

## **Welcome to MswLogo Help**

### **Menu Commands**

[File Menu](#)

[Bitmap Menu](#)

[Help Menu](#)

### **Glossary**

[Defined Terms](#)

### **Windows**

[Commander](#)

### **Procedures**

Doing Math ([Functions](#))

Commanding Turtle ([Graphics](#))

Controlling your program ([Control](#))

Managing numbers and words ([Lists](#))

Read and Writing Files ([Input/Output](#))

Finding problems ([Debug](#))

Writing and Editing programs ([Development](#))

Assigning labels to variables ([Properties](#))

Manipulation of Numbers and Words ([Variables](#))

Teaching Logo to do things ([Procedures](#))

Talking to DOS ([System](#))

Controlling sound devices ([MultiMedia](#))

Not working ([Unimplemented](#))

## **Doing Math (Functions)**

### **Working with Triangles (Trigonometry)**

Arctan

Cos

Sin

### **Comparing numbers (Logic)**

Both

Either

Equalp

Greaterp

Lessp

Not

### **Arithmetic**

Difference

Int

Maximum

Minimum

Pow

Product

Quotient

Random

Remainder

Rnd

Round

Sqrt

Sum

## **Commanding Turtle (Graphics)**

### **Moving The Turtle**

Back  
Forward  
Hideturtle  
Setxy  
Shownp  
Showturtle  
Xcor  
Ycor

### **Turning The Turtle**

Heading  
Left  
Right  
Setheading  
Towardsxy

### **Working with the Turtles' PEN**

Getpen  
Getpenred  
Getpengreen  
Getpenblue  
Pencolor  
Penred  
Pengreen  
Penblue  
Penwidth  
Pendown  
Penerase  
Penmode  
Penup  
Penreverse  
Textdraw  
Textsize  
Textweight  
Textfont

### **Filling shapes with colors**

Getfloodred  
Getfloodgreen  
Getfloodblue  
Floodcolor  
Floodred  
Floodgreen  
Floodblue  
Floodfill  
Block

### **Manipulating the bitmap**

Bitcut  
Bitfit  
Bitopen  
Bitpaste  
Bitsave

### **Adjusting the screen**

Clearscreen  
Clearpalette

Getscreenred  
Getscreengreen  
Getscreenblue  
Screencolor  
Screenred  
Screengreen  
Screenblue  
Scrunch  
Setscrunch  
Scrollx  
Scrolly  
Zoom  
Wipeclean

## Reading and Writing Files (Input/Output)

ascii  
cbreak  
char  
close  
fileprint  
filetype  
fileprint  
fileread  
filetype  
fileword  
fprint  
fput  
ftype  
keyp  
oflush  
openread  
openwrite  
print  
readchar  
readlist  
request

## Finding Problems (Debug)

continue

errpause

errquit

nostatus

pause

setipause

setqpause

status

trace

untrace

## Teaching Logo to do things (Procedures)

end  
mkmod  
rmmod  
output  
popmod  
pots  
pushmod  
show  
stop  
to  
toplevel

## Assigning labels to variables (Properties)

gprop

plist

pprop

pps

remprop



## Manipulation of Numbers and Words (Variables)

local

make

namep

numberp

thing

## Managing Numbers and Words (Lists)

butfirst

butlast

count

empty

first

item

last

list

lput

memberp

sentencep

sentence

word

## Controlling your Program (Control)

break  
go  
if  
iffalse  
iftrue  
is  
repcount  
repeat  
run  
test  
time  
wait  
wordp  
zerop

## Writing and Editing Programs (Development)

edit

erase

describe

help

type

## Controlling sound devices (MultiMedia)

mcicommand

## **Not Working (Unimplemented)**

turtle

## Talking to DOS (System)

goodbye  
unix

## **Glossary**

clipboard



## Commander

InputBox  
OutputBox  
ExecuteButton  
StatusButton  
TraceButton  
HaltButton  
ResetButton  
YieldButton  
PauseButton

**Input Box**

The input box is connected to the output/command-recall list box and to the execute button. The input box can be "filled" with anything from the list box. You can also edit the text in the input box. If what your typing doesn't fit it will scroll automatically. Once your command is in the box you need to execute it. You can do this by hitting ENTER or clicking on the execute button.

**Output/Command-Recall List Box**

The output/command-recall list box will record all output including what you type into the input box. You can select by clicking on the desired text, by typing the beginning of the desired string, or by using the arrow keys. If something went out of view use the scroll bar. You know how to do it.

**Execute Button**

The execute button executes what is in the input box and is also "PUSHED" when you hit ENTER key.

**Status Button**

This button pops up a status window telling you what LOGO is up to. Click it again to close the window.

**Trace Button**

The trace button turns on tracing for debugging your programs. Click again to disable tracing. You can turn tracing on or off even while Logo is running. See also command Trace and Untrace

**Halt Button**

The halt button immediately stops LOGO from processing any further. Logo is now waiting for a new command.

**Reset Button**

The reset button is like a ClearScreen command and resets LOGO.



**Yield Button**

The yield button asks LOGO to not let other programs use the computer while LOGO is working. The default is Yield. Note that the commander window itself is like another program. That is, if you hit the NoYield button while processing, the commander will lose control (That means HALT button won't work) until LOGO is idle again. Once Idle You can click the yield button again to enable yielding. The reason the yield button is here is that LOGO runs faster if it can keep the computer all to itself.

**Pause Button**

The pause stops LOGO so that you can examine variables. Once paused the pause button will show what depth you paused to. To continue you must issue a Continue command. You can also issue a Pause command within code to act as a "Break Point". You can think of pause and continue as sort of a Halt-n-Push and Pop-n-Continue of your state

## **File Menu**

The File menu includes commands that enable you to new to print and exit.

For more information, select the File menu command name.

ActiveArea

Print

PrintSetup

Exit

## **File Exit Command**

This is how you exit MswLogo.

## **File Active Area Command**

This allows you to select the work area to be printed or saved. The Active Area dialog box gives you 2 choices. Full image, which is the default, or Custom image. Full image prints or saves the entire bitmap. Custom image initializes the Extents (XLow, XHigh, YLow, YHigh) also to the full image. But the extents are now enabled and can be adjusted to your needs. The primary purpose of this option is performance and space. You no longer need to wait for the software to build a full image. It takes less time and less memory to print and disk space to save a partial image. As a side effect you can adjust where your image ends up on the page by selecting different extents.

## **File Print Setup Command**

This allows you to setup your printer before you print.

## **File Print Command**

This allows you to print your work on the printer.

See also [ActiveArea](#)

## **Bitmap Menu**

The Bitmap menu includes commands that enable you to load and save bitmap files.

For more information, select the Bitmap menu command name.

New

Load

Save

Save As



## **Bitmap New Command**

This will clear the work done on the screen and create a new environment to save things in.

## **Bitmap Load Command**

This allows you to read in work you already save in the past. The format of the file you save things in is known as a Microsoft Windows Bitmap (.BMP). You can interchange these files with other applications such as Paint. Note that these files can be BIG and can take a while to read or write.

## **Bitmap Save Command**

This allows you to save you work on the computer's disk so that the computer will not forget it. It also allows you to add more work to an existing piece of work. But REMEMBER if your image was generated with a LOGO program you really don't need to save it unless you want to use the image in another application such as Paint.

