LibreLogo Toolbar

From LibreOffice Help

LibreLogo is a simple, localized, Logo-like programming environment with turtle vector graphics for teaching of computing (programming and word processing), DTP and graphic design. See http://www.numbertext.org/logo/librelogo.pdf.

Contents

LibreLogo toolbar

The LibreLogo toolbar (View » Toolbars » Logo) contains turtle moving, program start, stop, home, clear screen, program editor/syntax highlighting/translating icons and an input bar (command line).

Turtle moving icons

They are equivalents of the Logo commands “FORWARD 10”, “BACK 10”, “LEFT 15”, “RIGHT 15”. Clicking on one of the icons will also focus the turtle shape scrolling the page to its position.

Start Logo program

Click on the icon “Start Logo program” to execute the text (or only the selected) text of the Writer document as a LibreLogo program. In an empty document an example program will be inserted and executed.

Click on the icon “Stop” to stop the program execution.

Home

Click on the icon “Home” to reset the position and settings of the turtle.

Clear screen

Click on the icon “Clear screen” to remove the drawing objects of the document.

Program editor/Syntax highlighting/Translating

The “magic wand” icon sets 2-page layout for program editing, expands and converts to uppercase the abbreviated, lowercase Logo commands in the Writer document. Change the language of the document (Tools » Options » Language Settings » Languages » Western) and click on this icon to translate the Logo program to the selected language.

Command line

Hit Enter in the command line to execute its content. To stop the program use the icon “Stop”.

Hold down the Enter to repeat the command line, for example, on the following command sequence:

```
FORWARD 200 LEFT 89
```

To reset the command line click triple in it or press Ctrl-A to select the previous commands, and type the new commands.

Graphical user interface of basic turtle settings

Turtle shape of LibreLogo is a normal fixed size drawing object. You can positionate and rotate it on standard way, too, using the mouse and the Rotate icon of the Drawing Object Properties toolbar. Modify Line Width, Line Color and Area Color settings of the turtle shape to set PENSIZE, PENCOLOR and FILLCOLOR attributes of LibreLogo.

Program editing

LibreLogo drawings and programs use the same Writer document. The LibreLogo canvas is on the first page of the Writer document. You can insert a page break before the LibreLogo programs and set page zoom using the “magic wand” icon of the Logo toolbar, also change the font size for a comfortable 2-page layout for LibreLogo programming: left (first) page is the canvas, right (second) page is the LibreLogo program editor.

LibreLogo programming language

LibreLogo is an easily localizable, Logo-like programming language, localized in several languages by LibreOffice native language communities. It is back-compatible with the older Logo systems in the case of the simple Logo programs used in education, eg.

```
TO triangle :size
REPEAT 3 [
FORWARD :size
LEFT 120
]
END

triangle 10 triangle 100 triangle 200
```

Differences from the Logo programming language

- List members are comma separated: POSITION [0, 0]
- Program blocks and lists are different
  - Program blocks need space or new line at parenthesization: REPEAT 10 [FORWARD 10 LEFT 36]
- Lists need close parenthesization: POSITION [0, 0], and not POSITION [0, 0]

- 1-line function declarations are not supported (TO and END need new lines).

Other features of LibreLogo

- The colon is optional before the variable names.
  TO triangle size
  REPEAT 3 [FORWARD size LEFT 120]
  END
- String notation supports also orthographical and Python syntax.
  PRINT "word ; original Logo syntax
  PRINT “Arbitrary text.” ; orthography, Writer
  PRINT 'Arbitrary text.' ; Python syntax
- Python list and string handling
  PRINT “text”[2] ; print “x”
  PRINT “text”[1:3] ; print “ex”
- Python-like FOR loop
- Python-like variable declaration:
  x = 15
  PRINT x
- There are no extra query functions:
  PRINT FILLCOLOR
  p = POSITION
  PRINT p
  REPEAT 10 [POSITION ANY POSITION p]
- Alternative parenthesization in function calls
  TO star size color
  FILLCOLOR color
  REPEAT 5 [LEFT 72 FORWARD size RIGHT 144 FORWARD size]
  FILL
  END

  star 100 “red”
  star (100, “green”)

star(100, “blue”)

