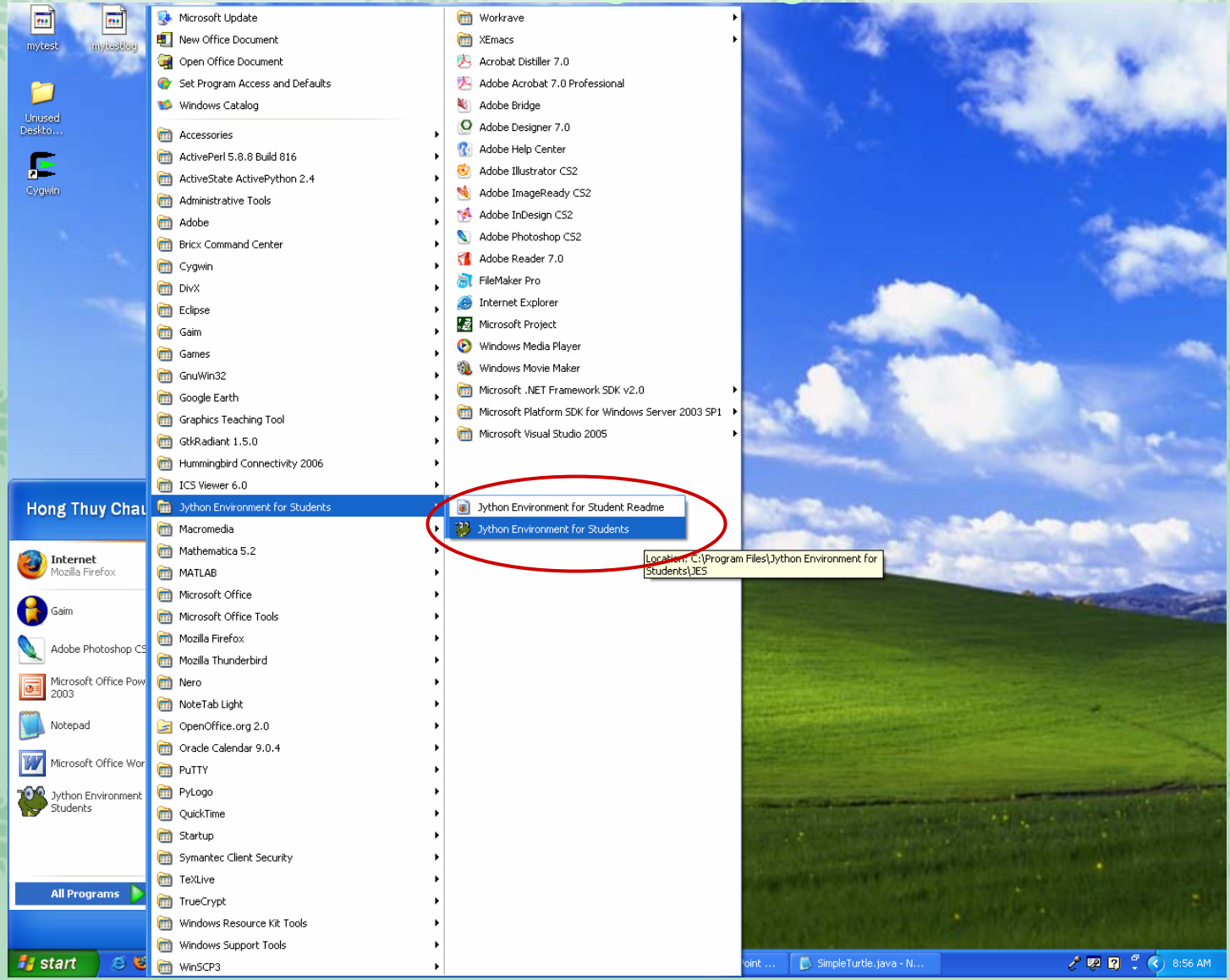
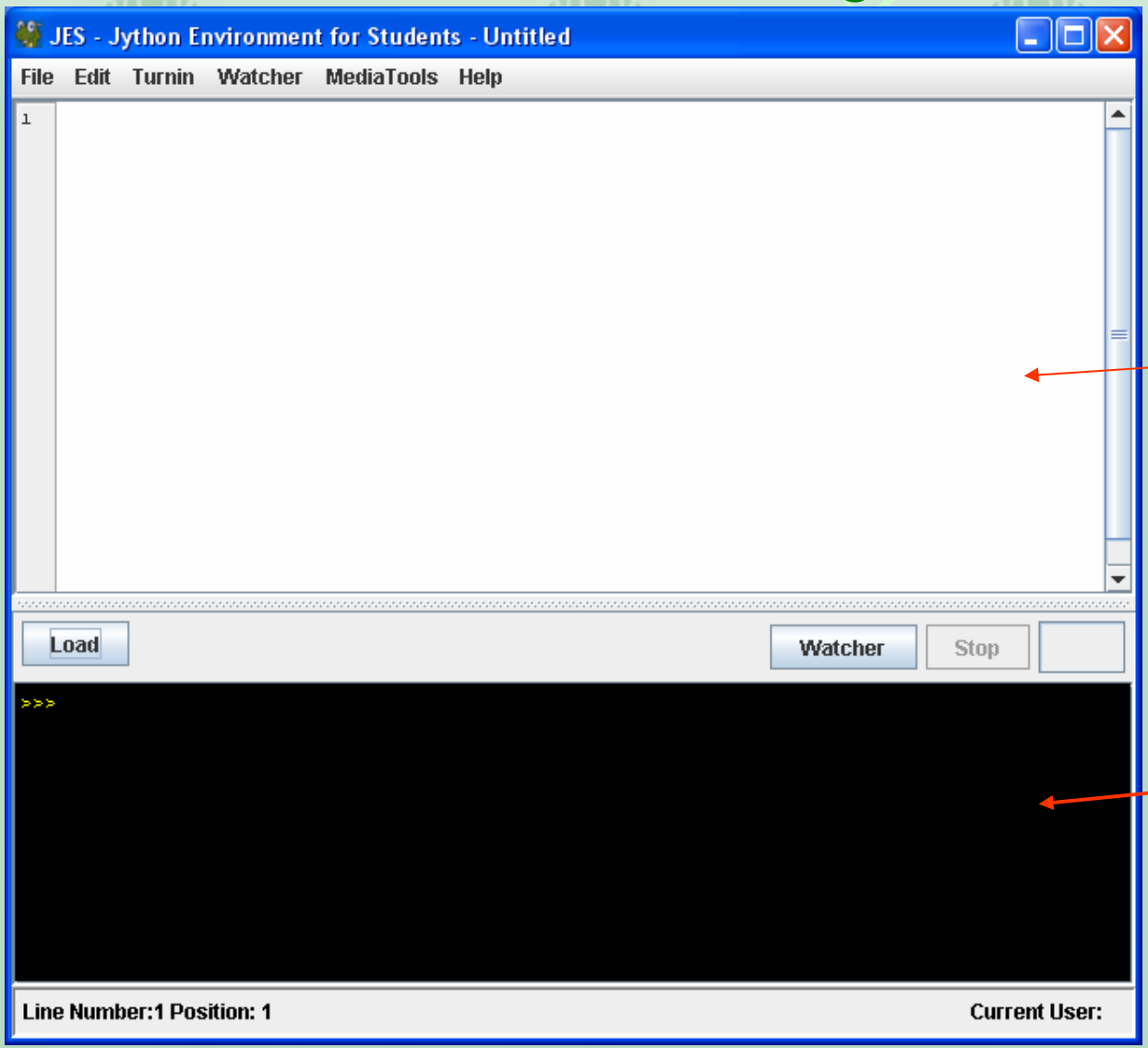
LOGO