The format of the file you save things in is known as a Microsoft Windows Bitmap (.BMP). You can interchange these files with other applications such as Paint. Note that these files can be BIG and can take a while to read or write.

See also [ActiveArea](#)

## **Bitmap Save As Command**

This allows you to save you work on the computer's disk so that the computer will not forget it.

It also allows you to add more work to an existing piece of work. This will also ask what name you want to give your work. But REMEMBER if your image was generated with a LOGO program you really don't need to save it unless you want to use the image in another application such as Paint.

The format of the file you save things in is known as a Microsoft Windows Bitmap (.BMP). You can interchange these files with other applications such as Paint. Note that these files can be BIG and can take a while to read or write.

See also [ActiveArea](#)

## **Help Menu**

The Help menu allows you to learn more about LOGO.

For more information, select the Help menu command name.

[Index](#)

[MCI](#)

[Using Help](#)

[About](#)

## **Help Index Command**

This is probably how you got here.

## **Help MCI Command**

This puts you into the MCI help file. It explains the syntax of the arguments to the Mcicommand.

## **Help Using Help Command**

This will explain how to use Microsoft Windows Help.



## **Help About Command**

This gives some details about the LOGO program like it's version.

arctan

arctan -- Operation, one input. Abbreviation: atan

The input must be numeric. The output is the arctangent, in degrees, of the input.

back

back -- Command, one input. Abbreviation: bk

The input is a number, a distance to move, as in the forward command. The difference is that the turtle moves backward, i.e., in the direction exactly opposite to the way it's pointing.

both

both -- Operation (predicate), two inputs. Abbreviation: and

The two inputs must both be either true or false. The output is true if both inputs are true; otherwise the output is false.

break

break -- Command, no inputs.

This command is only meaningful within the range of an if command within the range of a repeat command. It terminates the repeat immediately. If used in other contexts, the results may be strange.

butfirst

butfirst -- Operation, one input. Abbreviation: bf

The input may be any non-empty Logo object. If the input is a list, the output is a list equal to the input list with the first member removed. (If the input list has only one member, the output is the empty list, a list of zero members.) If the input is a word, the output is a word equal to the input word with the first letter removed. (If the input is a single-letter word, the output is the empty word.) If the input is empty, an error results.

butlast

butlast -- Operation, one input. Abbreviation: bl

The input may be any non-empty Logo object. If the input is a list, the output is a list equal to the input list with the last member removed. (If the input list has only one member, the output is the empty list, a list of zero members.) If the input is a word, the output is a word equal to the input word with the last letter removed. (If the input is a single-letter word, the output is the empty word.) If the input is empty, an error results.

ascii

ascii -- Command, one input.

The input must be a string of one character. The output is the ASCII value of the character.

Example:

ascii "a

ascii "b

ascii "c



char

char -- Command, one input.

The input must be a positive integer (ASCII value). The output is the word containing that single character.

Example:

char 97

char 98

char 99

`cbreak`

`cbreak` -- Command, one input.

The input must be either the word `on` or the word `off`. If the input is `on`, your screen is placed in `cbreak` mode, which means that what you type is made available to your program every character, rather than every line. This must be done before the `readchar` procedure, below, will work. This facility is good for writing video game programs or text editors. While in `cbreak` mode, `echo` is turned off also.

close

close -- Command, one input.

The input must be a file descriptor. The file is closed. This must be done when you've finished reading or writing the file.

Sample program:

```
make "fd openwrite "outfile  
fileprint :fd "Hello.  
close :fd
```

This will create a file named outfile containing the word Hello.

nostatus

nostatus -- Command, no inputs.

This command has the same effect as hitting the nostatus button. That is, it kills the popup status window.

status

status -- Command, no inputs.

This command has the same effect as hitting the status button. That is it pops up the status window.

continue

continue -- Command, no inputs. Abbreviation: co

This command is meaningful only when typed during an interactive Pause. It continues the current procedure from where it was paused.

cos

cos -- Operation, one input.

The input must be numeric. The output is the cosine of the input, taken in degrees, NOT radians.

count

count -- Operation, one input.

The input may be any Logo object. If the input is a list, the output is a number indicating the number of members in the list. (Note: only top-level members are counted, not members of members.) The count of the list

[[This is] [a list]]

is 2, not 4.) If the input is a word, the output is the number of letters (or other characters) in the word. Remember that in Logo a number is just a particular kind of word, so the output from count can be manipulated like any other Logo word.



clearscreen

clearscreen -- Command, no inputs. Abbreviation: cs

This command applies only to the display turtle. It erases everything on the Video screen, and restores the turtle to its initial position and heading (center of the screen, facing toward the top edge).

clearpalette

clearpalette -- Command, no inputs. Abbreviation: cp

This command clears the color palette. The color palette is filled by using the Pencolor, Screencolor, Floodcolor commands. The palette is only supported when windows is in 256 color mode. Once you run out of colors windows will choose the closest match. For example if the command (repeat 256 [penred repcount]) is issued it would fill the palette with 256 shades of red. At this point the palette would now be full. If you now wanted 256 shades of green they would NOT be granted (and matched to red). In order for them to be granted you have to let go of the 256 shades of red. This is done by "clearing the palette".

If you want a wide range of colors then select a wide range into the palette. For example the following would give you 216 colors covering a wide range. Once the few colors, left in the palette, are used, windows will have something reasonable to match further requests to (unlike the example above in which only shades of reds could be matched to).

```
repeat 6 [penr repcount*40; repeat 6 [peng repcount*40; repeat 6 [penb repcount*40]]]
```

Note also that loading in .BMP files uses up colors in the palette. Which again can be cleared using clearpalette. Clearing the screen does NOT clear the palette.

Note that when running 256 color mode with a palette that several advantages and disadvantages occur:

Printing a 256 color image on a mono printer will be less pleasing than printing in 16 color mode. This is because no "dithering" (mixing dots) is used in 256 color mode.

Running in 256 color mode is slower and takes more memory and may not be possible on some smaller machines or machines with limited video capabilities.

Running in 256 color does give much more pleasant colors and even allows manipulation of 256 .BMP files.

describe

describe -- Command, one input.

The input must be a word (the name of a Logo primitive procedure). Logo will then jump into Logo online help and search for the keyword.

Try this:

describe "sum

help

help -- Command, no inputs.

This command is here for compatibility and consistency with older versions of logo. Basically you can enter the "help" command to directly enter help. See also [Describe](#).

difference

difference -- Operation, two inputs. Abbreviation: diff Infix: -

The output of this procedure is the difference of the two inputs.

edit

edit - Command, zero or one input. Abbreviation: ed

The input to this command must be a word, which is the name of a procedure, or a list of words, each of which is the name of a procedure. (Unlike the to command, but like all other Logo procedures, the edit command evaluates its input, so you must use a quotation mark (") before the procedure name, if only one is given, to indicate that it is the name itself which is the input to edit; otherwise Logo would actually run the procedure to calculate the input to edit.) The procedure you name may or may not already be defined. Logo responds to the edit command by running the Windows text editor, editing the definition of the procedure(s) named in its input. (If a procedure was not previously defined, Logo creates an initial definition for it which contains the title line and the end line, which you must have.) You then edit the definition(s) with the editor. When you write the file and leave edit, Logo will use the edited file as the definition(s) of the procedure(s). You must not put anything in the file except procedure definitions; in other words, every non empty line in the file must be between a "to" line and an "end" line.

