Introduction:
In the first NetLogo lab you learned how to control turtles and patches using the Command Center and then how to control the interactions of patches and turtles by writing procedures. In this week's lab we'll write procedures to draw figures that represent some of the shapes of nature. In Netlogo procedures are either *commands* or *reporters. Commands* are actions that patches or agents carry out. *Reporters* are commands that return a value. For example `random 40` returns a random number between 0 and 39. Preexisting procedures in NetLogo are called *primitives.* An example of a simple procedure is given below:

```
to square
   clear-all
   crt 1 [ pendown ]          ; create a turtle for drawing
   repeat 4 [
      ask turtles[ fd 5
                   rt 90 ] ] ; draw a square
   clear-turtles              ; remove turtles
end
```

There are a couple of structural aspects of a procedure that you should notice. First, a procedure starts with a `to` and end with an `end`. Second, when you are programming it is important to indent, so that the code is easier to read. Please also get into the habit early of adding comments to your procedures so that it is clear to others what each component of your program does. Comments are added using semi-colons.

Look at the commands in the `square` procedure carefully make sure you understand why this procedure draws a square.

Geometrical Figure Procedures:
Open up a new netlogo file. Save your file with the naming convention

`Computer_Lab_05_lastname_firstname.nlogo`

Remember to save your file often as you work. Now, on the interface right click on the black world view and click edit. Change `max-pxcor` and `max-pycor` to 99 and put `patch size` to 3. This makes the world 100 by 100 patches Click on the procedures tab. Now type in the `square` procedure given above. I recommend that you type it line by line, as this will help you become aware of syntax subtleties.

Return to the interface and type `square` in the Command Center. Behold, a square appears. Try changing the number of turtles you create in the procedure and see what happens.

To save having to type `square` each time let's add a button to the interface that will call the square procedure when pressed.

1. Click on the button icon in the interface Toolbar
2. Click where you want the button to be in the empty white area of the Interface tab.
3. When the dialog box for editing the properties of the button opens, type `circle` in the box labeled "Commands".
4. Click OK and now you have a button that will draw a circle.

We can also add a slider to change the size of the square we draw. Click on the slider icon on the interface Toolbar. And click on the empty white area where you would like to place it. Then enter `step-size` in the Global Variable box, set the minimum to be 0 and the maximum to be 10 and the increment to be 0.1 and the value to be 5. This defines a new variable called `step-size`, which we will use to change the size of the circle.

In the `square` procedure replace the line `fd 5` with `fd step-size`. Now with your slider you can change the size of your square easily.

Polygon Procedure:
We could now create a procedure to make a pentagon, another to make a hexagon and so on, but let's instead make a procedure to make a polygon of any shape. For this you'll need to make a slider called `sides` which will specify the number of sides to your polygon. Now change your square procedure so it is called polygon (Change the button on the interface also). Change the `repeat 4` to `repeat sides` . You'll also need to change the turn angle so it turns the right number of degrees to complete the polygon with the right number of sides. Test out your procedure to make sure it works

Circle Procedure:
You'll probably notice that when you create a polygon with a large number of sides it starts to look like a circle. For example a circle procedure might be defined as follows

```
to circle
   clear-all
   crt 1 [ pendown ]          ; create a turtle for drawing
   repeat 36 [
      ask turtles[ fd 5
                   rt 10 ] ] ; draw a circle
   clear-turtles              ; remove turtles
end
```

although it is really a 36 sided polygon, with each side of length 5. How would you modify this procedure so it draws a circle with a given radius. (ie how does the length of the side need to depend on a specified radius?). Make a slider called `radius` and test it out.

Spiral Procedure:
Ok now we move from unnatural to natural shapes. We'll start with spirals. A spiral can be thought of as a circle that "misses", because its radius grows or shrinks. Let's adapt the above circle to draw a spiral. We need to find some way to make the turtle move a little bit more each time it takes a step.

We can do this as follows.

Define a new `turtles-own` variable called `step` by putting the command

```
turtles-own [ step ]
```

at the top of the procedures window. This defines a variable `step` for each turtle. Each turtle will have its own value for the variable `step`. The variable `step-size` we defined earlier is a Global variable, which means that it doesn't belong to any particular turtle.

Click on the procedures tab, copy and paste the `circle` procedure to create a new procedure and rename it `spiral` and then make the following changes:

```
to spiral
   clear-all
   crt 1 [  set step step-size
            pendown   ]                    ; create a turtle for drawing
   repeat 36 [
       ask turtles[ fd step
                    set step step * 1.03   ; draw a spiral which grows
                    rt 10  ]  ]            ; with factor 1.03 each repeat
   clear-turtles                           ; remove turtles
end
```

The line `set step step-size` initializes the variable `step` to be equal to the global variable `step-size` that is defined by the slider on the interface.

The line `step step * 1.03` makes the value of step grow by 3% once each repeat.

Add a `spiral` button and try it out. You can modify the procedure to make a longer spiral by increasing the number of repeats. You can change the growth rate by change the factor 1.03. If the spiral grows off the screen, you may want to set the initial step-size to be quite small.

