basic operators 4
II.2. Variables 4
II.3. Built-in variables 4
III. Functions 6
III.1. Implemented 6
III.2. Trigonometry 7
III.3. Exp/ln 8
III.4. Root 9
III.5. Partial function 9
III.6. Vectorization 10
III.7. User defined function 10
IV. Configuration 11
IV.1. Colors 11
IV.2. Example of a modified configuration 12
IV.3. Interact in the command line 12
V. Logic 14
V.1. Implemented operators 14
V.2. Example 14
VI. Plot 15
VI.1. Help 15
VI.2. Plot 15
VI.3. Terminal plotting 17
VII. Vectors computation 20
VIII. Matrices computation 20
IX. Exact math 22
IX.1. Rational exact math 22
IX.2. Symbolic computation 23
IX.3. Function differentiation 23
X. Non interactive use 25

XI. Syntax coloration 26
XII. Float display 26

I. Introduction

Calc is a fully-featured calculator written in Rust for education purpose, it was designed to be minimalistic but then went off the rails and a lot of feature where implemented.

Now Calc is a powerful calculator capable of exact rational computation, matrix and vectors algebra, symbolic computation, differentiation, with bindings to gnuplot and terminal plotting, alongside dozens of updates and currently (as of writing this manual) in version **3.3.3**.

If you prefer a website you may want to read **The Online Book (SOON)** which is always up to date.

I.1. Install

You can install it via cargo

```Bash
cargo install mini-calc
```

or via the source

```Bash
git clone https://github.com/vanilla-extracts/calc
cd calc
cargo build --release
./target/release/mini-calc
```

or alternatively, build/run using nix

```Bash
nix run github.com:vanilla-extracts/calc
```

Visit Calc to see all the install page

I.2. Contributors

Name	Role	Website
Charlotte THOMAS	Main developer/ Maintener	Personal Page
Léana 🏳‍⚧	Help, cleanup	Website/Blog

II. Usage

II.1. Basic operators

Calc have the basic operators which are

- `+` for the addition
- `-` for the substraction
- `*` for the multiplication
- `/` for the division (or for a rational)
- `^` for the exponentation

Warning

You shoudln't use the `^` operator with functions, it is not taken into account for differentiations

II.2. Variables

It also supports variable the syntax is

```
1  myvar = value
```

for example

```
1  var = (2+2)
```

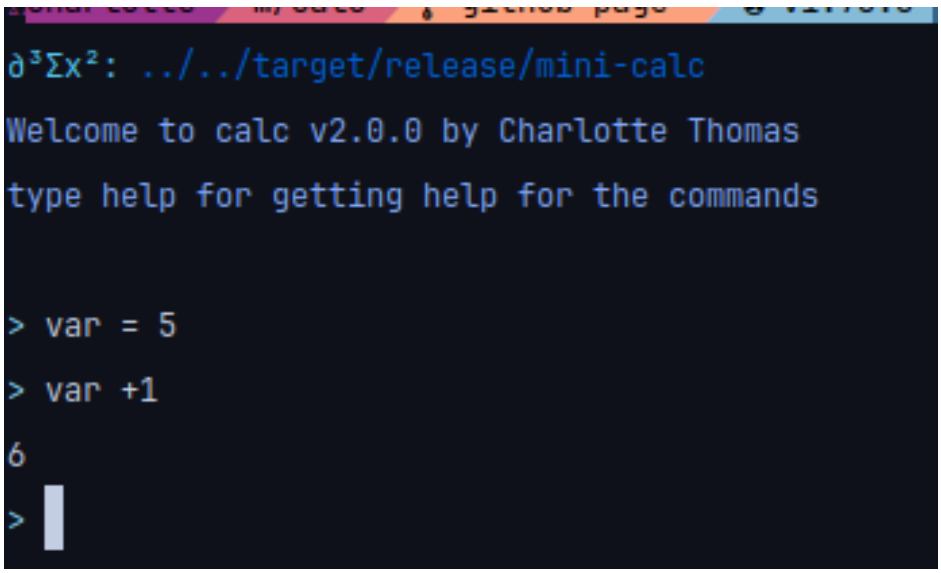

Figure 1: Example of setting a variable

II.3. Built-in variables

The following variables are built-in:

- `pi` is pi as a double precision float
- `e` is e as a double precision float

III. Functions

III.1. Implemented

The following functions are currently implemented:

Trigonometry

- `sin` (vectorized)
- `cos` (vectorized)
- `tan` (vectorized)

Hyperbolic trigonometry

- `sinh` (vectorized)
- `cosh` (vectorized)
- `tanh` (vectorized)

Reverse trigonometry

- `acos` (vectorized)
- `asin` (vectorized)
- `atan` (vectorized)

Exponentiation

- `exp` (vectorized)
- `ln` (alias: log) (vectorized)

Vectors

- `norm`

Matrices

- `det`
- `invert`

Plot

- `plot`
- `termplot`

Other

- `sqrt` (vectorized)
- `factorial` (alias: fact)
- `abs`
- `ceil`
- `floor`
- `round`

III.2. Trigonometry

For trigonometry, the input is assumed to be in radians, if it is in degrees you need to add `false` or `true` as a second argument, example shown bellow.

