

Calc Manual

Last updated, 2025-02-14

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
XII.1. Modes	26
XII.2. Toggle float	27
XII.3. Example	28
XII.4. Default float mode	28

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 were 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

```
1 cargo install mini-calc
```

Bash

or via the source

```
1 git clone https://github.com/vanilla-extracts/calc
2 cd calc
3 cargo build --release
4 ./target/release/mini-calc
```

Bash

or alternatively, build/run using nix

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

Bash

Visit **Calc** to see all the install page

I.2. Contributors

Name	Role	Website
Charlotte THOMAS	Main developer/ Maintainer	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 subtraction
- * for the multiplication
- / for the division (or for a rational)
- ^ for the exponentiation

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)
```

```
03Σx²: ../../target/release/mini-calc
Welcome to calc v2.0.0 by Charlotte Thomas
type help for getting help for the commands

> var = 5
> var +1
6
>
```

Figure 1: Example of setting a variable

II.3. Built-in variables

The following variables are built-in:

- `pi` is π 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.

```
[charlotte ➜ .../calc ➜ ⌂ add_functions ! ➜ @ v1.73.0 ➜ 16:03 ➜ ⏱ 11s
∂³Σx²: ../../target/release/mini-calc
Welcome to Calc v2.3.0 by Charlotte Thomas, type help for help
> help
Calc v2.3.0 Help
> info : show infos
> exit : exit the program
> help : print this help
> verbose : toggle the verbose
> version : prints the version

> cos(pi)
-1
> cos(180,true)
-1
> cos(180)
-0.5984600690578581
> |
```

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.



```
charlotte ➜ .../calc ➜ ↵ add_functions !? ➜ @ v1.73.0 ❤ 16:05 ➜ ⏱ 4s
d³Σx²: ../../target/release/mini-calc
Welcome to Calc v2.3.0 by Charlotte Thomas, type help for help
> exp(1)
2.718281828459045
> exp(2)
7.38905609893065
> exp(2,2)
4
> ln(e)
1
> ln(2,2)
1
>
```

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.

```
charlotte ➜ .../calc ➜ ⌂ stdlib ! ➜ v1.73.0 ➜ 22:36 ➜ ⏴ 2s
ð³Σx²: ../../target/release/mini-calc
Welcome to Calc v2.3.1 by Charlotte Thomas, type help for help
> sqrt(4)
2
> sqrt(8)
2.8284271247461903
> sqrt(8,3)
2
```

Figure 4: Usage of sqrt

III.5. Partial function

The calculator's language supports partial function.

```
>Welcome to Calc v2.7.0 by Charlotte Thomas, type help for help
:> log10(x) = log(x,10)
:> exp10(x) = exp(x,10)
:> log10(exp10(2))
2.00000
:> log10(exp10(3))
3.00000
:> log10(exp10(3.2))
3.20000
:> log10(exp10(3.4))
3.40000
:> cos_degrees(x) = cos(x,true)
:> cos_degrees(180)
-1.00000
:> cos_degrees(90)
0.00000
:> █
```

Figure 5: Example of a partial function

III.6. Vectorization

Functions have been vectorized.

```
Welcome to Calc v2.10.0 by Charlotte Thomas, type help for help
> sqrt([1,4,9,16,25])
*[1.00000,2.00000,3.00000,4.00000,5.00000]
> █
```

Figure 6: Example of a vectorized function

III.7. User defined function

You can define your own function

```
ð³Σx²: ../../target/release/mini-calc
Welcome to Calc v2.5.0 by Charlotte Thomas, type help for help
> test(x) = 2*x
> test(2)
4
> let(x,y) = x+y
> let(1,2)
3
> let(test(2),2)
6
> let(cos(pi),1)
0
> █
```

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

```
* mini-calc.toml  x |
1 general_color = 'purple'
2
3 [greeting]
4 greeting_message = 'Welcome to Calc %version% by %author%, type help for help'
5 greeting_color = 'blue'
6
7 [prompt]
8 prompt = '> '
9 prompt_color = 'cyan'
```

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

