ACSLogo

Command Reference

Version 1.5

Contents

+	*Add*	8
-	*Subtract*	8
*****	*Multiply*	8
/	*Divide*	8
<	*Less Than*	8
>	*Greater Than*	9
ABS	*Output the absolute value of a number*	9
AND	*Logical AND*	9
Arc	*Draw an Arc*	10
ArcCosine, ArcCos	*Output the angle for a Cosine*	10
ArCosh	*Inverse of Cosh*	10
ArcSine, ArcSin	*Output the angle for a Sine*	11
ArcTangent, ArcTan	*Output the angle for a Tangent*	11
ArSinh	*Inverse of Sinh*	11
ArTanh	*Inverse of Tanh*	11
ATan2	*Inverse of Tan taking two arguments*	11
ASCII	*Output a character's ASCII code*	12
Back	*Move the turtle backwards*	12
Background, Bg	*Output the background pen colour*	12
ButFirst	*Output all but the first element*	12
ButLast	*Output all but the last element*	13
Button?, ButtonP	*Output whether the left mousebutton is pressed*	13
CanvasSize	*Output the size of the canvas*	13
Catch	*Catch a Throw statement*	13
CD	*Change the Current Directory*	14
Char	*Output the character for an ASCII code*	14
Clean	*Clear the graphics window*	14
ClearScreen, CS	*Clear graphics and home the turtle*	14
CloseReadFile	*Close the file open for reading*	14
CloseWriteFile	*Close the file open for writing*	15
ColourAtPoint, ColorAtPoint	*Output the RGB values of a pixel*	15
Cosine, Cos	*Output the cosine of an angle*	15
Cosh	*Hyperbolic cosine*	15
Count	*Count the elements in an object*	15
CurrentPath	*Output the current path*	16
Date	*Output today's date*	16
Define	*Define a procedure*	16
Define?, DefineP	*Query existence of a procedure*	16

Difference	*Subtract two or more numbers*	17
Dir	*List the current directory*	17
Dot	*Draw a dot on the screen*	17
DrawImage	*Draw an image*	17
Empty?, EmptyP	*Test if an object has no elements*	18
Eof?, EofP	*Test for end-of-file*	18
Equal?, EqualP	*Test if two objects are equal*	19
Exp	*Exponential*	19
ExportEPS	*Export an EPS*	19
ExportPDF	*Export a PDF*	19
ExportTIFF	*Export a TIFF*	19
Fill	*Fill an area*	20
FillIn	*Fill an area*	20
FillCurrentPath	*Fill the current path*	20
FillPath	*Fill a path*	20
First	*Output the first element*	20
FirstPut	*Add an object to the start of another object*	21
FontFace, Font	*Return the name of the current font*	21
FontFaces, Fonts	*Return the names of available fonts*	21
FontFamilies	*Return the names of available font families*	21
FontFamily	*Return the name of the family of the current font*	22
FontTraits	*Return the traits of the current font*	22
Forward, FD	*Move the turtle forward*	22
FPrint	*Print to a file*	23
FReadChar	*Read a character from a file*	23
FReadChars	*Read characters from a file*	23
FReadList	*Read a line from a file into a list*	23
FReadWord	*Read a line from a file into a word*	23
FShow	*Write to a file*	23
FType	*Write to a file*	24
GetMouseChange	*Wait for the mouse button or a mouse move*	24
GetMouseClick	*Wait for the left mouse button to be pressed*	24
GetMouseMoved	*Wait for the mouse to be moved*	24
GetProp, GProp	*Retrieve a property for a name*	25
GraphicsType, GrType	*Draw Some Text*	25
Heading	*Output the turtle heading*	26
HideTurtle, HT	*Hide the turtle*	26
Home	*Home the turtle*	26
If	*Conditional processing*	26
Instruments	*Output available instruments*	26

