Zebraw

Zebraw is a lightweight and fast package for displaying code blocks with line numbers in typst, supporting code line highlighting. The term *Zebraw* is a combination of *zebra* and *raw*, for the highlighted lines will be displayed in the code block like a zebra lines.

Contents

```
Header and Footer 2
Colors 3
zebraw-init 4
Example ...... 5
```

Starting

Import zebraw package by #import "@preview/zebraw:0.4.0": * then follow with #show: zebraw to start using zebraw in the simplest way. To manually display some specific code blocks in zebraw, you can use #zebraw() function:

```
#grid(
  columns: (1fr, 1fr),
  [Hello], [world!],
)

1 #grid(
2   columns: (1fr, 1fr),
3   [Hello], [world!],
4 )

1 #grid(
2   columns: (1fr, 1fr),
3   [Hello], [world!],
4 )
```

Features

#zebraw(

Line Highlighting

You can highlight specific lines in the code block by passing the highlight-lines parameter to the zebraw function. The highlight-lines parameter can be a single line number or an array of line numbers.

```
1 #grid(
  // Single line number:
                                                2
                                                    columns: (1fr, 1fr),
 highlight-lines: 2,
                                                3
                                                    [Hello], [world!],
    typ
  #grid(
   columns: (1fr, 1fr),
   [Hello], [world!],
                                                1 = Fibonacci sequence
                                                2 The Fibonacci sequence is defined
                                                  through the
)
                                                3 recurrence relation F_n = F(n-1)
                                                  + F_{(n-2)}.
#zebraw(
                                                4 It can also be expressed in
 // Array of line numbers:
                                                  _closed form:_
 highlight-lines: (6, 7) + range(9, 15),
                                                6 \ F_n = round(1 / sqrt(5))
 = Fibonacci sequence
                                                 phi.alt^n), quad
 The Fibonacci sequence is defined
                                                   phi.alt = (1 + sqrt(5)) / 2 $
                                                7
through the
 recurrence relation F_n = F(n-1) +
F_{n-2}.
                                               9 #let count = 8
 It can also be expressed in _closed
                                              10 #let nums = range(1, count + 1)
form:_
                                              11 #let fib(n) = (
                                              12 if n \le 2 \{ 1 \}
 $ F n = round(1 / sgrt(5) phi.alt^n),
                                              13
                                                    else { fib(n-1) + fib(n-2) }
                                              14 )
   phi.alt = (1 + sqrt(5)) / 2 $
                                              15
                                              16 The first #count numbers of the
 #let count = 8
                                                  sequence are:
 #let nums = range(1, count + 1)
                                              17
 \#let fib(n) = (
                                              18 #align(center, table(
   if n \le 2 \{ 1 \}
    else { fib(n - 1) + fib(n - 2) }
                                              19
                                                    columns: count,
                                                    .. nums.map(n \Rightarrow $F_#n$),
                                              20
                                              21
                                                    .. nums.map(n \Rightarrow str(fib(n))),
 The first #count numbers of the
                                              22 ))
sequence are:
 #align(center, table(
   columns: count,
    .. nums.map(n \Rightarrow $F_#n$),
    ..nums.map(n \Rightarrow str(fib(n))),
  ))
)
```

#zebraw(

#zebraw(

is needed.

highlight-lines: (

to the highlight-lines parameter.

Comment

highlight-lines: (> The Fibonacci sequence is defined (1, [The Fibonacci sequence is through the recurrence relation defined through the recurrence relation $F_{n} = F_{n-1} + F_{n-2}$

You can add comments to the highlighted lines by passing an array of line numbers and comments

```
F_n = F_{n-1} + F_{n-2}
     It can also be expressed in _closed
 form: _ $ F_n = round(1 / sqrt(5))
 phi.alt^n), quad
     phi.alt = (1 + sqrt(5)) / 2 $]),
     // Passing a range of line numbers in
 the array should begin with `..
      .. range(9, 14),
     (13, [The first \mbox{\#count numbers of}
 the sequence.]),
   ), typ
   = Fibonacci sequence
   #let count = 8
   \#let nums = range(1, count + 1)
   \#let fib(n) = (
     if n \le 2 \{ 1 \}
     else \{ fib(n-1) + fib(n-2) \}
   #align(center, table(
     columns: count,
     .. nums.map(n \Rightarrow $F_#n$),
     ..nums.map(n \Rightarrow str(fib(n))),
   ))
 )
Comments can begin with a flag, which is ">" by default. You can change the flag by passing the
comment-flag parameter to the zebraw function:
```

