Logo Command reference

+ *Plus*

The addition operator.

```
3 + 2
5
```

- *Minus*

The subtraction operator. You must be careful to specify a space after the minus sign as otherwise it is taken as part of the following number: 7 -5 does not mean take away five from seven, but is two numbers, a seven followed by a minus five.

```
3 - 2
1
```

* *Multiplication*

The multiplication operator.

```
3 * 2
6
```

/ *Division*

The divide operator.

```
3 / 2
1.5
```

< *Less than*

The less than operator.

```
1 < 2
true
```

> *Greater than*

The greater than operator.

```
1 > 2
false
```

Abs

Output the absolute value of a number

Abs *number*

Output the absolute value of *number*.

```
Abs -2
2
Abs 2
2
```

And

Logical AND

And predicate1 predicate2

(And predicate1 predicate2 ...)

Outputs true if all predicates are true.

```
And “true 3 > 2
true
(And 1 < 3 5 = 5 (count []) = 0)
true
(And 1 < 0 5 = 5 (count []) = 0)
false
```

See also Not, Or, <, >.

Arc

Draw an arc

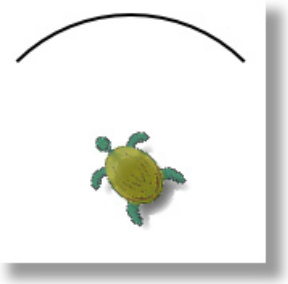
Arc *angle radius*

Draws an arc of radius *radius*, centred on the current turtle position and starting at the current heading, sweeping clockwise through angle *angle*. Draws a line if the pen is down.

The turtle heading and position do not change.

Example:

```
Left 45
Arc 90 100
```



See also PenUp, PenDown, Left, Right, Heading, SetHeading.

ArcCosine, ArcCos

ArcCosine

ArcCosine *number*

Output the angle whose cosine is *number*.

```
Cos 66
0.4067366430758
ArcCos 0.4067366430758
66
```

See also Cosine, Sine, Tangent, ArcSine, ArcTangent.

ArcSine, ArcSin

ArcSine

ArcSine *number*

Output the angle whose sine is *number*.

```
Sine 71
0.945518575599317
ArcSine 0.945518575599317
71
```

See also Cosine, Sine, Tangent, ArcCosine, ArcTangent.

ArcTangent, ArcTan

ArcTangent

ArcTangent *number*

Output the angle whose tangent is *number*.

```
Tan 60
1.73205080756888
ArcTan 1.73205080756888
60
```

See also Cosine, Sine, Tangent, ArcCosine, ArcSine.

ASCII

Output a character's ASCII code

ASCII *word*

Output the ASCII code of the first character of word .

```
Ascii "a
97
Ascii "ABC
65
```

Opposite of Char.

Back

Move the turtle backwards

Back *distance*

Move the turtle backwards *distance* pixels. Draws a line if the pen is down.

Example:

```
Back 100
```

Opposite of Forward. See also PenUp, PenDown

Background, Bg

Return the background pen colour

Background

Output the number of the background colour. This is the colour used to fill the screen when ClearScreen is called. At startup, the background colour is set to zero.

Example:

```
Background
0
```

See also SetBackground, Clean, ClearScreen, RGB, SetPenColor.

ButFirst

Output all but the first element

ButFirst *list*

ButFirst *word*

Output all but the first element of an object.

```
ButFirst “abcdefghijk
bcdefghijk
butfirst [thing1 thing2 thing3]
[thing2 thing3]
```

See also ButLast, Count, Empty?, First, FirstPut, Item, Last, LastPut, List, List?, Member?.

ButLast

Output all but the last element

ButLast *word*

ButLast *list*

Output all but the last element of an object.

```
ButLast “xyz
xy
ButLast [thing1 thing2 thing3]
[thing1 thing2 ]
```

See also ButFirst, Count, Empty?, First, FirstPut, Item, Last, LastPut, List, List?, Member?.

CD

Change the current directory

CD *path-name*

Changes the current directory to *path-name*, which can be either a word or a list. It’s generally better to use a list so that characters such as slashes in the name are not taken to be logo operators. A tilde (‘~’) can be used as shorthand for your home directory.

```
CD "~
Pwd
/Users/alan
```

path-name can either be absolute (starting with a slash) or relative, in which case the directory is changed relative to the current directory.

```
CD [/System]
Pwd
/System
```

```
CD [Library]
Pwd
/System/Library
```

To go up a level, use a double-dot '**..**'.

See also Pwd, Dir, OpenAppend, Openread, OpenWrite.

Char

Output the character for an ASCII code

char *number*

Output the character whose ASCII code is *number*.

```
Char 68
D
```

Opposite of ASCII.

Clean

Clear the graphics window

Clean

Clear the graphics screen without affecting the turtle. The graphics screen is filled with the current background pen colour.

See also ClearScreen, SetBackground, Background, RGB, SetPenColor.

ClearScreen, CS

Clear graphics and home the turtle

ClearScreen

Clear the graphics screen and set the turtle to its home position. The same as doing Clean followed by Home.

See also Clean, SetBackground, Background, RGB, Home, SetPenColor.

CloseReadFile

Close the file open for reading

CloseReadFile

Close the file that's been opened for reading with OpenRead .

See also CD, Dir, Pwd, OpenRead .

CloseWriteFile

Close the file open for writing

CloseWriteFile

Close the file that's been opened for writing with OpenWrite or OpenAppend .

See also CD, Dir, Pwd, OpenAppend, OpenWrite.

Cosine, Cos

Cosine

Cosine *angle*

Output the cosine of *angle*.

```
Cosine 60
0.5
```

See also Sine, Tangent, ArcCosine, ArcSine, ArcTangent.

Count

Count the elements in an object

Count *list*

Count *word*

Output the number of elements in an object.

```
Count "xyz
3
Count [thing1 thing2 thing3 thing4]
4
Count [[23 45 56 78] purple []]
3
```

See also ButFirst, ButLast, Empty?, First, FirstPut, Item, Last, LastPut, List, List?, Member?.