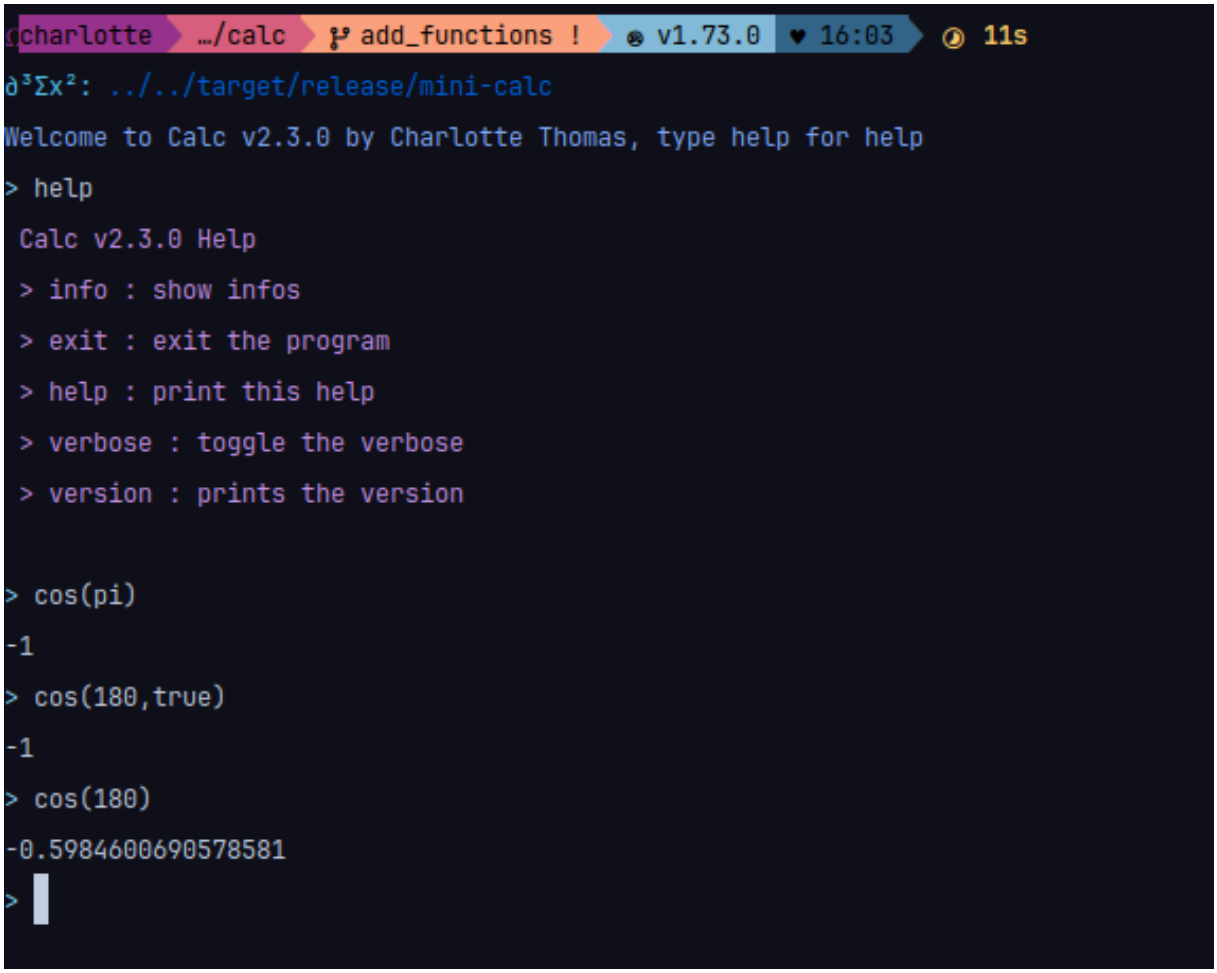

Figure 2: Usage of trigonometry

III.3. Exp/ln

If you use the exp function you can pass as a second argument the base you want to use if no second arguments are passed it will used the natural base.

Figure 3: Usage of exp/ln

III.4. Root

You can specify in second argument an integer to take the nth root, if not it take the square root.

Figure 4: Usage of sqrt

III.5. Partial function

The calculator's language supports partial function.

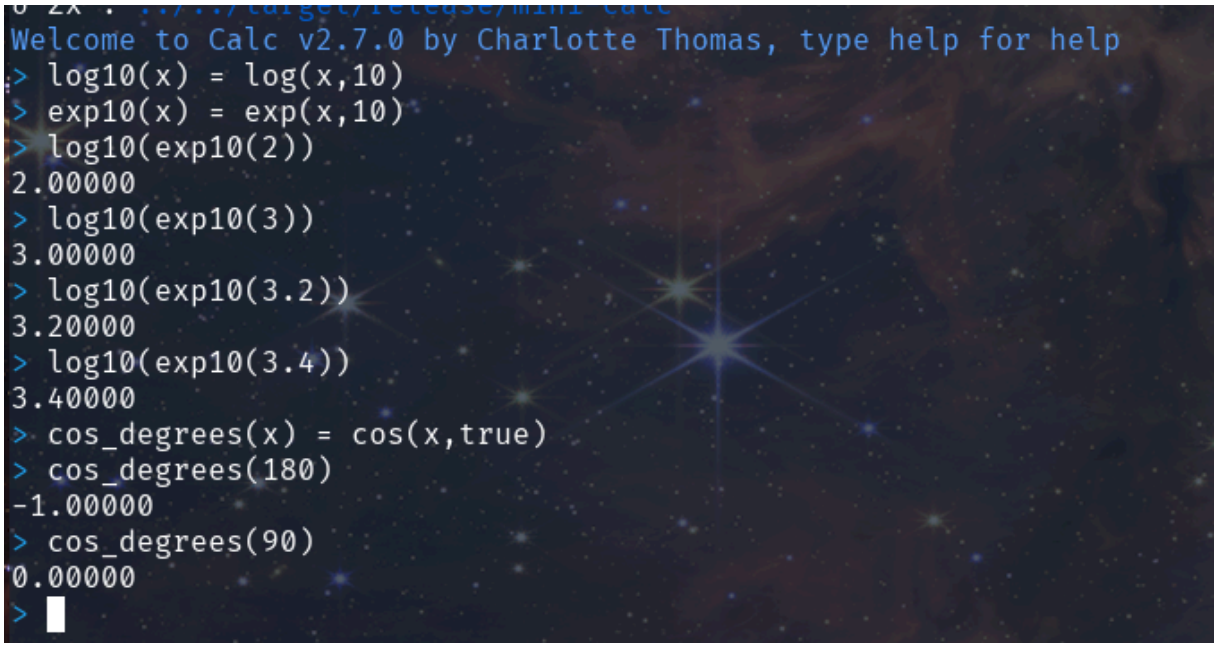

Figure 5: Example of a partial function

III.6. Vectorization

Functions have been vectorized.