```
mini-calc.toml
1 general_color = 'cyan'
2
3 [greeting]
4 greeting_message = 'Heya! This is calc! Version %version%, and I think coded by %author% but who knows!'
5 greeting_color = '#f7a8d8'
6
7 [prompt]
8 prompt = 'λπ: '
9 prompt_color = '#55cdf1'
```

Figure 9: Example of a modified config
it looks like

```
Heya! This is calc! Version v2.2.2, and I think coded by Charlotte Thomas but who knows!
λπ: 1+1
2
λπ: pi
3.141592653589793
λπ: e
2.718281828459045
λπ: 
```

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

```
Welcome to Calc v2.8.0 by Charlotte Thomas, type help for help
> config.
The greeting colour is set to blue which prints
Welcome to Calc %version% by %author%, type help for help
The prompt is > in cyan
Main color is purple which looks like
This is the general colour
If you've modified your config and it doesn't look good, the author (Charlotte Thomas) declines any responsibilities.

> config show
The greeting colour is set to blue which prints
Welcome to Calc %version% by %author%, type help for help
The prompt is > in cyan
Main color is purple which looks like
This is the general colour
If you've modified your config and it doesn't look good, the author (Charlotte Thomas) declines any responsibilities.

> config reload
Your configuration has been reloaded
> config set
You need more argument for this command
> config set general_color #f7a8d8
Greeting color has been set to #f7a8d8, reload for this to take effect
> config reload
Your configuration has been reloaded
> config set greeting_message Heya! %author% I love you!
Prompt has been updated to Heya! %author% I love you!, reload for this to take effect
> config reload
Your configuration has been reloaded
> config
The greeting colour is set to blue which prints
Heya! %author% I love you!
The prompt is > in cyan
Main color is #f7a8d8 which looks like
This is the general colour
If you've modified your config and it doesn't look good, the author (Charlotte Thomas) declines any responsibilities.

> exit

charlotte > ./calc ↵ github-page $ ⑧ v1.73.0 ⑨ 13:49 ⑩ 1m4s
ðΣx²: mini-calc
Heya! Charlotte Thomas I love you!
> config reset
Your config has been reseted to default settings
> exit

charlotte > ./calc ↵ github-page $ ⑧ v1.73.0 ⑨ 13:49 ⑩ 10s
ðΣx²: mini-calc
Welcome to Calc v2.8.0 by Charlotte Thomas, type help for help
> █
```

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()`

