StarLogo Complete Command List

(Edited and reformatted by Nicholas Gessler, 6 June 2001.)

Symbols [Observer, Turtle]

`number1` +, -, *, / `number2`
Basic math functions. Be sure to put a space between the numbers and the symbol.

`number1` ^ `number2`
Power function. Number1 ^ Number2 is equal to Number1 to the power of Number2

Symbols [Observer, Turtle]

`number1` =, !=, <, >, <=, >= `number2`
Equivalency operators. Again, a space is needed between the numbers and the symbol.

; [Observer, Turtle]
`;` is the comment character. Placing `;` either on a line by itself or at the end of a line in a procedure causes StarLogo to ignore everything it see until it reaches the next carriage return.

abs `number` [Observer, Turtle]
Reports the absolute value of `number`.

age [Turtle]
Returns the value of the caller's `age` variable.

age-at `xcor ycor` [Observer, Turtle]
Returns the value of the variable `age` of the turtles at `xcor ycor` relative to the caller.

age-of `turtle id number` [Observer, Turtle]
Returns the value of the variable `age` of the turtle with who number `turtle id number`.

age-towards `angle distance` [Observer, Turtle]
Returns a list of all the turtles and the values of their `age` variables which are at the patch `distance` away at angle `angle`.

`condition1` and `condition2` [Observer, Turtle]
Reports true if `condition1` and `condition2` report true.

ask-frogs `ilist` [Observer]
Asks all turtles of breed `frogs` to run `ilist`. The observer will wait for all of the turtles to finish before continuing.

ask-list-of-turtles `list-of-turtle-ids ilist` [Observer, Turtle]
Turtles whose `who` numbers (ID numbers) are in `list-of-turtle-ids` run `ilist`. The caller of this command waits for it to finish before continuing. If some invalid `who` numbers make up `list-of-turtle-ids`, they are ignored.

ask-patches `ilist` [Observer]
This observer command asks all of the patches to run the `ilist`. The observer will wait for the patches to finish before moving on.

ask-patch-at `xcor ycor ilist` [Observer, Turtle]
This command asks the patch which is `xcor` units in the x direction and `ycor` units in the y direction away from the caller to run the `ilist`. The caller of this command will wait for the patch to finish before moving on. The observer is considered to be located at 0 0.

ask-turtle `number ilist` [Observer]
Asks turtle with ID number `number` to perform `ilist`. The observer will wait for all of the turtles to finish before continuing.

ask-turtles `ilist` [Observer]
Asks all turtles to run `ilist`. The observer will wait for all of the turtles to finish before continuing.

atan `numerator denominator` [Observer, Turtle]
Trigonometry function. Returns the arctangent of the specified angle `numerator`/`denominator`. All angles are in degrees

bf (butfirst) `dlist` [Observer, Turtle]
Returns the value of `dlist` with the first item removed.

`condition1` bitand `condition2` [Observer, Turtle]
Reports the value of bitwise and on the inputs.

bitnot `condition1` [Observer, Turtle]
Reports the bitwise negation of its input.

`condition1` bitor `condition2` [Observer, Turtle]
Reports the value of bitwise or on the inputs.

`input1` bitxor `input2` [Observer, Turtle]
Reports the value of bitwise xor on the inputs.

bk, back `number` [Turtle]
Turtles move `number` steps backward

bl (butlast) `dlist` [Observer, Turtle]
Returns the value of `dlist` with the last item removed.

black `color` [Observer, Turtle]

`black` reports the number of its particular hue in the color table.
Its reference number is 0.

blue `color` [Observer, Turtle]

`blue` reports the number of its particular hue in the color table.
Its reference number is 105.

breed [Turtle]
Returns the turtle's breed.

breed-at `xcor ycor` [Observer, Turtle]
Reports the breed of the turtle `xcor` units in the x direction and `ycor` units in the y direction away from the caller.

breed-of `number` [Observer, Turtle]
Reports the breed of the turtle with ID Number `number`.

breed-towards `angle distance` [Observer, Turtle]
Reports the breed of the turtle `distance` away at an angle of `angle`.

brown `color` [Observer, Turtle]

`brown` reports the number of its particular hue in the color table.
Its reference number is 35.

ca, clearall [Observer]
Kills all turtles, sets all patches to black, and resets all patch variables.

case `variable [anything1 ilist1 anything2 ilist2]` [Observer, Turtle]
Checks if the first argument `variable` is equal to any of the `anythings` in the list. If it is equal, the corresponding `ilist` is executed and control skips to the end of the `case` statement. If nothing matches, no `ilists` are executed and control skips to the end of the `case` statement.
Sample usage:

```
case temperature
[
"cold" [show "boy, it's cold!"]
"warm" [show "mmmm. nice."]
```

```
"hot" ["wow, it's hot!"]
]
```

cc, clear-command-center [Observer, Turtle]
Clears the command center.

cg, cleargraphics [Observer]
Sets all patches to black

clearinfo, clear-info [Observer]
Clears the Information Window.

clearplot, clear-plot [Observer, Turtle]
Clears everything drawn by all plot pens, and resets all pens to (0, 0).

co, clear-output [Observer]
Clears all the text in the output window.

color [Turtle]
Returns the color of the turtle.

color-at `xcor ycor` [Observer, Turtle]
Reports the color of the turtle `xcor` units in the x direction and `ycor` units in the y direction away from the caller .

color-of `number` [Observer, Turtle]
Reports the color of the turtle with ID Number `number`.

color-towards `angle distance` [Observer, Turtle]
Reports the color of the turtle `distance` away at an angle of `angle`.

