

NetLogo		Version 1/15/18	
crt <i>n</i>	create <i>n</i> turtles (random headings)	if condition [commands]	if the condition is true, then execute the commands
cro <i>n</i>	create <i>n</i> turtles (equally distributed headings)		
ca cp cd	clear all clear patches clear drawing	ifelse condition [commands-1] [commands-2]	if the condition is true, then execute commands-1, otherwise execute commands-2
setxy <i>new-x new-y</i> move-to <i>agent</i>	move the turtle(s) to <i>xcor = new-x</i> and <i>ycor = new-y</i> moves the turtle on top of the agent (another turtle or patch)	while [<i>test</i>] [commands]	while the <i>test</i> is true, repeatedly do the commands. while [<i>xcor < 0</i>] [<i>fd 1</i> <i>set pcolor green</i>]
fd <i>n</i> bk <i>n</i>	forward <i>n</i> steps backward <i>n</i> steps	distancexy <i>xvalue yvalue</i>	Reports the distance between the turtle (or patch) and the point (<i>xvlaue</i> , <i>yvalue</i>)
sqrt <i>expression</i> <i>a mod b</i>	calculates the square root of the expression (e.g. <code>sqrt (xcor * xcor + ycor * ycor)</code>) will calculate the distance of this turtle from the origin calculates the remainder when <i>a</i> is divided by <i>b</i> . e.g. <code>13 mod 5</code> is 3	mouse-xcor and mouse-ycor mouse-down? mouse-inside?	- <i>mouse-xcor</i> and <i>mouse-ycor</i> are the coordinates (current position) of the mouse - <i>mouse-down?</i> is true if the left mouse button is pressed, false otherwise - <i>mouse-inside?</i> is true if the mouse cursor is inside the NetLogo visual area, false otherwise
rt <i>n</i>	rotate right <i>n</i> degrees	let <i>variable1 value1</i>	create variables used only in the current procedure
lt <i>n</i>	rotate left <i>n</i> degrees	globals [<i>global-variable-1 ...</i>]	create variables seen and modifiable throughout the program
pu	pen up	turtles-own [<i>property-1 ...</i>]	create properties for turtles
pd	pen down (draw)	patches-own [<i>property-1 ...</i>]	create properties for patches
set size <i>n</i>	change size of turtle	to procedure-name ... end	define a procedure
set color <i>n</i> (or) set color <i>color-word</i>	change color of turtle	to-report reporter-name ... report expression end	define a reporting procedure
repeat <i>n</i> []	repeat <i>n</i> times the commands in []	Common properties of a turtle	<i>who</i> , <i>xcor</i> , <i>ycor</i> , <i>color</i> , <i>shape</i> , <i>size</i> , <i>heading</i> , <i>label</i> , <i>label-color</i> , <i>pen-size</i> , <i>pen-mode</i> , <i>hidden?</i> , <i>breed</i>
set shape " <i>shape name</i> "	change shape of turtle	Common properties of a patch	<i>pxcor</i> , <i>pycor</i> , <i>pcolor</i> , <i>plabel</i> , <i>plabel-color</i>
"forever" button	continuously submits its commands	Math functions	<i>abs</i> , <i>cos</i> , <i>ceiling</i> , <i>floor</i> , <i>int</i> , <i>mod</i> , <i>pi</i> , <i>remainder</i> , <i>round</i> , <i>sin</i> , <i>sqrt</i> , <i>tan</i>
random <i>n</i> one-of	reports a number between 0 and <i>n-1</i> (inclusive) reports a random one of a list or agentset	Idioms	
set pcolor <i>n</i>	sets the color of the patch	set the color of turtle 12 to a random value	ask turtle 12 [set color random 140]
stamp	paints the ground underneath a turtle with the image of the turtle	create a "wiggling" procedure	to wobble [stepsize angle] rt random angle lt random angle fd stepsize end
lists and list functions	set fred [-8 3 "harry"] set label item 2 fred set shape one-of ["cow" "wolf" "ant"] set fred lput "harry2" fred	sample of a "collision" procedure: if there are 3 or more turtles on a patch, make them die from overcrowding	to overcrowding-check if count other turtles-here >= 2 [ask turtles-here [die]] end
Agentsets: turtles with [test] patches with [test] neighbors (8 neighbors) or neighbors4 (up,down,left,right)	Ask turtle 12 [let rich-neighbors neighbors with [earnings > 100] ask rich-neighbors [Lend-me-money]] ask patches [If count neighbors4 with [garbage > 100] > 2 [Move-to-different-neighborhood]]	Using values and lists created by: of min-one-of one-of n-of	Let list-of-xcors [<i>xcor</i>] of turtles if [<i>xcor</i>] of turtle 0 > 0 [...] if [<i>pcolor</i>] of patch-at 1 0 = red [sprout 1] Ask min-one-of turtles [distancexy 0 0] [die] Ask turtle 12 [move-to one-of patches with [pxcor > 0]] (recolor 5 randomly chosen blue patches red): Ask n-of 5 patches with [pcolor = blue] [set pcolor red]
face, facexy towards towardsxy	Ask turtle 12 [face turtle 2] Ask turtle 12 [facexy 3 -2] Ask turtle 12 [set heading towards turtle 2] Ask turtle 12 [set heading towardsxy mouse-xcor mouse-ycor]	List functions: item , length , sum , max , min , mean , median , etc.	print item 3 [4 12 45 -98 3] ; will print -98 print length [-5 98] ; will print 2 print sum [4 5 -8 25 0.8] ; will print 26.08 if max ([xcor] of turtles) > 8 [...] print mean [color] of turtles if median [grade] of students with [class = "MKS1"] < 65 [Fail-Teacher] ask patch 0 0 [let neighborhood-wealth sum [earnings] of neighbors4]
Relative patch: patch-at <i>dx dy</i> patch-ahead <i>how-far</i>	Ask turtles [ask patch-at 1 2 [set pcolor red]] Ask turtles [if red = [pcolor] of patch-ahead 1 [avoid-wall]]	Find turtle closest to mouse and "select" it.	Turtles-own [picked-up?] if mouse-down? [let closest min-one-of turtles with [distancexy mouse-xcor mouse-ycor] ask closest [if distancexy mouse-xcor mouse-ycor < size / 2 [set selected? True]]]