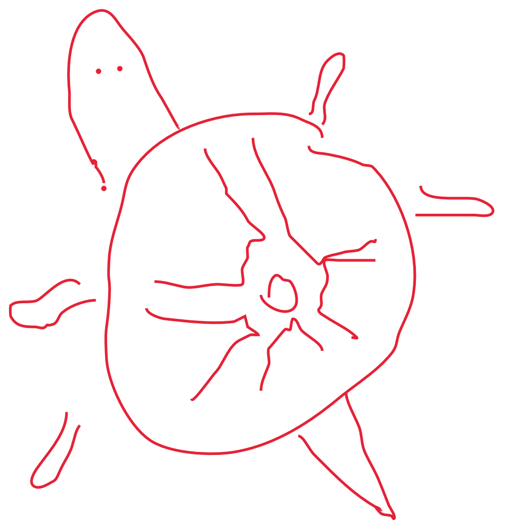
Quick intro to NetLogo

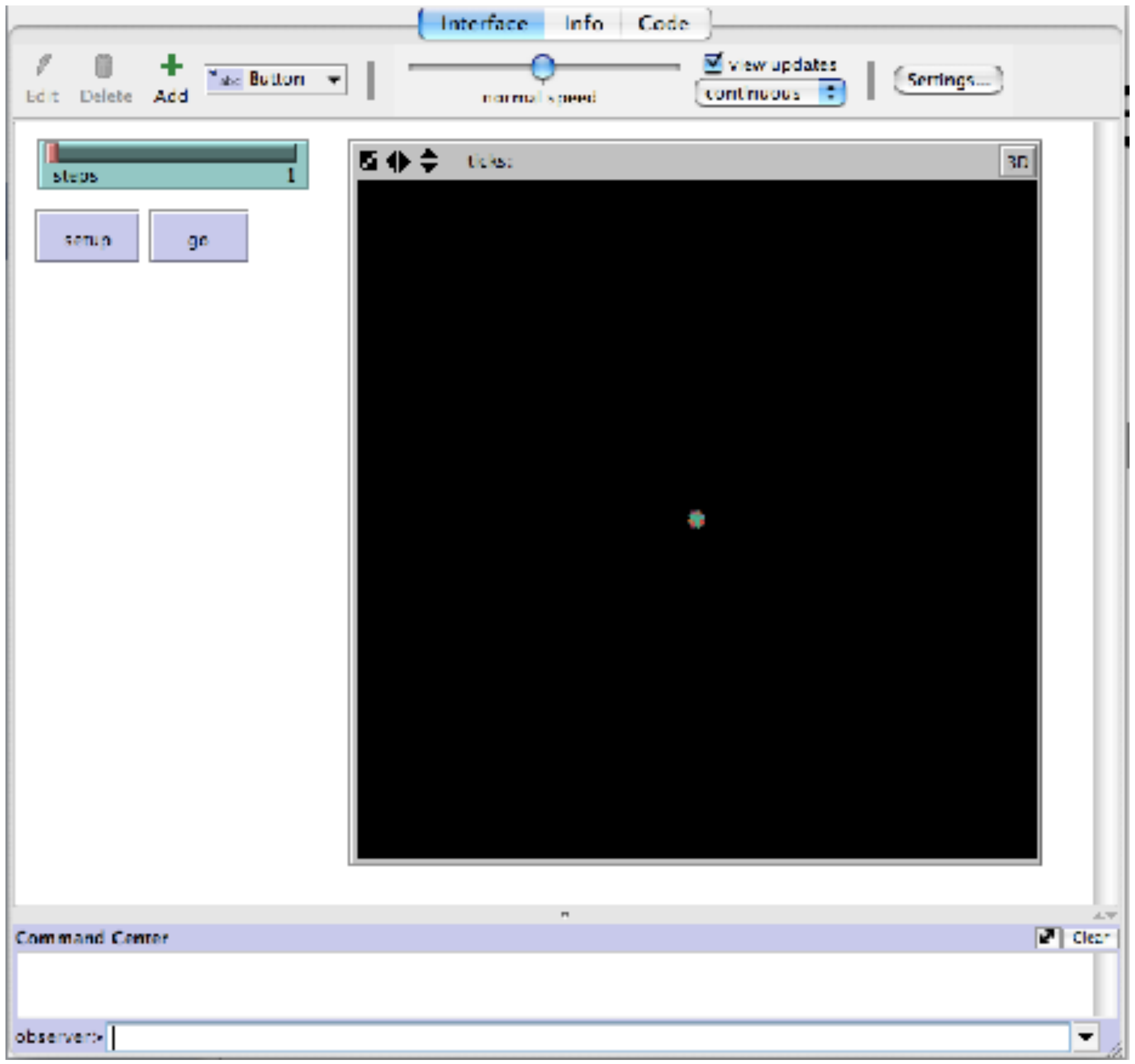
Intro to NetLogo



- Agents are called **turtles** (yep, turtles!)
- Environment is usually grid-based, each square in the grid is called a **patch**