cos `number` [Observer, Turtle]
Trigonometry function. Returns the cosine of the specified angle. All angles are in degrees.

count-color `color` [Observer, Turtle]
Returns the number of turtles that are the color `color`.

count-frogs [Observer, Turtle]
Returns the number of turtles whose `breed` is frogs.

count-frogs-at `xcor ycor` [Observer, Turtle]
Returns the number of turtles whose `breed` is frogs which are `xcor` units in the x direction and `ycor` units in the y direction away from the caller .

count-frogs-here [Turtle]
Returns the number of turtles whose `breed` is frogs which are on the current patch.

count-frogs-towards `angle distance` [Observer, Turtle]
Returns the number of turtles whose `breed` is frogs which are located at the patch `distance` away at angle `angle`.
count-frogs-with `[ilist]` [Observer, Turtle]
Returns the number of turtles whose `breed` is frogs and satisfy the conditions (must return a boolean of true or false) specified by the `ilist`.
Sample usage:
```
count-frogs-with [color = blue]
```

count-patches-with `ilist` [Observer, Turtle]
Returns the number of patches whose `ilist` is true.
Sample usage:
```
show count-patches-with [(distance 0 0) < 5]
```
This gives the number of patches inside a circle of radius 5 centered at the origin.

count-pc `color` [Observer, Turtle]
Returns the number of patches that are the color `color`.

count-turtles [Observer, Turtle]
Returns the number of turtles.

count-turtles-at `xcor ycor` [Observer, Turtle]
Returns the number of turtles `xcor` units in the x direction and `ycor` units in the y direction away from the caller .

count-turtles-here [Turtle]
Returns the number of turtles sitting on the caller's patch

count-turtles-towards `angle distance` [Observer, Turtle]
Returns the number of turtles `distance` away at angle `angle`.

count-turtles-with `[ilist]` [Observer, Turtle]
Returns the number of turtles whose `ilist` is true.
Sample usage: `show count-turtles-with [(distance 0 0) < 5]`
This gives the number of turtles inside a circle of radius 5 centered at the origin.

cp, clearpatches [Observer]
Sets all patches to black and resets all patch variables.

create-and-do, create-turtles-and do `number ilist` [Observer]
Creates `number` turtles and tells them to do `ilist`.

create-frogs `number` [Observer]
Creates `number` of turtles and assigns frog as their `breed`.

create-frogs-and-do `number list to run` [Observer]
Creates `number` of turtles and assigns frogs as their `breed`. The turtles then execute `list to run`.

crt, create-turtles `number` [Observer]
Creates `number` turtles.

ct, clear-turtles [Observer]
Clears (kills) all of the turtles.

cyan `color` [Observer, Turtle]
`cyan` reports the number of its particular hue in the color table.
Its reference number is 85.

die [Turtle]
Turtles die, meaning that they stop running all code and disappear forever.

diffuse `variable percentage` [Observer]
Makes each patch give 1/8 of `percentage` of `variable` to each neighboring patch.
Note:percentage should be expressed as a number between 0 and 1.

diffuse4 `variable percentage` [Observer]
Makes each patch give 1/4 of `percentage` of `variable` to each neighboring patch to the N, S, E and W.
Note:percentage should be expressed as a number between 0 and 1.

distance `xcor ycor` [Observer, Turtle]
Returns the distance from the caller to (`xcor`,`ycor`).

distance-nowrap `xcor ycor` [Observer, Turtle]
Returns the distance from the caller to `xcor ycor` without wrapping.

`number1` div `number2` [Observer, Turtle]
Returns the integer part of the answer to `number1 / number2`.

dlist [Observer, Turtle]
A list of data elements. This is not a command to type into StarLogo, rather, it is a

convention used in this documentation. See the `list` command.

dx [Turtle]
Returns the x-coordinate one step forward from the turtle's current position.

dy [Turtle]
Returns the y-coordinate one step forward from the turtle's current position.

e [Observer, Turtle]
Returns the value of e.

empty? `dlist` [Observer, Turtle]
Returns true if `dlist` is empty.

end [Observer, Turtle]
This command must be placed at the end of every procedure.

every `number list to run` [Observer, Turtle]
`every` is just like `loop`, in that it runs its `list to run` forever, but it waits `number` seconds between each iteration.

exp `number` [Observer, Turtle]
e raised to the power of `number`.

fd, forward `number` [Turtle]
Turtles move `number` steps forward.

first `dlist` [Observer, Turtle]
Returns the first item of `dlist`.

food [Observer, Turtle]
Returns the value of the patch variable `food`.
NOTE: This is a patch command. It must be called from within an `ask-patches` statement.

food-at `xcor ycor` [Observer, Turtle]
Returns the value of the variable `food` of the patch at `xcor ycor` relative to the caller.

food-towards `angle distance` [Observer, Turtle]
Returns the value of the`food` variable of the patch `distance` away at angle `angle`.

fput `item dlist` [Observer, Turtle]
Returns the value of `dlist` with `item` as its first element.

get-random-seed [Observer, Turtle]
Returns the current random seed.

globals `[variable-list]` [Observer]
Creates global variables. The `variable-list` contains one or more names.

grab `number ilist` [Observer, Turtle]
Caller instructs turtles with `who` number(s) `number(s)` to execute `ilist`. The `who` number of the turtles being `grab`bed are stored in `partner`, if there is one, or `partners`, if there are many. A turtle cannot `grab` itself.
Example: `grab one-of-turtles-here [setc red setc-of partner blue]`
This turns the caller red and the `grab`bed turtle blue. If there are no other turtles on the caller's patch, the `ilist` does not get executed and no turtles change color.
gray, grey `color` [Observer, Turtle]
`gray` reports the number of its particular hue in the color table.
Its reference number is 5.

