typst #zebraw(highlight-lines: range(3, 7), Fibonacci sequence header: [*Fibonacci sequence*], 1 # let count = 8``typst 2 #let nums = range(1, count + 1)#let count = 83 # let fib(n) = (#let nums = range(1, count + 1)if n <= 2 { 1 } 4 #let fib(n) = (else { fib(n - 1) + fib(n if n <= 2 { 1 } else { fib(n - 1) + fib(n - 2) } 2) } 6) #align(center, table(8 #align(center, table(columns: count, 9 columns: count, $..nums.map(n => F_#n$),$ 10 ..nums.map(n => \$F #n\$), ..nums.map(n => str(fib(n))), ..nums.map(n => str(fib(n))), 11 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: () numbering-offset int The offset of line numbers. The first line number will be numbering-offset + 1. Defaults to 0. Default: 0 header string or content The header of the code block. Default: none footer string or content The footer of the code block. Default: none inset dictionary The inset of each line. typst #zebraw(inset: (top: 6pt, bottom: 6pt), 1 # let count = 8```typst #let count = 82 #let nums = range(1, count + 1)#let nums = range(1, count + 1)3 # let fib(n) = (#let fib(n) = (if n <= 2 { 1 } if n <= 2 { 1 } else { fib(n - 1) + fib(n - 2) } else { fib(n - 1) + fib(n -2) } #align(center, table(6) columns: count, ..nums.map(n => \$F #n\$), ..nums.map(n => str(fib(n))),)) 8 #align(center, table(9 columns: count, $..nums.map(n \Rightarrow F #n$),$ 10 ..nums.map(n => str(fib(n))), 12)) Default: none background-color color or array The background color of the block and normal lines. typst #zebraw(background-color: (luma(240), luma(245), 1 # let count = 8luma(250), luma(245)), 2 #let nums = range(1, count + 1)``typst 3 # let fib(n) = (#let count = 84 if n <= 2 { 1 } #let nums = range(1, count + 1)else { fib(n - 1) + fib(n -5 #let fib(n) = (**2**) } if $n \le 2 \{ 1 \}$ else { fib(n - 1) + fib(n - 2) } 7 8 #align(center, table(#align(center, table(9 columns: count, columns: count, 10 ..nums.map(n => \$F_#n\$), $..nums.map(n => F_#n$),$ 11 ..nums.map(n => str(fib(n))), ..nums.map(n => str(fib(n))), 12)))) Default: none highlight-color color The background color of the highlighted lines. Default: none comment-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. typ #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 defined]relation $F_n = F_{n-1} + F_{n-2}$ through the recurrence relation \$F_n = 2 #let count = 8 $F_{(n-1)} + F_{(n-2)}$, 3 #let nums = range(1, count + 1)..range(9, 14), 4 # let fib(n) = ((13, [The first $\mbox{\#count numbers of the}$ if n <= 2 { 1 } sequence.]),), 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) = (10 columns: count, if n <= 2 { 1 } 11 ..nums.map(n => \$F_#n\$), else { fib(n - 1) + fib(n - 2) } ..nums.map(n => str(fib(n))), 12 13)) #align(center, table(> The first #count numbers of columns: count, the sequence. $..nums.map(n => F_#n$),$..nums.map(n => 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. typst #zebraw(lang: true, 1 #grid(lang-color: eastern, 2 columns: (1fr, 1fr), lang-font-args: (3 [Hello], [world!], font: "libertinus serif", 4) weight: "bold", 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. typ #zebraw(comment-flag: "", 1 = Fibonacci sequence highlight-lines: (The Fibonacci sequence is (1, [The Fibonacci sequence is defined defined through the recurrence through the recurrence relation \$F_n = $\text{relation } F_n = F_{n-1} + F_{n-2}$ $F_{(n-1)} + F_{(n-2)}$, 2 #let count = 8..range(9, 14), 3 #let nums = range(1, count + 1)(13, [The first \#count numbers of the 4 # let fib(n) = (sequence.]),), ```typ if n <= 2 { 1 } 6 else { fib(n - 1) + fib(n -= Fibonacci sequence 2) } #let count = 87) #let nums = range(1, count + 1)8 #let fib(n) = (9 #align(center, table(if n <= 2 { 1 } 10 columns: count, else { fib(n - 1) + fib(n - 2) } 11 \dots nums.map(n => \$F_#n\$), 12 ..nums.map(n => str(fib(n))), #align(center, table(13)) columns: count, The first #count numbers of the $..nums.map(n => F_#n$),$ sequence. ..nums.map(n => str(fib(n))),))) Default: none lang boolean or string or content Whether to show the language tab, or a string or content of custom language name to display. typ #zebraw(lang: true, 1 #grid(```typ columns: (1fr, 1fr), #grid(3 [Hello,], [world!], columns: (1fr, 1fr), 4) [Hello,], [world!],) **Typst** #zebraw(lang: strong[Typst], 1 #grid(```typ columns: (1fr, 1fr), #grid(3 [Hello,], [world!], columns: (1fr, 1fr), 4) [Hello,], [world!], Default: none dictionary comment-font-args The arguments passed to comments' font. Default: none lang-font-args dictionary The arguments passed to the language tab's font. typ #zebraw(lang: true, 1 <u>= Fibonacci sequence</u> comment-font-args: (font: "IBM Plex > The Fibonacci sequence is defined through Serif", style: "italic"), the recurrence relation $F_n = F_{n-1} + F_{n-2}$ lang-font-args: (font: "IBM Plex Sans", 2 #let count = 8 weight: "bold"), 3 #let nums = range(1, count + 1)highlight-lines: (4 #let fib(n) = ((1, [The Fibonacci sequence is defined if n <= 2 { 1 } through the recurrence relation \$F_n = else { fib(n - 1) + fib(n - $F_{(n-1)} + F_{(n-2)}$, 2) } ..range(9, 14), 7) (13, [The first \#count numbers of the sequence.]), 8 **),** 9 #align(center, table(10 columns: count, = Fibonacci sequence ..nums.map(n => \$F_#n\$), 11 #let count = 812 ..nums.map(n => str(fib(n))), #let nums = range(1, count + 1)#let fib(n) = (13)) if n <= 2 { 1 } > The first #count numbers of the sequence. else { fib(n - 1) + fib(n - 2) } #align(center, table(columns: count, $..nums.map(n => F_#n$),$..nums.map(n => str(fib(n))),)) Default: none numbering-font-args dictionary The arguments passed to the line numbers' font. typ #zebraw(numbering-font-args: (fill: blue), 1 = Fibonacci sequence ``typ 2 #let count = 8 = Fibonacci sequence 3 #let nums = range(1, count + 1)#let count = 84 # let fib(n) = (#let nums = range(1, count + 1)5 if $n \le 2 \{ 1 \}$ #let fib(n) = (6 else { fib(n - 1) + fib(n if n <= 2 { 1 } 2) } else { fib(n - 1) + fib(n - 2) } 7) 8 #align(center, table(9 #align(center, table(columns: count, 10 columns: count, $..nums.map(n => F_#n$),$..nums.map(n => \$F_#n\$), 11 ..nums.map(n => str(fib(n))), ..nums.map(n => str(fib(n))), 12)) 13))) Default: none extend boolean Whether to extend the vertical spacing. typ #zebraw(1 #grid(extend: false, ``typ columns: (1fr, 1fr), #grid(3 [Hello,], [world!], columns: (1fr, 1fr), 4) [Hello,], [world!],) Default: none numbering-separator Whether to show the numbering separator line. typ #zebraw(numbering-separator: true, 1 #grid(```typ 2 columns: (1fr, 1fr), #grid(3 [Hello,], [world!], columns: (1fr, 1fr), 4) [Hello,], [world!],) Default: none hanging-indent boolean Whether to show the hanging indent. typ #zebraw(hanging-indent: true, 1 This is a short line. ``typ Do a deer, a female deer. Ray, This is a short line. a drop of golden sun. Me, a Do a deer, a female deer. Ray, a drop name I call myself. Far, a of golden sun. Me, a name I call myself. long, long way to run. Sew, a Far, a long, long way to run. Sew, a needle pulling thread. La, a needle pulling thread. La, a note to note to follow sew. Tea, a follow sew. Tea, a drink with jam and drink with jam and bread. That bread. That will bring us back to do, oh, will bring us back to do, oh, oh, oh. oh, oh.) Default: none indentation int The amount of indentation, used to draw indentation lines. typ #zebraw(indentation: 2, 1 #grid(```typ columns: (1fr, 1fr), #grid(3 [Hello,], [world!], columns: (1fr, 1fr), 4) [Hello,], [world!], Default: none line-range array or dictionary Line range to show. Accepts an array of 2 integers [a, b) or a dictionary with keys named range and keep-offset. Defaults to [1, none). (none means the last line). Noticed that the line numbers are 1-based. typ #zebraw(line-range: (2, 4), 2 columns: (1fr, 1fr), ``typ [Hello,], [world!], #grid(columns: (1fr, 1fr), [Hello,], [world!], typ line-range: (range: (2, 4), keep-offset: 1 columns: (1fr, 1fr), false), [Hello,], [world!], ```typ #grid(columns: (1fr, 1fr), [Hello,], [world!],) Default: (1, none) length or relative block-width (Only for HTML) The width of the code block. Default: 42em wrap boolean (Only for HTML) Whether to wrap the code lines. Default: true body content The body.

Documentation

numbering: boolean,

highlight-lines: array int,
numbering-offset: int,
header: string content,
footer: string content,
inset: dictionary,

background-color: color array,

comment-flag: string content,
lang: boolean string content,
comment-font-args: dictionary,
lang-font-args: dictionary,
numbering-font-args: dictionary,

numbering-separator: boolean,

line-range: array dictionary,
block-width: length relative,

boolean

array or int

Lines to highlight or comments to show.

Whether to show the line numbers.

hanging-indent: boolean,

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

extend: boolean,

indentation: int,

wrap: boolean,
body: content
-> content

numbering: false,

columns: (1fr, 1fr),

[Hello,], [world!],

numbering

#zebraw(

```typ

#grid(

Default: none

highlight-lines

)

Block of code with highlighted lines and comments.

• zebraw()

Parameters zebraw(

zebraw

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

#grid(

)

columns: (1fr, 1fr),

[Hello,], [world!],

typ