// Comments can only be passed when

highlight-lines is an array, so at the

end of the single element array, a comma

```
form:
             F_n = \left\lfloor \frac{1}{\sqrt{5}} \phi^n \right\rceil, \quad \phi = \frac{1 + \sqrt{5}}{2}
 2 \#let count = 8
 3 \#let nums = range(1, count + 1)
 4 \#let fib(n) = (
    if n \le 2 \{ 1 \}
                         1) + fib(n - 2) }
      else { fib(n
 7 )
 9 #align(center, table(
10
      columns: count,
11
       .. nums.map(n \Rightarrow $F_#n$),
       .. nums.map(n \Rightarrow str(fib(n))),
12
13 ))
    > The first #count numbers of the
    sequence.
 1 = Fibonacci sequence
```

2 #let count = 8

4 #let fib(n) = (

4 #let fib(n) = (

if $n \le 2 \{ 1 \}$

columns: count,

columns: count,

.. nums.map(n \Rightarrow \$F_#n\$),

The fibonacci sequence is defined through the recurrence relation $F_n =$

.. nums.map($n \Rightarrow str(fib(n))$),

> The first #count numbers of the

else { fib(n - 1) + fib(n - 2) }

The Fibonacci sequence is defined

through the recurrence relation

if $n \leq 2 \{ 1 \}$

It can also be expressed in closed

1 = Fibonacci sequence

else { fib(n - 1) + fib(n - 2) } (6, [The Fibonacci sequence is defined through the recurrence relation → The Fibonacci sequence is $F_n = F_{n-1} + F_{n-2},$ defined through the recurrence

```
),
   comment-flag: "→→",
     typ
   = Fibonacci sequence
   #let count = 8
   #let nums = range(1, count + 1)
   \#let fib(n) = (
     if n \leq 2 \{ 1 \}
     else \{ fib(n-1) + fib(n-2) \}
   #align(center, table(
     columns: count,
     ..nums.map(n \Rightarrow $F_#n$),
     ..nums.map(n \Rightarrow str(fib(n))),
   ))
)
will be disabled as well):
#zebraw(
   highlight-lines: (
     (6, [The Fibonacci sequence is
 defined through the recurrence relation
 F_n = F_{n-1} + F_{n-2},
  ),
   comment-flag: "",
     typ
```

= Fibonacci sequence

#let count = 8

```
\text{relation } F_n = F_{n-1} + F_{n-2}
                                                        7)
                                                        8
                                                        9 #align(center, table(
                                                       10
                                                             columns: count,
                                                       11
                                                             .. nums.map(n \Rightarrow $F_#n$),
                                                             .. nums.map(n \Rightarrow str(fib(n))),
                                                       12
                                                       13 ))
To disable the flag feature, pass "" to the comment-flag parameter (the indentation of the comment
                                                        1 <u>= Fibonacci sequence</u>
                                                        2 \#let count = 8
                                                        3 \#let nums = range(1, count + 1)
```

3 #let nums = range(1, count + 1)

 $F_n = F_{n-1} + F_{n-2}$ #let nums = range(1, count + 1) 7) #let fib(n) = (8 if $n \leq 2 \{ 1 \}$ 9 #align(center, table(

10

10

11

12))

sequence.

 $F_{n-1} + F_{n-2}$

```
#align(center, table(
     columns: count,
     ..nums.map(n \Rightarrow $F_#n$),
     ..nums.map(n \Rightarrow str(fib(n))),
   ))
 )
Header and Footer
Usually, the comments passing by a dictionary of line numbers and comments are used to add a
header or footer to the code block:
 #zebraw(
   highlight-lines: (
     (header: [*Fibonacci sequence*]),
      .. range(8, 13),
     // Numbers can be passed as a string
 in the dictionary, but it's too ugly.
     ("12": [The first \#count numbers of
 the sequence.]),
     (footer: [The fibonacci sequence is
 defined through the recurrence relation
 F_n = F_{n-1} + F_{n-2},
   ), typ
   #let count = 8
```

