Collecting Rocks on Mars

The Moses Biological Computation Lab at the University of New Mexico does research on robots. The robots are *biologically inspired*; the robots simulate the behavior of ants, who are exceptionally good at finding and gathering resources as a group.

The goal of the lab's research is to deploy these robots on Mars and other distant planets and asteroids to gather resources such as rocks and ice. These materials can be used for building and creating a steady water supply. This will help us to colonize Mars.

In this exercise, you will create a NetLogo model that simulates the Moses Biological Computation Lab's Swarmie robots collecting rocks on Mars.

Follow the instructions step-by-step. **Don't skip ahead!**

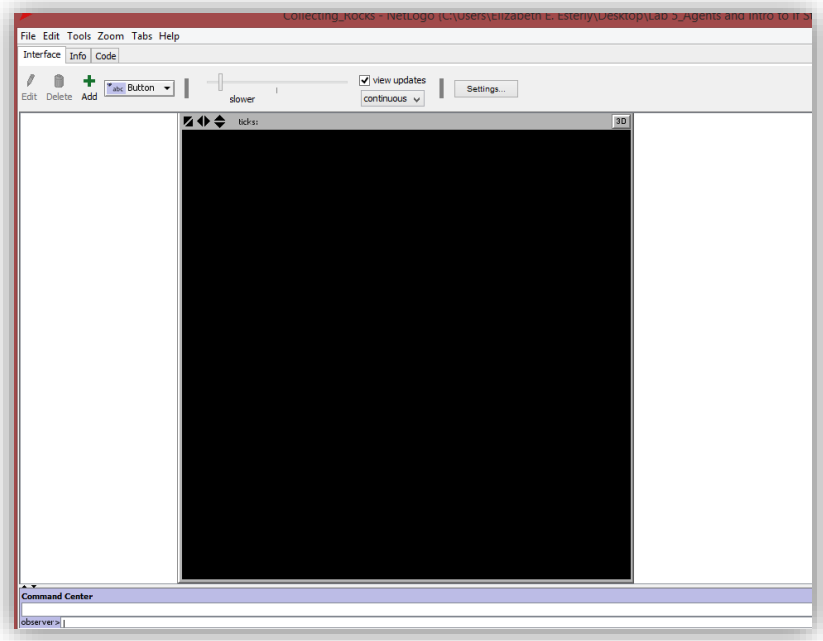
Some parts of the program are already done for you. While you are working**, don't change the code that's already been written, or the program may not run. Once you've finished the program,** you may experiment by changing the code.