Basic Setup

- NetLogo models have three tabs
  - Interface
  - Info
  - Code



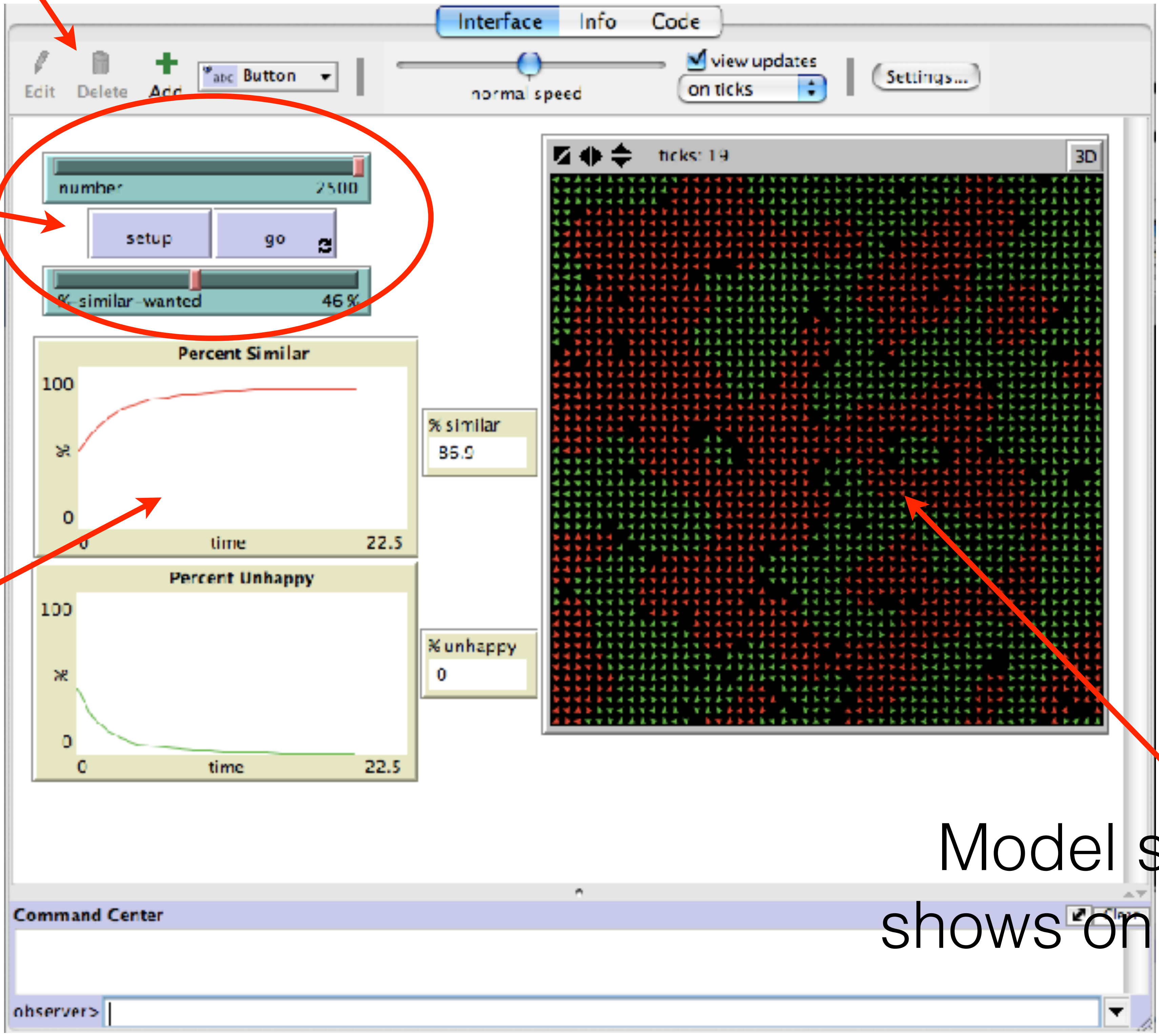
Interface

Add to & edit interface

Buttons & sliders to control model

Plots of model outputs

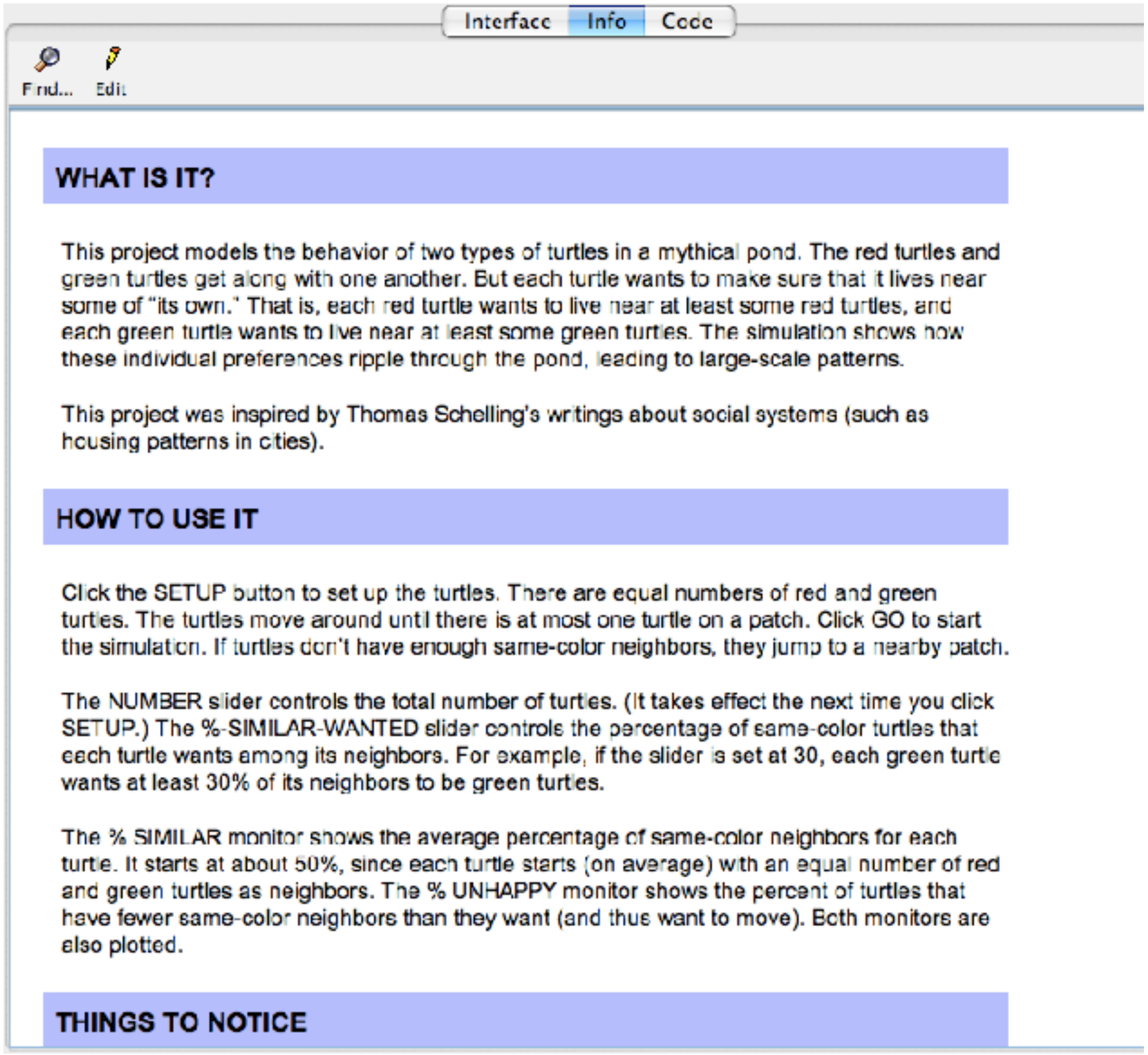
Model simulation shows on this screen



Interface

- Demonstrate how to make and edit buttons and sliders

Info



WHAT IS IT?