#let nums = range(1, count + 1)

else { fib(n - 1) + fib(n - 2) }

#let fib(n) = (if $n \leq 2 \{ 1 \}$

#align(center, table(columns: count,

else { fib(n - 1) + fib(n - 2) }

```
.. nums.map(n \Rightarrow $F_#n$),
11
      .. nums.map(n \Rightarrow str(fib(n))),
12
13 ))
Fibonacci sequence
 1 \#let count = 8
 2 #let nums = range(1, count + 1)
 3 \#let fib(n) = (
     if n \le 2 \{ 1 \}
     else { fib(n-1) + fib(n-2) }
 6)
 7
 8 #align(center, table(
```

```
.. nums.map(n \Rightarrow $F_#n$),
     ..nums.map(n \Rightarrow str(fib(n))),
   ))
 )
Or you can use header and footer parameters to add a header or footer to the code block:
 #zebraw(
   highlight-lines: (
      .. range(8, 13),
     (12, [The first \#count numbers of
 the sequence.]),
   ),
   header: [*Fibonacci sequence*],
    ``typ
   #let count = 8
   #let nums = range(1, count + 1)
   \#let fib(n) = (
     if n \leq 2 \{ 1 \}
     else \{ fib(n-1) + fib(n-2) \}
   #align(center, table(
     columns: count,
     .. nums.map(n \Rightarrow $F_#n$),
      ..nums.map(n \Rightarrow str(fib(n))),
   ))
   footer: [The fibonacci sequence is
 defined through the recurrence relation
 F_n = F_{n-1} + F_{n-2}
Language Tab
```

```
Fibonacci sequence
           1 \#let count = 8
             2 \#let nums = range(1, count + 1)
             3 \#let fib(n) = (
                                             if n \leq 2 \{ 1 \}
                                                             else { fib(n-1) + fib(n-2) }
             5
             6)
             7
           8 #align(center, table(
          9
                                                             columns: count,
10
                                                                 .. nums.map(n \Rightarrow $F_#n$),
                                                                 .. nums.map(n \Rightarrow str(fib(n))),
11
12 ))
                                      > The first #count numbers of the
                                      sequence.
   The fibonacci sequence is defined
   through the recurrence relation F_n=% \frac{1}{2}\left( \frac{1}{n}\right) \left( \frac{1}{n}
   F_{n-1} + F_{n-2}
```

typst

lang: true, typst #grid(columns: (1fr, 1fr),

#zebraw(

[Hello], [world!],

#zebraw(

Copyable

If copyable is set to true, line numbers will not be copied when copying exported code.

1 #grid(

1 #grid(

2

3

3

columns: (1fr, 1fr),

columns: (1fr, 1fr),

[Hello], [world!],

[Hello], [world!],

If lang is set to true, then there will be a language tab on the top right corner of the code block:

```
copyable: true,
     typst
   #grid(
    columns: (1fr, 1fr),
    [Hello], [world!],
 )
For comparison:
             copyable: false
   1 #grid(
   2 columns: (1fr, 1fr),
       [Hello], [world!],
   3
   4)
```

```
1 #grid(
                                               columns: (1fr, 1fr),
                                               3 [Hello], [world!],
                                               4)
However when a code block is copyable, it won't be able to cross page. Only line numbers will be
```

1 pub fn fibonacci_reccursive(n:

panic!("{} is negative!",

i32) → u64 {

2 if n < 0 {

}

3 n); copyable: true

#show: zebraw ```rust

}

PRs are welcome!

Theme

excluded for being selected.