green `color` [Observer, Turtle]
`green` reports the number of its particular hue in the color table.
Its reference number is 55.

hatch `ilist` [Turtle]
Turtles make exact copy of themselves, tell them to run `ilist`.
Note: Turtles will not run forever buttons while running `ilist`.

heading [Turtle]
Returns the direction that the turtle is facing.

heading-at `xcor ycor` [Observer, Turtle]
Reports the heading of the turtle `xcor` units in the x direction and `ycor` units in the y direction away from the caller.

heading-of `number` [Observer, Turtle]
Reports the heading of the turtle with ID Number `number`.

heading-towards `angle distance` [Observer, Turtle]
Reports the heading of the turtle `distance` away at an angle of `angle`.

home [Turtle]
Turtles go to coordinates (0,0). This is the same as using setxy 0 0.

ht, hideturtle [Turtle]
Turtles make themselves invisible.

if `condition ilist` [Observer, Turtle]
Do `ilist` if `condition` reports true.

if breed = frogs [Turtle]
Returns `true` if the `breed` of the turtle invoking this command is frogs.

ifelse `condition ilist1 ilist2` [Observer, Turtle]
Do `ilist1` if `condition` reports true, otherwise do `ilist2`.

ignore `whatever` [Observer, Turtle]
Ignores the statement immediately following it.

ilist [Observer, Turtle]
An instruction list: any list of valid instructions for whichever character is going to be executing them. This is not a command to type into StarLogo, rather, it is a convention used in this documentation.
Example: `fd 2 rt 90 fd 6` is a valid `ilist` for a turtle.

import-picture [Observer, Turtle]
Opens up a dialog box asking you to select a picture to import onto the Graphics Canvas. You must have a current version of QuickTime installed on your computer in order for this command to work.

import-picture-name `string` [Observer, Turtle]
Imports onto the Graphics Canvas the picture in the file specified by `string`. You must have a current version of QuickTime installed on your computer in order for this command to work.

info-name [Observer]
Returns the current filename of the information window. Returns false if no filename has been set yet.

inspect-turtle `number` [Observer]
Brings up the turtle monitor of the turtle whose who number is `number`.

int `number` [Observer, Turtle]
Reports the largest integer less than or equal to `number`.

item `number list` [Observer, Turtle]
Returns the element of `list` at the `number`'th position.

jump `number` [Turtle]
Turtles move `number` steps in the time it takes to make one normal step. For example, jump 15 and fd 1 both take the same amount of time to perform. fd 15, however, would take 15 times as long as jump 15.

kill `number` [Observer, Turtle]
Kills turtle with ID number `number`.

last `dlist` [Observer, Turtle]
Returns the last item of `dlist`.

leap `number` [Turtle]
Turtles jump `number` steps only if no other turtle is currently on the patch they will land on.

length `dlist` [Observer, Turtle]
Returns the length of `dlist`.
let `variable value` [Observer, Turtle]
Declares `variable` as a local variable and assigns `value` to it. All variables must have names which begin with a colon. See the Variables page for more information on local variables.
Example:
let [:myvar 6]

lime `color` [Observer, Turtle]
`lime` reports the number of its particular hue in the color table.
Its reference number is 65.

list `item1 item2` [Observer, Turtle]
Returns a new dlist with `item1` as its first element, and `item2` as its second.

list? `thing` [Observer, Turtle]
Returns true if `thing` is a list.

list-of-frogs [Observer, Turtle]
Returns a list of turtles of breed `frogs`.

list-of-frogs-at `xcor ycor` [Observer, Turtle]
Returns a list of turtles of breed `frogs` `xcor` units in the x direction and `ycor` units in the y direction away from the caller.

list-of-frogs-here [Turtle]

Returns a list of turtles of breeds `frogs` on the caller's patch.

list-of-frogs-towards `angle distance` [Observer, Turtle]
Returns a list of turtles of breed `frogs` at the patch `distance` away at angle `angle`.

list-of-frogs-with `condition` [Observer, Turtle]
Returns a list of turtles of breed `frogs` satisfying `condition`.

list-of-turtles [Observer, Turtle]
Returns a list of turtles.

list-of-turtles-at `xcor ycor` [Observer, Turtle]
Returns a list of turtles `xcor` units in the x direction and `ycor` units in the y direction away from the caller.

list-of-turtles-here [Turtle]
Returns a list of turtles on the caller's patch.

list-of-turtles-towards `angle distance` [Observer, Turtle]
Returns a list of turtles at the patch `distance` away at angle `angle`.

list-of-turtles-with condition [Observer, Turtle]
Returns a list of turtles satisfying `condition`.

ln `number` [Observer, Turtle]
Natural logarithm of `number`.

loop `ilist` [Observer, Turtle]
Do `ilist` forever.

lput `item dlist` [Observer, Turtle]
Returns the value of `dlist` with `item` as its last element.

lt, left `number` [Turtle]
Turtles turn left by `number` degrees.

magenta `color` [Observer, Turtle]
`magenta` reports the number of its particular hue in the color table.
Its reference number is 125.

max `number1 number2` [Observer, Turtle]
Reports the larger value of the two numbers.

maxnum [Observer, Turtle]
Returns the largest number possible without going into positive infinity.

