NetLogo Column 1		NetLogo Column 2	
crt *n* **cro** *n*	create *n* turtles (random headings) create n turtles (equally distributed headings)	**if *condition*** **[commands]**	if the condition is true, then execute the commands
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 *new-x new-y*
move-to *agent*
	move the turtle(s) to xcor = new-x and ycor = new-y
	moves the turtle on top of the agent (another turtle or patch)
		while [*test*]
		[commands]
			while the *test* is true, repeatedly do the commands.
			`while [ xcor < 0 ]`
			`[ fd 1`
			`  set pcolor green`
			`]`

fd *n* **bk *n***	forward *n* steps backward *n* steps	**distancexy *xvalue yvalue***	Reports the distance between the turtle (or patch) and the point (xvlaue, yvalue)
sqrt *expression* *a* **mod** *b*	calculates the square root of the expression (e.g. `sqrt (xcor * xcor + ycor * ycor))` will calculate the distance of this turtle from the origin calculates the remainder when a is divided by b. e.g. 13 mod 5 is 3	**mouse-xcor** and **mouse-ycor** **mouse-down?** **mouse-inside?**	- mouse-xcor and mouse-ycor are the coordinates (current position) of the mouse - mouse-down? is true if the left mouse button is pressed, false otherwise - mouse-inside? is true if the mouse cursor is inside the NetLogo visual area, false otherwise
rt *n*	rotate right *n* degrees	**let** variable1 value1	create variables used only in the current procedure
lt *n*	rotate left *n* degrees	**globals [*global-variable-1 ...*]**	create variables seen and modifiable throughout the program
pu	pen up	**turtles-own [*property-1 ...*]**	create properties for turtles
pd	pen down (draw)	**patches-own [*property-1 ...*]**	create properties for patches
set size *n*	change size of turtle	**to *procedure-name*** **...** **end**	define a procedure
set color *n* **(or)** **set color *color-word***	change color of turtle	**to-report *reporter-name*** **...** **report *expression*** **end**	define a reporting procedure
repeat *n* [ ]	repeat *n* times the commands in [ ]	**Common properties of a turtle**	who, xcor, ycor, color, shape, size, heading, label, label-color, pen-size, pen-mode, hidden?, breed
set shape "*shape name*"	change shape of turtle	**Common properties of a patch**	pxcor, pycor, pcolor, plabel, plabel-color
"forever" button	continuously submits its commands		
random *n* **one-of**	reports a number between 0 and n-1 (inclusive) reports a random one of a list or agentset	**<u>Idioms</u>**	
set pcolor *n*	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 wiggle [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**	Let list-of-xcors [xcor] of turtles if [xcor] of turtle 0 > 0 […] if [pcolor] of patch-at 1 0 = red [sprout 1] Ask min-one-of turtles [distancexy 0 0] [die] Ask turtles [move-to one-of patches with [pxcor > 0]]
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** dx dy **patch-ahead** how-far	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]]]

V. 1/11/17-a