#show: zebraw-init.with(..zebraw-

themes.zebra, lang: false)

```
pub fn fibonacci_reccursive(n: i32) \rightarrow
u64 {
    if n < 0 {
        panic!("{} is negative!", n);
    }
    match n {
        0 ⇒ panic!("zero is not a right
argument to fibonacci_reccursive()!"),
        1 \mid 2 \Rightarrow 1
        3 \Rightarrow 2,
        \_ \Rightarrow fibonacci\_reccursive(n - 1)
+ fibonacci_reccursive(n - 2),
  }
#show: zebraw-init.with(..zebraw-
themes.zebra-reverse, lang: false)
#show: zebraw
rust
```

```
5
       match n {
            0 ⇒ panic!("zero is not a
   right argument to
  fibonacci_reccursive()!"),
            1 \mid 2 \Rightarrow 1,
8
           3 \Rightarrow 2,
            _{-} \Rightarrow
   fibonacci_reccursive(n - 1) +
   fibonacci_reccursive(n - 2),
10 }
11 }
 1 pub fn fibonacci_reccursive(n:
```

```
i32) \rightarrow u64  {
                                                         if n < 0 {
                                                   3
                                                              panic!("{} is negative!",
                                                    n);
pub fn fibonacci_reccursive(n: i32) →
u64 {
                                                   4
                                                          }
                                                      match n {
                                                   5
   if n < 0 {
                                                              0 ⇒ panic!("zero is not a
        panic!("{} is negative!", n);
                                                     right argument to
    match n {
                                                     fibonacci_reccursive()!"),
       0 ⇒ panic!("zero is not a right
                                                 7
                                                      1 \mid 2 \Rightarrow 1,
argument to fibonacci_reccursive()!"),
                                                              3 \Rightarrow 2,
                                                   8
        1 \mid 2 \Rightarrow 1,
                                                               \rightarrow
        3 \Rightarrow 2,
                                                     fibonacci_reccursive(n - 1) +
        \_ \Rightarrow fibonacci\_reccursive(n - 1)
                                                     fibonacci_reccursive(n - 2),
+ fibonacci_reccursive(n - 2),
```

10

11 }

}

Customization

There are 3 ways to customize code blocks in your document:

- Manually render some specific blocks by #zebraw() function and passing parameters to it.
- By passing parameters to #show: zebraw.with() will affect every raw block after the #show rule, except blocks created manually by #zebraw() function.
- By passing parameters to #show: zebraw-init.with() will affect every raw block after the #show rule, including blocks created manually by #zebraw() function. By using zebraw-init without any parameters, the values will be reset to default.

Inset

Customize the inset of each line by passing a dictionary to the inset parameter:

```
#zebraw(
  inset: (top: 6pt, bottom: 6pt),
                                                1 #grid(
    `typ
                                                    columns: (1fr, 1fr),
  #grid(
   columns: (1fr, 1fr),
                                                    [Hello], [world!],
    [Hello], [world!],
                                                4 )
)
```

Colors

Customize the background color by passing a color or an array of colors to the backgroundcolor parameter.

```
#zebraw(
  background-color: luma(235),
    typ
  #grid(
   columns: (1fr, 1fr),
    [Hello], [world!],
)
#zebraw(
  background-color: (luma(235),
luma(245), luma(255)),
   ``typ
  #grid(
   columns: (1fr, 1fr),
   [Hello], [world!],
)
```

```
1 #grid(
2 columns: (1fr, 1fr),
3
    [Hello], [world!],
4 )
1 #grid(
  columns: (1fr, 1fr),
3
    [Hello], [world!],
4 )
```

Customize the highlight color by passing a color to the highlight-color parameter:

```
#zebraw(
 highlight-lines: 1,
 highlight-color: blue.lighten(90%),
    text
 I'm so blue!
              -- George III
```

```
1 I'm so blue!
              -- George III
```

Customize the comments' background color by passing a color to the comment-color parameter:

```
#zebraw(
  highlight-lines: (
   (2, "auto indent!"),
  comment-color: yellow.lighten(90%),
    text
  I'm so blue!
               -- George III
  I'm not.
              -- Hamilton
)
```

```
1 I'm so blue!
              -- George III
              > auto indent!
3 I'm not.
              -- Hamilton
```

Customize the language tab's background color by passing a color to the lang-color parameter. typst

```
#zebraw(
  lang: true,
  lang-color: teal,
    typst
  #grid(
    columns: (1fr, 1fr),
    [Hello], [world!],
)
```

```
1 #grid(
  columns: (1fr, 1fr),
   [Hello], [world!],
```

Font To customize the arguments of comments' font and the language tab's font, pass a dictionary to

comment-font-args parameter and lang-font-args parameter. Language tab will be rendered as comments if nothing is passed.