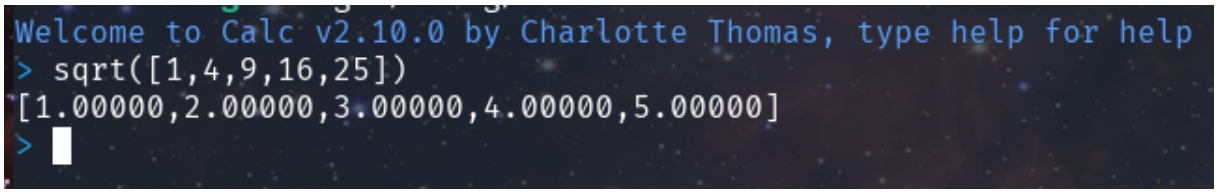

Figure 6: Example of a vectorized function

III.7. User defined function

You can defined your own function

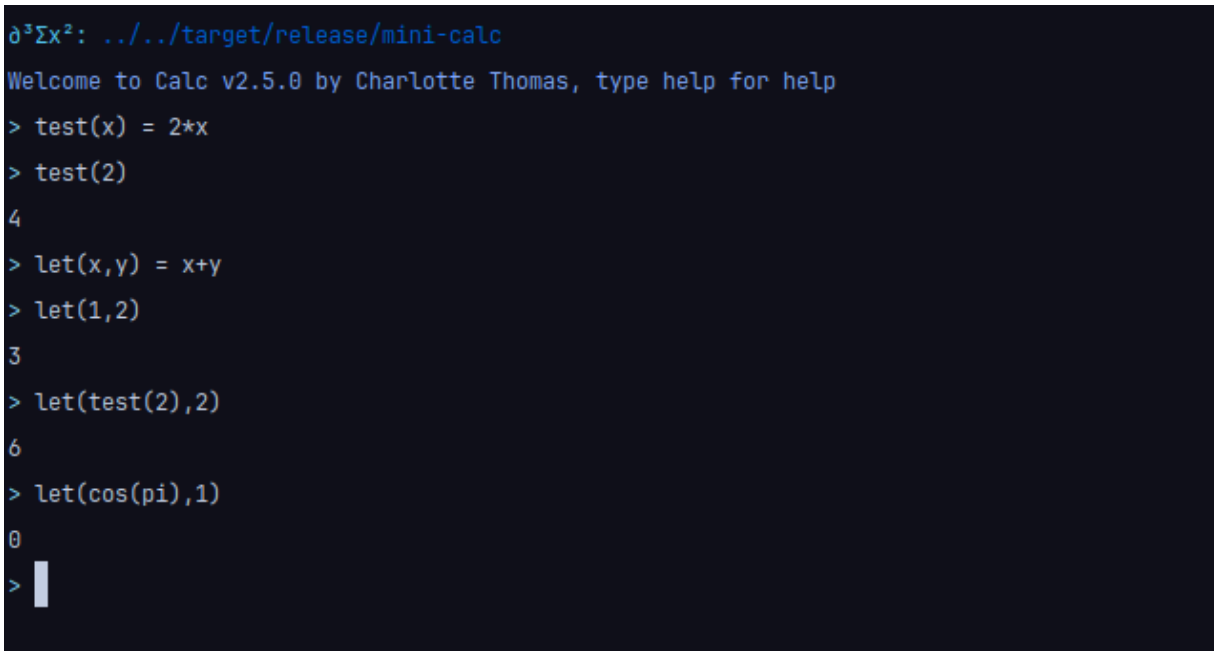

Figure 7: Definition of function

IV. Configuration

You can configure the general color, greeting message, greeting color, prompt and prompt color in a toml file found for example on linux in

```
1 ~/.config/mini-calc/mini-calc.toml
```

Bash

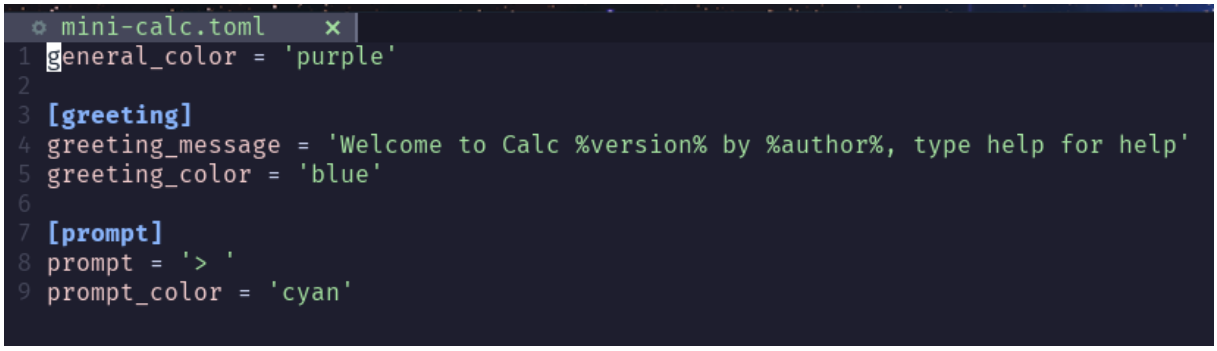

Figure 8: Example of the default configuration

IV.1. Colors

Available colors are

- blue
- black
- purple
- green
- cyan
- red
- yellow
- white
- an hexadecimal color (ex: #f7d8a8)

The default color (or if your colour can't be parsed) is cyan

IV.2. Example of a modified configuration

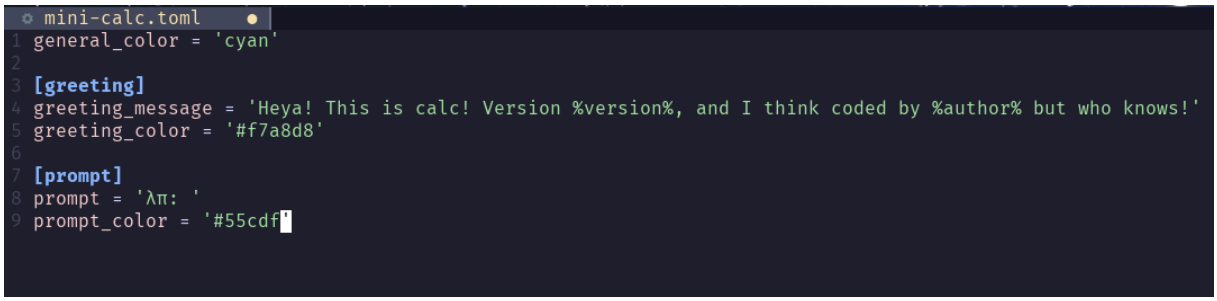

Figure 9: Example of a modified config

it looks like

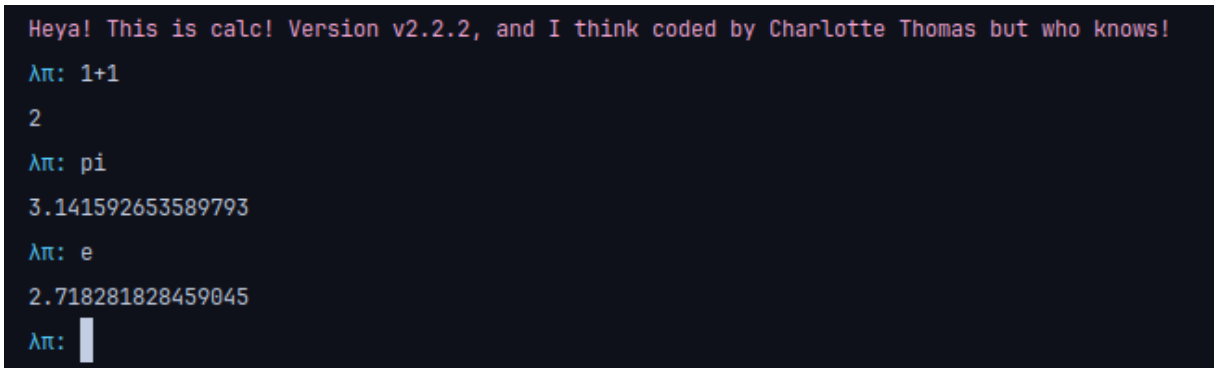

Figure 10: Modified configuration in action

IV.3. Interact in the command line

You can interact in the command line with the config, the commands are

- config: show the config help
- config reload: reload the config from the file
- config reset: reset the config
- config show: show the current config
- config set `<category> <value>`

categories are:

- greeting_message
- greeting_color
- prompt_color
- prompt
- general_color

Figure 11: Example of interaction in the command line of config

V. Logic

V.1. Implemented operators

The following operators have been implemented:

- or (alias: ||)
- and (alias: &&)
- geq (alias: >=)
- leq (alias: <=)
- gt (alias : >)
- lt (alias: <)

V.2. Example

```
Welcome to Calc v2.4.0 by Charlotte Thomas, type help for help
> true
true
> false
false
> true && false
false
> true || false
true
> 1 > 2
false
> 1 < (1+1)
true
> (1+2) >= cos(1)
true
> let = false
> !let
true
> !let && (1+2) >= cos(1)
true
>
```

Figure 12: Example of logic

VI. Plot

You can plot, the backend is provided by GNUPlot, so it should work great on linux and macos, the behaviour on windows is unknown.