CurrentPath

Output the current path

CurrentPath

The current path is the sequence of lines which have been generated by movements of the turtle or the GraphicsType command. The CurrentPath command outputs the current path as a list of commands, such as *moveto*, *lineto*, *curveto*, and *close*. The list can be used as input to StrokePath or FillPath.

```
Forward 100
Right 90
Forward 200
CurrentPath
[[moveto 0 0][lineto 0 100][lineto 200 100]]
```

See also StrokePath, FillPath, StrokeCurrentPath, FillCurrentPath, ReversePath, PathBounds,

Date

Output today's date

Date

Prints out today's date in the format *ccyy-mm-dd* where *ccyy* is the year, *mm* is the month number, and *dd* is the day of the month. For example:

```
Date
2003-05-20
```

See also Time

Define

Define a procedure

Define name [[*parameters*][*statements*]]

Create procedure name with parameters *parameters* and statements *statements*. For example:

```
Define "box [ [size] [ repeat 4 [forward :size right 90] ]
```

This is retained for compatibility with other versions of Logo. It's better to use the procedure window to create procedures. See also Define?.

Define?

Query existence of a procedure

Define? *name*

Outputs true if *name* is a procedure. E.g.:

```
Define? "box
true
```

See also Define.

Difference

Subtraction

Difference *number1 number2*

(Difference *number1 number2* ...)

Output the difference between two or more numbers.

```
Difference 100 80
20
```

```
(Difference 100 80 10)
10
```

See also +, -, *, /, Sum, Product, Quotient, Remainder.

Dir

List the current directory

Dir

List the contents of the current directory. In the resultant list, directories are signified by a trailing slash.

```
CD [/System]
Dir
.localized
Library/
```

Note that the view of the file system is the real view rather than the logical view seen in the Finder windows. Invisible files (starting with a '.') are listed.

See also Pwd, CD, OpenAppend, Openread, OpenWrite.

Dot

Draw a dot on the screen

Dot [*x y*]

Put a dot of the current colour at position *x y* on the screen. Note that positive y-values go up the screen.

Example:

```
Dot [100 80]
```

Empty?

Test if an object has no elements

Empty? *word*

Empty? *list*

Outputs true if an object has no elements.

```
Empty? "xyz
false
Empty? "
true
Empty? [[23 45 56 78] purple []]
false
Empty? ButFirst [abc]
true
```

See also ButFirst, ButLast, Count, First, FirstPut, Item, Last, LastPut, List, List?, Member?.

Equal?

Test if two objects are equal

Equal *object1 object2*

Outputs true if *object1* and *object2* are equal.

```
Equal? “xyz “xyz
true
Equal? [ ] [abc]
false
```

Exp

Exponential

Exp *number*

Returns e to the power of *number*.

```
Exp 1.8
6.04964746441295
```

See also Log

Fill

Fill an area

Fill

Fills an area with the current pen colour, starting from the current turtle position until a border of the current pen colour is hit. This may not work correctly if the “smooth lines” option is chosen (anti-aliasing) as the OSX graphics system draws lines with approximations to the current pen colour to get rid of jagged edges.

See also FillIn , FillPath ,

FillIn

Fill an area

FillIn

Fills an area with the current pen colour, starting from the current turtle position until a different coloured pixel from the start pixel is hit.

See also Fill, FillPath ,

FillCurrentPath

Fill the current path

FillCurrentPath

Fills an area bounded by the current path with the current pen colour. The current path is the sequence of lines which have been generated by movements of the turtle. A PenUp command or a command which moves the turtle without drawing a line, such as Clean or ClearScreen empties the path.

See also Fill , FillIn, StrokePath, FillPath, StrokeCurrentPath,

FillPath

Fill a path

FillPath [*path-commands*]

Fills an area bounded by the path specified by *path-commands*. Each element of path-commands is a list representing a path command such as *moveto*, *lineto*, *curveto*, or *close*.

The easiest way to create the list of path commands is to do some drawing, then save the path using CurrentPath.

See also Fill , FillIn, StrokePath, FillCurrentPath, StrokeCurrentPath, ReversePath, PathBounds,

First

Output the first element

First *word*

First *list*

Output the first element of a word or list.

```
First “xyz
x
First [[23 45 56 78] purple []]
[23 45 56 78]
```

See also ButFirst, ButLast, Count, Empty?, FirstPut, Item, Last, LastPut, List, List?, Member?.

FirstPut

Add an object to the start of another object

FirstPut *obj1 list*

FirstPut *obj1 word*

Add obj1 at the start of a list or word.

```
FirstPut “de “xyz
dexyz
FirstPut “bravo [apple tango]
[bravo apple tango]
FirstPut [New York] [London Paris Munich]
[[New York] London Paris Munich]
```

See also ButFirst, ButLast, Count, Empty?, First, Item, Last, LastPut, List, List?, Member?.

FontFace, Font

Return the name of the current font

FontFace

Returns the name of the current font as a list. This is the name of the font as known to the operating system, and consists of the font family name plus any attributes.

```
FontFace
[Helvetica]
setfonttraits [bold]
fontface
[Helvetica-Bold]
```

See also GraphicsType, TextBox, FontFamily, FontFamilies, FontFaces, SetFontFace, SetFontFamily, FontTraits, SetFontTraits.

FontFaces, Fonts

Return the name of available fonts

FontFaces

Returns the names of all available fonts as a list of lists.

```
FontFaces
[[LiSungLight][Bodoni-BoldItalic][ACaslonPro-Regular][CalistoMT-
Italic][LubalinGraph-Demi][TimesCYItalic][AGaramondPro-
Italic][FootlightMTLight][LucidaSans-Italic][DFLeiSho-SB-MP-RKSJ-
H][HiraMaruPro-W4][ArialMT][CapitalsRegular][HYSMyeongJoStd-Medium-
Acro][AGaramondPro-Regular][BaskOldFace][BookmanOldStyle-
Bold][CenturyGothic-Bold][Baskerville-BoldItalic][Mistral][AntiqueOlive-
Compact][Baskerville-Italic][ACaslonPro-Semibold][SIL-Kai-Reg-Jian][KozMinStd-
Heavy][Marigold][KinoMT]...
```

See also GraphicsType, TextBox, FontFamily, FontFamilies, FontFace, SetFontFace, SetFontFamily, FontTraits, SetFontTraits.