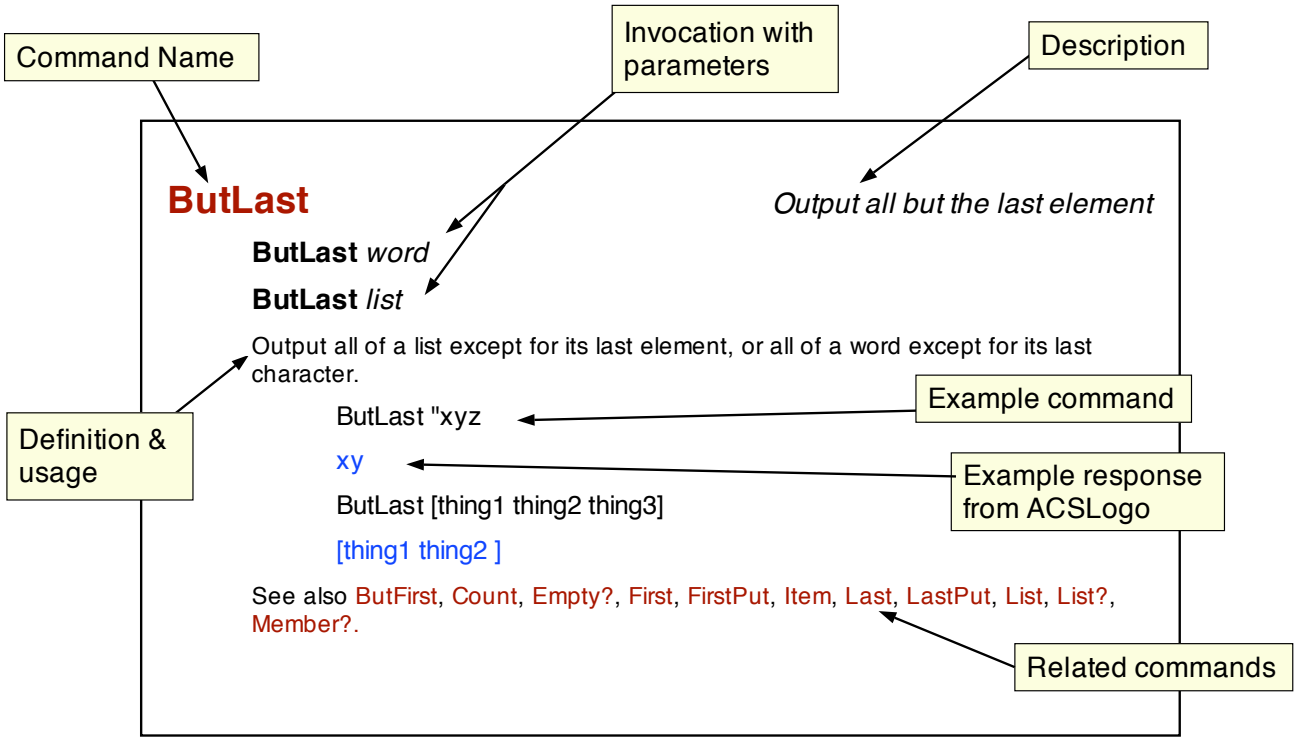
Integer	*Truncate to integer*	27
Item	*Output nth element of an object*	27
Last	*Output the last element of an object*	27
LastPut	*Append to a word or list*	27
Left	*Turn the turtle anticlockwise*	28
List	*Create a list*	28
List?, ListP	*Test if object is a list*	28
Local	*Declare a local variable*	29
Log, LN	*Natural logarithm*	29
Log10	*Base-10 logarithm*	29
LowerCase	*Convert to lowercase*	29
Make	*Set the value of a variable*	30
Member?, MemberP	*Test membership of a word or list*	30
Mouse	*Output mouse co-ordinates*	30
Name?, NameP	*Test existence of a variable*	30
Not	*Logical NOT*	31
Number?, NumberP	*Test whether an object is a number*	31
OpenAppend	*Open a file for appending to*	31
OpenRead	*Open a file for reading*	31
OpenTextAppend	*Open a text file for appending to*	32
OpenTextRead	*Open a text file for reading*	32
OpenTextWrite	*Open a unicode file for writing to*	32
OpenWrite	*Open a file for writing to*	33
Or	*Logical OR*	33
Output, Op	*Output result from a procedure*	33
PathBounds	*Output the bounding box of a path*	33
PathLength	*Output the length of a path*	34
Pen	*Output the pen state and colour*	34
PenColour, PenColor, PC	*Output the pen colour number*	34
PenDown, PD	*Put the pen into drawing state*	34
PenUp, PU	*Put the pen into non-drawing state*	34
PenWidth	*Output the size of the pen*	35
Pi	π	35
Play	*Play sounds*	35
Position, Pos	*Output the turtle's position*	35
Power	*Raise a number to a power*	36
Print	*Display objects*	36
Product	*Multiplication*	36
PropList, PList	*List properties for a name*	36
PutProp, PProp	*Set a property for a name*	37