Example:

```
*edit "myprog  
(do your edits)  
*edit (This will edit myprog again since it remembered)
```

If the edit command is given with no input, edit is given the same file as from the last time you used the edit command. This is a convenience for editing the same procedure(s) repeatedly.

If, while editing, you change your mind and want to leave edit without redefining anything, use the Close Window command say NO to saving changes instead of the normal YES. This special way of leaving edit tells Logo not to redefine your procedures.

either

either -- Operation (predicate), two inputs. Abbreviation: or

The two inputs must be either true or false. The output is true if at least one of the inputs is true; otherwise the output is false.

empty

empty -- Operation (predicate), one input.

The input can be any Logo object. The output is the word true if the input is of length zero (i.e., it is the empty word or the empty list). The output is the word false otherwise.



equalp

equalp -- Operation (predicate), two inputs. Infix: =

The two inputs to this procedure may be any Logo objects. If they are numbers, then the output is the word true if they are numerically equal, false if they are numerically unequal. If either input is not a number, then the output is the same as for the procedure is: it is true if the two inputs are identical, false if not. For example, the numbers 2 and 2.0 are numerically equal, but not identical.

erase

erase -- Command, one input. Abbreviation: er

As for the show command, the input is either a word, naming one procedure, or a list of words, naming more than one. The named procedures are erased, so that they are no longer defined.

pushmod

pushmod -- Command, one input. Abbreviation: push

This command pushes you down into a Logo Module. The parameter must be a word (the name of the module you want work in). You can use the Pots to list both procedures and modules. See also Popmod and Mkmod.

popmod

popmod -- Command, no inputs. Abbreviation: pop

This command pops you up out of a Logo Module. You cannot pop up any higher than the level you started logo at. See also [Pushmod](#) and Mkmod

mkmod

mkmod -- Command, one input. Abbreviation: mm

This command makes (creates) a Logo Module and then pushes (Pushmod) you into it. The parameter must be a word (the name of the module you want to make). Its purpose is to keep your procedures organized. You can use the Pots command to lists the procedures (and modules) in the current module. Once done with the module you can pop (Popmod) back out and push into another. You cannot pop out of the default module that you start up Logo in.

Try this:

```
*mkmod "myfoo
*to myprog1
>print "hello1
>end
*to myprog2
>print "hello2
>end
*pots
*pop
*pots
```

rmmod

rmmod -- Command, one input. Abbreviation: rm

This command removes (deletes) a Logo Module. The parameter must be a word (a name of a existing module). You cannot remove a module if you are pushed (Pushmod) into it. Nor can you remove it, if it contains any procedures or other modules (Pots must not return anything while pushed into the module you want to remove). So to be sure you can remove it, do a pots and check if it's there, then push into it, do pots again and confirm it's empty, pop (Popmod) back out and then remove it. See also Mkmod.

errpause

errpause -- Command, no inputs.

This command tells Logo that any errors which happen during procedure execution from now on should cause a pause, rather than a return to toplevel.

errquit

errquit -- Command, no inputs.

This command tells Logo that any errors which happen during procedure execution from now on should return to toplevel, rather than pausing. This is the initial state of affairs when you start Logo.



fileprint

fileprint -- Command, two inputs. Abbreviation: ffp

The first input must be a file descriptor previously output by openwrite. The second input is any object. The second input is printed (typed, fprinted, ftyped) into the file.

filetype

filetype -- Command, two inputs. Abbreviation: fifty

The first input must be a file descriptor previously output by openwrite. The second input is any object. The second input is printed (typed, fprinted, ftyped) into the file.

fileprint

fileprint -- Command, two inputs. Abbreviation: fip

The first input must be a file descriptor previously output by openwrite. The second input is any object. The second input is printed (typed, fprinted, ftyped) into the file.

fileread

fileread -- Operation, one input. Abbreviation: fird

The input must be a file descriptor, previously output by openread. The procedure reads one line from the file. The output is the line, in the form of a list. (That is, the output is the file line as if enclosed in square brackets in a program.) If the end of the file has been reached, the output is the empty word. If the file line contains mismatched brackets, trouble may result.

filetype

filetype -- Command, two inputs. Abbreviation: fity

The first input must be a file descriptor previously output by openwrite. The second input is any object. The second input is printed (typed, fprinted, ftyped) into the file.

fileword

fileword -- Operation, one input. Abbreviation: fiwd

The input must be a file descriptor, open for reading. The procedure reads one line from the file. The output is that line, in the form of a single word, including spaces and punctuation characters. If the end of the file has been reached, the output is the empty list.

first

first -- Operation, one input. Abbreviation: f

The input may be any non-empty Logo object. If the input is a list, the output is its first member. If the input is a word, the output is a single-letter word, namely the first letter of the input. If the input is empty (a word or list of length zero) an error results.

forward

forward -- Command, one input. Abbreviation: fd

The input is a number, the distance you would like the turtle to move. For a floor turtle, the unit of distance is however far the turtle can travel in 1/30 second. For a display turtle, the unit is one dot on the Video screen. (Note: on some displays, one dot horizontally may not be the same length as one dot vertically. The setscrunch command allows you to control the relative sizes so that squares come out square.) The turtle moves in whatever direction it is pointing when you use the command.



fprint

fprint -- Command, one input. Abbreviation: fp

The input is printed as by the print command, except that if it is a list (as opposed to a word) it is enclosed in square brackets. The name of the command is short for "full print".

fput

fput -- Operation, two inputs.

The first input may be any Logo object; the second must be a list. The output is a list which is identical with the second input except that it has an extra first member, namely, the first input.

ftype

ftype -- Command, one input. Abbreviation: fty

The input is printed as by the type command, except that if it is a list, it is enclosed in square brackets.

getpen

getpen -- Operation, no inputs.

The output is the turtle's current pen color (24 bit RGB).

getpenred

getpenred -- Operation, no inputs.

The output is the turtle's current red component of the pen color.

getpengreen

getpengreen -- Operation, no inputs.

The output is the turtle's current green component of the pen color.

getpenblue

getpenblue -- Operation, no inputs.

The output is the turtle's current green component of the pen color.

`getfloodred`

`getfloodred` -- Operation, no inputs.

The output is the turtle's current red component of the flood color.



`getfloodgreen`

`getfloodgreen` -- Operation, no inputs.

The output is the turtle's current green component of the flood color.

getfloodblue

getfloodblue -- Operation, no inputs.

The output is the turtle's current green component of the flood color.

getscreenred

getscreenred -- Operation, no inputs.

The output is the red component of the screen color.

getscreengreen

getscreengreen -- Operation, no inputs.

The output is the current green component of the screen color.

getscreenblue

getscreenblue -- Operation, no inputs.

The output is the current green component of the screen color.

`go`

`go` -- Command, one input.

This command can be used only inside a procedure. The input must be a number. The same number must appear at the beginning of some line in the same procedure. (This line number is otherwise ignored.) The next command executed will be the one on the indicated line in the definition. Note: there is always a better way to do it without using `go`.

goodbye

goodbye -- Command, no inputs. Abbreviation: bye

This command is used to leave Logo. It is the one of many ways out, including a crash.