1) Getting Started

1.1 Create a new folder called "NetLogo project" on the Desktop.

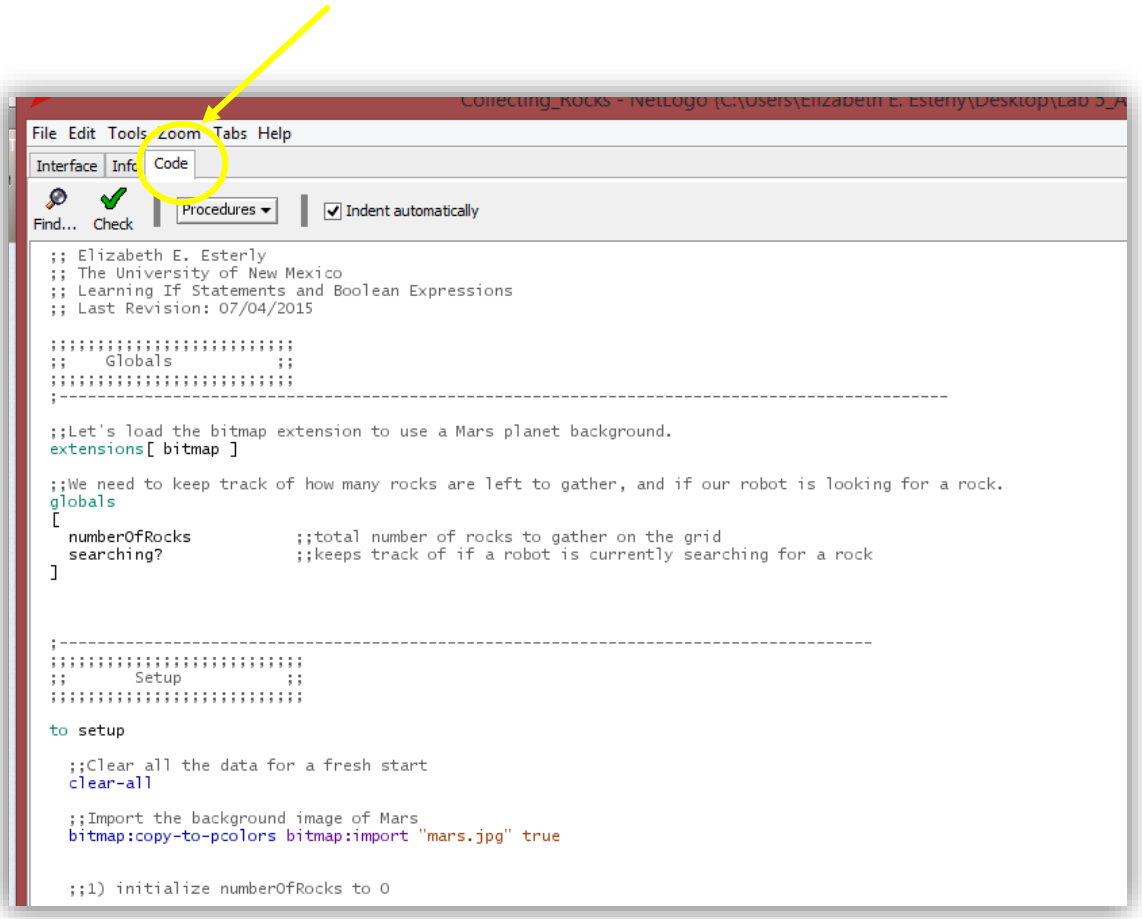
1.2 Download the .nlogo file and the mars.jpg file and put them in the folder.

1.3 Double-click the .nlogo file and it will launch in NetLogo.



1.4 Click the Code tab at the top of the screen.

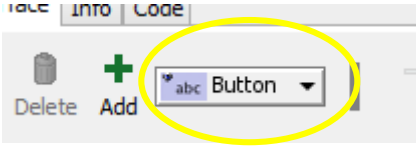
The gray code is comments. Comments are useful for explaining what a program does. NetLogo ignores comment code when it runs the program. The black and colorful code is used to run the program. Notice that some code in the Globals and Setup sections is already written for you.