Pwd	*Output current directory*	37
Quotient	*Division*	37
Random	*Random number*	37
ReadChar	*Read a single character*	38
ReadChars	*Read multiple characters*	38
ReadList	*Read characters into a list*	38
ReadWord	*Read characters into a word*	38
Remainder	*Remainder from division*	38
RemProp	*Remove a property*	38
Repeat	*Repeat a list of statements*	39
ReversePath	*Reverse a path*	39
RGB	*Output the RGB values for a colour number*	39
Right	*Turn the turtle clockwise*	39
Round	*Round to nearest integer*	40
Run	*Execute a list of statements*	40
Say	*Speak using the system voice synthesizer*	40
Sentence	*Create a list*	41
SetBackground, SetBG	*Set the background colour*	41
SetCanvasSize	*Set the window size*	41
SetClipPath	*Set the clipping path*	41
SetFontFace, SetFont	*Set the current font face*	42
SetFontFamily	*Set the current font family*	42
SetFontTraits	*Set the traits for the current font*	42
SetFullScreen	*Enable fullscreen mode*	43
SetHeading	*Set the turtle's heading*	43
SetLineCap	*Set the ending style for lines*	43
SetLineDash	*Set the dash pattern for lines*	44
SetPen	*Set the state of the pen*	44
SetPenColour, SetPenColor, SetPC	*Set the colour for drawing*	44
SetPenWidth	*Set the width of the drawing pen*	45
SetPosition, SetPos	*Set the position of the turtle*	45
SetRGB	*Set a colour's RGB values*	45
SetShadow	*Set the dropshadow for drawing*	45
SetTypeSize	*Set the size for type*	46
SetVoice	*Set the current voice*	46
SetWrap	*Set the wrap state of the turtle*	46
SetX	*Set the x position of the turtle*	47
SetY	*Set the y position of the turtle*	47
Shadow	*Display the shadow state*	47
Show	*Display objects*	47

Shown?	*Output the visibility of the turtle*	48
ShowTurtle, ST	*Show the turtle*	48
Sine, Sin	*Sine*	48
Sinh	*Hyperbolic Sine*	48
Snap	*Capture an animation frame*	48
SqRt	*Square Root*	49
Stop	*Return from a procedure*	49
StrokeCurrentPath	*Stroke the current path*	49
StrokePath	*Stroke a path*	50
Sum	*Addition*	50
Tangent, Tan	*Tangent*	50
Tanh	*Hyperbolic Tangent*	50
Text	*Output a procedure as a list*	51
TextBox	*Output list describing text size*	51
Thing	*Output the value of a variable*	51
Throw	*Throw to a corresponding Catch*	51
Time	*Output the current time*	52
Towards	*Output required heading*	52
Type	*Print object*	52
UpperCase	*Convert to upper case*	52
Voice	*Output the name of the current voice*	53
Voices	*Output the names of available voices*	53
Wait	*Wait for a specified duration*	53
WaitForSpeech	*Wait for speech to finish*	53
Word	*Concatenate words*	53
Word?, WordP	*Test if object is a word*	54
Wrap	*Test if wrap is turned on*	54
XPos	*Output the turtle's x co-ordinate*	54
YPos	*Output the turtle's y co-ordinate*	54

Format

The definition of each command has this format:



+ *Add*

expression1 **+** *expression2*

The addition operator.

```
3 + 2
```

5

- *Subtract*

expression1 **-** *expression2*

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

* *Multiply*

expression1 ***** *expression2*

The multiplication operator.

```
3 * 2
```

6

/ *Divide*

expression1 **/** *expression2*

The division operator.

```
3 / 2
```

1.5

< *Less Than*

expression1 **<** *expression2*

The less than operator. Returns true if *expression1* is less than *expression2*. Returns false otherwise.

1 < 2

true

2 < 1

false

>

Greater Than

expression1* > *expression2

The greater than operator. Returns true if *expression1* is greater than *expression2*, otherwise false.