This project models the behavior of two types of turtles in a mythical pond. The red turtles and green turtles get along with one another. But each turtle wants to make sure that it lives near some of "its own." That is, each red turtle wants to live near at least some red turtles, and each green turtle wants to live near at least some green turtles. The simulation shows how these individual preferences ripple through the pond, leading to large-scale patterns.

This project was inspired by Thomas Schelling's writings about social systems (such as housing patterns in cities).

HOW TO USE IT

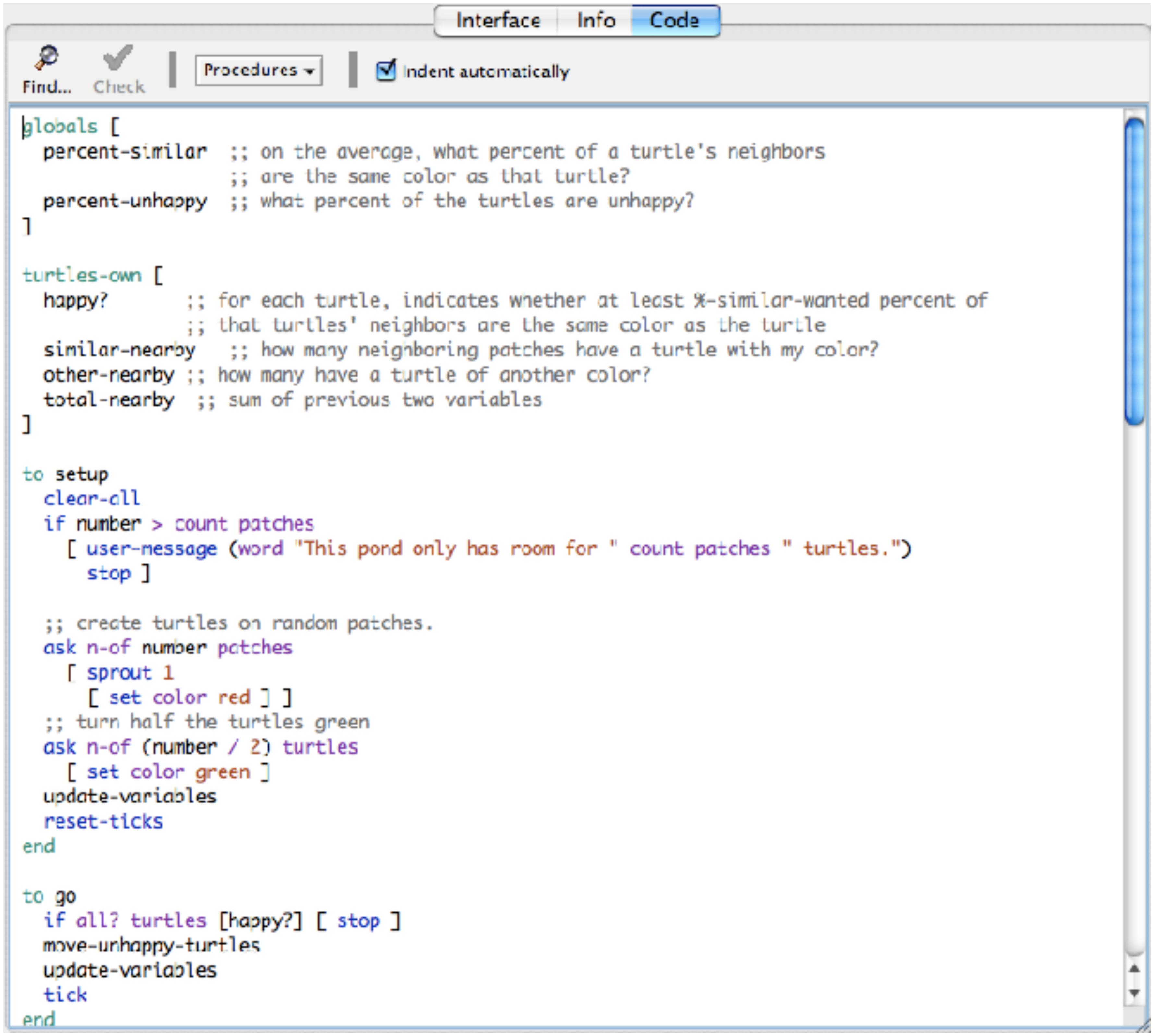
Click the SETUP button to set up the turtles. There are equal numbers of red and green turtles. The turtles move around until there is at most one turtle on a patch. Click GO to start the simulation. If turtles don't have enough same-color neighbors, they jump to a nearby patch.