#zebraw(

```
highlight-lines: (
   (2, "columns ... "),
 lang: true,
 comment-color: white,
 comment-font-args: (
    font: "IBM Plex Sans",
    style: "italic"
 ),
typst
  #grid(
   columns: (1fr, 1fr),
   [Hello], [world!],
)
#zebraw(
```

```
1 #grid(
2 columns: (1fr, 1fr),
    > columns...
   [Hello], [world!],
3
4 )
```

typst

typst

```
highlight-lines: (
   (2, "columns ... "),
  ),
  lang: true,
  lang-color: eastern,
  lang-font-args: (
   font: "libertinus serif",
    weight: "bold",
   fill: white,
  comment-font-args: (
    font: "IBM Plex Sans",
    style: "italic"
 ),
typst
  #grid(
    columns: (1fr, 1fr),
    [Hello], [world!],
)
```

```
2 columns: (1fr, 1fr),
    > columns...
3
    [Hello], [world!],
4 )
```

1 #grid(

disabled.

Extend Extend at vertical is enabled at default. When there's header or footer it will be automatically

```
#zebraw(
  extend: false,
    typst
  #grid(
   columns: (1fr, 1fr),
    [Hello], [world!],
```

```
1 #grid(
    columns: (1fr, 1fr),
3
    [Hello], [world!],
4 )
```