```
Welcome to Calc v2.9.0 by Charlotte Thomas, type help for help
> plot()
> plot(): displays help
> plot(f): plot f
> plot(f,title,xlabel,ylabel): plot f with title,xlabel,ylabel
> plot(f,mode): plot f with the mode=LINE|LINEMARKS|MARKS(default)
> plot(f,title,xlabel,ylabel,mode): plot f with title,xlabel,ylabel and mode
> plot(f,start,end,step,mode): plot f between start and end with steps and mode
> plot(f,start,end,step,title,xlabel,ylabel,mode): combines
```

Figure 13: Help of plot

VI.2. Plot

VI.2.1. Default

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

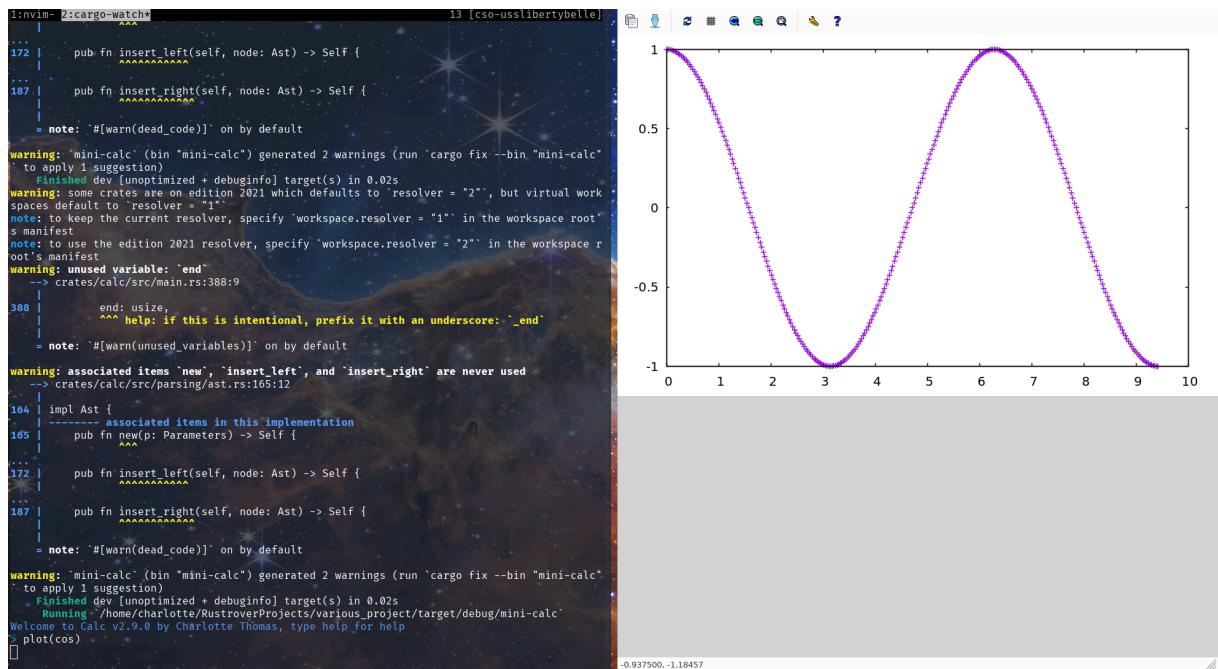


Figure 14: Plot of the cos function with default values

VI.2.2. Options

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

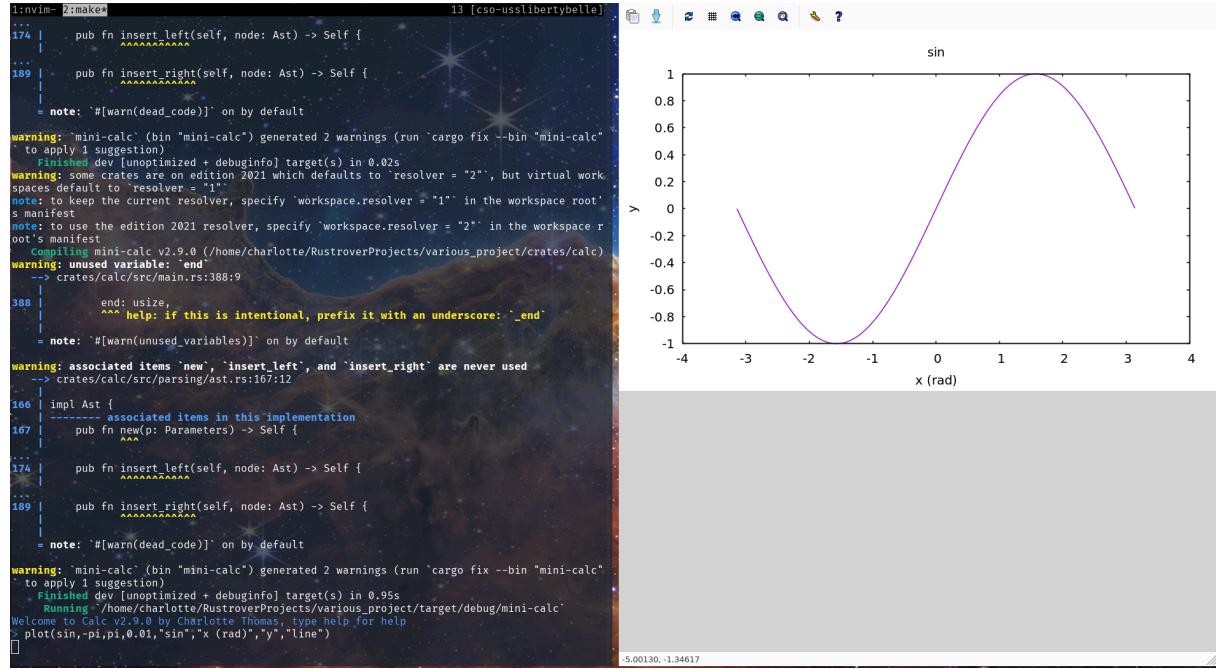


Figure 15: Plot with overloading of default values

VI.2.3. Plot your own function

You can plot your own defined functions!

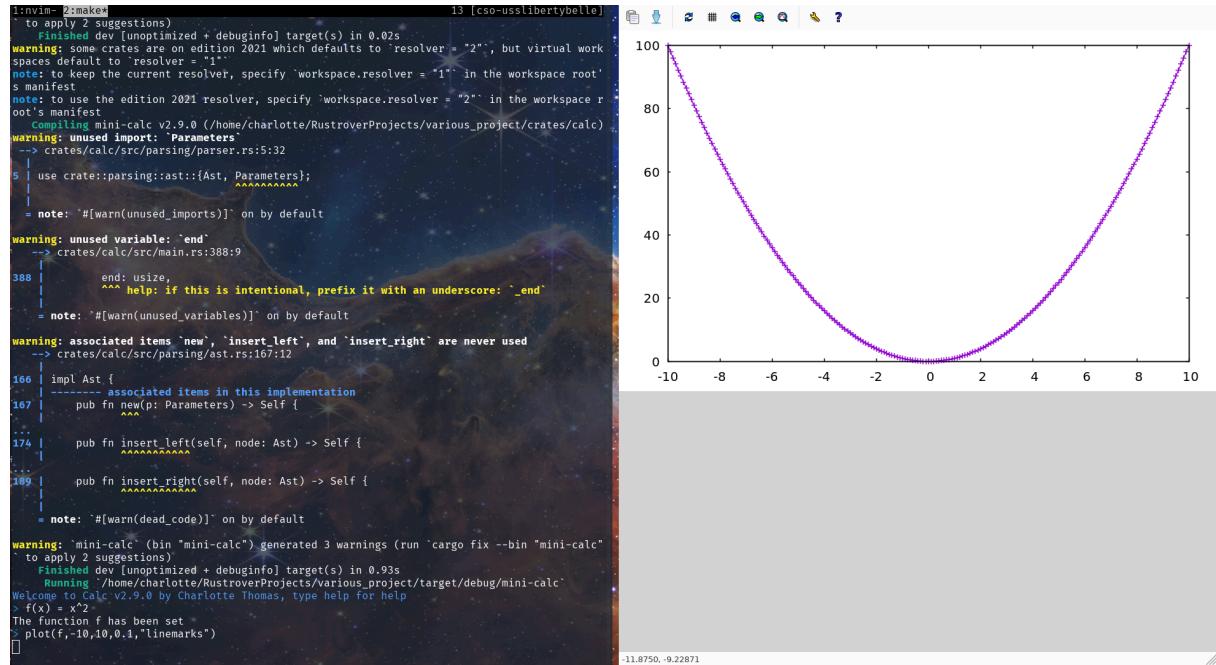