The NUMBER slider controls the total number of turtles. (It takes effect the next time you click SETUP.) The %-SIMILAR-WANTED slider controls the percentage of same-color turtles that each turtle wants among its neighbors. For example, if the slider is set at 30, each green turtle wants at least 30% of its neighbors to be green turtles.

The % SIMILAR monitor shows the average percentage of same-color neighbors for each turtle. It starts at about 50%, since each turtle starts (on average) with an equal number of red and green turtles as neighbors. The % UNHAPPY monitor shows the percent of turtles that have fewer same-color neighbors than they want (and thus want to move). Both monitors are also plotted.

THINGS TO NOTICE

Code



```
globals [
  percent-similar  ;; on the average, what percent of a turtle's neighbors
                   ;; are the same color as that turtle?
  percent-unhappy  ;; what percent of the turtles are unhappy?
]

turtles-own [
  happy?         ;; for each turtle, indicates whether at least %-similar-wanted percent of
                 ;; that turtles' neighbors are the same color as the turtle
  similar-nearby   ;; how many neighboring patches have a turtle with my color?
  other-nearby ;; how many have a turtle of another color?
  total-nearby  ;; sum of previous two variables
]

to setup
  clear-all
  if number > count patches
    [ user-message (word "This pond only has room for " count patches " turtles.")
      stop ]

  ;; create turtles on random patches.
  ask n-of number patches
    [ sprout 1
      [ set color red ] ]
  ;; turn half the turtles green
  ask n-of (number / 2) turtles
    [ set color green ]
  update-variables
  reset-ticks
end

to go
  if all? turtles [happy?] [ stop ]
  move-unhappy-turtles
  update-variables
  tick
end
```