comment-color color The background color of the comments. When it's set to none, it will be rendered in a lightened highlight-color. Default: none lang-color color The background color of the language tab. The color is set to none at default and it will be rendered in comments' color. Default: none comment-flag string or content The flag at the beginning of comments. Default: ">" lang boolean Whether to show the language tab. Default: true comment-font-args dictionary The arguments passed to comments' font. Default: (:) dictionary lang-font-args The arguments passed to the language tab's font. Default: (:) extend boolean Whether to extend the vertical spacing. Default: true copyable boolean Whether to enable "copyable" feature. Default: false body content The body zebraw Block of code with highlighted lines and comments. #zebraw(rust highlight-lines: (Calculate Fibonacci number using $(9, "50 \Rightarrow 12586269025"),$ reccursive function 1 pub fn fibonacci_reccursive(n: lang: true, i32) → u64 { header: "Calculate Fibonacci number using if n < 0 { 2 reccursive function", 3 panic!("{} is negative!", n); pub fn fibonacci_reccursive(n: i32) → u64 { } if n < 0 { 5 match n { panic!("{} is negative!", n); 0 ⇒ panic!("zero is not a right argument to match n { fibonacci_reccursive()!"), $0 \Rightarrow panic!("zero is not a right)$ 7 $1 \mid 2 \Rightarrow 1,$ argument to fibonacci_reccursive()!"), 8 $3 \Rightarrow 2$, $1 \mid 2 \Rightarrow 1$, 9 \Rightarrow fibonacci_reccursive(n - 1) fibonacci_reccursive(n - 1) + + fibonacci_reccursive(n - 2), fibonacci_reccursive(n - 2), } **>** 50 ⇒ 12586269025 10 11 } **Parameters** highlight-lines: array int, header: string content, footer: string content, inset: dictionary, background-color: color array, highlight-color: color, comment-color: color, lang-color: color, comment-flag: string content, lang: boolean, comment-font-args: dictionary, lang-font-args: dictionary, extend: boolean, copyable: boolean, body: content) -> content highlight-lines array or int Lines to highlight or comments to show. #zebraw(Fibonacci sequence highlight-lines: range(3, 7), header: [*Fibonacci sequence*], 1 #let count = 82 #let nums = range(1, count + 1) #let count = 8 3 #let fib(n) = (#let nums = range(1, count + 1) if $n \le 2 \{ 1 \}$ #let fib(n) = (if $n \le 2 \{ 1 \}$ 5 else { fib(n-1) + fib(n-1)2) } else { fib(n-1) + fib(n-2) } 6) #align(center, table(8 #align(center, table(columns: count, 9 columns: count, .. nums.map(n \Rightarrow \$F_#n\$), .. nums.map(n \Rightarrow \$F_#n\$), ..nums.map(n \Rightarrow str(fib(n))), 10)) 11 .. nums.map(n \Rightarrow str(fib(n))), 12)) footer: [The fibonacci sequence is The fibonacci sequence is defined defined through the recurrence relation through the recurrence relation $F_n = F_{n-1} + F_{n-2}$ $F_n = F_{n-1} + F_{n-2}$ Default: () header string or content The header of the code block. Default: none footer string or content The footer of the code block. Default: none dictionary inset The inset of each line. #zebraw(inset: (top: 6pt, bottom: 6pt), 1 #let count = 8 typst #let count = 8 2 #let nums = range(1, count + 1) #let nums = range(1, count + 1) #let fib(n) = (3 #let fib(n) = (if $n \le 2 \{ 1 \}$ if $n \leq 2 \{ 1 \}$ else { fib(n - 1) + fib(n - 2) } else { fib(n - 1) + fib(n -2) } #align(center, table(columns: count, 6) ..nums.map(n \Rightarrow \$F_#n\$), ..nums.map(n \Rightarrow str(fib(n))), 8 #align(center, table() columns: count, .. nums.map(n \Rightarrow \$F_#n\$), .. nums.map(n \Rightarrow str(fib(n))), 12)) Default: none background-color color or array The background color of the block and normal lines. #zebraw(background-color: (luma(240), 1 #let count = 8 luma(245), luma(250), luma(245)), 2 #let nums = range(1, count + 1) typst 3 #let fib(n) = (#let count = 8 4 if $n \le 2 \{ 1 \}$ #let nums = range(1, count + 1) 5 else { fib(n-1) + fib(n-1)#let fib(n) = (2) } if $n \leq 2 \{ 1 \}$ else { fib(n - 1) + fib(n - 2) } 6) 7 8 #align(center, table(#align(center, table(9 columns: count, columns: count, 10 .. nums.map(n \Rightarrow \$F_#n\$), .. nums.map(n \Rightarrow \$F_#n\$), .. nums.map(n \Rightarrow str(fib(n))), ..nums.map(n \Rightarrow str(fib(n))), 12)))) Default: none highlight-color color The background color of the highlighted lines. Default: none comment-color color The background color of the comments. The color is set to none at default and it will be rendered in a lightened highlight-color. #zebraw(highlight-color: yellow.lighten(80%), 1 = Fibonacci sequence comment-color: yellow.lighten(90%), > The Fibonacci sequence is highlight-lines: (defined through the recurrence (1, [The Fibonacci sequence is relation $F_n = F_{n-1} + F_{n-2}$ defined through the recurrence relation 2 #let count = 8 $F_n = F_{n-1} + F_{n-2}$ 3 #let nums = range(1, count + 1).. range(9, 14), 4 #let fib(n) = ((13, [The first $\mbox{\#count numbers of}$ the sequence.]), if $n \le 2 \{ 1 \}$), typ else { fib(n - 1) + fib(n -2) } = Fibonacci sequence 7) #let count = 8 8 #let nums = range(1, count + 1) 9 #align(center, table(#let fib(n) = (columns: count, 10 if $n \le 2 \{ 1 \}$.. nums.map(n \Rightarrow \$F_#n\$), else { fib(n-1) + fib(n-2) } 11 12 .. nums.map(n \Rightarrow str(fib(n))), 13)) #align(center, table(> The first #count numbers of columns: count, the sequence. ..nums.map(n \Rightarrow \$F_#n\$), ..nums.map(n \Rightarrow str(fib(n))),))) Default: none lang-color color The background color of the language tab. The color is set to none at default and it will be rendered in comments' color. #zebraw(typst lang: true, 1 #grid(lang-color: eastern, lang-font-args: (columns: (1fr, 1fr), font: "libertinus serif", [Hello], [world!], weight: "bold", 4) fill: white), typst #grid(columns: (1fr, 1fr), [Hello], [world!],) Default: none comment-flag string or content The flag at the beginning of comments. The indentation of codes will be rendered before the flag. When the flag is set to "", the indentation before the flag will be disabled as well. #zebraw(comment-flag: "", 1 <u>= Fibonacci sequence</u> highlight-lines: (The Fibonacci sequence is (1, [The Fibonacci sequence is defined through the recurrence defined through the recurrence relation relation $F_n = F_{n-1} + F_{n-2}$ $F_n = F_{n-1} + F_{n-2},$ 2 #let count = 8.. range(9, 14), 3 #let nums = range(1, count + 1)(13, [The first $\mbox{\#count numbers of}$ 4 #let fib(n) = (the sequence.]),), typ 5 if $n \le 2 \{ 1 \}$ else { fib(n - 1) + fib(n -6 = Fibonacci sequence 2) } #let count = 8 7) #let nums = range(1, count + 1) 8 #let fib(n) = (9 #align(center, table(if $n \le 2 \{ 1 \}$ 10 columns: count, else { fib(n - 1) + fib(n - 2) } 11 .. nums.map(n \Rightarrow \$F_#n\$), .. nums.map(n \Rightarrow str(fib(n))), 12 #align(center, table(13)) columns: count, The first #count numbers of the ..nums.map(n \Rightarrow \$F_#n\$), sequence. ..nums.map(n \Rightarrow str(fib(n))),))) Default: none lang boolean Whether to show the language tab on the top-right corner of the code block. #zebraw(typ lang: true. 1 #grid(typ columns: (1fr, 1fr), #grid(columns: (1fr, 1fr), 3 [Hello,], [world!], [Hello,], [world!], 4)) Default: none comment-font-args dictionary The arguments passed to comments' font. Default: none lang-font-args dictionary The arguments passed to the language tab's font. #zebraw(typ lang: true, comment-font-args: (font: "IBM Plex 1 <u>= Fibonacci sequence</u> Serif", style: "italic"), > The Fibonacci sequence is defined through lang-font-args: (font: "IBM Plex Sans", the recurrence relation $F_n = F_{n-1} + F_{n-2}$ weight: "bold"), 2 #let count = 8 highlight-lines: (3 #let nums = range(1, count + 1)(1, [The Fibonacci sequence is 4 #let fib(n) = (defined through the recurrence relation if n ≤ 2 { 1 } 5 $F_n = F_{n-1} + F_{n-2},$ 6 else { fib(n-1) + fib(n-1).. range(9, 14), (13, [The first $\mbox{\#count numbers of}$ 2) } the sequence.]), 7)), typ 8 9 #align(center, table(= Fibonacci sequence 10 columns: count, #let count = 8 .. nums.map(n \Rightarrow \$F_#n\$), 11 #let nums = range(1, count + 1).. nums.map($n \Rightarrow str(fib(n))$), #let fib(n) = (12 13)) if $n \le 2 \{ 1 \}$ else $\{ fib(n-1) + fib(n-2) \}$ > The first #count numbers of the sequence. #align(center, table(columns: count, ..nums.map(n \Rightarrow \$F_#n\$), ..nums.map(n \Rightarrow str(fib(n))),))) Default: none extend boolean Whether to extend the vertical spacing. #zebraw(1 #grid(extend: false, columns: (1fr, 1fr), 2 typ #grid(3 [Hello,], [world!], columns: (1fr, 1fr), 4) [Hello,], [world!],) Default: none copyable boolean Whether to enable "copyable" feature. If copyable feature is enabled, lines of code are displayed together in a block to allow copying code in an export. However when this feature is enabled, the code block cannot be displayed across pages. #zebraw(copyable: true, 1 #grid(typ columns: (1fr, 1fr), #grid(3 [Hello,], [world!], columns: (1fr, 1fr), 4) [Hello,], [world!],) body content The body.