VI.1. Help

To display the help just type `plot()`

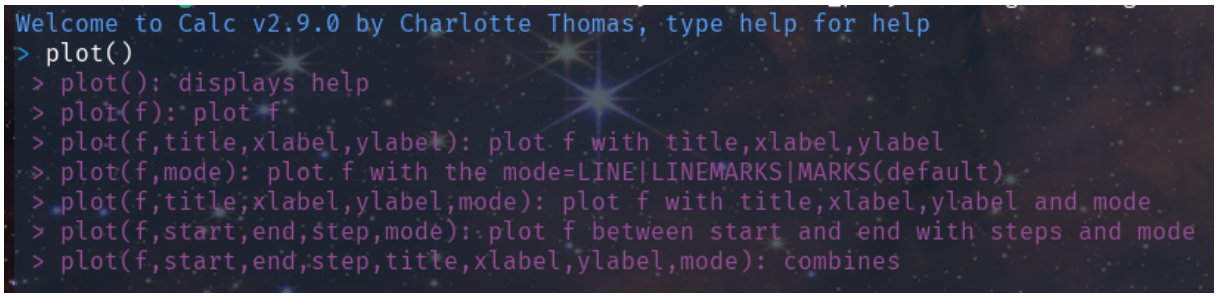

Figure 13: Help of plot

VI.2. Plot

VI.2.1. Default

It's easy to plot a function just type `plot(fn)`

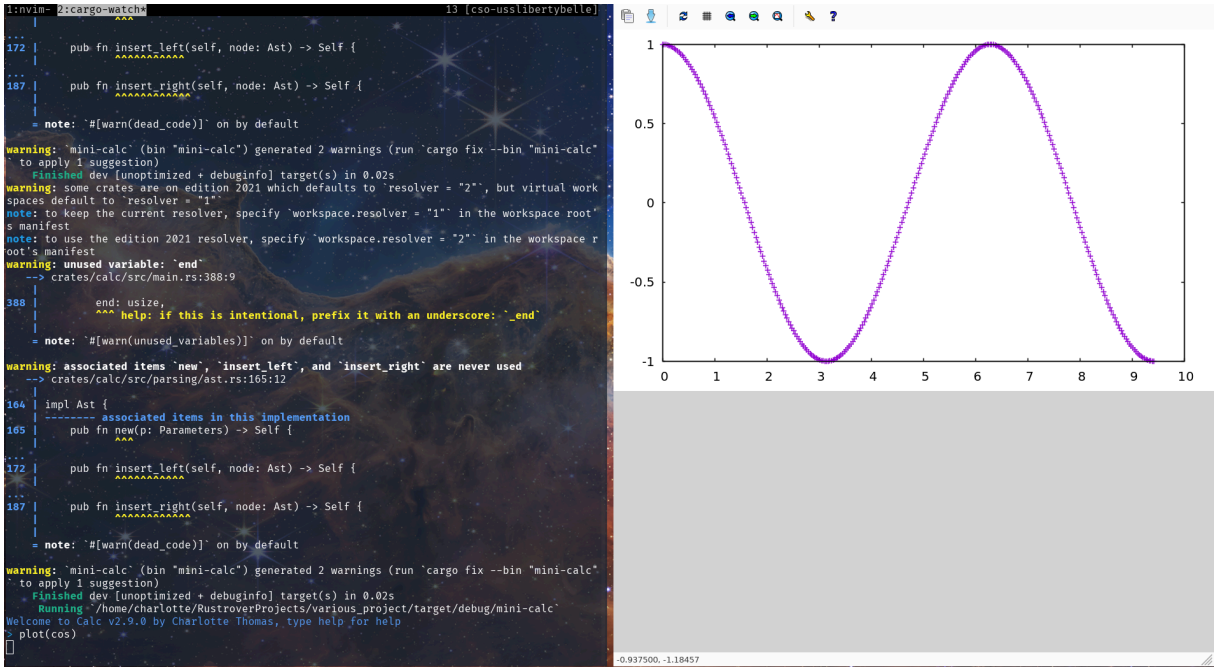

Figure 14: Plot of the cos function with default values

VI.2.2. Options

A more difficult operation is with values, `plot(sin,-pi,pi,0.01,"sin","x(rad)","y","line")`.

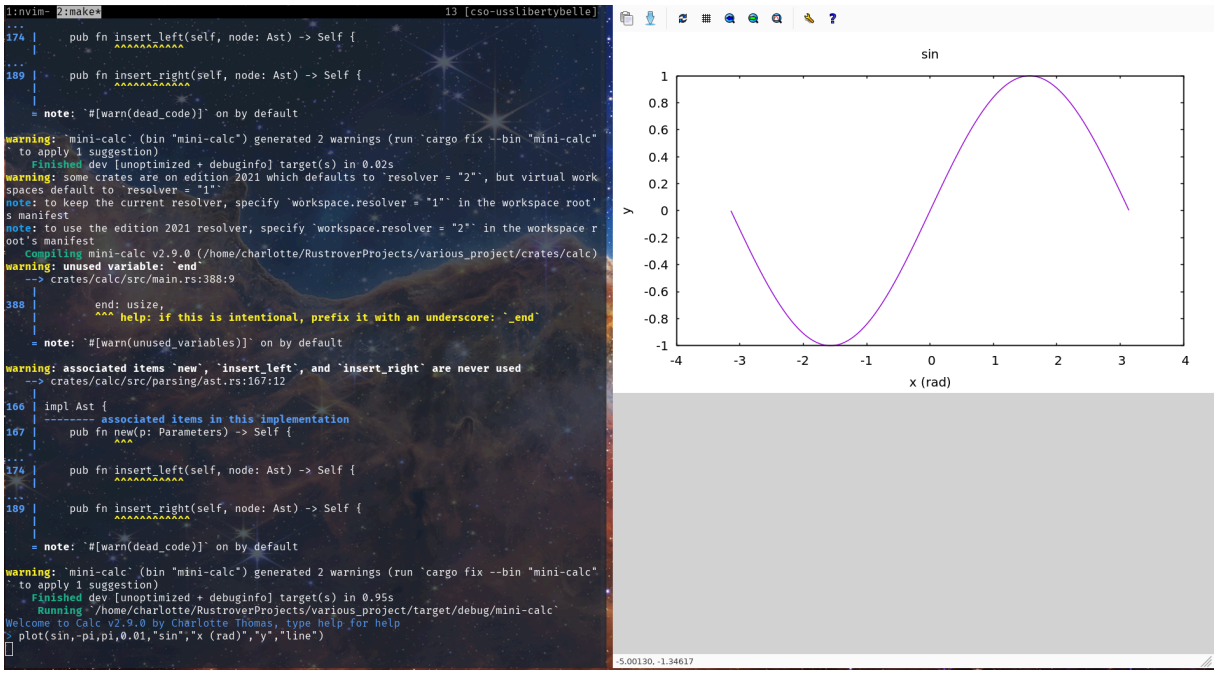

Figure 15: Plot with overloading of default values

VI.2.3. Plot your own function

You can plot your own defined functions!