FontFamilies

Return the name of available font families

FontFamilies

Returns the name of all available font families as a list of lists.

```
FontFamilies
[[Gujarati MT][Footlight MT Light][Andale Mono][Albertus MT][Gurmukhi
MT][Didot][Gloucester MT Extra  Condensed][Geneva CY][Matura MT Script
Capitals][Monaco CY][Charcoal][Corsiva Hebrew][Silom][Mistral][Courier  New]
[Garamond][Chicago][Marigold][#HeadLineA][Hiragino Kaku Gothic Std][STSong Std
Acro][Beijing][DecoType Naskh]...
```

See also GraphicsType, TextBox, FontFamily, FontFace, FontFaces, SetFontFace, SetFontFamily, FontTraits, SetFontTraits.

FontFamily

Return the name of the family of the current font

FontFamily

Returns the name of the family of the current font. A font family may have several different font faces corresponding to it — for instance bold and italic versions. The name is returned as a list as it may contain spaces or characters which are Logo operators.

```
FontFace
[Helvetica]
FontFamily
[Helvetica]
SetFontTraits [bold italic]
FontFace
[Helvetica-BoldOblique]
FontFamily
[Helvetica]
```

See also GraphicsType, TextBox, FontFamilies, FontFace, FontFaces, SetFontFace, SetFontFamily, FontTraits, SetFontTraits.

FontTraits

Return the traits of the current font

FontTraits

FontTraits returns a list containing the traits of the current font. Traits are the style attributes of the font, and can currently be *bold* or *italic*. If there are no attributes, *plain* is returned.

```
FontFace
[Helvetica]
FontTraits
[plain]
SetFontTraits [bold italic]
FontTraits
[bold italic]
```

See also GraphicsType, TextBox, FontFamilies, FontFamily, FontFace, FontFaces, SetFontFace, SetFontFamily, SetFontTraits.

Forward, FD

Move the turtle forward

Forward *distance*

Move the turtle forward *distance* pixels. If the pen is down, a line is drawn.

See also Back, PenUp, PenDown.

FPrint

Print to a file

FPrint *object*

(FPrint *object1* ...)

Similar to Print, but writes out to a file rather than to the main window. Prints out one or more objects to the file currently open for writing. An object may be a number, word, or list. If an object is a list, the outermost brackets are not printed. Writes a new line afterwards.

```
OpenWrite [thing.txt]
FPrint [Bit at the start]
```

See also OpenWrite, CloseWriteFile, FShow, FType, Print.

FReadChar

Read a character from a file

FReadChar

Read a single character from the file which is currently open for reading and outputs it.

```
OpenRead [thing.txt]
FReadChar
B
```

See also OpenRead, CloseReadFile, FReadChars, FReadList, FReadWord, ReadChar.

FReadChars

Read characters from a file

FReadChars *count*

Read *count* characters from the file which is currently open for reading and outputs them as a word.

```
OpenRead [thing.txt]
FReadChars 10
Bit at the
```

See also OpenRead, CloseReadFile, FReadChar, FReadList, FReadWord, ReadChars.

FReadList

Read a line from a file into a list

FReadList

FReadList reads a line of characters from the file currently open for reading and outputs the characters as a list.

See also OpenRead, CloseReadFile, FReadChar, FReadChars, FReadWord, ReadList.

FReadWord

Read a line from a file into a word

FReadWord

ReadWord accepts characters typed in the main (text) window until carriage return is pressed, then outputs the characters as a word. The insertion point becomes a blue colour while it's waiting for input.

See also OpenRead, CloseReadFile, FReadChar, FReadChars, FReadList, ReadWord.

FShow

Write to a file

FShow *object*

(FShow *object1* ...)

Similar to Show, but writes out to a file rather than to the main window. Prints *object* to file currently open for writing, then starts a new line. if *object* is a list, the outermost brackets are printed.

```
OpenWrite [thing.txt]
FShow [Bit at the start]
```

See also OpenWrite, CloseWriteFile, FPrint, FType, Show.

FType

Write to a file

FType *object*

(FType *object1* ...)

Similar to Type, but writes out to a file rather than to the main window. Prints out one or more objects to the file currently open for writing. An object may be a number, word, or list. If an object is a list, the outermost brackets are not printed. Does not write a new line.

```
OpenWrite [thing.txt]
FType [Bit at the start]
```

See also OpenWrite, CloseWriteFile, FShow, FPrint, Type.

GraphicsType, GrType

Draw some text

GraphicsType *word*

GraphicsType *list*

Prints *word* or *list* to the graphics screen at the current pen position with the current pen colour. The command does not change the turtle position. It does not print the outermost brackets of a list.

Example:

```
grtype “start
setpencolour 2
left 45
grtype [the end]
```

See also Show, Print, Type, SetTypeSize, TextBox, StrokePath.

Heading

Output the turtle heading

Heading

Output the heading angle of the turtle. This is the angle the turtle faces, and the angle it will move in if the Forward

command is used. A heading of zero is facing straight up.The heading increases in a clockwise direction - a heading of 90 is pointing to the right, 180 is pointing straight down, 270 is pointing to the left. When the heading reaches 360, it is reset to zero.

```
ClearScreen
Right 45
Heading
45
ClearScreen
Left 1
Heading
359
```

See also SetHeading, Right, Forward, Left.

HideTurtle, HT

Hide the turtle

HideTurtle

Hide the turtle. Its position remains the same. Drawing happens in the same way, and the turtle's position is affected by drawing commands in the same way as when it is showing.

See also ShowTurtle.

Home

Home the turtle

Home

Move the turtle to the middle of the screen (position [0 0]) and set its heading to zero (pointing straight up).

See also ClearScreen, Position, SetPosition, Heading, SetHeading.