`gprop`

`gprop` -- Operation, two inputs.

Both inputs must be words. The first is a name, and the second is a property name. The output is the value of the indicated property of the indicated object. It is not an error if there is no such property; the output in that case is the empty list.



greaterp

greaterp -- Operation (predicate), two inputs. Infix: >

The output of this procedure is the word true if the first input is numerically strictly greater than the second input. Otherwise the output is the word false.

heading

heading -- Operation, no inputs.

The output is the turtle's current heading in degrees. This operation works only with the display turtle.

hideturtle

hideturtle -- Command, no inputs. Abbreviation: ht

This command applies only to the display turtle. It erases the display of the turtle itself from the screen, so that only the lines drawn when the turtle moves are visible. The display is faster when the turtle is hidden. Also, once a graphics program is debugged, it may be prettier to watch without the turtle visible.

if

if -- Command or operation, two or three inputs.

The first input to the if procedure must be either the word true or the word false. Typically, it is the output from a predicate. The second and (optional) third inputs are lists containing instruction lines. The second input is executed if the first input is true. The third input, if any, is executed if the first input is false:

```
to greet :person
  if equalp :person [Ronald Reagan] [print [Hi, turkey!]] "
    [print sentence "Hi, :person]
end
```

In that example, the first input to if is the output from the expression equalp :person [Ronald Reagan].

The if procedure can be used as an operation, producing a value. In this case, the third input is required:

```
print if equalp :person "Reagan ["Loser] ["Winner]
```

iffalse

iffalse -- Command, one input. Abbreviation: iff

The input must be an instruction list. It is run if the most recent test command saved a false value.

iftrue

iftrue -- Command, one input. Abbreviation: ift

The input must be an instruction list. It is run if the most recent test command saved a true value.

is

is -- Operation (predicate), two inputs.

The inputs can be any Logo objects. The output is the word true if the two inputs are identical. That is, they must be of the same type (both words or both lists), they must have the same count, and their members (if lists) or their characters (if words) must be identical. The output is the word false otherwise. (Note: this is an exception to the convention that names of predicates end with the letter "p".)

item

item -- Operation, two inputs. Abbreviation: nth

The first input must be a positive integer less than or equal to the count of the second input. If the second input is a word, the output is a word of length one containing the selected character from the word. (Items are numbered from 1, not 0.) If the second input is a list, the output is the selected member of the list.



keyp

keyp -- Operation (predicate), no inputs.

This procedure outputs true if there is a character waiting to be read from the screen, if you are in cbreak mode. If not, it outputs true if there is an entire line waiting to be read.

last

last -- Operation, one input. Abbreviation: l

The input may be any non-empty Logo object. If the input is a list, the output is its last member. If the input is a word, the output is a single-letter word, namely the last letter of the input. If the input is empty (a word or list of length zero) an error results.

left

left -- Command, one input. Abbreviation: lt

The input is a number, the number of degrees of angle through which the turtle should turn counterclockwise. This command does not change the position of the turtle, but merely its heading (the direction in which it points). The turn will be only approximately correct for the floor turtle, because of mechanical errors. For the display turtle, the angle will be perfectly reproducible, although it may not look quite right on the screen because of the difference in size between horizontal and vertical dots. Nevertheless, a display turtle program will work in the sense that when the turtle is supposed to return to its starting point, it will do so.

lessp

lessp -- Operation (predicate), two inputs. Infix: <

The output of this procedure is the word true if the first input is numerically strictly less than the second input. Otherwise the output is the word false.

list

list -- Operation, two inputs.

The output is a list of two elements, namely, the two inputs. The inputs may be words or lists.

local

local -- Command, one input.

The input must be a word. A variable with that word as its name is created, local to the currently running procedure (that is, local to the procedure in which the local command is used).

lput

lput -- Operation, two inputs.

The first input may be any Logo object; the second must be a list. The output is a list which is identical with the second input except that it has an extra last member, namely, the first input.

make

make -- Command, two inputs.

The first input is the name of a variable (that is, it must be a word); the second is any Logo object. The effect of the command is to assign the second input as the value of the variable named by the first input.



int

int -- Operation, one input.

The input must be a number. The output is the integer component of the number (everything to the right of the decimal point is removed).

maximum

maximum -- Operation, two inputs. Abbreviation: max

The output of this procedure is equal to whichever of the two inputs is numerically greater.

memberp

memberp -- Operation (predicate), two inputs.

If the second input is a word, the first input must be a word of length one (a single character), and the output is true if and only if the first input is contained in the second as a character. If the second input is a list, the first input can be any Logo object, and the output is true if and only if the first input is a member of the second input. (Note that this is member, not subset.)

minimum

minimum -- Operation, two inputs. Abbreviation: min

The output of this procedure is equal to whichever of the two inputs is numerically smaller.

namep

namep -- Operation (predicate), one input.

The input must be a word. The output is true if that word is the name of a variable which has a value assigned to it, false otherwise.

not

not -- Operation (predicate), one input.

The input must be either true or false. The output is true if the input is false, and vice versa.

numberp

numberp -- Operation (predicate), one input.

The input may be any Logo object. The output is the word true if the input is a number, false if not.

oflush

oflush -- Command, no inputs.

Normally, when you tell Logo to print something, the printing is not always done right away. Instead, Logo remembers everything you tell it to print, and the printing is done all at once the next time Logo is waiting for you to type something. This arrangement makes Logo much faster than it would be if everything were printed immediately. The oflush command tells Logo to print whatever you've previously asked for right away, without waiting.



openread

openread -- Operation, one input. Abbreviation: openr

The input to this procedure is a word, which must be a filename. It can contain slashes to indicate directory names. If the file can be opened for reading, the output from the procedure is a file descriptor, which should be stored in a variable for use in reading the file. If the file cannot be opened, an error results.

openwrite

openwrite -- Operation, one input. Abbreviation: openw

The input must be a filename. The file is opened for writing (replacing any previous version), if allowed, and the output is a file descriptor, for use by file printing commands. If the file cannot be opened, an error results.

output

output -- Command, one input. Abbreviation: op

This command is used in a user procedure which is meant to be an operation. The input to this command becomes the output from the user procedure. Please don't be confused by the fact that the user procedure is an operation, while the output primitive procedure is a command used in that procedure. Example:

```
to nickname :person
  if equalp :person [Peter Parker] [output "Spiderman]
  if equalp :person [Lamont Cranston] [output "Shadow]
  output first :person
end
```

pause

pause -- Command, no inputs.

This command is meaningful only within a procedure. It causes a pause. Once paused the pause button will show what depth you paused to. To continue you must issue a Continue command. You can also issue a PauseButton while a procedure is executing. You can think of pause and continue as sort of a Halt-n-Push and Pop-n-Continue of your state

Try this:

```
to mybreaktest
>print "Hello1
>pause
>print "Hello2
>end
```

```
mybreaktest
mybreaktest
mybreaktest
continue
continue
continue
```

pencolor

pencolor -- Command, three inputs. Abbreviation: penc

This command sets the color of the drawing pen. Each input represents how much Red Green and Blue you want in the color. Each input has a range of 0-255. By mixing different amounts of colors you can create 16.7 million different colors !!! . But there's a catch, if you want all those colors the penwidth must be greater than 1. If the penwidth is 1 the colors will change in steps. That is 0-100 will be no red, 101-200 might be medium red and 201-255 will be bright red.