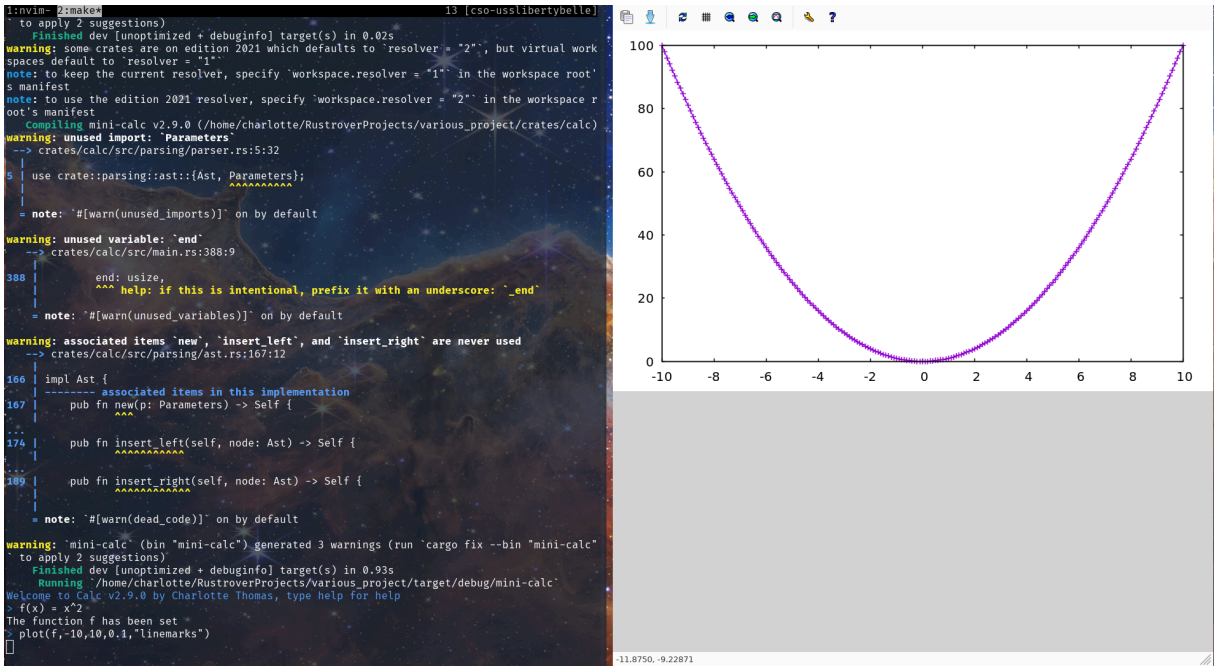

Figure 16: Plot of an user-defined function

VI.3. Terminal plotting

You can plot *right* into your terminal

VI.3.1. Default

The best example to show it is the square function between -5 and 5 with a 0.1 step. The x axis is automatically scaled but not the y axis for now.