If

Conditional processing

If *condition* [*true-statements*] [*false-statements*]

If condition *condition* is true, execute *true-statements*, otherwise execute *false-statements*. Both of the lists must be present, although either can be empty.

```
ClearScreen
Right 45
If Heading > 180 [print [pointing left]] [print [pointing right]]
pointing right
```

Instruments

Output available instruments

Instruments

Return a list of the descriptions of the instruments available for use by Play. The first description corresponds to

instrument number 1, etc.

```
Instruments
[ [Acoustic Grand Piano] [Bright Acoustic Piano] [Electric Grand Piano]
[Honkytonk Piano] [Electric Piano] [Chorused Piano] [Harpsichord] [Clavi]
[Celesta] [Glockenspiel] [Music Box] [Vibraphone]...
```

See also Play.

Integer, Int

Truncate to integer

Integer *number*

Truncates *number* to just its integer portion.

```
Integer 1.8
1
Integer 100 / 3
33
```

See also Round

Item

Output n^{th} element of an object

Item *number list*

Item *number word*

Outputs element *number* of an object.

```
Item 3 “abcdef
c
Item 2 [apple tango]
tango
Item 2 [[New York] London Paris Munich]
London
```

See also ButFirst, ButLast, Count, Empty?, First, FirstPut, Last, LastPut, List, List?, Member?.

Last

Output the last element of an object

Last *list*

Last *word*

Output the last element of an object.

```
Last “abcdef
f
Last [apple tango]
tango
```

See also ButFirst, ButLast, Count, Empty?, First, FirstPut, Item, LastPut, List, List?, Member?.

LastPut

Append to a word or list

LastPut *object list*

LastPut *object word*

Output object appended to the end of a list or word.

```
LastPut “xyz “abcdef
abcdefxyz
LastPut “bravo [apple tango]
[apple tango bravo]
LastPut [New York] [London Paris Munich]
[London Paris Munich [New York]]
```

See also ButFirst, ButLast, Count, Empty?, First, FirstPut, Item, Last, List, List?, Member?.

Left

Turn the turtle anticlockwise

Left *angle*

Rotate the turtle anti-clockwise through *angle* degrees.

```
Home
Heading
0
Left 30
Heading
330
Left 25
Heading
305
```

See also Right, Heading, SetHeading, Forward.

List

Create a list

List *object1 object2*

(List *object1 object2* ...)

Output a list consisting of object1, object2, ...

```
List “xyz “abcdef
[xyz abcdef ]
List “bravo [apple tango]
[bravo [apple tango] ]
(List “New “York “London “Paris “Munich)
[New York London Paris Munich ]
```

See also ButFirst, ButLast, Count, Empty?, First, FirstPut, Item, Last, LastPut, List?, Member?.

List?

Test if object is a list

List? *object*

Output **true** if *object* is a list.

```
List? “xyz
false
List? [apple tango]
true
```

See also ButFirst, ButLast, Count, Empty?, First, FirstPut, Item, Last, LastPut, List, Member?.

Local

Declare a local variable

Local *name*

(Local *name1* ...)

Declare a name as local to a procedure - effectively a local variable.

```
Local “thething
Make “thething 5
Make “thething :thething + 1
:thething
6
```

See also Make, Thing.

Log, LN

Natural logarithm

Log *number*

Returns the natural logarithm of *number*.

```
Log 6.04964746441295
1.8
```

See also Log10, Exp

Log10

Base-10 logarithm

Log10 *number*

Returns the base-10 logarithm of *number*.

```
Log10 1000
3
```

See also Log

LowerCase

Convert to lowercase

LowerCase *list*

LowerCase *word*

Output *list* or *word* with all upper case characters converted to lower case.

```
LowerCase  “ABC
abc
LowerCase   [aHGF   8768 HHHH a]
[ahgf   8768 hhhh a]
```

See also UpperCase .

Make

Set the value of a variable

Make *name object*

Give variable *name* the value *object*. Creates the variable if it doesn’t exist.

```
Make “thething 5
Make “thething :thething + 1
:thething
6
```

See also Local, Thing.

Member?

Test if object is a member of a word or list

Member *object1 object2*

Output true if *object1* is a member of *object2*.

```
Member?  “y “xyz
true
Member?  “s “xyz
false
Member? “tango [apple tango lima]
true
Member? “tango [apple [tango lima]]
false
Member? [tango  lima] [apple [tango lima]]
true
```

See also ButFirst, ButLast, Count, Empty?, First, FirstPut, Item, Last, LastPut, List, List?.

Not

Logical NOT

Not *predicate*

Output **false** if *predicate* is true, otherwise **true**.

```
not “true
false
not 3 < 2
true
```

See also And, Or, <, >.

Number?

Test an object to see if it’s a number

Number? *object*

Output **true** if *object* is a number, otherwise **false**.

```
Number? 5.6
true
Number? [abc]
false
Number? “xyz
false
Number? “155.6
true
```

OpenAppend

Open a file for appending to

OpenAppend *file-name*

Open file *file-name* in the current directory for appending, i.e. written data will be added on the end. *file-name* may be a word or a list. Only one file can be open for writing at a time.

After opening, the file can be written to using commands FPrint, FShow, and FType. The file should then be closed using CloseWriteFile.

```
OpenAppend [thing.txt]
FPrint [Bit at the end]
CloseWriteFile
```

See also CD, Dir, Pwd, OpenRead, OpenWrite, CloseWriteFile, FPrint, FShow, FType.

OpenRead

Open a file for reading

OpenRead *file-name*

Open file *file-name* in the current directory for reading. Only one file can be open for reading at a time. After opening, the file can be read from using commands FReadChar, FReadChars, FReadList, FReadWord. The file should then be closed using CloseReadFile.

```
OpenRead [thing.txt]
Make "var FReadList
CloseReadFile
:var
[Bit at the start]
```

See also CD, Dir, Pwd, OpenAppend, OpenWrite, CloseReadFile, FReadChar, FReadChars, FReadList, FReadWord.

OpenWrite

Open a file for writing to

OpenWrite *file-name*

Open file file-name in the current directory for writing to. Previous contents of the file are overwritten. *file-name* may be a word or a list. Only one file can be open for writing at a time.