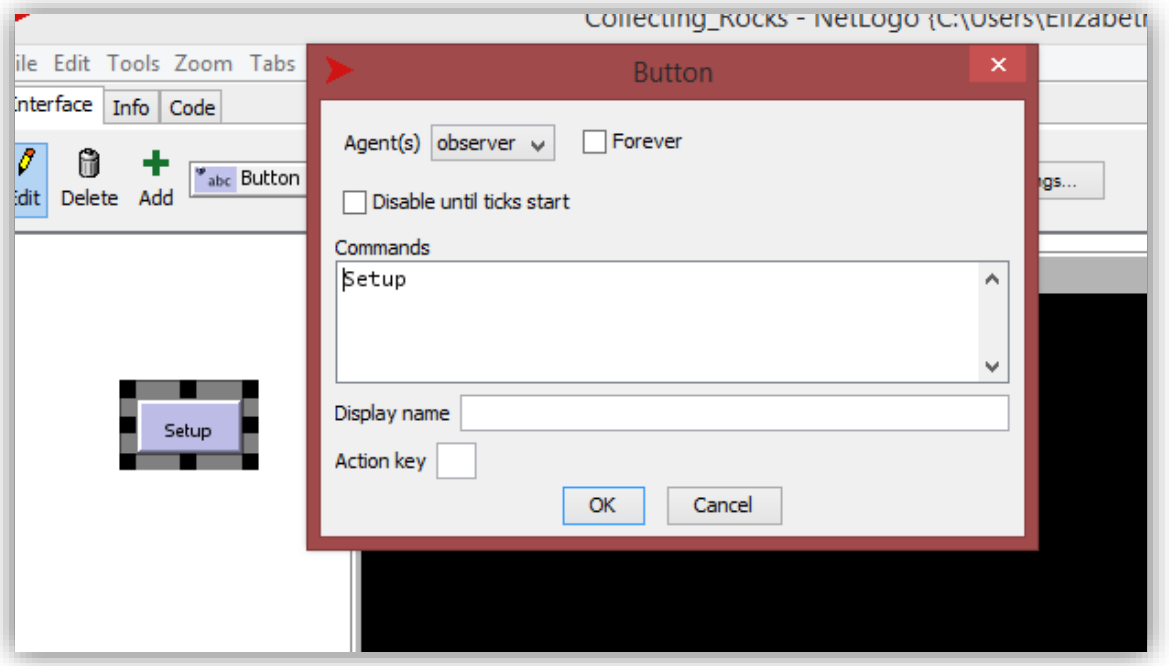
2) Setup the program

2.1 Click the Interface tab.

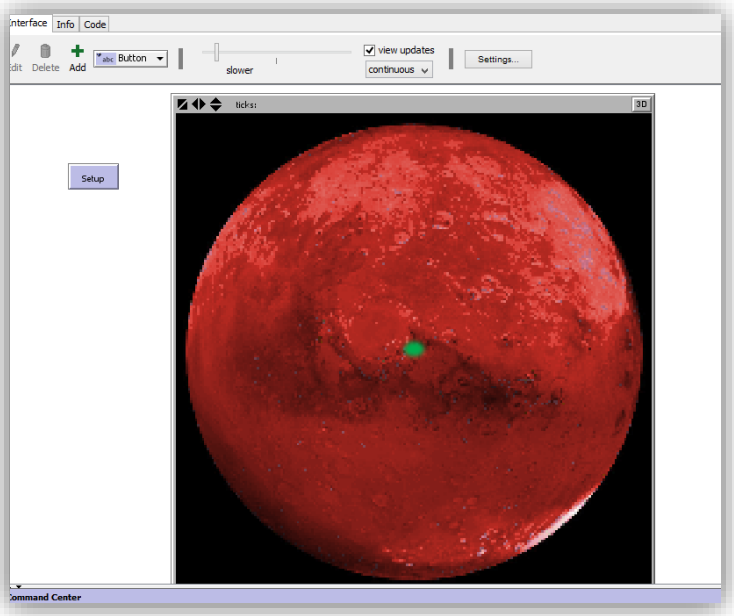
2.2 Create a new button by clicking the button next to the Add button and selecting Button from the drop-down menu (first option).



2.3 The cursor changes to crosshairs. Click to the left of the black display panel. A button appears and a dialogue screen comes up. Type Setup in the Commands box and click OK.



2.4 Click your new Setup button. You should see an image of Mars appear in the window after about 20 seconds. The green circle in the middle is your base.



2.5 Now it's time to write some code! Add the following code into your program in the Setup section under the comments. The comments are there to guide you. Read the comments to understand what your code is doing.

Be careful to type the code exactly as it is written here, including dashes, quotes, and brackets, or it may not work.

```
;;1) Set numberOfRocks to 0 to start
set numberOfRocks 0


;;2) Set robots to start in search mode
set searching? true


;;3) Here we create a turtle. NetLogo's default model is a turtle.
;;Change its shape from a turtle to a robot.
;;Set its size to 8, so you can see it more clearly.
set-default-shape turtles "robot"
create-turtles 1
[
  set size 8
]
```

2.6 Save your program.

2.7 Go back to the Interface tab. Click the Setup button again. A small robot will appear on your base.

2.8 Let's add some rocks for the robot to pick up. Go back to the Code tab and add the following code:

```
;;4) Let's set a maximum of 25 random patches to the color yellow.
;; pcolor means patch color.
;;The yellow patches will represent the rocks that we are gathering.
;;We don't want to make rocks that are off the planet, so we check if the random
;;patch selected is black. We won't put a rock there if it is.
;; != means does not equal.
;;When we add a rock, we also add 1 to the numberOfRocks variable to keep track of how
;;many rocks we have.
ask n-of 25 patches
[
  if pcolor != black
  [
    set pcolor yellow
    set numberOfRocks ( numberOfRocks + 1 )
  ]
]
```

2.9 Go back to the Interface tab. Click the Setup button again. You should see some randomly placed yellow rocks along with the robot.