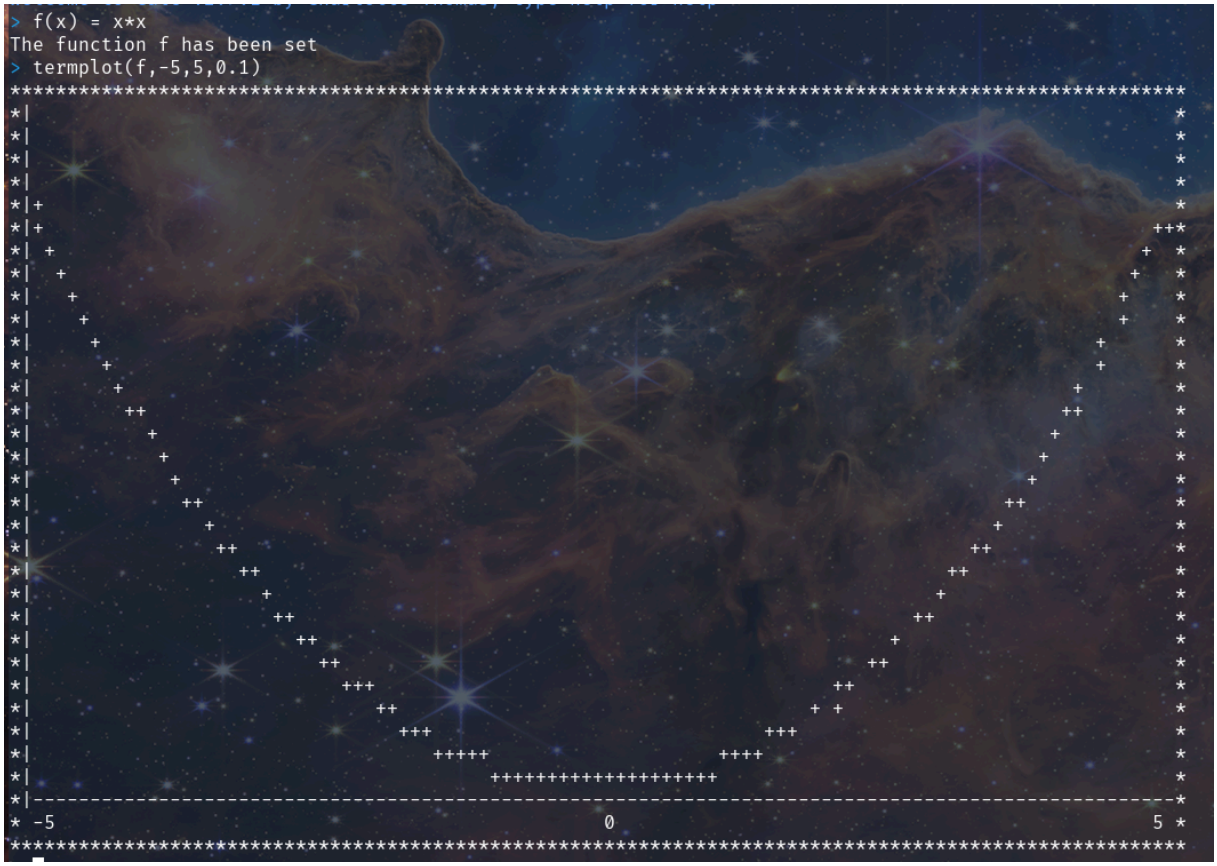

Figure 17: Terminal plotting of an user defined function

VI.3.2. Options

The terminal supports labels as options

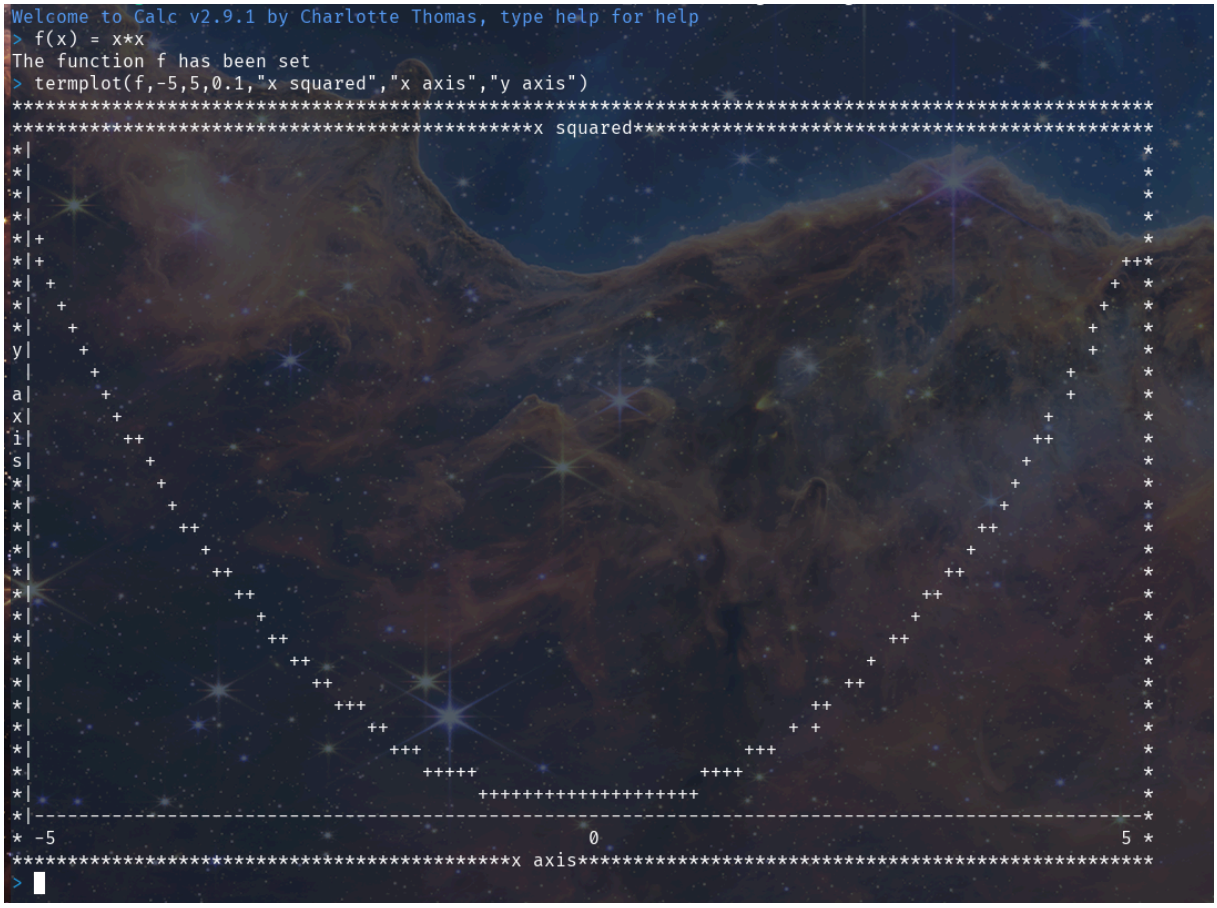

Figure 18: Terminal plotting with options

VI.3.3. Auto y scaling

It scales automatically in y too!