Figure 16: Plot of an user-defined function

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.

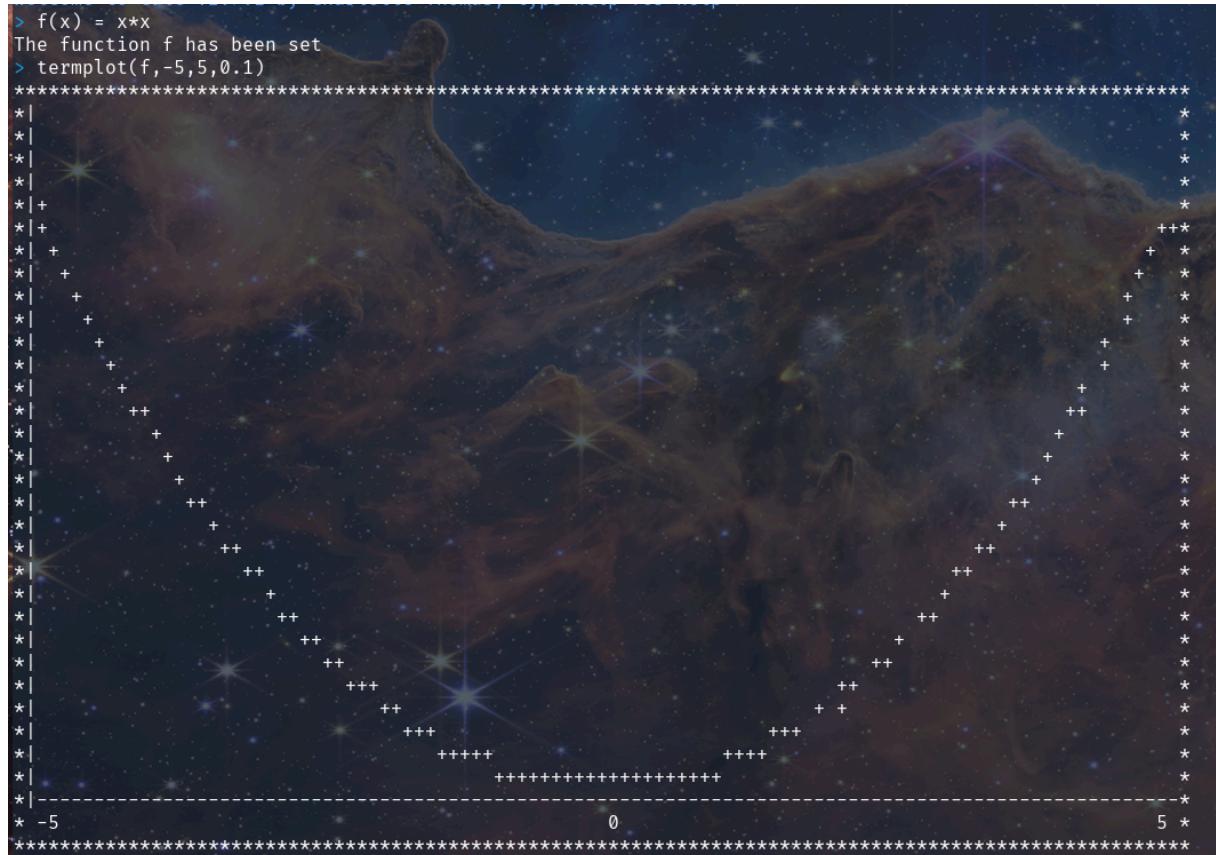


Figure 17: Terminal plotting of an user defined function

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!

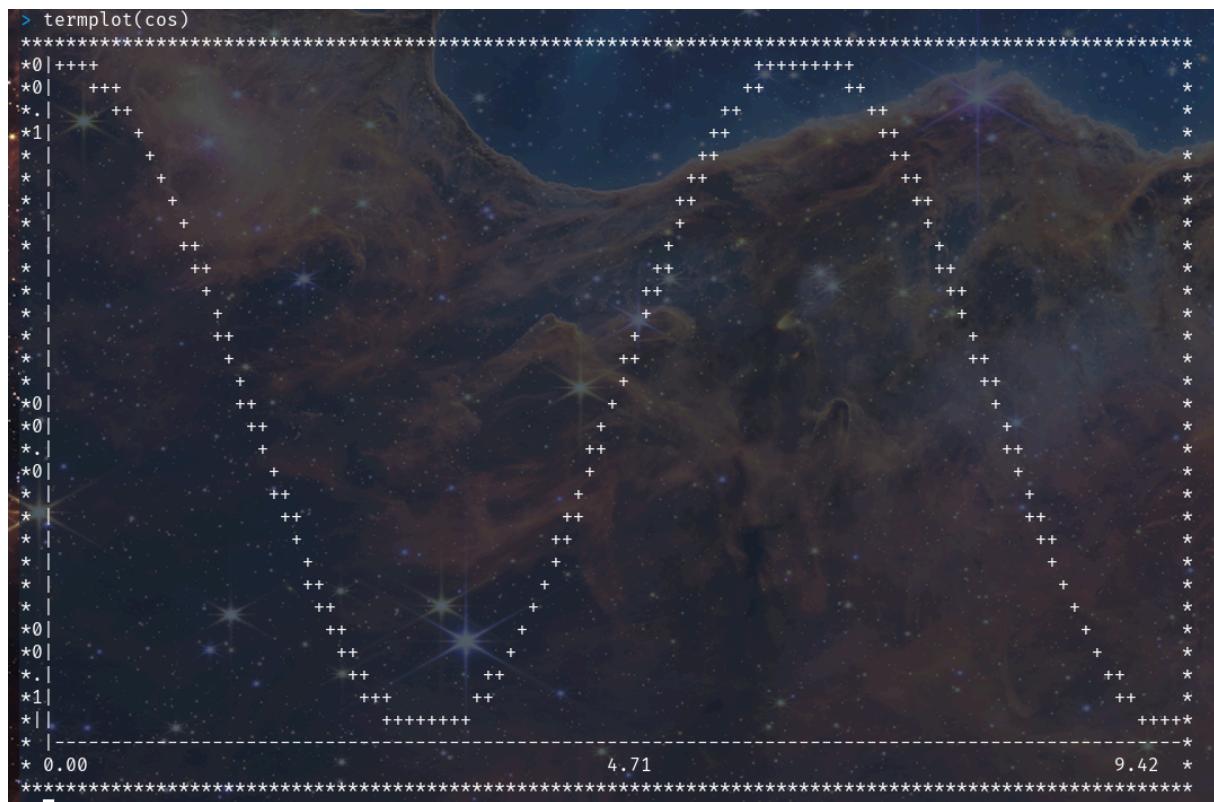


Figure 19: Example of a plot with y auto-scaling

VII. Vectors computation

You can compute vectors using these functions,

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