max-of-frogs-with `pred-ilist ilist` [Observer, Turtle]
Reports the highest value of `ilist` when run over the turtles of breed `frogs` that satisfy `pred-ilist`. If there are no `frogs` in which `pred-ilist` is true, returns `minnum`, the smallest number possible without going into negative infinity. Note: This command can also be executed by patches, for example within an `ask-patches` statement.

max-of-patches-with `pred-ilist ilist` [Observer, Turtle]
Reports the highest value of `ilist` when run over the turtles that statisfy `pred-ilist`. If there are no patches in which `pred-ilist` is true, reports `minnum`, the smallest number possible without going into negative infinity. Note: This command can also be executed by patches, for example within an `ask-patches` statement.

max-of-turtles-with `pred-ilist ilist` [Observer, Turtle]
Reports the highest value of `ilist` when run over the turtles that satisfy `pred-ilist`. If there are no turtles in which `pred-ilist` is true, returns `minnum`, the smallest number possible without going into negative infinity. Note: This command can also be executed by patches, for example within an `ask-patches` statement.

member? `item dlist` [Observer, Turtle]
Returns true if `item` is a member of `dlist`.

min `number1 number2` [Observer, Turtle]
Reports the smaller value of the two numbers.

minnum [Observer, Turtle]
Returns the smallest number possible without going into negative infinity.

min-of-frogs-with `pred-ilist ilist` [Observer, Turtle]
Reports the lowest value of `ilist` when run over the turtles of breed `frogs` that satisfy `pred-ilist`. If there are no `frogs` in which `pred-ilist` is true, returns `maxnum`, the largest number possible without going into positive infinity. Note: This command can also be executed by patches, for example within an `ask-patches` statement.

min-of-patches-with `pred-ilist ilist` [Observer, Turtle]
Reports the lowest value of `ilist` when run over the turtles that statisfy `pred-ilist`. If there are no patches in which `pred-ilist` is true, reports `maxnum`, the largest number possible without going into positive infinity. Note: This command can also be executed by patches, for example within an `ask-patches` statement.

min-of-turtles-with `pred-ilist ilist` [Observer, Turtle]

Reports the lowest value of `ilist` when run over the turtles that satisfy `pred-ilist`. If there are no turtles in which `pred-ilist` is true, returns `maxnum`, the largest number possible without going into positive infinity. Note: This command can also be executed by patches, for example within an `ask-patches` statement.

`number1` mod `number2` [Observer, Turtle]
Modulo function. Number1 mod Number2 is equal to the remainder when Number1 is divided by Number2. The answer to mod is always positive.

mouse-xcor [Observer, Turtle]
Returns the value of the xcor where the mouse was most recently.
Example: run this code in a forever button:
```
seth towards mouse-xcor mouse-ycor fd .5
```
Turtles will follow the mouse (pointer tool) around the graphics canvas.

mouse-ycor [Observer, Turtle]
Returns the value of the ycor where the mouse was most recently.
Example: run this code in a forever button:
```
seth towards mouse-xcor mouse-ycor fd .5
```
Turtles will follow the mouse (pointer tool) around the graphics canvas.

myself [Turtle]
During a `count-turtles-with` command, `myself` reports the who number of the turtle which called it.
Example: `count-turtles-with [(distance xcor-of myself ycor-of myself) < 5]` counts all turtles within a radius of 5 units of the caller.

nobody [Observer, Turtle]
An alias for -1

not `condition1` [Observer, Turtle]
Reports true if `condition1` reports false.

nsum `variable1 variable2` [Observer]
For each patch, takes the sum of `variable1` from all neighboring patches and places it in `variable2`

nsum4 `variable1 variable2` [Observer]
For each patch, takes the sum of `variable1` from the non-diagonal (N,E,S,W) neighboring patches and places it in `variable2`

number? `thing` [Observer, Turtle]
Returns true if `thing` is a number.

one-of-frogs [Observer, Turtle]
Returns a random turtle of breed `frogs`.

one-of-frogs-at `xcor ycor` [Observer, Turtle]
Returns a random turtle of breed `frogs` `xcor` units in the x direction and `ycor` units in the y direction away from the caller.

one-of-frogs-here [Turtle]
Returns a random turtle of breed `frogs` on the caller's patch other than the caller.

one-of-frogs-towards `angle distance` [Observer, Turtle]
Returns a random turtle of breed `frogs` at the patch `distance` away at angle `angle`.

one-of-turtles [Observer, Turtle]
Returns a random turtle.

one-of-turtles-at `xcor ycor` [Observer, Turtle]
Returns a random turtle `xcor` units in the x direction and `ycor` units in the y direction away from the caller.

one-of-turtles-here [Turtle]
Returns a random turtle on the caller's patch other than the caller.

one-of-turtles-towards `angle distance` [Observer, Turtle]
Returns a random turtle at the patch `distance` away at angle `angle`.

`condition1` or `condition2` [Observer, Turtle]
Reports true if either `condition1` or `condition2` reports true.

orange `color` [Observer, Turtle]
`orange` reports the number of its particular hue in the color table.
Its reference number is 25.

output `something` [Observer, Turtle]
Exits the current procedure and returns `something`

output-name [Observer]
Returns the current filename of the output window. Returns false if no filename has been set yet.

partner [Observer, Turtle]
Returns the ID number of the turtle being grabbed, or -1 if no turtle is being grabbed.

partners [Observer, Turtle]
Returns a dlist of the turtles being grabbed, or [ ] if no turtles are being grabbed.

patches-own `[variable list]` [Observer, Turtle]
Defines a set of variables to be properties of patches. The `variable list` contains one or more names.

pc, patchcolor [Turtle]
Reports the color of the patch the turtle is on.
Please see the color reference for more info on colors.