After opening, the file can be written to using commands FPrint, FShow, and FType. The file should then be closed using CloseWriteFile.

```
OpenWrite [thing.txt]
FPrint [Bit at the start]
FPrint [Bit in the middle]
CloseWriteFile
```

See also CD, Dir, Pwd, OpenRead, OpenAppend, CloseWriteFile, FPrint, FShow, FType.

Or

Logical OR

Or *predicate1 predicate2*

(Or *predicate1 predicate2* ...)

Output true if any of the predicates is true.

```
or "true "false
true
(or (3 > 4) (5 > 6) (99 < 100))
true
```

See also And, Not, <, >.

Output, Op

Output result from a procedure

Output *object*

Return *object* as the output of a procedure.

See also Stop.

PathBounds

Output the bounding box of a path

PathBounds *path-list*

Outputs a list representing the bounding box of *path-list* - a list of statements representing a path.

The output from PathBounds is in the form [*min-x-coord min-y-coord width height*].

```
Forward 45 Right 90 Forward 60 Right 90 Forward 50
PathBounds CurrentPath
[0 -5 60 50]
```

See also CurrentPath, FillPath, StrokePath, ReversePath.

Pen

Output the pen state and colour

Pen

Outputs a list containing the pen state and pen colour.

```
PenDown
SetPenColor 3
Pen
[PenDown 3]
```

See also PenDown, PenUp, SetPenColor, PenColor.

PenColour, PenColor, PC

Output the pen colour number

PenColour

Output the Pen colour number. This is the colour number which will be used when lines are drawn or fills are done. At startup, the Pen colour is set to 1.

```
SetPenColor 3
PenColor
3
```

See also PenDown, PenUp, SetPenColor, Pen.

PenDown, PD

Put the pen into drawing state

PenDown

Put the pen into draw state. If the turtle moves, it will draw a line.

See also PenUp, PenColor.

PenUp, PU

Put the pen into non-drawing state

PenUp

Put the pen into non-draw state.

See also PenDown, PenColor.

PenWidth

Output the size of the drawing pen

PenWidth

Output the size of the drawing pen. This determines the width of lines that are drawn by the turtle.

```
SetPenWidth 10
PenWidth
10
```

See also SetPenWidth.

Pi π

Pi

Output the mathematical constant Pi, the ratio of the circumference of a circle to its diameter.

```
Pi
3.14159265358979
```

Play

Play Sounds

Play *music-part-list*

Play [*music-part-list1 music-part-list2* ...]

Play the notes specified by one or more *music-part-lists*. Each music-part-list can be thought of as a piece for a particular instrument, and is of the form:

[*instrument chord1 chord2*...]

The music-part-lists are played simultaneously. Each chord is of the form:

[*duration loudness note1 note2*...]

duration is the duration of the chord and is specifed in ticks. Each tick is one sixtieth of a second. Loudness is the loudness of the note. Loudness is followed by one or more notes which are played together to form a chord. The note is specified as a number between 0 and 127. Number 60 is middle C.

Example 1. Play middle C for a second on a grand piano:

```
Play [1 [60 80 60]]
```

Example 2. Play an arpeggio on a harpsichord. Each note lasts for half a second:

```
Play [7 [30 80 60][30 80 64][30 80 67][30 80 72]]
```

Example 3. Play a chord lasting for two seconds:

```
Play [1 [120 80 60 64 67 72]]
```

To get a list of available instruments, see Instruments.

Position, Pos

Output the turtle's position

Position

Output a list containing the turtle's x and y co-ordinates.

```
Home
Position
[0 0]
Forward 100
Position
[0 100]
```

See also Home, ClearScreen, Forward, Back, SetPosition, SetX, SetY.

Power

Power

Power *number1 number2*

Returns *number1* to the power of *number2*.

```
Power 2 3
8
Power 2 0.5
1.4142135623731
```

Print

Display objects

Print *object*

(Print *object1* ...)

Prints out one or more objects to the main text window. If an object is a list, the outermost brackets are not printed. Writes a new line afterwards.

```
Print “abc
abc
print [the end of the line]
the end of the line
(print “abc “def “ghi)
abc def ghi
```

See also Show, GraphicsType, Type.

Product

Multiplication

Product *number1 number2*

(Product *number1 number2*...)

Output the product of the input numbers.

```
Product 100 10
1000
(Product 3 4 5)
60
```

See also +, -, *, /, Sum, Difference, Quotient, Remainder.

Pwd

Output current directory

Pwd

Output the current directory, similar to the Unix command (stands for Print Working Directory).

```
cd "~
Pwd
/Users/alan
```

See also CD, Dir, OpenAppend, OpenRead, OpenWrite.

Quotient

Division

Quotient *number1 number2*

(Quotient *number1 number2* ...)

Output the quotient of the input numbers.

```
Quotient 100 3
33.3333333333333
(Quotient 3 4 5)
0.15
```

See also +, -, *, /, Sum, Difference, Product, Remainder.

Random

Random number

Random *number*

Output a random integer between zero and *number*.

```
Random 55
45
Random 1000 * 1000
800899
```

ReadChar

Read a single character

ReadChar

ReadChar waits for a character to be typed in the main (text) window, then outputs the character as a word. The insertion point becomes a blue colour while it's waiting for input.

See also ReadChars, ReadList, ReadWord, FReadChar.

ReadChars

Read mutiple characters

ReadChars *count*

Output a word containing *count* characters read from the keyboard. The insertion point becomes a blue colour while it's waiting for input.

See also ReadChar, ReadList, ReadWord, FReadChars.

ReadList

Read characters into a list

ReadList

ReadList accepts characters typed in the main (text) window until carriage return is pressed, then outputs the characters as a list. The insertion point becomes a blue colour while it's waiting for input.

See also ReadChar, ReadChars, ReadWord.

ReadWord

Read characters into a word

ReadWord

ReadWord accepts characters typed in the main (text) window until carriage return is pressed, then outputs the characters as a word. The insertion point becomes a blue colour while it's waiting for input.

See also ReadChar, ReadChars, ReadList.

Remainder

Remainder

Remainder *number1 number2*

Output the remainder when *number1* is divided by *number2*.

```
Remainder 25 20
5
```

See also +, -, *, /, Sum, Difference, Product, Quotient.