Try this and watch what happens:

```
repeat 256 [ht; pd; penw 2; penc recount-1 0 0; fd 100; bk 100; rt 1]
```

See if you can get green and blue to do the same thing.

Note the common colors:

penc 000 000 000	black
penc 255 255 255	white
penc 128 128 128	gray
penc 255 000 000	Red
penc 000 255 000	Green
penc 000 000 255	Blue

Also see [Clearpalette](#)

penred

penred -- Command, one input. Abbreviation: penr

This command sets the red color component of the drawing pen. See [Pencolor](#) for details.

pengreen

pengreen -- Command, one input. Abbreviation: peng

This command sets the green color component of the drawing pen. See [Pencolor](#) for details.

penblue

penblue -- Command, one input. Abbreviation: penb

This command sets the blue color component of the drawing pen. See [Pencolor](#) for details.



penwidth

penwidth -- Command, one input. Abbreviation: penw

This command sets the width of the drawing pen. Note that when the pen width is 1 (default) that it draws in "solid" colors. Logo is limited to only 20 Solid colors (currently). So it will find the closest color it can. One the width is greater than 1 Logo with mix colors to simulate the desired color.

floodcolor

floodcolor -- Command, three inputs. Abbreviation: fc

This command sets the color of the flood pen. Each input represents how much Red Green and Blue you want in the color. Each input has a range of 0-255. By mixing different amounts of colors you can create 16.7 million different colors !!! But there's a catch, if you want all those colors the penwidth must be greater than 1. If the penwidth is 1 the colors will change in steps. That is 0-100 will be no red, 101-200 might be medium red and 201-255 will be bright red.

Note the common colors:

fc 000 000 000	black
fc 255 255 255	white
fc 128 128 128	gray
fc 255 000 000	Red
fc 000 255 000	Green
fc 000 000 255	Blue

Also see [Clearpalette](#)

floodred

floodred -- Command, one input. Abbreviation: fr

This command sets the red color component of the flood pen. See [Floodcolor](#) for details.

floodgreen

floodgreen -- Command, one input. Abbreviation: fg

This command sets the green color component of the flood pen. See [Floodcolor](#) for details.

floodblue

floodblue -- Command, one input. Abbreviation: fb

This command sets the blue color component of the flood pen. See [Floodcolor](#) for details.

pendown

pendown -- Command, no inputs. Abbreviation: pd

This command tells the turtle to lower its pen, so that later commands will draw lines when the turtle moves.

penerase

penerase -- Command, no inputs. Abbreviation: pe

This command tells the turtle to "lower its eraser", so that lines previously drawn will be erased when retraced by the turtle. It only works with the display turtle. The commands penup, pendown, penerase, and penreverse are mutually exclusive; whichever was most recently used is the one which affects the turtle.

floodfill

floodfill -- Command, no inputs. Abbreviation: ff

This command tells the turtle to flood an area with the "floodcolor". The Turtle must be inside a shape for it to work. The floodfill command will flood an area bounded by "color" of the currently selected pencolor.

Try this:

```
(draw a square)
cs
pd
penc 0 0 0
repeat 4 [fd 100; rt 90]
(now move inside the square, but don't draw into it)
pu
rt 45
fd 50
(choose a floodcolor red)
fc 255 0 0
(now flood it)
ff
(now flood it with black)
fc 0 0 0
ff
(now flood it with green)
fc 0 0 255
ff
```

Why didn't the last flood, flood with green. Because it only floods up to the color of the pen (default of black). Since it's all black you can't get past it. Note also that Windows mixes colors and sometimes it may BLOCK the flooding when you think it should not. Also note that currently some printers cannot handling this flooding.



block

block -- Command, two inputs.

This command will "fill" in a solid block of color specified by Floodcolor. LOGO will fill starting at the turtles' position with a width of the first argument and a height of the second argument..Don't confuse this with floodfill which must have a boundary and will not cross that boundary.

Try this:

```
floodcolor 100 100 100  
block 100 200  
floodfill
```

bitcut

bitcut -- Command, two inputs.

This command will "cut" out part of the image and into the LOGO's memory. Later at anytime you can "paste" (Bitpaste) it back into the image. LOGO can only remember one thing to cut (it will forget what you cut when you cut again). LOGO will cut starting at the turtles' position with a width of the first argument and a height of the second argument..

Try this:

```
cs
floodcolor 100 100 100
floodfill
bitcut 100 100
cs
bitpaste
```

bitfit

bitfit -- Command, two inputs.

This command will "fit" the currently "cut" (Bitcut) image into the specified dimensions. Later at anytime you can "paste" (Bitpaste) it back into the image. LOGO will fit the "cut" image to a width of the first argument and a height of the second argument. The original "cut" image is replaced by this newly "fit" image. You can permanently "scale" your image with bitfit. Where as zoom only views it temporarily at a different scale.

bitload

bitload -- Command, one input.

This command is the same as the Bitmap menu Load command. Its one input must be a word which describes the bitmap file to load.

Try this:

```
bitload "myfile.bmp
```

bitsave

bitsave -- Command, one input.

This command is the same as the Bitmap menu Save As command. Its one input must be a word which describes the bitmap file to save.

Try this:

```
bitsave "myfile.bmp
```

bitpaste

bitpaste -- Command, no inputs.

This command will "paste" back into the image what was "cut" ([Bitcut](#)). LOGO will always "paste" at the location of the turtle.

Note: You can write higher order procedures such as bitcopy by building logo procedures using the bit-primitive commands in logo.

penmode

penmode -- Operation, no inputs.

This operation outputs one of the words penup, pendown, penerase, or penreverse, depending on the current state of the turtle's pen.

penup

penup -- Command, no inputs. Abbreviation: pu

This command tells the turtle to raise its pen from the paper, so that it does not leave a trace when it moves. In the case of the display turtle, there is no physical pen to move mechanically, but the effect is the same: any forward or back commands after this point do not draw a line. The turtle starts with its pen down.



plist

plist -- Operation, one input.

The input must be a word, which is a name. The output is the property list of the specified name. Note: the output is actually a copy of the property list. The real property list is not a Logo list. Any later changes to the properties of the specified name will not change the list which was output by an earlier plist.

pots

pots -- Command, no inputs.

This command types at your screen the title lines of all procedures (and Modules) you've defined in the current Module. The name is an abbreviation for "print out titles".

pow

pow -- Operation, two inputs.

The inputs must be numbers. If the first is negative, the second must be an integer. The output is the first number raised to the power of the second input.

pprop

pprop -- Command, three inputs.

The first input, which must be a word, is a name with which a property list is associated. The second input, which must be a word, is the name of a property. The third input can be any Logo object. It becomes the value of the specified property of the specified name.

pps

pps -- Command, no inputs.