pc-ahead [Turtle]
Reports the color of the patch one space ahead in the direction that the turtle is facing.
Please see the color reference for more info on colors.

pc-at `xcor ycor` [Observer, Turtle]
Reports the color of the patch `xcor` units in the x direction and `ycor` units in the y direction away from the caller.
Please see the color reference for more info on colors.

pc-towards `angle distance` [Observer, Turtle]
Reports the color of the patch `distance` away at a relative heading of `angle`.
Please see the color reference for more info on colors.

pd, pendown [Turtle]
Turtles put down their "pens," meaning that they draw when they move, leaving a trail behind them. The color of the pen is that of the turtle. Note: Only `fd` and `bk` will draw a line.

pendown? [Turtle]
Returns true if the turtle's pen is down, otherwise false.

pendown?-at `xcor ycor` [Observer, Turtle]
Reports whether the pen of the turtle `xcor` units in the x direction and `ycor` units in the y direction away from the caller is down.

pendown?-of `number` [Observer, Turtle]
Reports whether the pen of the turtle with ID Number `number` is down.

pendown?-towards `angle distance` [Observer, Turtle]
Reports whether the pen of a turtle `distance` away at an angle of `angle` is down.

pi [Observer, Turtle]
Returns the value of pi.

pick `list` [Observer, Turtle]
Returns a random element of the `list`. Returns `false` if the list is empty.

pink `color` [Observer, Turtle]
`pink` reports the number of its particular hue in the color table.
Its reference number is 135.

play-sound `string_of_sound_file` [Observer, Turtle]
Plays the .au sound specified in `string_of_sound_file`. Note: The sound file must be in the Sounds folder, and the string should not contain the .au extension.

plot `y-value` [Observer, Turtle]
Plots a point at `y-value`, then increments the `x-value` of the plot pen by 1. The pen will rescale itself when it reaches `plot-xmax`.

plot-title [Observer, Turtle]
Returns the title of the plot.

plot-xlabel [Observer, Turtle]
Returns the value of the label of the x-axis of the graph.

plot-xmax [Observer, Turtle]
Returns the maximum x-value of the plot.

plot-xmin [Observer, Turtle]
Returns the minimum x-value of the plot.

plot-ylabel [Observer, Turtle]
Returns the value of the label of the y-axis of the graph.

plot-ymax [Observer, Turtle]
Returns the maximum y-value of the plot.

plot-ymin [Observer, Turtle]
Returns the minimum y-value of the plot.

position `item dlist` [Observer, Turtle]
Returns the position of `item` in the `dlist`.

pp `integer` [Observer, Turtle]

Selects a plot pen (e.g. `pp 1`). Subsequent plotting commands are executed by the selected plot pen. There are ten plot pens, numbered 1 through 10.

ppc [Observer, Turtle]
Returns the current plot pen color.

ppd [Observer, Turtle]
Puts the plot pen down, so that it draws connected graphs.

ppreset [Observer, Turtle]
Clears everything this plot pen has drawn and resets its position to (0, 0).

ppu [Observer, Turtle]
Lifts up the plot pen, so that it plots points individually (not connected).

ppxcor [Observer, Turtle]
Returns the x-value of the current plot pen.

ppycor [Observer, Turtle]
Returns the y-value of the current plot pen.

pred-list [Observer, Turtle]
Any group of commands which can return a boolean value. This is not a command to type into StarLogo, rather, it is a convention used in this documentation.
Example: `sum-of-turtles-with (color = red) [age]` Here, `color = red` is the `pred-ilist`.

print `text or variables` [Observer]
Prints the `text or variables` in the output window followed by a carriage return.

project-name [Observer]
Returns the current filename of the project. Returns false if no filename has been set yet.

pstamp `color` [Observer]
Allows patches to set the color of the turtle that is on it to `color`.

pstamp-at `xcor ycor color` [Observer]
Allows patches to set the color of the turtle `xcor` patches in the x-direction and `ycor` patches in the y-direction away to `color`.

pstamp-towards `radius angle color` [Observer]
Allows patches to set the color of the turtle `radius` units away at an angle of `angle` to `color`.

pu, penup [Turtle]
Turtles pick up their "pens," meaning that they no longer draw when they move.

random `number` [Observer, Turtle]
Returns a random number between 0 and `number`, including 0 but not `number`, based on a uniform distribution.

random-gaussian `number` [Observer, Turtle]
Returns a random number with mean 0 and standard deviation `number`.

red `color` [Observer, Turtle]
`red` reports the number of its particular hue in the color table.
Its reference number is 15.

repeat `number ilist` [Observer, Turtle]
Do `ilist` `number` times.

resett, reset-timer[Observer, Turtle]
Resets the timer.

round `number` [Observer, Turtle]
Reports the integer closest to `number`.

rt, right `number` [Turtle]
Turtles turn right by `number` degrees.

save-info [Observer]
Saves the information window.

save-info-as [Observer]
Saves the information window using a different name than presently specified. A dialog box will appear to ask for the new file name.

save-output [Observer]
Saves output window to a file.

save-output-as [Observer]
Saves output window to a file. A dialog box will appear to ask for the new file name.

save-project [Observer]
Saves the current project to a file.

save-project-as [Observer]
Saves the current project using a different name than presently specified. A dialog box will appear to ask for the new file name

scale-color `color variable limit1 limit2` [Turtle]
Turtles set their color to a shade of `color` based on their value of `variable`. `limit1` and `limit2` determine the amount of gradation.
Example:
```
scale-color blue energy 0 20
```
Turtles turn one of twenty shades of blue. Turtles with lower energy turn darker blue.
```
scale-color blue energy 20 0
```
Turtles turn one of twenty shades of blue. Turtles with lower energy turn lighter blue.