Repeat

Repeat a list of statements

Repeat *number* [*statements*]

Run statements *statements number* times.

```
Repeat 5 [Print Random 100]
52
85
86
47
8
```

See also Run.

ReversePath

Reverse a path

ReversePath *path-list*

Given *path-list*, a list representing a sequence of path-drawing commands, outputs a list representing the path drawn backwards.

```
Forward 45 Right 90 Forward 60 Right 90 Forward 50
CurrentPath
[[moveto 0 0][lineto 0 45][lineto 60 45][lineto 60 -5]]
ReversePath CurrentPath
[[moveto 60 -5][lineto 60 45][lineto 0 45][lineto 0 0]]
```

See also CurrentPath, FillPath, StrokePath, PathBounds.

RGB

Output the RGB values for a colour number

RGB *colour-number*

Output the red, green, and blue components of colour *colour-number*. Each component is a number between 0.0 and 1.0. Because of the way colours are held within Mac OSX, the values returned from RGB may be slightly different from those set with SetRGB.

```
RGB 0
[1 1 1]
RGB 1
[0 0 0]
```

See also SetRGB, PenColor, SetPenColor, Pen.

Right, Rt

Turn the turtle clockwise

Right *angle*

Rotate the turtle clockwise through *angle* degrees.

```
Home
Heading
0
Right 30
Heading
30
Right 25
Heading
55
```

See also Left, Heading, SetHeading, Forward.

Round

Round to nearest integer

Round *number*

Round a number to the nearest integer.

```
Round 1.1
1
Round 1.5
2
```

See also Integer.

Run

Execute a list of statements

Run [*statements*]

Execute statements *statements*.

```
Run [Abs -6]
6
Right 45 Forward 100
Make “mylist Firstput “Sum Position
:mylist
[Sum 70.7106857299805 70.7106628417969]
Run :mylist
141.421348571777
```

See also Repeat.

Say

Speak using the system voice synthesizer

Say *list*

Say *word*

Speaks its parameter using the operating system voice synthesizer. The command returns immediately — i.e., it does not wait for the speech to finish before continuing. If you need it to, use WaitForSpeech. You can change the voice used by Say by using the SetVoice command.

```
Say [Needle in the Hay]
```

See also Voice, Voices, SetVoice, WaitForSpeech.

Sentence

Create a list

Sentence *object1 object2*

(Sentence *object1 object2* ...)

Output a list containing the input objects. Strips the outermost brackets of any object which is a list.

```
Sentence “abc “def
[abc def]
Sentence “abc [def]
[abc def]
Sentence “abc [[def]]
[abc [def]]
```

See also List.

SetBackground, SetBG

Set the background colour

SetBackground *colour-number*

Set the background colour to *colour-number*. This colour will be used when ClearScreen or Clean is called.

```
SetBackground 3
Background
3
```

See also Background, SetPenColor, RGB, SetRGB, Clean, ClearScreen.

SetFontFace, SetFont

Set the current font face

SetFontFace [*font-name*]

Set the current font face (the terms font and font face are interchangeable). Note that the name of the font face is enclosed in list brackets. This is in case the name contains unusual characters. A list of available font faces is given by the FontFaces function. The name of the current font face (the one currently used for drawing) is given by the FontFace function. The name of the font face will generally contain the font family name plus any font traits.

```
SetFontFace [Baskerville-BoldItalic]
```

See also GraphicsType, TextBox, FontFamilies, FontFamily, FontFace, FontFaces, SetFontFamily, FontTraits, SetFontTraits.

SetFontFamily

Set the current font family

SetFontFamily [*font-family-name*]

Sets the current font family to *font-family-name*. A font family name is a generic name which often applies to several fonts. The corresponding font names will have attributes such as *Bold*, *Italic*, *Light*, appended. Note that the name of the font family is enclosed in list brackets. This is in case the name contains spaces or other characters such as hyphens. A list of available font families is given by the FontFamilies function. The name of the current font family (the one currently used for drawing) is given by the FontFamily function.

```
SetFontFamily [Baskerville]
```

See also GraphicsType, TextBox, FontFamilies, FontFamily, FontFace, FontFaces, SetFontFace, FontTraits, SetFontTraits.

SetFontTraits

Set the traits of the current font

SetFontTraits *trait-list*

Set the traits of the current font. The available traits for a font are *bold* and *italic*. The arguments to SetFontTraits are *bold*, *unbold* (to turn off *bold*), *italic*, *unitalic* (to turn off *italic*), and *plain*. Plain is a lack of the other traits.

```
FontFace
[Helvetica]
SetFontTraits [bold italic]
FontFace
[Helvetica-BoldOblique]
SetFontTraits [plain]
FontFace
[Helvetica]
```

See also GraphicsType, TextBox, FontFamilies, FontFamily, FontFace, FontFaces, SetFontFace, SetFontFamily, FontTraits.

SetHeading

Set the turtle's heading

SetHeading *angle*

Set the turtle heading to *angle*. The heading is the direction in which the turtle is pointing. Straight up is a heading of zero. The heading increases as you go clockwise - straight down is 180.

```
SetHeading 45
Heading
45
```

See also Heading, Left, Right.

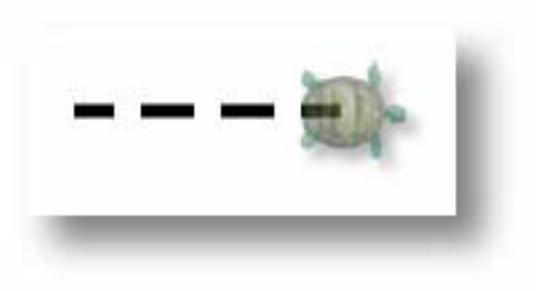
SetLineDash

Set the dash pattern for lines

SetLineDash [*phase drawn-dash-1 empty-dash-1...*]

Sets the line dash pattern for drawn lines. *drawn-dash-1* is the length, in pixels, of the first, drawn, part of the line. *empty-dash-1* is the length of the first blank part of the line. The pattern is then repeated. *phase* is how far into the pattern the line starts drawing.

```
SetPenWidth 5
SetLineDash [5 20 10]
Right 90 Forward 100
```



See also SetPenWidth, Forward.