LibreLogo commands

Basic syntax

Case sensitivity

Commands, color constants are case insensitive:

```
PRINT “Hello, World!”
print “Hello, World, again!”
```

Variable names are case sensitive:

```
a = 5
A = 7
PRINT a
PRINT A
```

Program lines

Lines of a LibreLogo program are paragraphs in the LibreOffice Writer document. A program line can contain multiple commands:

```
PRINT “Hello, World!” PRINT “LibreLogo”
```

Comments

Lines or line parts are comments from a semicolon to the end of the line (paragraph):

```
 ; some comments
PRINT 5 * 5 ; some comments
```

Break program lines to multiple paragraphs

It’s possible to break a program line for more paragraphs using the character tilde at the end of the line:

```
PRINT “This is a very long ” + ~
“warning message”
```

Turtle moving

FORWARD (fd)

```
FORWARD 10 ; move forward 10pt (1pt = 1/72 inch)
FORWARD 10pt ; see above
FORWARD 0.5in ; move forward 0.5 inch (1 inch = 2.54 cm)
FORWARD 1" ; see above
FD 1mm
FD 1cm
```

BACK (bk)

```
BACK 10 ; move back 10pt
```

LEFT (lt)

```
LEFT 90 ; turn counterclockwise 90 degrees
LEFT 90° ; see above
LT 3h ; see above (clock position)
LT any ; turn to a random position
```

RIGHT (rt)

```
RIGHT 90 ; turn clockwise 90 degrees
```

PENUP (pu)

```
PENUP ; turtle will move without drawing
```

PENDOWN (pd)

```
PENDOWN ; turtle will move with drawing
```

POSITION (pos)

```
POSITION [0, 0] ; turn and move to the top-left corner
POSITION PAGESIZE ; turn and move to the bottom-right corner
POSITION [PAGESIZE[0], 0] ; turn and move to the top-right corner
POSITION ANY ; turn and move to a random position
```

HEADING (seth)

```
HEADING 0 ; turn north
HEADING 12h ; see above
HEADING [0, 0] ; turn to the top-left corner
HEADING ANY ; turn to a random direction
```

Other turtle commands

HIDETURTLE (ht)

```
HIDETURTLE ; hide turtle (until the showturtle command)
```

SHOWTURTLE (st)

```
SHOWTURTLE ; show turtle
```

HOME

```
HOME ; reset initial turtle settings and position
```

CLEARSCREEN (cs)

```
CLEARSCREEN ; remove drawing objects of the document
```

FILL and CLOSE

```
FILL ; close and fill the actual line shape or points
CLOSE ; close the actual line shape or join the actual points
```

Example: filling a regular triangle:

```
FORWARD 50 LEFT 120 FORWARD 50 FILL
```

Example: drawing a regular triangle:

```
FORWARD 50 LEFT 120 FORWARD 50 CLOSE
```

Pen settings

PENSIZE (ps)

```
PENSIZE 100 ; line width is 100 points
PENSIZE ANY ; equivalent of PENSIZE RANDOM 10
```

PENCOLOR/PENCOLOUR (pc)

```
PENCOLOR “red” ; set red pen color (by color name, see color constants)
PENCOLOR [255, 255, 0] ; set yellow color (RGB list)
PENCOLOR 0xffff00 ; set yellow color (hexa code)
PENCOLOR 0 ; set black color (0x000000)
PENCOLOR ANY ; random color
PENCOLOR [5] ; set red color (by color identifier, see color constants)
PENCOLOR “invisible” ; invisible pen color for shapes without visible outline
PENCOLOR “~red” ; set random red color
```

PENTRANSPARENCY

```
PENTRANSPARENCY 80 ; set the transparency of the actual pen color to 80%
```

PENCAP/LINECAP

```
PENCAP “none” ; without extra line end (default)
PENCAP “round” ; rounded line end
PENCAP “square” ; square line end
```