scale-pc `color variable limit1 limit2` [Observer]
Patches set their color to a shade of `color` based on their value of `variable`. `limit1` and `limit2` determine the amount of gradation.
Example:
```
scale-color green density 0 20
```
Patches turn one of twenty shades of green. Patches with lower density turn darker green.
```
scale-color blue energy 20 0
```
Patches turn one of twenty shades of green. Patches with lower density turn lighter green.
NOTE: This is a patch command. It must be called from within an `ask-patches` or `ask-patch-at` statement

screen-half-height [Observer, Turtle]
Returns the half the width of the screen.

screen-half-width [Observer, Turtle]
Returns half of the width of the screen.

screen-height [Observer, Turtle]
Returns the height of the screen. The heights is always an odd number, and the lowest possible value is 3. To change the `screen-height`, drag the mouse over the patchcancas and use the handles to resize it with the mouse.

screen-width [Observer, Turtle]
Returns width of the screen. The width is always an odd number, and the lowest possible value is 3. To change the `screen-width`, drag the mouse over the patchcancas and use the handles to resize it with the mouse.

se, sentence `anything1 anything2` [Observer, Turtle]
Returns a list. If `anything1` and `anything2` are lists, it appends the two lists. If `anything1` is a list and `anything2` is not, it puts `anything2` at the end of the first list. If

`anything1` is not a list and `anything2` is a list, it puts `anything1` at the front of the second list. If both `anything1` and `anything2` are not lists, it creates a new list containing both `anything1 and anything2`.

set `variable value` [Observer, Turtle]
Assigns `value` to `variable`, where `variable` is a local variable which has already been declared. For more information on local variables, see the Variables page.

setage `anything` [Turtle]
Sets the turtle's `age` variable to `anything`.

setage-at `xcor ycor anything` [Observer, Turtle]
Sets the variable `age` of turtles at `xcor ycor` relative to the caller to `anything`.

setage-of `turtle id number anything` [Observer, Turtle]
Sets the variable `age` of turtle with who number `turtle id number` to `anything`.

setage-towards `angle distance anything` [Observer, Turtle]
Sets the variable `age` of turtles `distance` away at angle `angle` relative to the caller to `anything`.

setbreed `breedname` [Turtle]
Turtles set their breed to `breedname`.

setbreed-at `xcor ycor breedname` [Observer, Turtle]
Sets the breed of the turtle `xcor` units in the x direction and `ycor` units in the y direction away from the caller to `breedname`.
See the breeds reference for more information.

setbreed-of `number breedname` [Observer, Turtle]
Sets the breed of the turtle with ID Number `number` to `breedname`.
See the breeds reference for more information.

setbreed-towards `angle distance breedname` [Observer, Turtle]
Sets the breed of the turtle `distance` away in the direction `angle` to `breedname`.
See the breeds reference for more information.

setc, setcolor `colorname (or colornumber)` [Turtle]
Turtles set their color to `colorname (or color number)`.

setc-at `xcor ycor number` [Observer, Turtle]
Sets the color of the turtle `xcor` units in the x direction and `ycor` units in the y direction away from the caller to `number`.

setc-of `number1 number2` [Observer, Turtle]
Sets the color of the turtle with ID `number1` to color `number2`.

setc-towards `angle distance number` [Observer, Turtle]
Sets the color of the turtle `distance` away if the heading was `angle` to `number`.

setfood `anything` [Observer, Turtle]
Sets the variable `food` of patches to `anything`.
NOTE: This is a patch command. It must be called from within an `ask-patches` statement

setfood-at `xcor ycor anything` [Observer, Turtle]
Sets the variable `food` of the patch at `xcor ycor` relative to the caller to `anything`.

setfood-towards `angle distance anything` [Observer, Turtle]
Sets the variable `food` of the patch `distance` away at angle `angle` relative to the caller to `anything`.

seth, setheading `direction` [Turtle]
Turtles set their heading to a `direction` from 0 to 359 degrees.

seth-at `xcor ycor number` [Observer, Turtle]
Sets the heading of the turtle `xcor` units in the x direction and `ycor` units in the y direction away from the caller to `number`.

seth-of `number1 number2` [Observer, Turtle]
Sets the heading of the turtle with ID `number1` to `number2`.

seth-towards `angle distance number` [Observer, Turtle]
Sets the heading of the turtle `distance` away if the heading was `angle` to `number`.

set-info-name `name` [Observer]
Sets the name of the information window to `name`.

setitem `number list elt` [Observer, Turtle]
Sets the element at the `number`'th position in `list` to `elt`.

set-output-name `name` [Observer]
Sets the name of the output window to `name`.

setpc, setpatchcolor `color` [Observer, Turtle]
This command has two different uses. It can be run as a turtle command, in which case it will set the color of the patches which are underneath turtles to `color`. It can also be run as

a patch command, inside either an `ask-patches` or `ask-patch-at` statement. In these cases, the patches being asked will set their color to `color`.

setpendown?-at `xcor ycor boolean` [Observer, Turtle]
Sets the pendown state of the turtle `xcor` units in the x direction and `ycor` units in the y direction away from the caller to `boolean`.

setpendown?-of `number boolean` [Observer, Turtle]
Sets the pendown state of the turtle with ID Number `number` to `boolean`.

setpendown?-towards `angle distance boolean` [Observer, Turtle]
Sets the pendown state of a turtle `distance` away at an angle of `angle` to `boolean`.