SetPen

Set the state of the pen

SetPen [*penstate colour-number*]

Set the state of the pen to *penstate* and its colour to colour-number. Penstate is PENUP or PENDOWN.

```
SetPen [penup 7]
Pen
[PenUp 7]
```

See also Pen, SetPenColor, PenUp, PenDown, SetRGB.

SetPenColour, SetPenColor, SetPC

Set the foreground colour

SetPenColor *colour-number*

Set the drawing colour to *colour-number*. This is the colour number used to draw lines and do fills. Use SetRGB to set this colour number to a particular colour value.

```
SetPenColor 3
PenColor
3
```

See also Pen, SetPen, PenColor, RGB, SetRGB.

SetPenWidth

Set the width of the drawing pen

SetPenWidth *width*

Set the width of the pen to *width*. New lines are drawn with this width.

```
PenWidth
1
SetPenWidth 10
PenWidth
10
```

See also PenWidth.

SetPosition, SetPos

Set the position of the turtle

SetPosition [*x y*]

Move the turtle to position *x,y*. If the pen is down, a line is drawn in the current colour.

```
SetPosition [100 100]
Position
[100 100]
```

See also Position, Forward, Back, Home, ClearScreen.

SetRGB

Set a colour's RGB values

SetRGB colour-number [*red green blue*]

SetRGB colour-number [*red green blue opacity*]

Set red, green, and blue components of colour colour-number to *red*, *green*, *blue*. Each component is a number

between 0.0 and 1.0. 0.0 means that none of that component is present, while 1.0 means all of it is present. So, [1.0 0.0 0.0] is a bright red and [0.0 0.0 1.0] is a bright blue. Black is [0.0 0.0 0.0] and white is [1.0 1.0 1.0].

If *opacity* is specified, the colour has the specified opacity - 0.0 is completely transparent, 1.0 is completely opaque.

```
SetRGB 3 [0.6 0.7 0.8]
RGB 3
[0.6 0.7 0.8]
```

See also Position, Forward, Back, Home, ClearScreen.

SetShadow

Set the dropshadow for drawing

SetShadow [*x-offset y-offset radius*]

SetShadow [*x-offset y-offset radius colour-number*]

SetShadow [ ]

Set the dropshadow for all subsequent drawing. The x-offset and y-offset determine how far the shadow is offset from the originating drawing. Radius determines how much the shadow is blurred — i.e., how far it spreads. If colour-number is specified, that colour is used to draw the shadow, otherwise a black colour with opacity of 0.3 is used.

If SetShadow is called with an empty list, dropshadow drawing is turned off.

Example:

```
SetShadow [15 -15 5]
SetTypeSize 146
SetPenColour 2
GraphicsType “abc
```

See also SetRGB, SetPenColour.

SetTypeSize

Set the size for type

SetTypeSize *type-size*

Set the size of type in the graphics window (displayed by GraphicsType).

Example:

```
PenUp
SetTypeSize 24
GraphicsType [24 type]
```

```
Forward 32
SetTypeSize 46
GraphicsType [46 type]
```

See also GraphicsType, TextBox, StrokePath

SetVoice

Set the current voice

SetVoice [*voice-name*]

Sets the current voice used in speech to *voice-name*. A list of available voices is given by Voices.

```
SetVoice [com.apple.speech.synthesis.voice.Vicki]
```

See also Voice, Voices, Say, WaitForSpeech.

SetX

Set the x position of the turtle

SetX *x*

Set the x-co-ordinate of the turtle to *x*. Draws a line if the pen is down.

```
Home
SetX -100
Position
[-100 0]
```

See also Position, Forward, Back, SetPosition, SetY, Home, ClearScreen.

SetY

Set the y position of the turtle

SetY *y*

Set the y-co-ordinate of the turtle to *y*. Draws a line if the pen is down.

```
Home
SetY -100
Position
[0 -100]
```

See also Position, Forward, Back, SetPosition, SetX, Home, ClearScreen.

Show

Display objects

Show *object*

(Show *object1* ...)

Print *object* to the main text window, then start a new line. if *object* is a list, the outermost brackets are printed.

```
Show [the end of the line]
[the end of the line]
(Show “abc “def “ghi)
abc def ghi
```

See also Print, GraphicsType, Type, FShow.

Shown?

Output the visibility of the turtle

Shown?

Output true if the turtle is visible, otherwise false.

```
HideTurtle
Shown?
false
```

See also ShowTurtle, HideTurtle.

ShowTurtle, ST

Show the turtle

ShowTurtle

Show the turtle if it is hidden.

```
ShowTurtle
Shown?
true
```

See also Shown?, HideTurtle.

Sine, Sin

Sine

Sine angle

Output the sine of angle.

```
Sine 60
0.866025403784439
```

See also Cosine, Tangent, ArcCosine, ArcSine, ArcTangent.

Snap

Capture an animation frame

Snap

Captures the current graphics view to the current movie if one is being created, otherwise does nothing.

By choosing **Create Movie...** from the Special menu, then using **Snap** to capture several frames, then choosing **Finish Movie** , you can create an animation.

SqRt

Square Root

Sqrt *number*

Output the square root of *number*.

```
Sqrt 2
1.4142135623731
```

Stop

Return from a procedure

Stop

Return from a procedure without returning a result.

See also Output.

StrokeCurrentPath

Stroke the current path

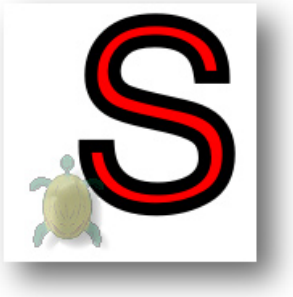
StrokeCurrentPath

Stroke, i.e. draw a line along, the current path with the current pen colour. The current path is the sequence of lines which have been generated by movements of the turtle or the GraphicsType command. A PenUp command or a command which moves the turtle without drawing a line, such as Clean or ClearScreen empties the path.