PENJOINT/LINEJOINT

```
PENJOINT “rounded” ; rounded line joint (default)
PENJOINT “miter” ; sharp line joint
PENJOINT “bevel” ; bevel line joint
PENJOINT “none” ; without line joint
```

PENSTYLE

```
PENSTYLE “solid” ; solid line (default)
PENSTYLE “dotted” ; dotted line
PENSTYLE “dashed” ; dashed line

 ; custom dot–dash pattern specified by a list with the following arguments:
 ; – number of the neighbouring dots
 ; – length of a dot
 ; – number of the neighbouring dashes
 ; – length of a dash
 ; – distance of the dots/dashes
 ; – type (optional):
 ; 0 = dots are rectangles (default)
```

```
 ; 2 = dots are squares (lengths and distances are relative to the pensize)

PENSTYLE [3, 1mm, 2, 4mm, 2mm, 2] ; ...——...——...——
```

Fill settings

FILLCOLOR/FILLCOLOUR (fc)

```
FILLCOLOR “blue” ; fill with blue color, see also PENCOLOR
FILLCOLOR “invisible” CIRCLE 10 ; unfilled circle
FILLCOLOR [“blue”, “red”] ; gradient between red and blue
FILLCOLOR [[255, 255, 255], [255, 128, 0]] ; between white and orange
FILLCOLOR [“blue”, “red”, 1, 0, 0] ; set axial gradient (with the required rotation and
border settings), possible values: 0-5 = linear, axial, radial, elliptical, square and
rectangle gradients
FILLCOLOR [“red”, “blue”, 0, 90, 20] ; linear with 20% border, rotated with 90 degrees from
the actual heading of the turtle
FILLCOLOR [“red”, “blue”, 0, 90, 20, 0, 0, 200, 50] ; from 200% to 50% intensity
FILLCOLOR [ANY, ANY, 2, 0, 0, 50, 50] ; radial gradient with random colors and 50-50%
horizontal and vertical positions of the center
```

FILLTRANSPARENCY

```
FILLTRANSPARENCY 80 ; set the transparency of the actual fill color to 80%
FILLTRANSPARENCY [80] ; set linear transparency gradient from 80% to 0%
FILLTRANSPARENCY [80, 20] ; set linear transparency gradient from 80% to 20%
FILLTRANSPARENCY [80, 20, 1, 90] ; set axial transparency gradient rotated with 90 degrees
from the actual heading of the turtle
FILLTRANSPARENCY [80, 20, 2, 0, 20, 50, 50] ; set radial transparency gradient from outer
80% to inner 20% transparency with 20% border and with 50-50% horizontal and vertical
positions of the center
```

FILLSTYLE

```
FILLSTYLE 0 ; fill without hatches (default)
FILLSTYLE 1 ; black single hatches (horizontal)
FILLSTYLE 2 ; black single hatches (45 degrees)
FILLSTYLE 3 ; black single hatches (-45 degrees)
FILLSTYLE 4 ; black single hatches (vertical)
FILLSTYLE 5 ; red crossed hatches (45 degrees)
FILLSTYLE 6 ; red crossed hatches (0 degrees)
```

```
FILLSTYLE 7 ; blue crossed hatches (45 degrees)
FILLSTYLE 8 ; blue crossed hatches (0 degrees)
FILLSTYLE 9 ; blue triple crossed
FILLSTYLE 10 ; black wide single hatches (45 degrees)

 ; custom hatches specified by a list with the following arguments:
 ; – style (1 = single, 2 = double, 3 = triple hatching)
 ; – color
 ; – distance
 ; – degree

FILLSTYLE [2, “green”, 3pt, 15°] ; green crossed hatches (15 degrees)
```

Drawing objects

CIRCLE

```
CIRCLE 100 ; draw a circle shape (diameter = 100pt)
```

ELLIPSE

```
ELLIPSE [50, 100] ; draw an ellipse with 50 and 100 diameters
ELLIPSE [50, 100, 2h, 12h] ; draw an elliptical sector (from 2h clock position to 12h)
ELLIPSE [50, 100, 2h, 12h, 2]  ; draw an elliptical segment
ELLIPSE [50, 100, 2h, 12h, 3]  ; draw an elliptical arc
```