```
δ³Σx²: mini-calc
Welcome to Calc v2.6.0 by Charlotte Thomas, type help for help
> [1,2,3] + [1,2,3]
[2,4,6]
> [1,2,3] - [1,2,3]
[0,0,0]
> [1,2,3] * [1,2,3]
14
> -[1,2,3]
[-1,-2,-3]
```

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

```
Welcome to Calc v2.7.0 by Charlotte Thomas, type help for help
> let = [[1,2],[3,4]]
> det(let)
-2.00000
> invert(let)
[[[-2.00000,1.00000],[1.50000,-0.50000]]
> transpose(let)
[[[1,3],[2,4]]
> [[1,2,3],[1,2,3]] * [[1,2],[3,4],[3,2]]
[[7,10],[7,10]]
> ]
```

Figure 21: Example of matrix computation

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

```
> [[1,2],[3,4]]  
+---+  
| 1 2 |  
| 3 4 |  
+---+  
> a = [[1,2,3],[4,0,6],[7,8,9]]  
a = +----+  
| 1 2 3 |  
| 4 0 6 |  
| 7 8 9 |  
+----+  
> a  
+----+  
| 1 2 3 |  
| 4 0 6 |  
| 7 8 9 |  
+----+  
> invert(a)  
+-----+  
| -4/5 1/10 1/5 |  
| 1/10 -1/5 1/10 |  
| 8/15 1/10 -2/15 |  
+-----+  
> invert([[1,2,3,4],[5,0,6,0],[7,0,8,0],[9,10,11,12]])  
+-----+  
| 0 -4 3 0 |  
| -3/4 17/4 -13/4 1/4 |  
| 0 7/2 -5/2 0 |  
| 5/8 -15/4 11/4 -1/8 |  
+-----+  
> █
```

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