Code

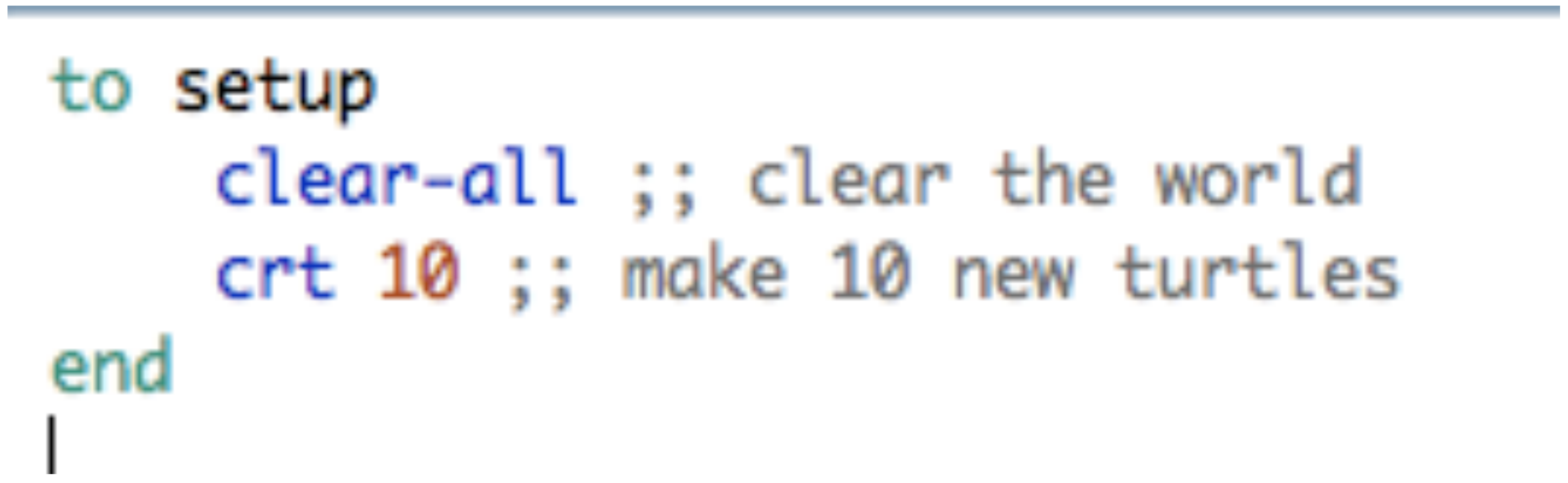
- Connects to & drives the interface tab
- Logo-language
- This is where you set up environment, rules, turtles/agents, etc.

Coding in NetLogo

- **Procedures** - basic building block of code, similar to a function in python
- Always starts with `to` and ends with `end`
- Examples—the **setup** and **go** buttons on the interface are defined by procedures
- **Primitives** - procedures built in to NetLogo

Coding in NetLogo

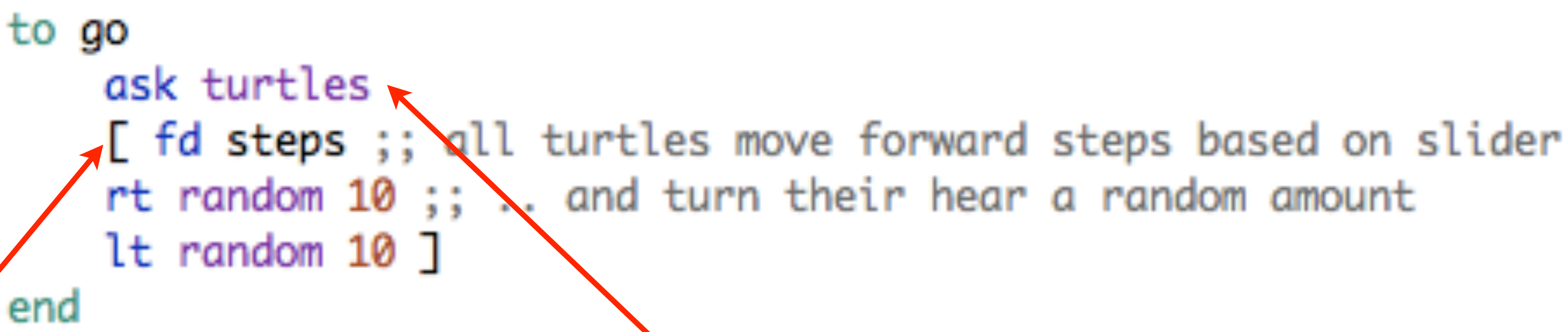
- **Setup** - sets up model to get ready to run it (e.g. setting parameters, initial conditions, etc.)



```
to setup
    clear-all ;; clear the world
    crt 10 ;; make 10 new turtles
end
```

Coding in NetLogo

- **Go** - runs the model. This is often where you call the **procedures** for all the agent rules, etc.