All properties of all names are listed on your video screen.

print

print -- Command, one input. Abbreviation: pr

The input, which may be a word or a list, is printed on the video screen, followed by a new line character. (That is, the video screen is positioned at the beginning of a new line after printing the object.) If the object is a list, any sub-lists are delimited by square brackets, but the entire object is not delimited by brackets.

textdraw

textdraw -- Command, one input. Abbreviation: td

The input, which may be a word or a list, is printed on the screen. If the object is a list, any sub-lists are delimited by square brackets, but the entire object is not delimited by brackets. You can print any logo object (numbers, lists and strings). Note that the handle of the string (the origin) is the top-left corner of the string. Another thing to be aware of is that the capabilities of the text changes depending on the device (screen or printer), the size, the turtle heading (direction) and the font. In other words sometime the text can be drawn at the turtle heading and sometimes it cannot. Sometimes what is on the screen will not be exactly what you print.

The size of the text is determined by the command textsize.

The weight (how bold) of the text is determined by the command textweight

The color of the text is determined by pencolor.

The position of the text is determined by the location of the turtle.

The font of the text is determined by textfont

The angle of the text is determined by the heading (direction) of the turtle.

Try this:

```
ts 50
pu
repeat 36 [fd 250; textdraw heading; bk 250; rt 10]
pd
```

textfont

textfont -- Command, one input. Abbreviation: tf

The input, must be a word or list that describes a font. A font determines what your characters look like on the screen. The available fonts depend on you your computer. Some common ones (Default Windows 3.1) are:

```
textfont "System
textfont "Fixedsys
textfont "Terminal
textfont [Ms Sans Serif]
textfont "Courier
textfont [MS Serif]
textfont "Roman
textfont "Script
textfont "Modern
textfont [Small Fonts]
textfont "BorlandTE
textfont "BorlandTEi
textfont "Preview
textfont "Arial
textfont [Courier New]
textfont [Times New Roman]
textfont "Windings
textfont "Symbol
```

Note, if you mistype the font name textfont will list what fonts are available.



textsize

textsize -- Command, one input. Abbreviation: ts

The input, which must be a number, is used to determine the size of text drawn on the screen with textdraw.

textweight

textweight -- Command, one input. Abbreviation: ts

The input, which must be a number, is used to determine the weight (how bold) of text drawn on the screen with `textdraw`. A weight of 400 is considered normal text.

product

product -- Operation, two inputs. Infix: \*

The output of this procedure is the product of the two inputs (numbers).

penreverse

penreverse -- Command, no inputs. Abbreviation: px

This command tells the display turtle to lower its "reversing pen" thereafter, when the turtle moves, it turns on any points which were off, and turns off any points which were on. The commands penup, pendown, penerase, and penreverse are mutually exclusive; whichever was most recently used is the one which affects the turtle.

quotient

quotient -- Operation, two inputs. Infix: /

The output of this procedure is the quotient of the two inputs. If both inputs are integers, the output is also an integer; the remainder of the division is lost. If either input is not an integer, the quotient can include a fractional part. Therefore, these two are not the same:

quotient 2 3  
quotient 2.0 3

random

random -- Operation, no inputs.

The output from this procedure is an integer between 0 and 9, i.e., a single digit. It is chosen randomly, so the output may be different each time the procedure is used.

readchar

readchar -- Operation, no inputs. Abbreviation: rc

This operation waits for you to type a single character at your screen. The output is a word containing only that character. This works only if you have turned on cbreak mode.

readlist

readlist -- Operation, no inputs. Abbreviation: rl

Logo waits for a line to be typed by the user. The contents of the line are made into a list, as though typed within square brackets as part of a Logo instruction. (The user should not actually type brackets around the line, unless s/he desires a list of one element, which is a list itself.) That list is the output from the operation.



remainder

remainder -- Operation, two inputs. Abbreviation: mod Infix: "

The inputs to this procedure must be integers. The output is also an integer, and is the remainder of dividing the first input by the second.

remprop

remprop -- Command, two inputs.

The inputs must be words, as for gprop. The specified property is removed from the specified name.

repcount

repcount -- Operation, no inputs.

This operation may be used only within the range of a repeat command. It outputs the number of repetitions which have been done, including the current one. That is, it outputs 1 the first time through, 2 the second time, and so on.

repeat

repeat -- Command, two inputs.

The first input must be a positive number. The second is an instruction list, as for the run command. The list is run repeatedly, the number of times specified by the first input: See also [repcount](#).

```
repeat 5 [print "hello]
```

request

request -- Operation, no inputs.

A question mark is printed on the video screen as a prompt. Then Logo waits for a line to be typed by the user, as for readlist.

right

right -- Command, one input. Abbreviation: rt

The input is a number; the turtle turns through the specified number of degrees clockwise.

rnd

rnd -- Operation, one input.

The input must be a positive integer. The output is a randomly selected integer between 0 and one less than the input.

round

round -- Operation, one input.

The input must be a number. The output is the closest integer.



run

run -- Command or operation, one input.

The input must be a list, containing a Logo instruction line. The list is run as if you typed it directly to Logo. Example:

```
to while :condition :cmd
  if not run :condition [stop]
  run :cmd
while :condition :cmd
end
```

```
make "number 1
while [:number < 5] [print :number; make "number :number+1]
```

The run procedure can be used as an operation, if its input is a Logo expression which produces a value, instead of a complete instruction:

```
print run [sum 2 3]
```

scrunch

scrunch -- Operation, no inputs.

This operation is used only for display turtles. It outputs a number, which is the scrunch factor (or aspect ratio) by which vertical motion is multiplied before it is displayed. This number is changed using the setscrunch command.

sentence

sentence -- Operation, two inputs. Abbreviation: se

The two inputs may be words or lists. The output is a list formed from the two inputs in this way: if either input is a word, that word becomes a member of the output list; if either input is a list, the members of that input become members of the output. Here are some examples:

first input	second input	output
"hello	"test	[hello test]
"goodbye	[cruel world]	[goodbye cruel world]
[a b]	[c d]	[a b c d]
[]	"garply	[garply]

If an input is the empty list, as in the last example above, it contributes nothing to the output.

sentencep

sentencep -- Operation (predicate), one input.

The input can be any Logo object. The output is the word true if the input is a sentence, i.e., a list of words. The output is the word false if the input is a word, or if any member of the input is a list.

screencolor

screencolor -- Command, three inputs. Abbreviation: sc

This command sets the color of the background. Each input represents how much Red Green and Blue you want in the color. Each input has a range of 0-255. By mixing different amounts of colors you can create 16.7 million different colors !!!.

Note the common colors:

sc 000 000 000	black
sc 255 255 255	white
sc 128 128 128	gray
sc 255 000 000	Red
sc 000 255 000	Green
sc 000 000 255	Blue

Also see [Clearpalette](#)

screenred

screenred -- Command, one input. Abbreviation: sr

This command sets the red color component of the background. See [Screencolor](#) for details.

screengreen

screengreen -- Command, one input. Abbreviation: sg

This command sets the green color component of the background. See [Screencolor](#) for details.

screenblue

screenblue -- Command, one input. Abbreviation: sb

This command sets the blue color component of the background. See [Screencolor](#) for details.



setheading

setheading -- Command, one input. Abbreviation: seth

The input must be a number. The turtle's heading is set to the input, taken in degrees. Zero points straight up, as the turtle starts out; positive headings are clockwise from zero. This command applies only to the display turtle.

setipause

setipause -- Command, no inputs.

This command tells Logo that from now on, the system interrupt character should pause, and the system quit character should return to toplevel. This is the initial state of affairs when you start Logo.

setqpause

setqpause -- Command, no inputs.