setplot-title `name` [Observer, Turtle]
Sets the title of the plot to be `name`.

setplot-xlabel `label` [Observer, Turtle]
Sets the label of the x-axis of the graph to `label`.

setplot-xmax `value` [Observer, Turtle]
Sets the maximum value of the x-axis to be `value`.

setplot-xmin `value` [Observer, Turtle]
Sets the minimum value of the x-axis to be `value`.

setplot-xrange `value1 value2` [Observer, Turtle]
Sets the minimum value of the xy-axis to be `value1`, and the maximum value of the x-axis to be `value2`.

setplot-ylabel `label` [Observer, Turtle]
Sets the label of the y-axis of the graph to `label`.

setplot-ymax `value` [Observer, Turtle]
Sets the maximum value of the y-axis to be `value`.

setplot-ymin `value` [Observer, Turtle]
Sets the minimum value of the y-axis to be `value`.

setplot-yrange `value1 value2` [Observer, Turtle]
Sets the minimum value of the y-axis to be `value1`, and the maximum value of the y-axis to be `value2`.

setppc `color (or number)` [Observer, Turtle]

Sets the color of the current plot pen to be `color`.

setshown?-at `xcor ycor boolean` [Observer, Turtle]
Sets the visibility of the turtle `xcor` units in the x direction and `ycor` units in the y direction away from the caller to `boolean`.

setshown?-of `number boolean` [Observer, Turtle]
Sets the visibility of the turtle with ID Number `number` to `boolean`.
setshown?-towards `angle distance boolean` [Observer, Turtle]
Sets the visibility of a turtle `distance` away at an angle of `angle` to `boolean`.

settime `anything` [Observer]
Sets the value of the `globals` variable `time` to `anything`.

setx, setxcor `number` [Turtle]
Turtles set their x-coordinate to `number`.

setxcor-at `xcor ycor number` [Observer, Turtle]
Sets the x-coordinate of the turtle `xcor` units in the x direction and `ycor` units in the y direction away from the caller to `number`.

setxcor-of `number1 number2` [Observer, Turtle]
Sets the x-coordinate of the turtle with ID `number1` to `number2`.

setxcor-towards `angle distance number` [Observer, Turtle]
Sets the x-coordinate of the turtle `distance` away if the heading was `angle` to `number`.

setxy `number1 number2` [Turtle]
Turtles set their x-coordinate to `number1` and y-coordinate to `number2`.

sety, setycor `number` [Turtle]
Turtles set their y-coordinate to `number`.

setycor-at `xcor ycor number` [Observer, Turtle]
Sets the y-coordinate of the turtle `xcor` units in the x direction and `ycor` units in the y direction away from the caller to `number`.

setycor-of `number1 number2` [Observer, Turtle]
Sets the y-coordinate of the turtle with ID Number `number1` to `number2`.

setycor-towards `angle distance number` [Observer, Turtle]
Sets the y-coordinate of the turtle `distance` away if the heading was `angle` to `number`.

set-count-plot-pens `number` [Observer]
Sets the number of plot pens to `number`.

set-info-name `name` [Observer]
Sets the name of the information window to `name`.

set-output-name `name` [Observer]
Sets the name of the output window to `name`.

set-project-name `name` [Observer]
Set the name of the current project to `name`.

set-random-seed `integer` [Observer]
Sets the random see to `integer`. Integers allowed are in the range -(2^31) to (2^31 -1).

show `anything` [Observer, Turtle]
Types `anything` in the command center.

shown? [Observer, Turtle]
Returns true if the turtle is currently visible, otherwise false.

shown?-at `xcor ycor` [Observer, Turtle]
Reports whether the turtle at (`xcor ycor`) is visible.

shown?-of `number` [Observer, Turtle]
Reports whether the turtle with ID `number` is visible.

shown?-towards `angle distance` [Observer, Turtle]
Reports whether the turtle `distance` away at an angle of `angle` is visible.

sin `number` [Observer, Turtle]
Trigonometry functions. Returns the sine of the specified angle. All angles are in degrees

sky `color` [Observer, Turtle]
`sky` reports the number of its particular hue in the color table.
Its reference number is 95.

sprout `list to run` [Observer, Turtle]
Each patch creates a turtle, which then executes `list to run`.
NOTE: This is a patch command. It must be called from within an `ask-patches` or `ask-patch-at` statement.

sqrt `number` [Observer, Turtle]

Finds the square root of `number`.

st, showturtle [Turtle]
Hidden turtles make themselves visible.

stamp `color` [Observer, Turtle]
Sets color of patch under turtle to `color`

stamp-at `xcor ycor color` [Observer, Turtle]
Sets the color of the patch `xcor` units in the x direction and `ycor` units in the y direction away from the caller to `color`.

stamp-towards `angle distance color` [Turtle]
Sets the color of the patch `distance` away, if the heading were `angle`, to `color`.

startup [Observer]
This is a reserved observer procedure name. Anything within the `startup` procedure will be executed each time the project is opened.

step [Turtle]
A synonym for `fd 1`.

stop [Observer, Turtle]
Tells the current procedure to stop running.

stopall [Observer, Turtle]
Tells all everything to stop running, including procedures, buttons, monitors, and command centers.

sum-of-frogs `ilist` [Observer, Turtle]
Reports the total of evaluating `ilist` in every frog and adding it up.

sum-of-frogs-with `pred-ilist ilist` [Observer, Turtle]
Reports the total value of `ilist` when run over the turtles of breed `frogs` that satisfy `pred-ilist`. If there are no `frogs` in which `pred-ilist` is true, returns 0. Note: This command can also be executed by patches, for example within an `ask-patches` statement. Example: `sum-of-frogs-with (color = green) [age]` returns the sum of all of the ages of the green `frogs`.