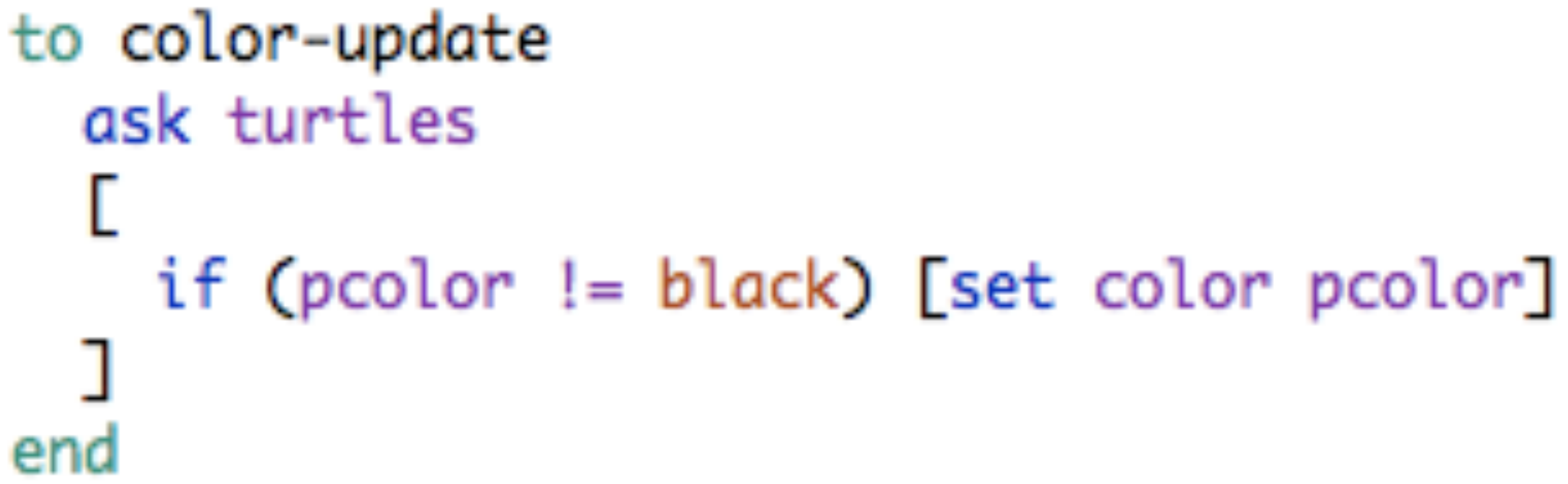


```
to go
    ask turtles
    [ fd steps ;; all turtles move forward steps based on slider
    rt random 10 ;; .. and turn their hear a random amount
    lt random 10 ]
end
```