2.10 Now we'll also add some clusters of rocks. Go to the Code tab and add the following code:

```
;;5) Let's make a maximum of 10 additional clusters of rocks for the robot to pick up.
;;We don't want to make rocks that are off the planet, so we check for black patches like before.
;;This time we also check for yellow patches, because we don't want to put a rock on top of
;;another rock.
;;If the randomly selected patch is not black and not yellow, then we make it yellow (add a rock)
;;and add 1 to our numberOfRocks variable like before.
;;This time, we ask the patches to the North, South, East, and West to become rocks also and add
;;those to the variable numberOfRocks also.
ask n-of 10 patches
[
  if pcolor != black and pcolor != yellow
  [
    set pcolor yellow
    set numberOfRocks ( numberOfRocks + 1)
    ask neighbors4
    [
      set pcolor yellow
      set numberOfRocks ( numberOfRocks + 1)
    ]
  ]
]
```

2.11 Go back to the Interface tab. Click the Setup button again. You should see some extra clusters of rocks appear.

2.12 Save your program.

3) Let's go!

3.1 Click the Interface tab.

3.2 Follow the same procedure as you did for the Setup button to create a new button called "Go".

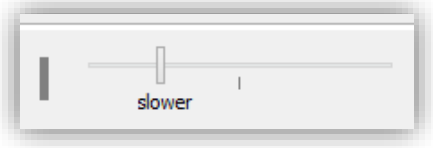
3.3 It's time to make the robot move! We are going to build the Go set of instructions one piece at a time. The Go instructions are broken down into several parts. First, we'll have the robots do a "wiggle walk." Add the following code to your program:

```
;;1) make the robots move
wiggle
```

3.2 But this doesn't give any specific instructions of what a wiggle is! We need to tell the program how to "wiggle." We can do that by writing a *procedure* (also called a method or a function). Scroll down to the wiggle procedure. Write it by adding the following code:

```
;-------------------------------------------------------------------------------
;;;;;;;;;;;;;;;;;;;;
;;    wiggle      ;;
;;;;;;;;;;;;;;;;;;;;

to wiggle

  ;; 1) turn right 0 - 40 degrees
  right random 40


  ;; 2) turn left 0 - 40 degrees
  left random 40


  ;; 3) turn around and face the origin if we hit the edge of the planet
  ;; (the patch color is black at the edge of the planet)
  if pcolor = black [ facexy 0 0 ]


  ;; 4) go forward one patch
  forward 1

end
```

3.3 Click the Interface tab, Click the Setup button and wait for the model to finish setting up. When it's done, click the new Go button. The robot moves! You can change the robot's movement speed by adjusting the slider near the top of the screen:



3.4 Go to the Tools dropdown menu at the top of the screen and select Halt. This stops your program.

3.5 Great! The robot is moving. But now we need to gather rocks and bring them back to the base. In **2.5**, we set the robots to start in searching mode. When a robot finds a rock, we need to turn off searching mode so it will return to the base. Let's set this up by adding the following code in the Go section:

```
;;2) ask the turtles (robots) if they are searching or not

ifelse searching?

;;3) tell them what to do based on the answer
[look-for-rocks]
[return-to-base]
```

This ifelse statement can be broken down. If the robot is in search mode, it has not yet found a rock, and it should look for rocks. Else (otherwise) it has found a rock and should return to the base.

3.6 Save your program.

4) Controlling the Robot: Picking up Rocks

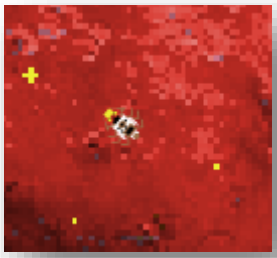
4.1 The robot is walking around, but it's not recognizing rocks yet. Let's make it do that first. Assuming this robot has sensors all around it, it could detect a rock in any direction. So we will "ask neighbors" if there's a rock. Ask neighbors means checking the 8 patches around the robot. And remember, our rocks are yellow. So the robot is checking if any of the 8 patches around it are yellow.

If there is a rock, we need to turn off search mode, and remove the rock from the list of rocks to pick up.