There is a nice way of getting NetLogo to draw a spiral whenever and wherever you click the mouse. This is how you modify the `spiral` procedure:

```
to spiral
  if mouse-down?[                          ; draw a spiral if the mouse
                                           ; is down
    crt 1 [ setxy mouse-xcor mouse-ycor    ; move the turtle to the
                                           ; location of the pointer
            set step step-size
            pendown]                       ; create a turtle for drawing
    repeat 36 [
       ask turtles[ fd step
                    set step step * 1.03   ; draw a spiral which grows
                    rt 10  ] ]             ; with factor 1.03 each repeat
    clear-turtles                    ; remove turtles
  wait 0.1  ]                        ; wait 0.1 seconds for mouse to
                               ; unclick, then close the if statement
end
```

Now back in the interface right click on the `spiral` button and check the box marked `forever` and press ok. This will make sure the procedure keeps checking to see if you have clicked the mouse. Press the spiral button to start the procedure and then click somewhere on the screen. You may have noticed that we removed `clear-all` from the above procedure. This is because the repeat loop would keep erasing our drawing if we didn't. However, you will want to be able to remove the drawing from time to time or else life will get quite cluttered. So at this point you should make a button to clear everything from the screen and use it when you need to start with a clean screen.

Finally, we can enhance the beauty of this spiral by letting the color change as the spiral grows. After the `fd step` command inside the repeat loop of your procedure add the line

```
set color color + 10
```

Try it out. If you want to give specific initial values for the color you can do this in the part of the procedure where you create your turtle.

Lab Assignment

Enhancing Spiral:

1. Add a slider called `growth-factor` that will allow the growth factor (currently 1.03) to change between 1 and 1.05 in increments of 0.001. Repleace the 1.03 in your procedure with the variable `growth-factor`.
2. Add a `number` slider for your spiral procedure that will allow the number or repeats (currently 36) to range from 0 to 500 in increments of 5. Adjust the repeat loop accordingly. If you choose a large number of repeats you should set your growth factor close to 1 or else you'll get a mess!
3. Add a slider that will allow you to change the `angle` that the turtle turns through. At the moment that is 10 degrees, make the slider vary from 5 to 175 degrees in steps of 0.5. See what happens when you modify this angle so that it is close to an angle that corresponds to a regular polygon (eg 45 degrees or 72 degrees).
4. It is possible to get the turtle drawing the spiral to stamp its image on the screen as it moves along. The primitive is called `stamp`. Change the shape of the turtle when you create it to be a circle (or create some other shape like a chamber in a shell or a petal in a flower), and then have the turtle `stamp` after it makes its `fd step`. Also, make the `size` of the turtle change by the same growth factor as the `step` length.
5. Now experiment with this spiral a little. The curve that is drawn shows what is called the generating spiral, which is the spiral that we are intentionally constructing. You may also notice that the turtle stamps align themselves naturally into either lines or other spirals. These emergent spirals are called parastiches and we will deal with them more next week. The colors will also line themselves up into spirals, and this is because Netlogo provides only a limited number of colors and they eventually repeat. For a fixed number of repetitions try to find an `angle` for which the stamps seem to fill the space most efficiently. You may need to adjust the step-size to see this. What is that angle?

Branching:

We will be discussing branching structures in the next few weeks. We can write a branching procedure that is remarkably similar to the spiral program you wrote in the tutorial.

1. Copy and paste the procedure above for `spiral` and rename the new copy `branch`. Here are the key differences:
    a. Remove the `stamp` commands.
    b. Instead of turning right after each step have the turtle produce a bud of a new branch at some `angle` to the main branch. You can do this by replacing the `rt angle` command with a `hatch 1 [rt angle]` command. (`hatch` is a command that makes a turtle create a copy of itself).
    c. Instead of new branches getting longer with each step they get smaller (say by a decay factor). Add a `decay-factor` slider on the interface which varies between 0.1 and 1 in steps of 0.001, and adjust the `set step` command in your repeat loop accordingly.

    d. In the branch procedure you do not need to repeat very many times and if you do the computer will crash! So add a new slider `branching-number` that only ranges between 1 and 10 and then adjust your repeat loop accordingly.
2. Make the changes suggested above and add a `branch` button to your interface. Try it out (you may need to make the initial `step-size` larger, or perhaps create a new slider called `branch-size` and adjust your procedure accordingly – that way you won't affect your spirals). You'll probably get a fairly rudimentary looking tree. Let's enhance it.
3. To make your branch a bit more symmetric make the turtle also hatch another bud on the other side. (eg `hatch 1 [lt angle])` Try it out. What types of branches can you create (ferns, palms, etc?)
4. You will actually get a somewhat more realistic branch if you let the turtles take an additional `step` immediately after producing its left and right buds. Do this.
5. Add the following enhancements to your model
    a. Branches are usually brown. When you first create your turtle make it brown and also set the `pen-size` to 5.
    b. In the repeat loop have `pen-size` decrease by the `decay-factor` each step.
    c. Branches usually have green leaves at the end. Before clearing the turtles at the end of the procedure ask the turtles to change their color to green and get them to `stamp` their shape on the canvas. You may want to create a leaf shape for them first and change the size or your turtle to be small enough to get a nice pattern.
6. After adding the enhancements try them out for a variety of settings. You should get remarkably different branches with small changes in these variables.
7. To make your branch procedure even more realistic instead of branching at a fixed angle, branch at a random angle. You can do this by replacing `rt angle` with `rt random-float angle` (and like-wise with the left branch).
8. Do the same with step size. Instead of `fd step` do fd `random-float step`. Note: In computing a float is another name for a decimal number. Thus a command like `random 4` randomly reports one of the whole numbers 0,1,2,3. `random-float 4` gives any decimal number (up to 16 decimal places) between 0 and 4.

When you have finished upload your program to our moodle site.