1 > 2

false

3 > 2

true

ABS

Output the absolute value of a number

ABS *number*

Outputs the absolute value of *number*: if *number* is positive, outputs *number* unchanged; if *number* is negative, outputs *number* negated.

Abs -2

2

Abs 2

2

AND

Logical AND

AND *predicate1 predicate2*

(AND *predicate1 predicate2…)*

Outputs true if all predicates are true — outputs false if any predicate is false.

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

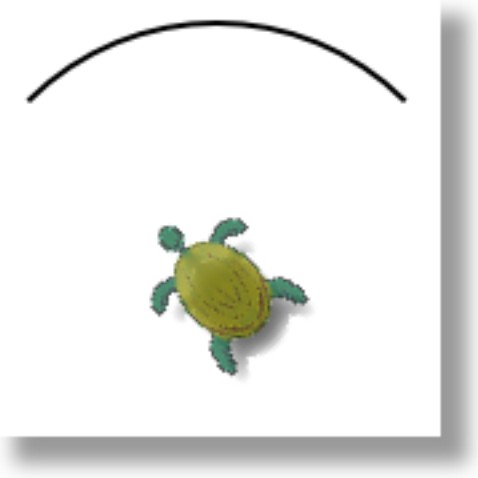
Draw an Arc

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.

```
Left 45
Arc 90 100
```



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

ArcCosine, ArcCos

Output the angle for a Cosine

ArcCosine *number*

Output the angle whose cosine is *number*.

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

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

ArCosh

Inverse of Cosh

ArCosh *number*

The inverse of Cosh.

```
ArCosh 1.32461
45.0001
```

See also ArSinh, ArTanh, Cosh, Sinh, Tanh.

ArcSine, ArcSin

Output the angle for a Sine

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

Output the angle for a Tangent

ArcTangent *number*

Output the angle whose tangent is *number*.

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

See also Cosine, Sine, Tangent, ArcCosine, ArcSine, ATan2.

ArSinh

Inverse of Sinh

ArSinh *number*

Output the angle whose hyperbolic sin is *number*.

```
ArSinh 0.868671
45
```

See also ArCosh, ArTanh, Cosh, Sinh, Tanh.

ArTanh

Inverse of Tanh

ArTanh *number*

Output the angle whose hyperbolic tangent is *number*.

```
ArTanh 0.780714
59.9999
```

See also ArCosh, ArSinh, Cosh, Sinh, Tanh.

ATan2

Inverse of Tan taking two arguments

ATan2 *y x*

The inverse of the tangent, taking two arguments. For a position, [*x,y*], *ATan2 y x* gives the angle of the line drawn from the origin to *x,y*.

```
Home Forward 100
ATan2 Last Position First Position
0
```

```
Home Right 45 Forward 100
ATan2 Last Position First Position
45
Home Right 45 Back 100
ATan2 Last Position First Position
-135
```

See also Tangent, ArcTangent.

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
```

See also Char.

Back

Move the turtle backwards

Back *distance*

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

```
Back 100
```

Opposite of Forward. See also PenUp, Pendown.

Background, Bg

Output the background pen colour

Background

Output the number of the background colour. This is the colour used to fill the graphics window when ClearScreen is called. At startup, the background colour is set to zero. This is initially white, but can be changed with SetRGB.

```
Background
0
```

See also SetBackground, Clean, ClearScreen, RGB, SetRGB, SetPenColour.

ButFirst

Output all but the first element

ButFirst *word*

ButFirst *list*

Output all of a list except for its first element, or all of a word except for its first character.

```
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 of a list except for its last element, or all of a word except for its last character.

ButLast "xyz

xy

ButLast [thing1 thing2 thing3]

[thing1 thing2]

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

Button?, ButtonP

Output whether the left mousebutton is pressed

Button?

Outputs true if the left mouse button is down in the Graphics window, otherwise false.

See also Mouse, GetMouseMoved, GetMouseClick, GetmouseChange.

CanvasSize

Output the size of the canvas

CanvasSize

Outputs the width and height of the canvas (drawing area) as a list.

CanvasSize

[800 800]

See also SetCanvasSize.

Catch

Catch a Throw statement

Catch *name* [*statements*]

Catch executes statements *statements*. If within the scope of these statements (i.e. in the statements or a procedure called by the statements), a condition called *name* is thrown using the Throw statement, condition passes immediately back to the statement following the catch statement.