SQUARE

```
SQUARE 100 ; draw a square shape (size = 100pt)
```

RECTANGLE

```
RECTANGLE [50, 100] ; draw a rectangle shape (50×100pt)
RECTANGLE [50, 100, 10] ; draw a rectangle with rounded corners
```

POINT

```
POINT ; draw a point with size and color of the pen
```

CLOSE can join the last points, FILL can fill the shape defined by points. For example, it’s easy to draw a “flat” star starting from its center:

```
PENUP
REPEAT 5 [
FORWARD 80
POINT
BACK 80
RIGHT 36
FORWARD 50
POINT
BACK 50
RIGHT 120
] FILL
```

LABEL

```
LABEL “text” ; print text in the turtle position
LABEL 'text' ; see above
LABEL "text ; see above (only for single words)
```

TEXT

```
CIRCLE 10 TEXT “text” ; set text of the actual drawing object
```

Font settings

FONTCOLOR/FONTCOLOUR

```
FONTCOLOR “green” ; set font color
```

FONTFAMILY

```
FONTFAMILY “Linux Libertine G” ; set font (family)
FONTFAMILY “Linux Libertine G:smcp=1” ; set also font feature (small caps)
FONTFAMILY “Linux Libertine G:smcp=1&onum=1” ; small caps + old figures
```

FONTSIZE

```
FONTSIZE 12 ; set 12pt
```

FONTWEIGHT

```
FONTWEIGHT “bold” ; set bold font
FONTWEIGHT “normal” ; set normal weight
```

FONTSTYLE

```
FONTSTYLE “italic” ; set italic variant
FONTSTYLE “normal” ; set normal variant
```

PICTURE (pic)

PICTURE is for

- shape grouping;
- starting new line shapes;
- saving SVG images and SVG/SMIL animations;
- keeping the consistency of positions and line shapes at the left border.

Shape grouping

```
 ; PICTURE [ LibreLogo_commands ]
PICTURE [ FORWARD 100 CIRCLE 100 ] ; tree-like grouped shape
```

See also “Group” in LibreOffice Writer Help.

```
TO tree location
PENUP POSITION location HEADING 0 PENDOWN
PICTURE [ FORWARD 100 CIRCLE 100 ] ; tree-like grouped shape
END

PICTURE [ tree [30, 50] tree [100, 50] ] ; grouped shapes in a grouped shape
```

Starting new line shapes

```
PICTURE ; start a new line shape
FORWARD 10 PICTURE FORWARD 10 ; two line shapes
```

Saving SVG images

```
PICTURE “example.svg” [ CIRCLE 5 ] ; save the picture as an SVG image file in the user
folder
PICTURE “Desktop/example.svg” [ FORWARD 100 CIRCLE 5 ] ; as above, with a relative path
PICTURE “/home/user/example.svg” [ CIRCLE 5 ] ; absolute path for Unix/Linux
PICTURE “C:\example.svg” [ CIRCLE 5 ] ; absolute path for Windows
```

Saving SVG/SMIL animations (drawings with SLEEP commands)

```
PICTURE “animation.svg” [ CIRCLE 5 SLEEP 1000 CIRCLE 99 ] ; save as an SVG/SMIL animation (see also SLEEP)
PICTURE “animation2.svg” [ CIRCLE 5 SLEEP 1000 CIRCLE 99 SLEEP 2000 ] ; as above, but using SLEEP after the last object will result looping: after 2 seconds the SVG animation restarts in SMIL-conformant browsers
```

Consistency at the left border

Use picture to keep the consistency of positions and line shapes at the left border of Writer:

```
PICTURE [ CIRCLE 20 POSITION [-100, 100] CIRCLE 20 ]
```

Loops

REPEAT

```
 ; REPEAT number [ commands ]
```

```
REPEAT 10 [ FORWARD 10 LEFT 45 CIRCLE 10 ] ; repeat 10 times
```

```
 ; number is optional
```

```
REPEAT [ POSITION ANY ] ; endless loop
```