This command tells Logo that from now on, the system quit character should pause, and the system interrupt character should return to toplevel. This is the reverse of the usual interpretation. This command is provided for people whose systems or keyboards make one of these characters easier to type than the other. In particular, under Eunice there is only an interrupt character, not a quit character.

setscrunch

setscrunch -- Command, one input. Abbreviation: setscrunch

This command is used only for display turtles. The input must be a number. The vertical component of turtle motion is multiplied by this number before each motion is taken. If squares come out too wide on your screen, you should increase the number; if too tall, you should decrease it. (You can also use setscrunch to deform the turtle's motion on purpose, so for example a circle program will draw an ellipse instead.) The initial scrunch value depends on the Video screen you are using.

scrollx

scrollx -- Command, one input.

This command allows LOGO to control the scrollers of the "Screen" window. The argument is the amount to change the scroller by (delta). A positive number scrolls to the right and negative number scrolls to the left.

Try this:

```
repeat 10 [scrollx 10]
```

scrolly

scrolly -- Command, one input.

This command allows LOGO to control the scrollers of the "Screen" window. The argument is the amount to change the scroller by (delta). A positive number scrolls down and negative number scrolls to the up.

Try this:

```
repeat 10 [scrollx 10]
```

zoom

zoom -- Command, one input.

This command allows LOGO to control the scale of the "Screen" window. The argument is the amount to zoom (scale) by. A number greater than 1.0 makes things bigger (e.g. 2.0 makes it 2 times bigger), a number smaller than 1.0 makes things smaller (0.5 makes 1/2 as big). If an existing image is on the screen when you zoom it will be stretched or squeezed (this takes time by the way) according to the zoom (it may look a little jagged). If you "draw" while zoomed things will not be as jagged. Even though things may appear jagged LOGO remembers everything as if zoom was normal (1.0) and only prints in normal. Once you return to zoom of 1.0 your image will not be stretched or squeezed to fit again zoom of 1.0. In other words in a zoom of 1.0 lines never get jagged even if you drew it at zoom 0.5 or 2.0.

NOTE: Zoom works best if you choose a zoom that is a "power" of two. For example 2, 4, 8, 1/2 (0.5), 1/4 (0.25), 1/8 (0.125) etc.

setxy

setxy -- Command, two inputs.

The two inputs must be numbers. The turtle is moved to the point on the screen whose x (horizontal) coordinate is the first input, and whose y (vertical) coordinate is the second input. The center of the screen, where the turtle starts, has both coordinates zero. If the pen is down, this command draws a line. This command applies only to the display turtle.



show

show -- Command, one input. Abbreviation: po

The input to this command is a word or a list of words. Each word must be the name of a procedure. The command prints out the definition(s) of the procedure(s) on your screen. (The abbreviation po stands for printout, which is the name used for this command in some other versions of Logo.)

shownp

shownp -- Operation (predicate), no inputs.

This predicate applies only to the display turtle. It outputs the word true if the turtle is visible on the Window, false otherwise.

sin

sin -- Operation, one input.

The input must be numeric. The output is the sine of the input, taken in degrees, not radians.

sqrt

sqrt -- Operation, one input.

The input must be a nonnegative number. The output is its square root.

showturtle

showturtle -- Command, no inputs. Abbreviation: st

This command applies only to the `display turtle`. It restores the display of the turtle, after the `hideturtle` command has been used.

stop

stop -- Command, no inputs.

This command is used in user procedures which are meant to be commands. It stops the user procedure. (Note that it does not stop all running procedures. If user procedure A runs user procedure B, a stop command in procedure B returns to procedure A, which continues after the point where procedure B was invoked.)

sum

sum -- Operation, two inputs. Infix: +

The output of this procedure is the sum of the two inputs.

test

test -- Command, one input.

The input must be the word true or the word false. The command remembers its input for use in a later iftrue or iffalse command. This is an alternative to if which is useful if several instructions are to be made conditional on the same condition. The remembered truth value is local to the current procedure, if any.



thing

thing -- Operation, one input.

The input must be a word, and must be the name of a variable. The output is the value of the variable. These are equivalent:

```
:foo  
thing "foo
```

to

to -- Command, special form, see below.

This command takes a variable number of inputs. The first is the name of a procedure to be defined. The rest, if any, must be preceded by colons, and are the names of variables to be used as inputs to the procedure. Logo responds to the to command by printing a "greater than" sign (>) prompt, to show you that you are defining a procedure rather than entering commands to be executed immediately. You type the instruction lines which make up the definition. When you are done with the definition, type the special word end on a line by itself. For example:

```
^*to twoprint :thing
-^>print :thing
-^>print :thing
-^>end
-^*
-
```

This example shows the definition of a procedure named twoprint, which has one input, named thing. The procedure you are defining with the to command must not already be defined.

end

end -- Command, no inputs.

The end command is only valid within a procedure. It depicts the "end" of your procedure and **MUST** exist in every procedure. See the To command for an example.

toplevel

toplevel -- Command, no inputs. Abbreviation: top

This command stops all running procedures. The user at the screen is prompted to type another command. This can be used when a user procedure discovers some error condition and wants to abort the entire program, for example.

towardsxy

towardsxy -- Operation, two inputs.

This operation applies only to the display turtle. The two inputs must be numbers, which are the x and y coordinates of a point on the Video screen. The output is a number which is the heading to which the turtle must be set, in order to point towards that point from its current position. Note: this operation does not actually move or turn the turtle. You can use it as the input to setheading if that is what you want.

trace

trace -- Command, no inputs.

This command is used for debugging your Logo programs. After you use this command, every time a user-defined procedure starts or stops, a message is typed at your screen naming the procedure and its inputs or its output, if any. The message is indented according to the depth in procedure calls. See also the Button trace.

turtle

turtle -- Command, one input. Abbreviation: tur

The input is the name of a turtle. You can only control one turtle at a time, so using this command a second time releases the turtle previously selected. The names of floor turtles are numbers like 0 and 1. If you are using a graphics display screen (not just a text screen), you can control the display turtle by using the word `display` (or the abbreviation `dpy`) as the turtle name. (As usual, the word must be preceded by a quotation mark.) If you use a graphics primitive without selecting a turtle, Logo assumes that you want to use the display turtle. But once you select a floor turtle, you must say `turtle "display` explicitly to switch to the display. The word `off` as input to the turtle command releases a floor turtle, if you have one, or turns off the graphics display if you have been using the display turtle. This also happens when you leave Logo.

type

type -- Command, one input.

The input, which may be a word or a list, is printed on the screen, without a new line character. (That is, the screen remains positioned at the end of the object after printing it.) Brackets are used as with the print command.



unix

unix -- Command, one input.

This command is not currently implemented.

untrace

untrace -- Command, no inputs.

This command turns off the trace messages started by the `trace` command.

wait

wait -- Command, one input.

The input must be a positive integer. Logo waits that many seconds before continuing.

time

time -- Command, no inputs.

This command outputs the time on the system as a list.

wipeclean

wipeclean -- Command, no inputs. Abbreviation: clean

This command applies only to the display turtle. It erases everything on the Video screen, but does not change the turtle's position or heading.

word

word -- Operation, two inputs.