```
Welcome to Calc v2.11.0 by Charlotte Thomas, type help for help
> 1/3 + 1/3
2/3
> abs(-1/4)
1/4
> 1/4 + 1/8
3/8
> 4/9 * 1/2
2/9
> 5 * 1/3
5/3
> 5/1/3
5/3
> 5/(1/3)
15
> █
```

Figure 23: Example of rational computations

```
Welcome to Calc v2.11.1 by Charlotte Thomas, type help for help
> invert([[1,2,3],[4,0,6],[7,8,9]])
[[-4/5,1/10,1/5],[1/10,-1/5,1/10],[8/15,1/10,-2/15]]
> █
```

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!

```
> 2*x+y+x  
val: op = ((3)x)+(y)  
> (x+x)*y  
val: op = ((2)x)*(y)  
> (x+x)*y**x  
val: op = ((2)x2)*(y)  
> █
```

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

```
mini-calc ~/D/d/calc
> diff(sin)
cos((1)x)
> diff(cos)
val: op = (sin((1)x))*(-1)
> diff(ln)
val: var = 1/x
> diff(exp)
exp((1)x)
>
```

Figure 26: Example of a built-in function differentiation

```
mini-calc ~/D/d/calc
> f(x) = x*x*x
fun: f(( x )) = ((( x ) * ( x )) * ( x ))
> diff(f)
val: var = (3)x^2
> g(x) = x*x*x + x*x + x + 1
fun: g(( x )) = ((((( x ) * ( x )) * ( x )) + (( x ) * ( x ))) + ( x ) + ( 1 ))
> g(2)
val: int = 15
> diff(g)
val: op = (((3)x^2)+(2x))+1
> h(x) = sin(cos(x))
fun: h(( x )) = sin(cos(( x )))
> diff(h)
val: op = (((sin(x))*(1))*(-1))*(cos(cos(x)))
>
```

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.

```
charlotte ➜ .../calc ➜ ⌂ uninteractive ! @ v1.74.1 ◽ 13:53
ð³Σx²: target/debug/mini-calc "1 * 4"
4

charlotte ➜ .../calc ➜ ⌂ uninteractive ! @ v1.74.1 ◽ 13:53
ð³Σx²: target/debug/mini-calc "1^2"
1.0000000000

charlotte ➜ .../calc ➜ ⌂ uninteractive ! @ v1.74.1 ◽ 13:53
ð³Σx²: target/debug/mini-calc "1+3"
4