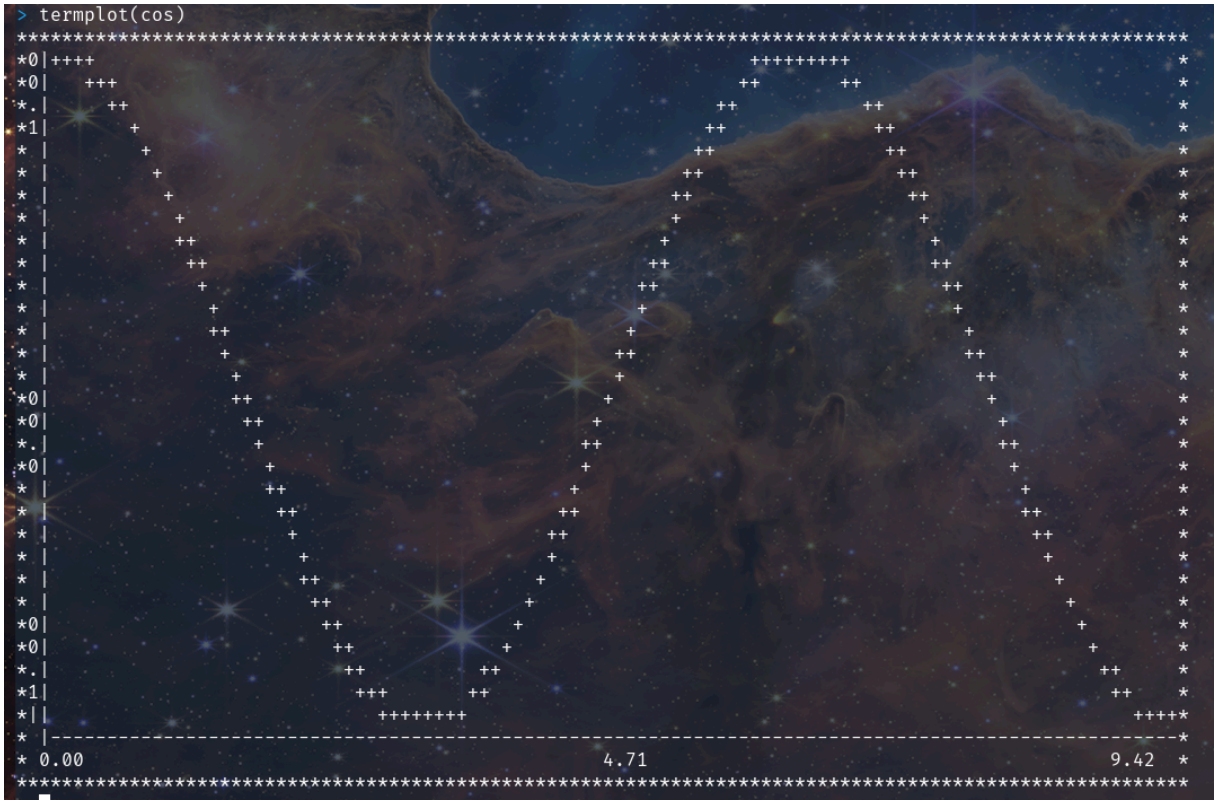

Figure 19: Example of a plot with y auto-scaling

VII. Vectors computation

You can compute vectors using these functions,

- add vectors
- dot product (* operator)
- norm function

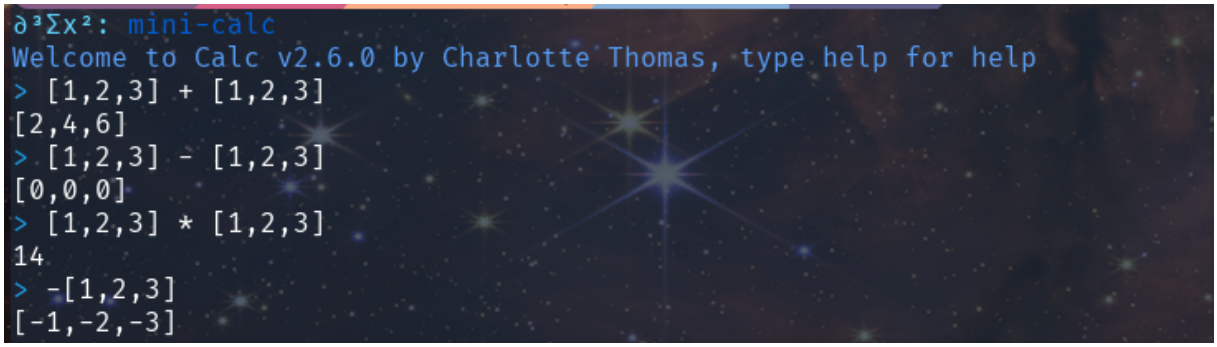

Figure 20: Example of vector computation

VIII. Matrices computation

As of 2.7.0 matrix algebra was added to the calculator you can

- add matrices
- multiply compatible matrices

functions added

- transpose
- invert
- 2023-11-26 14:49

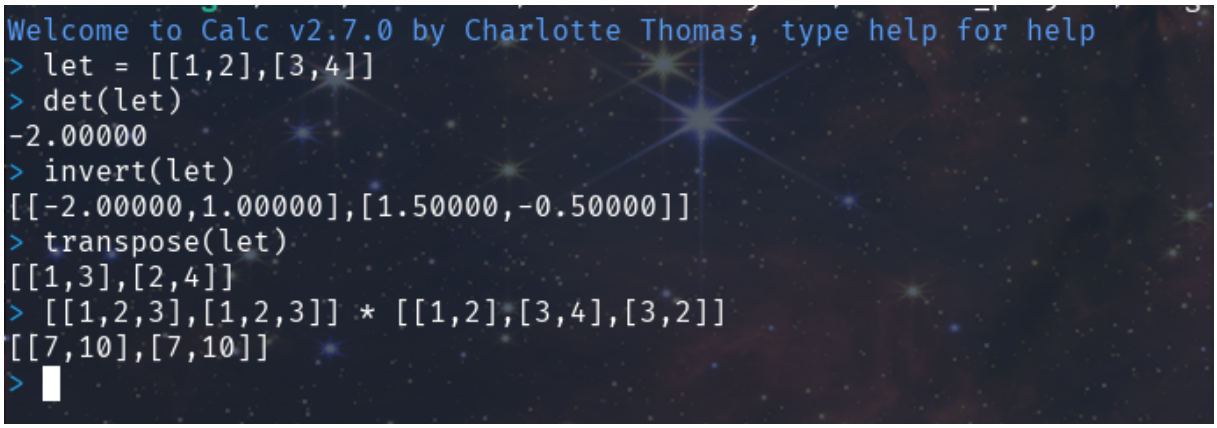

Figure 21: Example of matrix computation

And as of 2.11.5 they are pretty printed with each column being aligned.

Figure 22: Pretty printed matrix

IX. Exact math

IX.1. Rational exact math

As of 2.11.0 rational exact math was added, supports for

- rational operations
- rational reduction
- rationalization of floats (precision: 10 digits)

IX.1.1. Examples

Figure 23: Example of rational computations

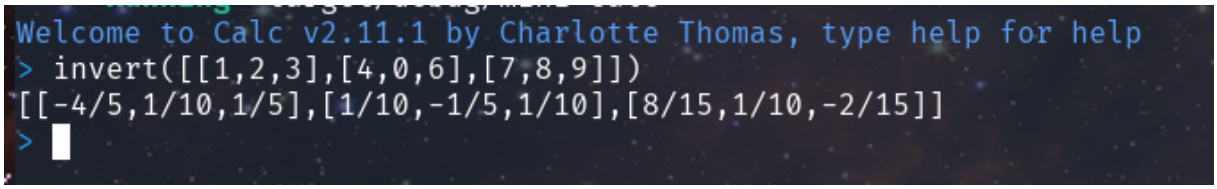

Figure 24: Example of rational in matrices

IX.2. Symbolic computation

As of 3.0.0, the support for symbolic reduction has been added to calc. It supports multi-variable multi-operation reductions, you can reduce expressions using it, fair warning, it uses *plenty* of parentheses!

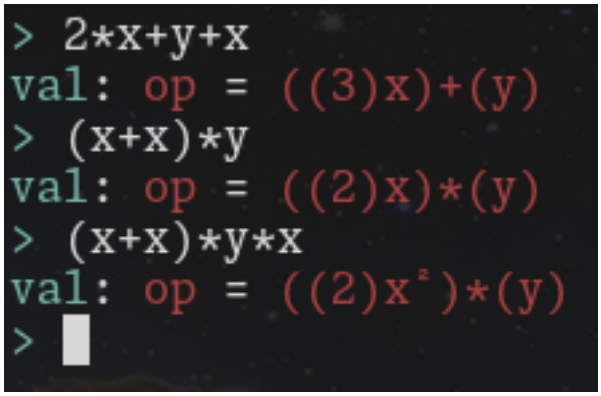

Figure 25: Example of a multi-variable reduction

IX.3. Function differentiation

As of 3.2.0, the calculator can differentiate known functions (function constructed using the standard functions). It supports both built-in and user-defined functions.

Warning

Beware as of 3.3.3 there is some bugs to iron out. It doesn't differentiates vars (differentiation of $x * x$ works but not x^2.). And differentiation of function referencing each other (example $v(x) = f(x) + x * x$) doesn't work either.

Fixed

As of 3.3.4 functions referencing each other works with **diff**

IX.3.1. Examples

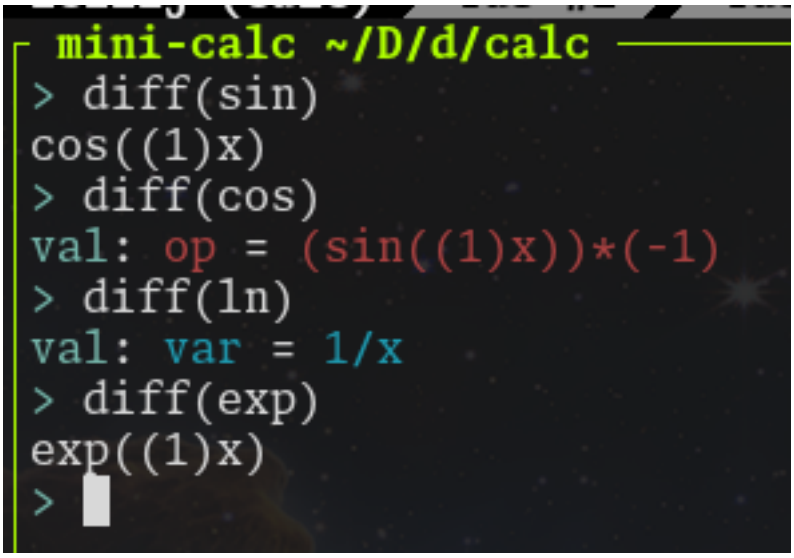

Figure 26: Example of a built-in function differentiation

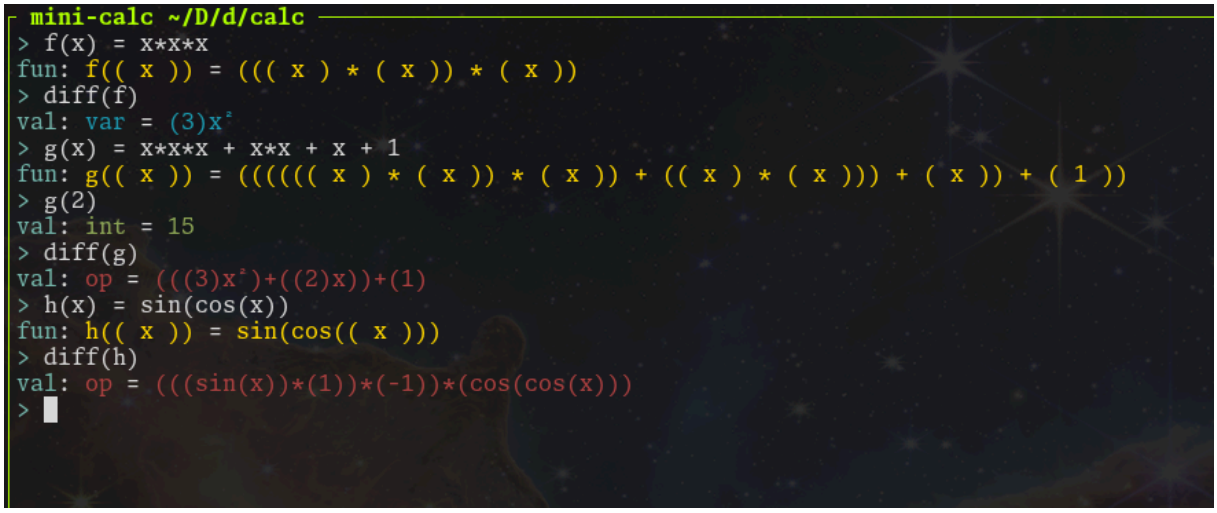

Figure 27: Example of a user-defined function differentiation

X. Non interactive use

With version 2.12.0 non interactive use was added to the calculator if you have a quick computation to run and don't want to start the REPL.