The main use of this is to outline some text, for example:

```
SetPenColour 2
SetTypeSize 156
GraphicsType “S
SetPenWidth 8
```

```
SetPenColour 1
StrokeCurrentPath
```



See also FillPath , SetTypeSize, GraphicsType, FillCurrentPath, StrokePath, CurrentPath,

StrokePath

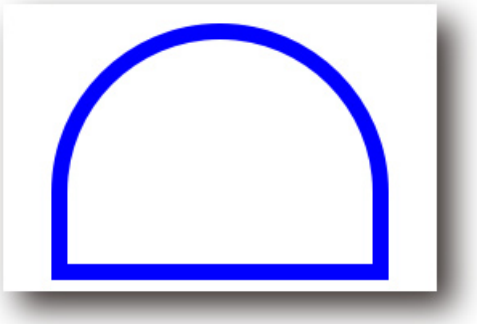
Stroke a path

StrokePath [*path-commands*]

Stroke, i.e. draw a line along, the path represented by *path-commands*. Each element of path-commands is a list representing a path command such as *moveto*, *lineto*, *curveto*, or *close*.

The easiest way to create the list of path commands is to do some drawing, then save the path using CurrentPath.

```
strokepath [[moveto -100 0] [curveto 0 100 -100 55 -55 100] [curveto 100 0 55 100
100 55] [lineto 100 -50][lineto -100 -50][close]]
```



See also FillPath , SetTypeSize, GraphicsType, FillCurrentPath, StrokeCurrentPath, CurrentPath, ReversePath, PathBounds,

Sum

Addition

Sum *number1 number2*

(Sum *number1 number2* ...)

Output the sum of the input numbers.

```
Sum 100 80
180
(Sum 100 80 10)
190
```

See also +, -, *, /, Difference, Product, Quotient, Remainder.

Tangent, Tan

Tangent

Tangent *angle*

Output the tangent of *angle*.

```
Tan 45
1
```

See also Cosine, Sine, ArcCosine, ArcSine, ArcTangent.

Text

Output a procedure as a list

Text *name*

Output procedure *name* as a list of lists. See Define.

TextBox

Output list describing text size

TextBox *string*

TextBox *list*

Output a list describing the size of the parameter if printed by GraphicsType. The list is of the form [*x y w h*], where *x,y* is the co-ordinate of the bottom-left corner, *w* is the box width, and *h* is the box height.

The textbox is not a bounding box - the height of the box is the line-height of the text, and the width includes letter spacing on either side:

For Example:

```
SetTypeSize 72
TextBox [The End]
[400 400 250.22265625 86]
```

See also SetTypeSize, GraphicsType, StrokePath.

Thing

Output value of a variable

Thing *name*

Output the value of variable *name*. An alternative to using ':' to access the value of a variable.

```
Make "var 333
thing "var
333
:var
333
```

See also Make.

Time

Output the current time

Time

Outputs the current time of day in the format *hh:mm:ss:ttt* where *hh* is the hour, *mm* is the minutes past the hour, *ss* is seconds past the minute, and *tt* is thousands of a second.

By calling time at the start and end of a procedure, you can determine how long the procedure takes.

```
Time
19:25:41:693
```

See also Date

Towards

Output required heading

Towards [*x y*]

Output the angle which the turtle's heading must be set to to point towards position *x,y*.

```
Home
Towards [100 100]
45
```

See also Heading, SetHeading, Position.

Type

Print object

Type *object*

(Type *object1 object2* ...)

Print an object or objects - do not start a new line. Remove outer brackets for a list.

```
Type [the end of the line]
the end of the line
```

See also Print, GraphicsType, Show, FType.

UpperCase

Convert to upper case

UpperCase *list*

UpperCase *word*

Output *list* or *word* with all lower case characters converted to upper case.

```
UpperCase  “abc
ABC
UpperCase   [h4j5jlllABAB 8]
[H4J5JLLLABAB 8]
```

See also LowerCase .

Voice

Output the name of the current voice

Voice

Output the name of the current voice used in speech (by the Say command) as a list.

```
Voice
[com.apple.speech.synthesis.voice.Vicki]
```

See also Voices, SetVoice, Say, WaitForSpeech.

Voices

Output the names of available voices

Voices

Output a list containing the names of all voices available for speech.

```
Voices
[[com.apple.speech.synthesis.voice.Agnes] [com.apple.speech.synthesis.voice.Alber
t] [com.apple.speech.synthesis.voice.BadNews] [com.apple.speech.synthesis.voice.B
ahh] [com.apple.speech.synthesis.voice.Bells]...
```

See also Voice, SetVoice, Say, WaitForSpeech.

Wait

Wait for a specified duration

Wait *duration*

Waits (i.e., does nothing) for *duration* ticks, where a tick is one sixtieth of a second. The following statement waits for two seconds:

```
Wait 120
```

WaitForSpeech

Wait for speech to finish

WaitForSpeech

If something is being spoken (effected by the Say command), waits for the speech to finish before proceeding. This stops consecutive Say commands from overlaying each other.

```
Say [The End of the World]
WaitForSpeech
Say [Is Nigh]
```

See also Voice, Voices, SetVoice, Say.

Word

Concatenate words

Word *word1 word2*

(Word *word1 word2* ...)

Output a word consisting of the input words concatenated.

```
Word "abc "def
abcdef
(Word "abc "def "ghi)
abcdefghi
```

See also Word?, Sentence.

Word?

Test if object is a word

Word? *object*

Output **true** if *object* is a word.

```
Word? “abc
true
Word? [a b c]
false
```

See also Word.

XPos

Output the turtle’s x co-ordinate

XPos

Output the turtle’s x co-ordinate.

```
Home
Left 90 Forward 100
XPos
-100
```

See also Position, Forward, Back, SetPosition, SetX, Home, ClearScreen, YPos.

YPos

Output the turtle’s y co-ordinate

YPos

Output the turtle’s y-co-ordinate.

```
Home
Forward 100
YPos
100
```

See also Position, Forward, Back, SetPosition, SetY, Home, ClearScreen, XPos.