Documentation

Initialize the zebraw block in global.

highlight-color: color,
comment-color: color,
lang-color: color,

background-color: color array,

comment-flag: string content,

comment-font-args: dictionary,
lang-font-args: dictionary,

inset: dictionary,

lang: boolean,

extend: boolean,
copyable: boolean,
body: content
) -> content

The inset of each line.

background-color

Default: luma(245)

highlight-color

dictionary

zebraw-init()zebraw()

zebraw-init

Parameters

inset

zebraw-init(

The default value of most parameters are none for it will use the default value in zebraw-init.

Default: (top: 0.34em, right: 0.34em, bottom: 0.34em, left: 0.34em)

color or array

The background color of the block and normal lines.

color

The background color of the highlighted lines.

Default: rgb("#94e2d5").lighten(70%)

Example

```
rust
Calculate Fibonacci number using reccursive function
 1 pub fn fibonacci reccursive(n: i32) → u64 {
 2
       if n < 0 {
           panic!("{} is negative!", n);
           > to avoid negative numbers
       }
 4
       match n {
           0 ⇒ panic!("zero is not a right argument to fibonacci_reccursive()!"),
 6
           1 \mid 2 \Rightarrow 1
 8
           3 \Rightarrow 2,
            _ ⇒ fibonacci_reccursive(n - 1) + fibonacci_reccursive(n - 2),
 9
           > 50 ⇒ 12586269025
10
11 }
```