REPCOUNT

Loop variable (also in the FOR and WHILE loops).

```
REPEAT 100 [ FORWARD REPCOUNT LEFT 90 ]
```

FOR IN

Loop for the list elements:

```
FOR i IN [1, 5, 7, 9, 11] [
FORWARD i
LEFT 90
]
```

Loop for the characters of a character sequence:

```
FOR i IN “text” [
```

```
LABEL i
FORWARD 10
]
```

WHILE

```
WHILE TRUE [ POSITION ANY ] ; endless loop
WHILE REPCOUNT <= 10 [ FORWARD 50 LEFT 36 ] ; as REPEAT 10 [ ... ]
```

BREAK

Stop the loop.

```
REPEAT [ ; endless loop
POSITION ANY
IF REPCOUNT = 100 [ BREAK ]  ; equivalent of the REPEAT 100 [ ... ]
]
```

CONTINUE

Jump into the next iteration of the loop.

```
REPEAT 100 [
POSITION ANY
IF REPCOUNT % 2 = 0 [ CONTINUE ]
CIRCLE 10 ; draw circles on every 2nd positions
]
```

Conditions

IF

```
 ; IF condition [ true block ]
 ; IF condition [ true block ] [ false block ]

IF a < 10 [ PRINT “Small” ]
IF a < 10 [ PRINT “Small” ] [ PRINT “Big” ]
```

AND, OR, NOT

Logical operators.

```
IF a < 10 AND NOT a = 5 [ PRINT “0, 1, 2, 3, 4, 6, 7, 8 or 9” ]
```

```
IF a < 10 AND a != 5 [ PRINT “0, 1, 2, 3, 4, 6, 7, 8 or 9” ] ; as above
```

Subroutines

TO, END

New word (or procedure).

```
TO triangle
REPEAT [ FORWARD 100 RIGHT 120 ] FILL
END

REPEAT 10 [ triangle PENUP POSITION ANY PENDOWN ]
```

OUTPUT

Return value of the function.

```
TO randomletter
OUTPUT RANDOM “qwertzuiopasdfghjklyxcvbnm”
END

PRINT randomletter + randomletter + randomletter ; print 3-letter random character sequence
```

STOP

Return from the procedure.

```
TO example number
IF number < 0 [ STOP ]
PRINT SQRT number ; print square root
]

example 100
example -1 ; without output and error
example 25
```

Default variables

ANY

Default random value of colors, etc.

```
PENCOLOR ANY ; random pen color
```

TRUE

Logical value.

```
WHILE TRUE [ POSITION ANY ] ; endless loop
PRINT TRUE ; print true
```

FALSE

Logical value.

```
WHILE NOT FALSE [ POSITION ANY ] ; endless loop
PRINT FALSE ; print false
```

PAGESIZE

```
PRINT PAGESIZE ; print list of the page sizes in points, eg. [595.30, 841.89]
```

PI/π

```
PRINT PI ; print 3.14159265359
```

Input/Output

PRINT

```
PRINT “text” ; print “text” in a dialog box
PRINT 5 + 10 ; print 15
```

INPUT

```
PRINT INPUT “Input value?” ; ask and print a string by a query dialog box
PRINT FLOAT (INPUT “First number?”) + FLOAT (INPUT “Second number?”) ; simple calculator
```

SLEEP

```
SLEEP 1000 ; wait for 1000 ms (1 sec)
```

GLOBAL

Set global variables used in procedures.

```
GLOBAL about
about = “LibreLogo”

TO example
PRINT about
GLOBAL about ; when we want to add a new value
about = “new value for the global variable”
END

example
PRINT about
```

Functions

RANDOM

```
PRINT RANDOM 100 ; random float number (0 <= x < 100)
PRINT RANDOM “text” ; random letter of the “text”
PRINT RANDOM [1, 2] ; random list element (1 or 2)
```

INT

```
PRINT INT 3.8 ; print 3 (integer part of 3.8)
PRINT INT RANDOM 100 ; random integer number (0 <= x < 100)
PRINT INT “7” ; convert the string parameter to integer
```