Make “i 1

Catch “bod [repeat 100 [print :i make “i :i + 1 if :i > 3 [throw “bod] [ ]] print [got to end]] print “done

1

2

3

done

CD

Change the Current Directory

CD *pathname*

Changes the current directory to *pathname*, 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
```

pathname 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
```

See also Pwd, Dir.

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, SetPenColour.

ClearScreen, CS

Clear graphics and home the turtle

ClearScreen

Clear the graphics screen and set the turtle to its home position. The graphics screen is filled with the current background pen colour. The same as doing Clean followed by Home.

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

CloseReadFile

Close the file open for reading

CloseReadFile

Close the file that was opened for reading with OpenRead.

See also CD, Dir, Pwd, OpenRead, OpenTextRead, CloseWriteFile.

CloseWriteFile

Close the file open for writing

CloseWriteFile

Close the file that was opened for writing with OpenWrite or OpenAppend.

See also CD, Dir, Pwd, OpenWrite, OpenTextWrite, OpenAppend, OpenTextAppend, CloseReadFile.

ColourAtPoint, ColorAtPoint

Output the RGB values of a pixel

ColourAtPoint [*x y*]

Output the RGB values of the pixel at position x,y as a list.

```
SetPenWidth 20
SetPenColour 2
Forward 20
ColourAtPoint [0 10]
[1 0 0]
```

See also Dot, RGB, SetRGB.

Cosine, Cos

Output the cosine of an angle

Cosine *angle*

Output the cosine of *angle*.

```
Cosine 60
0.5
```

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

Cosh

Hyperbolic cosine

Cosh *angle*

Output the hyperbolic cosine of *angle*.

```
Cosh 45
1.32461
```

See also ArCosh, ArSinh, ArTanh, Sinh, Tanh.

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, PathLength, SetClipPath.

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?, DefineP

Query existence of a procedure

Define? *name*

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

Define? "box

true

See also Define.

Difference

Subtract two or more numbers

Difference *number1 number2*

(**Difference** *number1 number2…)*

Output the difference between two or more numbers. This is the same as using the minus sign.

```
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.

```
Dot [100 80]
```

DrawImage

Draw an image

DrawImage *file-path* [*width height*]

Draw an image from file *file-path* with width *width* and height *height*.

The file-path parameter can be either a fully qualified path in the file system, or partially qualified relative to the current working directory. See CD and PWD for more information. file-path should generally be enclosed in list brackets, so that slashes and possibly other characters which are part of the file-path are not interpreted as operators by Logo - otherwise, it can be a string.

The second parameter is a list of two numbers, width and height, either or both of which can be absent. If both are present, the image is drawn with the width and height specified. If both are absent, the image is drawn at its native size. If one is absent, the other is inferred from the one specified and the aspect ratio of the image.

The image is drawn at the current turtle position, rotated by the turtle heading.

```
Right 45
```

CD [~/Volumes/Ext500/Pictures]

DrawImage [IMG_0007.JPG] [150]

Empty?, EmptyP

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?.

Eof?, EofP

Test for end-of-file

Eof?

Returns *true* if the file open for reading has been read to the end.

```
OpenRead "myfile.txt
Eof?
false
FReadChars 10000
Eof?
true
```

See also OpenRead, OpenTextRead, FReadChar, FReadChars, FReadList, FReadWord.

Equal?, EqualP

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*

Outputs e to the power of *number*.

```
Exp 1.8
6.04964746441295
```

See also Log.

ExportEPS

Export an EPS

ExportEPS *file-name*

Export the contents of the graphics window as an EPS (Encapsulated Postscript) file to *file-name* in the current directory.The same as choosing **Export/EPS...** from the **File** menu.

```
ExportEPS "thing.eps
```

See also ExportPDF, ExportTiff.

ExportPDF

Export a PDF

ExportPDF *file-name*

Export the contents of the graphics window as an Acrobat PDF to *file-name* in the current directory.The same as choosing **Export/PDF...** from the **File** menu.

```
ExportPDF "thing.pdf
```

See also ExportEPS, ExportTiff.

ExportTIFF

Export a TIFF

ExportTIFF *file-name*

Export the contents of the graphics window as a TIFF bitmap file to *file-name* in the current directory.The same as choosing **Export/Graphics...** from the **File** menu.