Scroll down to the look-for-rocks procedure and enter the following code:

```
;---------------------------------------------------------------------------------------
;;;;;;;;;;;;;;;;;;;;
;; look-for-rocks ;;
;;;;;;;;;;;;;;;;;;;;

to look-for-rocks
  ;;1) Ask the 8 patches around the robot if the patch color is yellow

  ask neighbors
  [
    if pcolor = yellow
    [

  ;;2) If it is, take one rock away, and set search mode to false.
  ;;   Change the patch color to red where we removed the rock.

      set numberOfRocks ( numberOfRocks - 1)
      set searching? false
      set pcolor red
    ]
  ]

  ;;3) If they're not searching anymore, that means they found a rock.
  ;;Change the robot's shape to the one holding a rock.

  if not searching?
  [
    set shape "robot with rock"
  ]

end
```

4.2 Click the Interface tab. Click the Setup button and wait for the model to finish setting up. When it's done, click the Go button. The robot picks up a rock when it finds one!



4.3 Go to Tools → Halt to stop the program.

4.4 Save your program.

5) Controlling the Robot: Returning to Base

5.1 The final step is to make the robot return to the base to drop off the rock when it finds one. The base is the green patches in the middle of Mars. It is placed at the origin (0, 0), so we'll have the robot face the origin as it moves to guide it towards the base. The robot will know it found the base when it sees a green patch. When it finds the base, we'll have it drop off the rock and turn search mode on again, so it looks for more rocks.

We'll use an ifelse statement like we did in the Go procedure. If the robot found the base, we change the model to the one without the rock (drop the rock) and turn search mode on again. Else, we haven't found the base yet, so the robot should face the base.

Scroll down to the return-to-base procedure and enter the following code:

```
;---------------------------------------------------------------------------------
;;;;;;;;;;;;;;;;;;;;
;; return-to-base ;;
;;;;;;;;;;;;;;;;;;;;

to return-to-base

;;1) If the patch color is green, we found the base.
  ifelse pcolor = green
  [

;;2) Change the robot's shape to the one without the rock,
    set shape "robot"


;;3) and start searching again.
    set searching? true
  ]


;;4)otherwise, we didn't find the base yet--face the base
  [ facexy 0 0 ]

end
```

5.2 Click the Interface tab. Click the Setup button and wait for the model to finish setting up. When it's done, click the Go button. The robot picks up a rock when it finds one, and then returns it to the base!

5.3 Save your program.

6) Make Your Model More Awesome

Here are a few things you can do:

6.1 Add multiple robots.

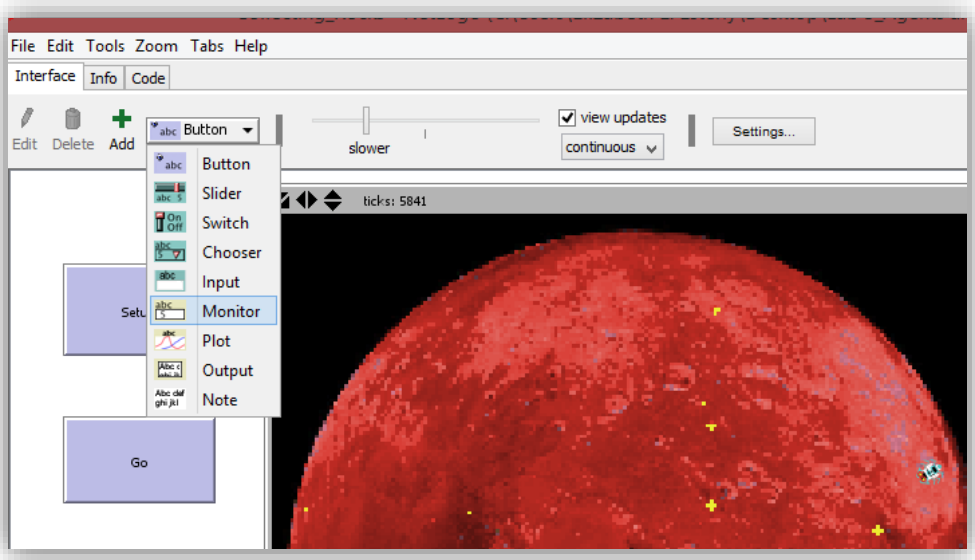
This one is easy. In the Setup block, change `create-turtles 1` to `2`, or, `3`, or `20`…however many you want! Click Setup and then Go on the Interface tab to try it out.

```
;;3) Here we create a turtle. NetLogo's default model is a turtle.
;;Change its shape from a turtle to a robot.
;;Set its size to 8, so you can see it more clearly.
set-default-shape turtles "robot"
create-turtles 2
[
  set size 8
]
```