sum-of-patches `ilist` [Observer, Turtle]
Reports the total of evaluating `ilist`in every patch and adding it up.
Example usage: `show sum-of-patches [food]`
This gives the total amount of food in the world.

sum-of-patches-with `pred-ilist ilist` [Observer, Turtle]
Reports the total value of `ilist` when run over the turtles that statisfy `pred-ilist`. If there are no patches in which `pred-ilist` is true, reports 0. Note: This command can also be executed by patches, for example within an `ask-patches` statement.
Example: `sum-of-patches-with (pc = green or pc = blue) [density]` returns the sum of the densities of all patches which are blue or green.

sum-of-turtles `ilist` [Observer, Turtle]
Reports the total of evaluating `ilist`in every turtle and adding it up.
Example usage:
```
show sum-of-turtles [weight * weight]
```
This gives the sum of every turtle's (weight squared).
sum-of-turtles-with `pred-ilist ilist`
Reports the total value of `ilist` when run over the turtles that satisfy `pred-ilist`. If there are no turtles in which `pred-ilist` is true, returns 0. Note: This command can also be executed by patches, for example within an `ask-patches` statement.
Example: `sum-of-turtles-with (color = red) [age]` returns the sum of the ages of all the red turtles.

tan `number` [Observer, Turtle]
Trigonometry functions. Returns the tangent of the specified angle. All angles are in degrees.

time [Observer]
Returns the value of the `globals` variable `time`.

timer [Observer, Turtle]
Reports the curent value of the timer.

to [Observer, Turtle]
When you create a procedure, the first line of the procedure must be `to procedureName` where `procedureName` is a one-word title for your procedure. `procedureName` cannot be any command already recognized by StarLogo (this includes both built-in commands and the procedure names of other procedures you may already have written).For example: `to go fd 1 rt 90 end`

towards `xcor ycor` [Observer, Turtle]
Returns the angle from the callers xcor and ycor to the specified absolute position specified by `xcor` and `ycor`.

towards-nowrap `xcor ycor` [Observer, Turtle]
Returns the angle from the callers xcor and ycor to the specified absolute position specified

by `xcor` and `ycor` without wrapping.

to-delimited-string `ilist` [Observer, Turtle]
Takes a list of instructions (variables, reporters, strings) and separates them with commas and then concatenates them, returning the concatenated string. Example:

```
globals [mystring one]
setone "abc"
setmystring to-string [one "two"]
```

The value of `mystring` is now "`abc,two`".

to-string `ilist` [Observer, Turtle]
Takes a list of instructions (variables, reporters, strings) and concatenates them, returning the concatenated string. Example:

```
globals [mystring one]
setone "abc"
setmystring to-string [one "two"]
```

The value of `mystring` is now "`abctwo`".
turquoise `color` [Observer, Turtle]
`turquoise` reports the number of its particular hue in the color table.
Its reference number is 75.

turtles-own `[variable list]` [Observer, Turtle]
Defines a set of variables to be properites of turtles. The `variable list` contains one or more names.

type `text or variables` [Observer, Turtle]
Prints the `text or variables` in the output window. No carriage return is printed.

viewplot, view-plot [Observer, Turtle]
Opens the Plot Window.

violet, purple `color` [Observer, Turtle]
`violet` reports the number of its particular hue in the color table.
Its reference number is 115.

wait `number` [Observer, Turtle]
Caller waits `number` seconds before continuing.

wait-until `predicate` [Observer, Turtle]
Caller waits until `predicate` is true.

white `color` [Observer, Turtle]
`white` reports the number of its particular hue in the color table.

Its reference number is 9.

who [Turtle]
Returns the ID number of the turtle.

word? `thing` [Observer, Turtle]
Returns true if `thing` is a word or a string.

xcor [Turtle]
Returns the x-coordinate of the turtle

xcor-at `xcor ycor` [Observer, Turtle]
Reports the x-coordinate of the turtle `xcor` units in the x direction and `ycor` units in the y direction away from the caller.

xcor-of `number` [Observer, Turtle]
Reports the x-coordinate of the turtle with ID `number`.

xcor-towards `angle distance` [Observer, Turtle]
Reports the x-coordinate of the turtle `distance` away if the heading was `angle`.

`condition1` xor `condition2` [Observer, Turtle]
Reports the value of `condition1` xor `condition2`.

ycor [Turtle]
Returns the y-coordinate of the turtle

ycor-at `xcor ycor` [Observer, Turtle]
Reports the y-coordinate of the turtle `xcor` units in the x direction and `ycor` units in the y direction away from the caller.

ycor-of `number` [Observer, Turtle]
Reports the y-coordinate of the turtle with ID `number`.

ycor-towards `angle distance` [Observer, Turtle]
Reports the y-coordinate of the turtle `distance` away if the heading was `angle`.

yellow `color` [Observer, Turtle]
`yellow` reports the number of its particular hue in the color table.
Its reference number is 45.

<u>Colors:</u>

Alphabetically

0
black
105
blue
35
brown
85
cyan
5
gray
55
green
65
lime
125
magenta
25
orange
135
pink
15
red
95
sky
75
turquoise
115
violet
9
white
45
yellowNumerically

0
black
5
gray
9
white
15

red
25
orange
35
brown
45
yellow
55
green
65
lime
75
turquoise
85
cyan
95
sky
105
blue
115
violet
125
magenta
135
pink