```
ExportTIFF "thing.tif
```

See also ExportPDF, ExportEPS.

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, FillCurrentPath, 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, FillCurrentPath, 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, CurrentPath, StrokePath, FillPath, StrokeCurrentPath, ReversePath, PathBounds, PathLength.

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, CurrentPath, StrokePath, StrokeCurrentPath, FillCurrentPath, ReversePath, PathBounds, PathLength.

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 *object word*

FirstPut *object list*

Add *object* 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 names of available fonts

FontFaces

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

```
FontFaces
[[LiSungLight][Bodoni-BoldItalic][ACaslonPro-Regular][CalistoMTItalic][LubalinGraph-Demi]
[TimesCYItalic][AGaramondPro-Italic][FootlightMTLight][LucidaSans-Italic][DFLeiSho-SB-
MP-RKSJH][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 names 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 StdAcro][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 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

Reads 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 output 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

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

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 the 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, OpenTextWrite, 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 *object* to the file currently open for writing, then starts a new line. if *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, OpenTextWrite, CloseWriteFile, FPrint, FShow, Type.

GetMouseChange

Wait for the mouse button or a mouse move

GetMouseChange

Wait until the left mouse button is pressed in the graphic window, or the mouse is moved with the graphic window active, then output a list containing the mouse co-ordinates and whether the left mouse button is down.

```
GetMouseChange
[[-175 195]false]
```

See also Button?, Mouse, GetMouseMoved, GetMouseClick.

GetMouseClick

Wait for the left mouse button to be pressed

GetMouseClick

Wait for the left mouse button to be pressed, then output the co-ordinates of the mouse as a list.

```
GetMouseClick
[20 -30]
```

See also Button?, Mouse, GetMouseMoved, GetMouseChange.

GetMouseMoved

Wait for the mouse to be moved

GetMouseMoved

Wait for the left mouse to be moved while the graphics window is active, then output the co-ordinates of the mouse as a list.

```
GetMouseMoved
[20 -30]
```

See also Button?, Mouse, GetMouseClick, GetMouseChange.

GetProp, GProp

Retrieve a property for a name

GetProp *name property*

Retrieve a property for a name which has been previously assigned with PutProp. If there is no such property, the empty list is returned.

```
PutProp "fred "address [5 Letsby Avenue]
PutProp "fred "age 47
GetProp "fred "address
[5 Letsby Avenue]
GetProp "fred "age
47
GetProp "fred "height
[ ]
```

See also PutProp, PropList, RemProp.

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.

```
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
```

Instructions

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

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 nth 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*

Outputs 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?, ListP

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, Name?.

Member?, MemberP

Test membership 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?.

Mouse

Output mouse co-ordinates

Mouse

Output the co-ordinates of the mouse as a list.

```
Mouse
[0 30]
```

See also Button?, GetMouseMoved, GetMouseClick, GetMouseChange.

Name?, NameP

Test existence of a variable

Name? *word*

Output *true* if *word* is the name of a variable, otherwise *false*.

```
Name? "thething
false
Make "thething 5
Name? "thething
true
```

See also Make, Thing.

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?, NumberP

Test whether an object is 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 Pwd, CD, Dir, 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 reads single bytes and interpret them as characters — if the file is a unicode text file, this may not give the result you want — then use OpenTextRead instead.

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

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

OpenTextAppend

Open a text file for appending to

OpenTextAppend *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. Characters are written to the file as unicode (multiple byte) characters. 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.

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

See also Pwd, CD, Dir, OpenTextRead, OpenTextWrite, CloseWriteFile, FPrint, FShow, FType.

OpenTextRead

Open a text file for reading

OpenTextRead *file-name*

Open file *file-name* in the current directory for reading. OpenTextRead differs from OpenRead in that it will attempt to work out if the file is a unicode file. If it is, the characters will be interpreted correctly as unicode characters.

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.

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

See also Pwd, CD, Dir, OpenRead, OpenTextAppend, OpenTextWrite, CloseReadFile, FReadChar, FReadChars, FReadList, FReadWord.