The two inputs must be words. The output is a word which is the concatenation of the two inputs. There is no space or other separation of the two inputs in the output.

wordp

wordp -- Operation (predicate), one input.

The input can be any Logo object. The output is the word true if the input is a word. The output is the word false if the input is a list.

`xcor`

`xcor` -- Operation, no inputs.

The output is the turtle's current x (horizontal) coordinate. The operation works only with the display turtle.



ycor

ycor -- Operation, no inputs.

The output is the turtle's current y (vertical) coordinate. This operation works only with the display turtle.

mcicommand

mcicommand -- Operation, one input. Abbreviation: mci

The input must be a list. It may or may not output a list depending on the context. The MCI interface is very powerful. It opens the door to letting Logo control any Windows MultiMedia device. These include Sound cards (with or without MIDI interfaces), CD-ROM players and more.

The MCI command is designed to let YOU (the programmer) write procedures to manipulate MultiMedia devices. You can now link sounds to the steps of drawing a picture. You can narrate your own slide show. You can even ask your user questions in your own voice.

### **Current limitations:**

The MCI interface allows you to start a device and optionally "wait" for it to finish or "notify" you when it has finished the request. The "wait" works fine, but "notify" is NOT implemented yet.

The "real" MCI online help is supplied by Microsoft. I'm not allowed at this time to distribute it. But I'm asking. Mean while you may have to live with a substitute that I have written.

Try this:

```
to soundit
print mci [open c:/windows/tada.wav type waveaudio alias wa1]
print mci [open c:/windows/ding.wav type waveaudio alias wa2]
mci [seek wa1 to start]
mci [play wa1 wait]
repeat 2\
  [
    mci [seek wa2 to start];\
    mci [play wa2 wait];\
  ]
mci [close wa1]
mci [close wa2]
end
```

Note: That the "SPEAKER" sound card emulator does NOT work with MCI.

Note: MCI is only available to **Microsoft Windows 3.1** systems.

zerop

zerop -- Operation (predicate), one input.

The input must be a number. The output is the word true if the input is numerically equal to zero, false otherwise.

## **Windows Keys**

The keyboard topics below come from Help for Windows. You can create similar keyboard topics for your application's Help. Choose from the following list to review the keys used in Windows:

[Cursor Movement Keys](#)

[Dialog Box Keys](#)

[Editing Keys](#)

[Help Keys](#)

[Menu Keys](#)

[System Keys](#)

[Text Selection Keys](#)

[Window Keys](#)

## Cursor Movement Keys

<b>Key(s)</b>	<b>Function</b>
DIRECTION key	Moves the cursor left, right, up, or down in a field.
End or Ctrl+Right Arrow	Moves to the end of a field.
Home or CTRL+Left Arrow	Moves to the beginning of a field.
PAGE UP or PAGE DOWN	Moves up or down in a field, one screen at a time.


## Dialog Box Keys

<b>Key(s)</b>	<b>Function</b>
TAB	Moves from field to field (left to right and top to bottom).
SHIFT+TAB	Moves from field to field in reverse order.
ALT+letter	Moves to the option or group whose underlined letter matches the one you type.
DIRECTION key	Moves from option to option within a group of options.
ENTER	Executes a command button. Or, chooses the selected item in a list box and executes the command.
ESC	Closes a dialog box without completing the command. (Same as Cancel)
ALT+DOWN ARROW	Opens a drop-down list box.
ALT+UP or DOWN ARROW	Selects item in a drop-down list box.
SPACEBAR	Cancels a selection in a list box. Selects or clears a check box.
CTRL+SLASH	Selects all the items in a list box.
CTRL+BACKSLASH	Cancels all selections except the current selection.
SHIFT+ DIRECTION key	Extends selection in a text box.
SHIFT+ HOME	Extends selection to first character in a text box.
SHIFT+ END	Extends selection to last character in a text box

## Editing Keys

<b>Key(s)</b>	<b>Function</b>
Backspace	Deletes the character to the left of the cursor. Or, deletes selected text.
Delete	Deletes the character to the right of the cursor. Or, deletes selected text.

## Help Keys

<b>Key(s)</b>	<b>Function</b>
F1	<p>Gets Help and displays the Help Index for the application. If the Help window is already open, pressing F1 displays the "Using Windows Help" topics.</p> <p>In some Windows applications, pressing F1 displays a Help topic on the selected command, dialog box option, or system message.</p>
SHIFT+F1	<p>Changes the pointer to  so you can get Help on a specific command, screen region, or key. You can then choose a command, click the screen region, or press a key or key combination you want to know more about.</p> <p>(This feature is not available in all Windows applications.)</p>



## Menu Keys

<b>Key(s)</b>	<b>Function</b>
Alt	Selects the first menu on the menu bar.
Letter key	Chooses the menu, or menu item, whose underlined letter matches the one you type.
Alt+letter key	Pulls down the menu whose underlined letter matches the one you type.
LEFT or RIGHT ARROW	Moves among menus.
UP or DOWN ARROW	Moves among menu items.
Enter	Chooses the selected menu item.

## System Keys

The following keys can be used from any window, regardless of the application you are using.

<b>Key(s)</b>	<b>Function</b>
Ctrl+Esc	Switches to the Task List.
Alt+Esc	Switches to the next application window or minimized icon, including full-screen programs.
Alt+TAB	Switches to the next application window, restoring applications that are running as icons.
Alt+PrtSc	Copies the entire screen to Clipboard.
Ctrl+F4	Closes the active window.
F1	Gets Help and displays the Help Index for the application. (See <a href="#">Help Keys</a> )

## Text Selection Keys

<b>Key(s)</b>	<b>Function</b>
SHIFT+LEFT or RIGHT ARROW	Selects text one character at a time to the left or right.
SHIFT+DOWN or UP	Selects one line of text up or down.
SHIFT+END	Selects text to the end of the line.
SHIFT+HOME	Selects text to the beginning of the line.
SHIFT+PAGE DOWN	Selects text down one window. Or, cancels the selection if the next window is already selected.
SHIFT+PAGE UP	Selects text up one window. Or, cancels the selection if the previous window is already selected.
CTRL+SHIFT+LEFT or RIGHT ARROW	Selects text to the next or previous word.
CTRL+SHIFT+UP or DOWN ARROW	Selects text to the beginning (UP ARROW) or end (DOWN ARROW) of the paragraph.
CTRL+SHIFT+END	Selects text to the end of the document.
CTRL+SHIFT+HOME	Selects text to the beginning of the document.

## Window Keys

<b>Key(s)</b>	<b>Function</b>
ALT+SPACEBAR	Opens the Control menu for an application window.
ALT+Hyphen	Opens the Control menu for a document window.
Alt+F4	Closes a window.
Alt+Esc	Switches to the next application window or minimized icon, including full-screen programs.
Alt+TAB	Switches to the next application window, restoring applications that are running as icons.
Alt+ENTER	Switches a non-Windows application between running in a window and running full screen.
DIRECTION key	Moves a window when you have chosen Move from the Control menu. Or, changes the size of a window when you have chosen Size from the Control menu.



## **Clipboard**

This is a topic that describes the Windows term "clipboard". If you click the "clipboard" term within the Copying Text or Glossary topic, this Help topic will be displayed in a pop-up window.

This topic is also tagged with the keyword "clipboard," for use with the WinHelp Search option.

