Chapter 2 Exercise Hints and Solutions

Agent-based and Individual-Based Modeling: A Practical Introduction, 2nd Edition

Exercise 3

We used the statement `ask n-of 20 patches in-radius 5` to identify the patches to turn red; the statement is executed four times to create the four clusters. The primitive `n-of 20 patches` chooses 20 patches randomly from among *all* patches, including those that are already red. So sometimes we are turning a patch red when it is already red, so the total number of red patches is less than 80.

(We could make sure each cluster turns exact 20 more patches from black to red by changing the statement to `ask n-of 20 (patches in-radius 5) with [pcolor != red]`. This will work even if the clusters are very close together: it turns out there are 81 patches within a radius of 5 of any patch.)

Exercise 4

We also provide a NetLogo file "MushroomHunt-Exercise4.nlogo" that has these changes all implemented.

- There is only one hunter instead of two.
  In the `setup` procedure, change:
  ```
  crt 2
  ```
  to:
  ```
  crt 1
  ```

- The hunter's initial heading is always 45 degrees.
  In the `setup` procedure, insert within the `crt` block of commands (e.g., just after `pen-down`):
  ```
  set heading 45
  ```

- Instead of four clusters that each have 20 mushrooms, there are eight clusters of 10 mushrooms. Each cluster has a radius of only three patches.
  In the `setup` procedure, change:
  ```
  set num-clusters 4
  ```
  to:
  ```
  set num-clusters 10
  ```
  Change:
  ```
  ask n-of 20 patches in-radius 5
  ```
  to:
  ```
  ask n-of 10 patches in-radius 3
  ```

- Whenever the hunter finds a mushroom, it writes "I found one!" to the Command Center.
  In the `search` procedure, change:
  ```
  set pcolor yellow
  ```

to:
```
set pcolor yellow
show "I found one!"
```

(The primitives `print`, `type`, and `write` could also be used, but `show` is preferred because it includes a carriage return and tells us which turtle found a mushroom.)

- When the hunter has not recently found a mushroom, it turns by a random angle between -45 and +45 degrees instead of between -10 and +10.
  In the `search` procedure, change:
  ```
  [right (random 21) - 10]
  ```
  to:
  ```
  [right (random 91) - 45]
  ```

- The hunter starts in the lower left corner instead of in the middle.
  In the `setup` procedure, insert within the `crt` block of commands (e.g., just *before* `pen-down`, so the pen is not down until the turtle has moved to the corner):
  ```
  setxy min-pxcor min-pycor
  ```

  Note that this puts the turtle in the center of the lower-left patch, not absolutely in the corner. This statement could also be `setxy -13 -13` because the world extends from -13 to +13, but using `min-pxcor` and `min-pycor` is preferred because it will work even if the world size is changed.

- The hunter counts how many mushrooms it catches. This requires a new turtle variable that must be incremented each time a mushroom is found. Instead of writing "I found one!" to the command center, the hunter now writes out how many total it has found.

  - Add a new turtle variable by changing the `turtles-own` statement to:
    ```
    turtles-own
    [
      time-since-last-found
      num-found
    ]
    ```

  - In `setup`, initialize the value of num-found by adding this to the `crt` block of commands (e.g., after `set heading 45`):
    ```
    set num-found 0
    ```
    (The code will work without this initialization because variables are given a default initial value of zero, but it is a good habit to initialize all variables.)

  - In the `search` procedure, change:
    ```
    set pcolor yellow
    ```
    to:
    ```
    set pcolor yellow
    set num-found num-found + 1
    show word "So far I found: " num-found
    ```

(Again, there are other ways to write the output statement but using `word` is preferred as it is most flexible; we use it often later.)

Exercise 6

Solutions are in parentheses:

- Round a number (e.g., the mean size of turtles) off to 3 decimal places. (`precision` (*not* `round`))
- Identify all the turtles that have the highest value of the variable `age`. (`with-max [age]`)
- Identify the turtle with lowest value of `age`. (`min-one-of [age]`)
- Move a turtle to the patch with coordinates -3, 5. (`move-to patch -3 5`)
- Move a turtle three patches left and five patches up. (`move-to patch-at -3 5`)
- Sort the turtles by their size. (`sort-by` and `sort-on` (*not* `sort`, which sorts turtles by who number))
- Find the first 10 characters in a text string. (`substring`)
- Save the entire Interface as a graphics file. (`export-interface`)
- Remove all the turtles (using one primitive). (`clear-turtles` (`ask turtles [die]` also removes all the turtles but uses two primitives and does not reset the who counter to zero))
- Open a pop-up menu to let the user select an input file. (`user-file`)
- Execute a set of commands every 4 seconds. (`every`)
- Have a turtle leave a permanent mark at its current location. (`stamp`)

- Move a turtle to the one of its four neighboring patches that has the lowest value of the variable `risk`.
  (`downhill4 risk` (`move-to min-one-of neighbors4 [risk]` also works))