OpenTextWrite

Open a unicode file for writing to

OpenTextWrite *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. Characters are written to the file as unicode (multiple byte) characters. 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.

```
OpenTextWrite [thing.txt]
```

FPrint [Bit at the start]

FPrint [Bit in the middle]

CloseWriteFile

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

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 Pwd, CD, Dir, 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, StrokePath, FillPath, StrokeCurrentPath, FillCurrentPath, ReversePath,

PathLength.

PathLength

Output the length of a path

PathLength *path-list*

Outputs the length 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
PathLength CurrentPath
155
```

See also CurrentPath, StrokePath, FillPath, StrokeCurrentPath, FillCurrentPath, 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 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

Raise a number to a 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.

PropList, PList

List properties for a name

PropList *name*

Output a list of the properties and values associated with *name*. The list is an alternating list of properties and their values, and the order of the properties in the list is undefined.

```
PutProp "fred "address [5 Letsby Avenue]
PutProp "fred "age 47
PropList "fred
[ age 47 address [5 Letsby Avenue]]
```

See also GetProp, PutProp, RemProp.

PutProp, PProp

Set a property for a name

PutProp *name property object*

Create property *property* for name *name* and give it a value of *object*. A number of different properties can be assigned to *name*. If *name* already has property *property*, the old value is replaced with *object*.

```
PutProp "fred "address [5 Letsby Avenue]
PutProp "fred "age 47
GetProp "fred "address
[5 Letsby Avenue]
```

See also GetProp, PropList, RemProp.

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 multiple 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 from division

Remainder *number1 number2*

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

Remainder 25 20

5

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

RemProp

Remove a property

RemProp *name property*

Remove a property for a name which has previously been assigned with PutProp.

PutProp "fred "age 47

GetProp "fred "age

47

RemProp "fred "age

GetProp "fred "age

[ ]

See also PutProp, PropList, GetProp.

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, StrokePath, FillPath, StrokeCurrentPath, FillCurrentPath, PathBounds, PathLength.

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, ColourAtPoint.

Right

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 *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.

SetCanvasSize

Set the window size

SetCanvasSize [*width height*]

Set the size of the canvas on which the turtle draws, effectively changing the size of the graphics window. The same as choosing **Canvas Size...** from the **Format** menu.

```
SetCanvasSize [400 400]
```

SetClipPath

Set the clipping path

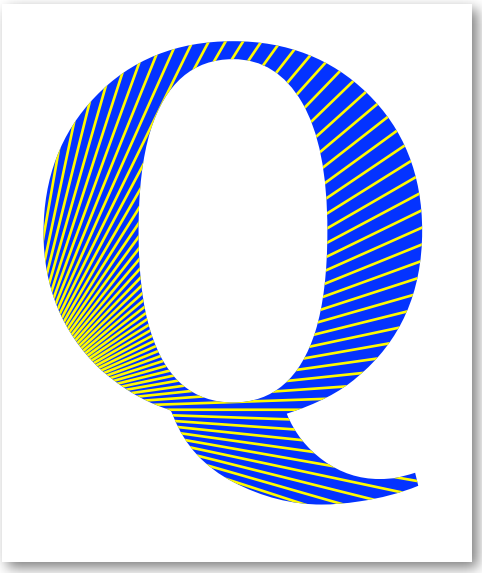
SetClipPath *path-command-list*

Confine all drawing to the path defined by *path-command-list*. The canvas will not be affected by any drawing which takes place outside that area.

The best way to create the list is to do some drawing, and then save the path into a list using CurrentPath. See that command for more on the subject. Setting the Clipping Path to the empty list resets the Clipping Path so that drawing is not clipped. ClearScreen also resets the Clipping Path.

```
ClearScreen
```

```
SetTypeSize 200
SetFont [Times-Bold]
GrType "Q
Make "ps CurrentPath
SetPC 3
FillPath :ps
SetPC 4
SetClipPath :ps
Make "angle 0
Repeat 60 [Home Right :angle Fd 300
        Make "angle :angle + 2]
SetClipPath [ ]
```



See also CurrentPath, StrokePath, StrokeCurrentPath, FillPath, FillCurrentPath, ReversePath, PathBounds, PathLength.

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 FontFamily.