FLOAT

```
 ; convert the string parameter to float number
PRINT 2 * FLOAT “5.5” ; print 11.0
```

STR

```
 ; convert the number parameter to string
PRINT “Result: ” + STR 5 ; print “Result: 5”
PRINT 10 * STR 5 ; print 5555555555
```

SQRT

```
PRINT SQRT 100 ; print 10, square root of 100
```

SIN

```
PRINT SIN 90 * PI/180 ; print 1.0 (sinus of 90° in radians)
```

COS

```
PRINT COS 0 * PI/180 ; print 1.0 (cosinus of 0° in radians)
```

LOG10

```
PRINT LOG10 100 ; print 2.0 (common logarithm of 100)
```

ROUND

```
PRINT ROUND 3.8 ; print 4 (rounding 3.8)
PRINT ROUND RANDOM 100 ; random integer number (0 <= x <= 100)
```

ABS

```
PRINT ABS -10 ; print 10, absolute value of -10
```

COUNT

```
PRINT COUNT “text” ; print 4, character count of “text”
PRINT COUNT [1, 2, 3] ; print 3, size of the list
```

SET

```
 ; Convert list to Python set
PRINT SET [4, 5, 6, 6] ; print {4, 5, 6}
PRINT SET [4, 5, 6, 6] | SET [4, 1, 9] ; print {1, 4, 5, 6, 9}, union
PRINT SET [4, 5, 6, 6] & SET [4, 1, 9] ; print {4}, intersection
PRINT SET ([4, 5, 6, 6]) - SET [4, 1, 9] ; print {5, 6}, difference
PRINT SET [4, 5, 6, 6] ^ SET [4, 1, 9] ; print {1, 5, 6, 9}, symmetric difference
```

RANGE

```
 ; Python-like list generation
PRINT RANGE 10 ; print [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
PRINT RANGE 3 10 ; print [3, 4, 5, 6, 7, 8, 9]
PRINT RANGE 3 10 3 ; print [3, 6, 9]
```

```
FOR i IN RANGE 10 50 10 [ ; loop for [10, 20, 30, 40]
FORWARD i
LEFT 90
]
```

LIST

```
 ; remove the repeating elements of a list using set and list conversion
PRINT LIST (SET [1, 3, 5, 5, 2, 1]) ; print [1, 3, 5, 2]
```

TUPLE

Conversion to Python tuple (non-modifiable list)

```
PRINT TUPLE [4, 5]
```

SORTED

It returns with a sorted list.

```
PRINT SORTED [5, 1, 3, 4] ; print [1, 3, 4, 5]
```

SUB

Substitute character sequences using regex (regular expression) patterns.

```
PRINT SUB (“t”, “T”, “text”) ; print “Text”, replacing “t” with “T”
PRINT SUB (“(.)”, “\\1\\1”, “text”) ; print “tteexxtt”, doubling every characters
```

SEARCH

Search character sequences patterns using regex patterns.

```
IF SEARCH (“\w”, word) [ PRINT “Letter in the word.” ]
```

FINDALL

Find all character sequences in the input string matching the given regex pattern.

```
PRINT FINDALL(“\w+”, “Dogs, cats.”) ; print [“Dogs”, “cats”], the list of the words.
```

MIN

```
PRINT MIN [1, 2, 3] ; print 1, the lowest element of the list
```

MAX

```
PRINT MAX [1, 2, 3] ; print 3, the greatest element of the list
```

Color constants

```
PENCOLOR “SILVER” ; set by name
PENCOLOR [1] ; set by identifiers
PENCOLOR “~SILVER” ; random silver color
```

Identifier	Name
0	BLACK
1	SILVER
2	GRAY/GREY
3	WHITE
4	MAROON
5	RED
6	PURPLE
7	FUCHSIA/MAGENTA
8	GREEN
9	LIME
10	OLIVE
11	YELLOW
12	NAVY
13	BLUE
14	TEAL
15	AQUA
16	PINK
17	TOMATO
18	ORANGE
19	GOLD
20	VIOLET
21	SKYBLUE
22	CHOCOLATE
23	BROWN
24	INVISIBLE

Category: EN

EN