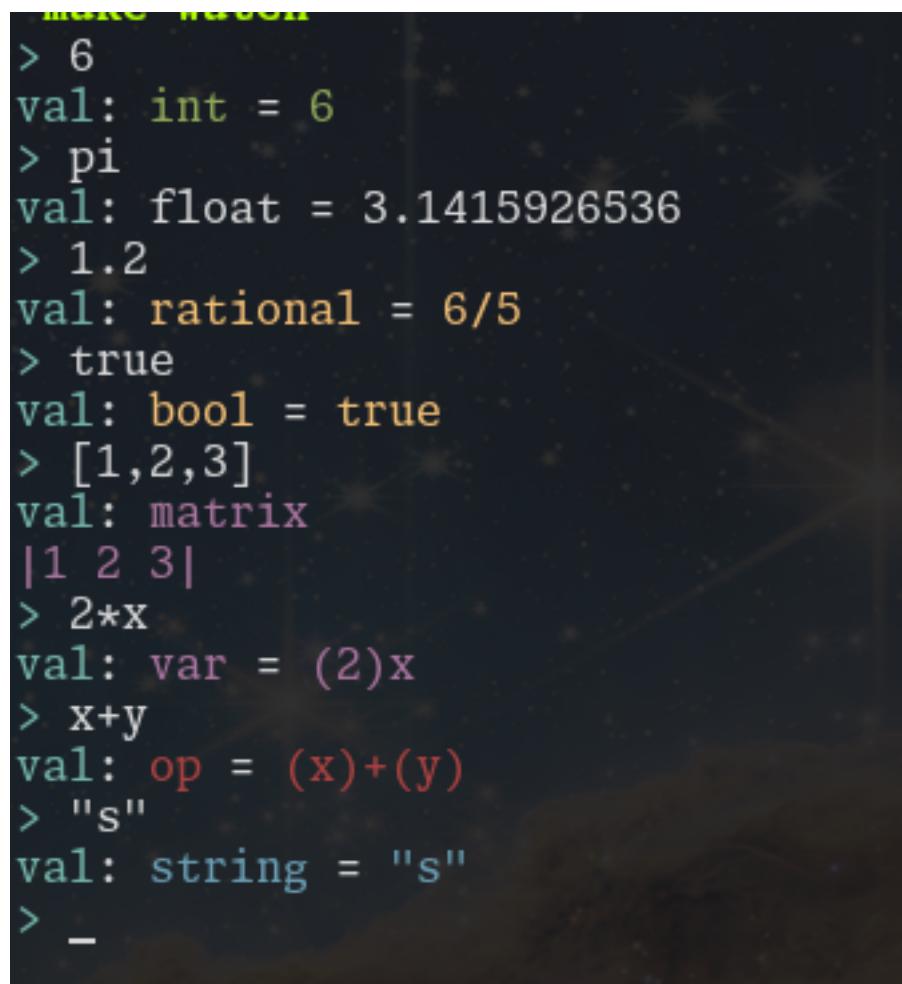
charlotte ➜ .../calc ➜ ⌂ uninteractive ! @ v1.74.1 ◽ 13:54
ð³Σx²: target/debug/mini-calc 1 + 3
4

charlotte ➜ .../calc ➜ ⌂ uninteractive ! @ v1.74.1 ◽ 13:54
ð³Σx²: target/debug/mini-calc 1 + 4
5
```

Figure 28: Example of non interactive use

XI. Syntax coloration

In 3.3.0 the coloration was added.



The screenshot shows a terminal window with syntax coloration applied to the output of a command-line interface. The colors used include green for numbers (6, 3.1415926536), blue for floating-point numbers (1.2, 6/5), orange for booleans (true), purple for lists ([1,2,3]), red for matrices (|1 2 3|), pink for variables (2*x, x+y), brown for operators (op = (x)+(y)), and yellow for strings ("s"). The prompt (>) and the word 'val:' are also colored.

```
> 6
val: int = 6
> pi
val: float = 3.1415926536
> 1.2
val: rational = 6/5
> true
val: bool = true
> [1,2,3]
val: matrix
|1 2 3|
> 2*x
val: var = (2)x
> x+y
val: op = (x)+(y)
> "s"
val: string = "s"
> -
```

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.

XII.1. Modes

For example in normal mode if you type

```
1 > 1.5
```

It outputs:

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

In exact mode, that

```
1 > 1.5
```

Outputs:

```
1 val: rational = 3/2
```

And finally in science mode, this

```
1 > 150.2
```

Outputs

```
1 val: float = 1.502*10^2
```

XII.2. Toggle float

There is a `toggle_float` command in the REPL now.

Its usage is easy and it is auto-completed.

```
1 toggle_float <normal|science|exact>
```

It then displays a message telling you in which mode you toggled.

By default it is in exact mode.

XII.3. Example

```
> toggle_float normal
You toggled the float mode to : normal mode.
Example: 1.5=1.5000000000
> 15021.345
val: float = 15021.3450000000
> toggle_float exact
You toggled the float mode to : exact mode.
Example: 1.5=3/2
> 15021.345
val: rational = 3004269/200
> toggle_float science
You toggled the float mode to : science mode.
Example: 1500.1=1.5001*103
> 15021.345
val: float = 1.5021345000*104
> █
```

Figure 30: The different float modes

XII.4. Default float mode

As of v3.4.2 you can alter the default float mode. By default it is exact

You can alter it by modifying the config file directly or by using the config set command.

```
1 > config set float_mode exact|normal|science
```

And then reload the config with

```
1 config reload
```