```
SetFontFamily [Baskerville]
```

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

SetFontTraits

Set the traits for 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, FontTraits, SetFontFamily.

SetFullScreen

Enable fullscreen mode

SetFullScreen *boolean-value*

Set the graphics window to fullscreen (true) or normal mode (false).

SetFullScreen "true

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

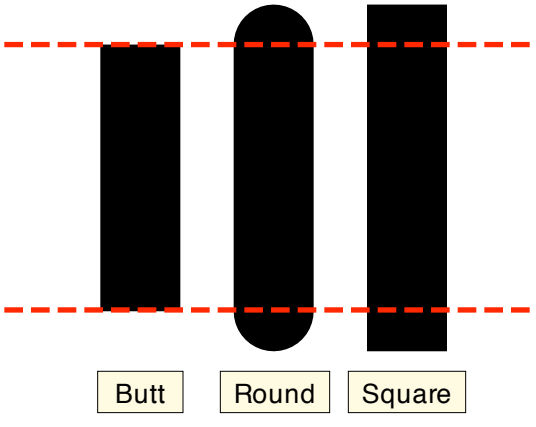
See also Heading, Left, Right.

SetLineCap

Set the ending style for lines

SetLineCap *line-end-style*

Sets the end style for lines to *line-end-style*, which can be *butt* (the default), *round*, or *square*. The following diagram shows how the line ending varies for each of the options. The red dashed lines show where the lines end.



Note that in the case of *round* and *square*, the line endings extend beyond the end of the line. These effects are only noticable for thick lines.

```
SetPenWidth 25
SetLineCap "round
Forward 100
```

See also SetPenWidth, Forward, SetLineDash.

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, SetLineCap, 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 colour for drawing

SetPenColour *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.

```
SetPenColour 3
PenColour
3
```

See also Pen, SetPen, PenColour, 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 RGB, ClearScreen, ColourAtPoint.

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.

```
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).

```
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.

SetWrap

Set the wrap state of the turtle

SetWrap "truel"false

Sets the wrap state of the turtle, i.e., whether the turtle is able to leave the canvas area. If wrap is true, when the turtle goes out one side of the canvas, it comes in the other.

```
SetWrap "true
Position
[0  0]
Forward 300
Position
[0  300]
Forward 300
Position
[0  -200]
```

See also Wrap, Position.

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.

Shadow

Display the shadow state

Shadow

Displays the x offset, y offset, and blur radius of the current dropshadow setting used for drawing. Returns the empty list if shadows are turned off.

```
Shadow
[10 10 5]
```

See also SetShadow.

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.

Sinh

Hyperbolic Sine

Sinh *angle*

Output the hyperbolic sine of *angle*.

```
Sinh 45
0.868671
```

See also ArCosh, ArSinh, ArTanh, Cosh, Tanh.

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 CurrentPath, StrokePath, FillPath, FillCurrentPath, ReversePath, PathBounds, PathLength.

StrokePath

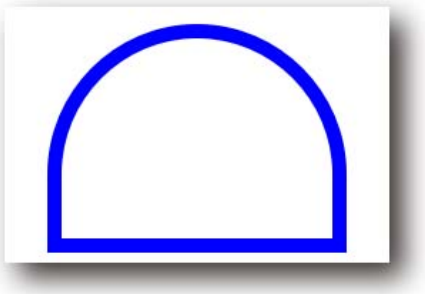
Stroke a path

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 CurrentPath, FillPath, StrokeCurrentPath, FillCurrentPath, ReversePath, PathBounds, PathLength.

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, ATan2.

Tanh

Hyperbolic Tangent

Tanh *angle*

Output the hyperbolic tangent of *angle*.

```
Tanh 60
0.780714
```

See also ArCosh, ArSinh, ArTanh, Cosh, Sinh.

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 the 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, Name?.

Throw

Throw to a corresponding Catch

Throw *name*

The purpose of Throw is to pass control back to an encompassing Catch statement.

```
Make "i 1
Catch "bod [repeat 100 [print :i make "i :i + 1 if :i > 3 [throw "bod] [ ]] print [got to end]] print "done
1
```

2

3

done

See also Catch.

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 without starting a new line. Removes 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.Albert]
[com.apple.speech.synthesis.voice.BadNews] [com.apple.speech.synthesis.voice.Bahh]
[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?, WordP

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.

Wrap

Test if wrap is turned on

Wrap

Output **true** if *wrap is on*.

```
Wrap
false
```

See also SetWrap.

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