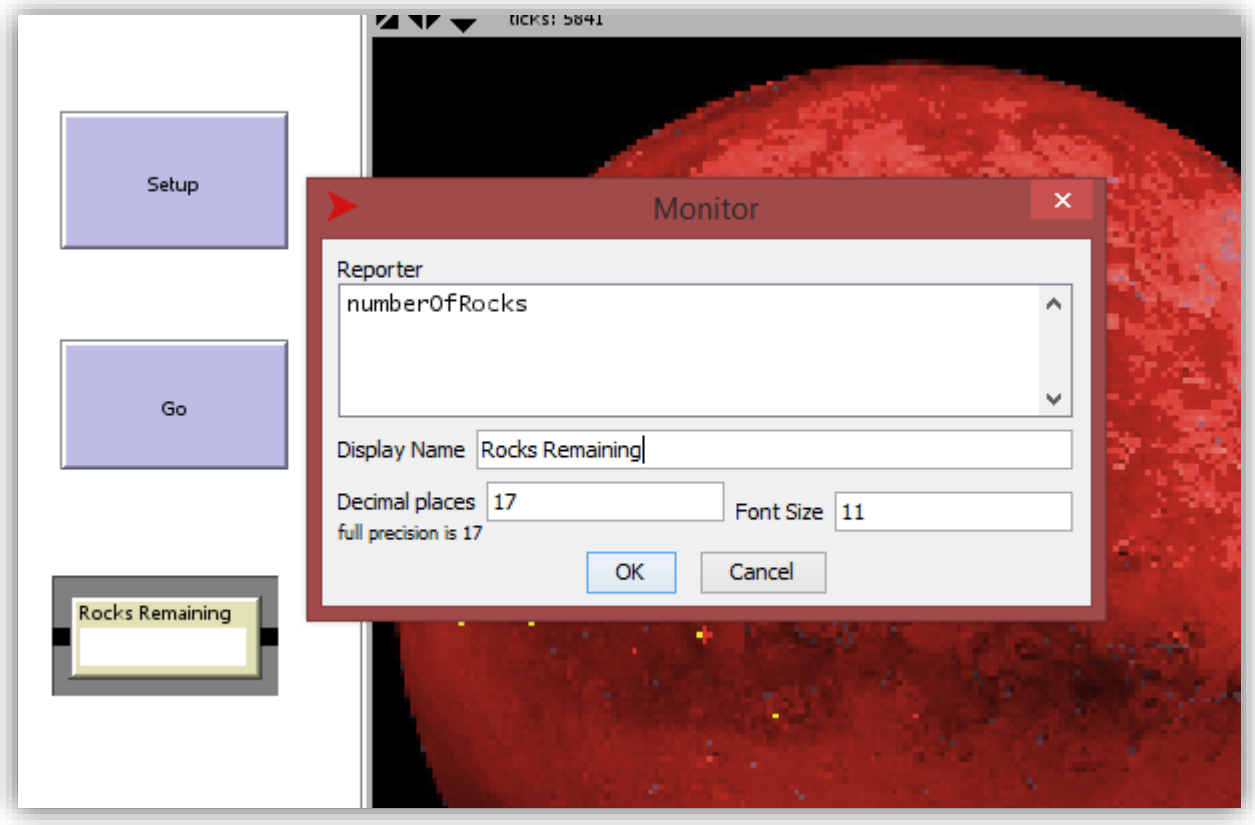
This will make 2 robots!

6.2 Make a monitor! You can keep track of how many rocks the robots have left to gather.

Click on the Interface tab. Click on the drop-down menu next to Add and select Monitor.



6.3 The cursor changes to crosshairs. Click below your Go button. A small window appears and a dialogue screen comes up. Type numberOfRocks in the Reporter box and Rocks Remaining in the Display Name box. Click OK.



6.4 Click Setup and then Go on the Interface tab to try it out.

6.5 Change the size and position of your buttons. Right-click any of your buttons and click Select. You'll be able to resize and move your buttons.

6.6 Load a model from the Library. NetLogo has tons of pre-made models available for you to play with. Go to Files → Models Library to open one. Just make sure you save your program before you do!

That's it! Thanks for participating.

NetLogo is free to download, so I hope you'll continue programming at home!

If you have any questions, you can always email me at elizabethesterly@gmail.com !