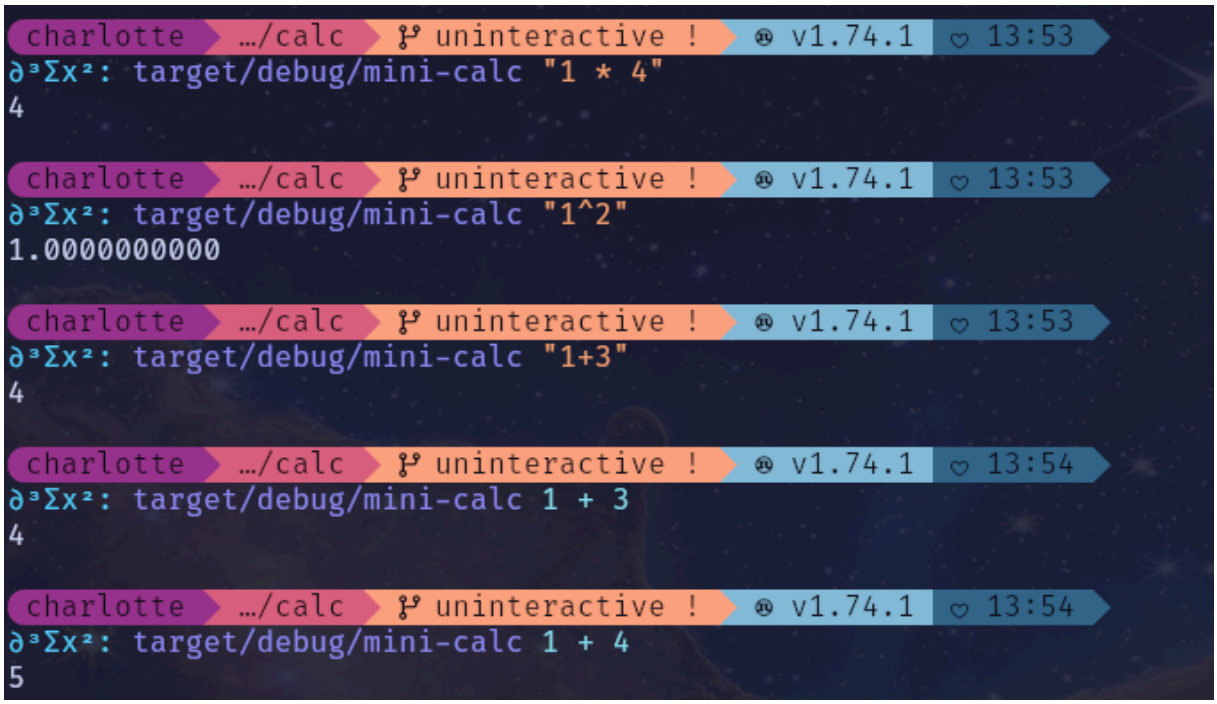

Figure 28: Example of non interactive use

XI. Syntax coloration

In 3.3.0 the coloration was added.

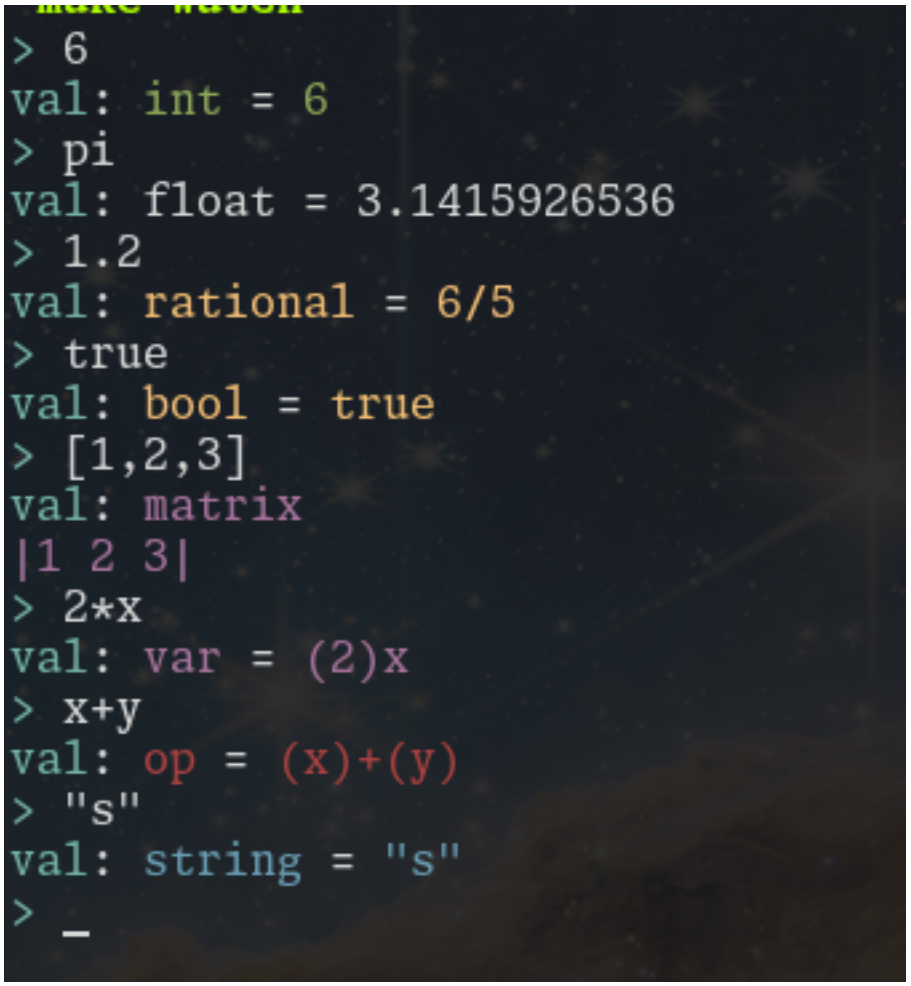

Figure 29: Syntax coloration as of 3.3.3

XII. Float display

With v3.4.0 there is now a way to chose how you want your floats to be displayed.

For example in normal mode if you type

```
1 > 1.5
```

you will see

```
1 val: float = 1.5000000000
```