Note brackets for ask

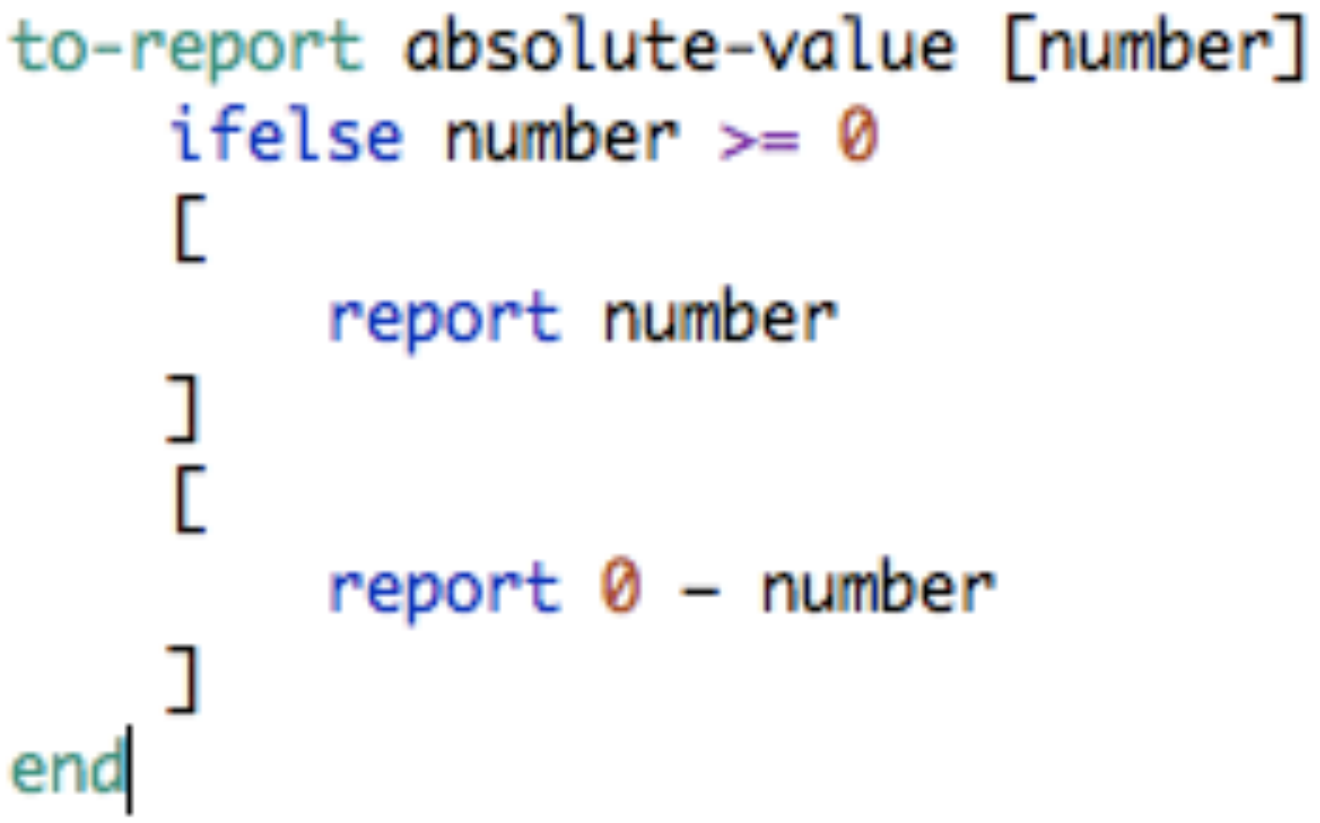
Ask is how you get turtles to do stuff. In this case, they move forward and then turn

Commands



```
to color-update
  ask turtles
  [
    if (pcolor != black) [set color pcolor]
  ]
end
```

Reporters



```
to-report absolute-value [number]
    ifelse number >= 0
    [
        report number
    ]
    [
        report 0 - number
    ]
end
```

Primitives

- Some useful ones for today:
- `color` = turtle/agent color
- `pcolor` = color of patch the turtle is on
- `fd 2` = move turtle forward 2 steps
- `if` (condition)[do this]
- `set` variable value, e.g. `set x 2` will set x = 2

Variables

- Global variables can be defined either by
  - Adding a slider in the interface tab
  - Adding them to the list of globals at the beginning:
    `globals` [parameter]
- `Set` - sets variable value,
  e.g. `set cat-color grey`

Examples for using variables

- `ask turtles [set color red]`
- `ask patches [set pcolor red]`
- `ask turtles [set pcolor red]`
- `ask turtle 5 [set color green]`
  or
  `set color-of turtle 5 green`
- `ask turtles with [color = red] [set color green]`

If statements

-
  ```
  ask turtles
  [
       if (attribute > threshold)
    [
            set color green
            fd 1
       ]
  ]
  ```

Useful resources

- NetLogo quick guide: https://ccl.northwestern.edu/netlogo/resources/NetLogo-4-0-QuickGuide.pdf
- NetLogo programming guide: http://ccl.northwestern.edu/netlogo/docs/programming.html
- NetLogo dictionary: http://ccl.northwestern.edu/netlogo/docs/dictionary.html