Starting Logo



Understanding the Screen



Where you will write programs.

You can just type methods or commands in here.

Ex: t.forward()

A Little Logo History

What is LOGO?

- A programming language developed at the MIT Artificial Intelligence Lab
- It's easy to learn, but once you know the basics you can do extremely complicated things
- Logo code has been used in telecommunications, multimedia software and robotics

What is the TURTLE?

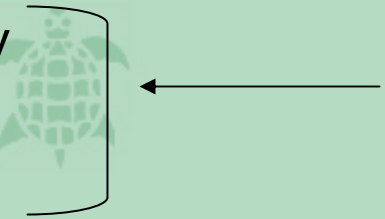
- Originally, the turtle was a robotic creature that sat on the floor and was directed to move around by a user typing commands into a computer
- Today, it is an icon on the Logo screen (in our version of Logo the turtle is represented by a triangle)

Creating your Turtle

- Before doing anything, you want to create you turtle. However, a turtle needs a World to live in. So do all of these steps first.

  Type these commands and hit enter after each one:

```
import ModelDisplay
import World
import Turtle
w = World()
t = Turtle(w)
```



Don't worry much about these lines. They simply allow you to use the turtle methods we will be using later…

Basic Logo Commands

forward(number) You can put a number inside the parentheses. The larger the number the farther the turtle will go. If you don't put a number, the turtle will move 100 steps.

t.forward(40)

backward(number) You can insert a number here, too, as you did for forward. This time the turtle will move backwards.

t.backward(33)

turn(number) This command turns the turtle the specified number of degrees. Use a negative number to turn left, and a positive number to turn right.

t.turn(-33)

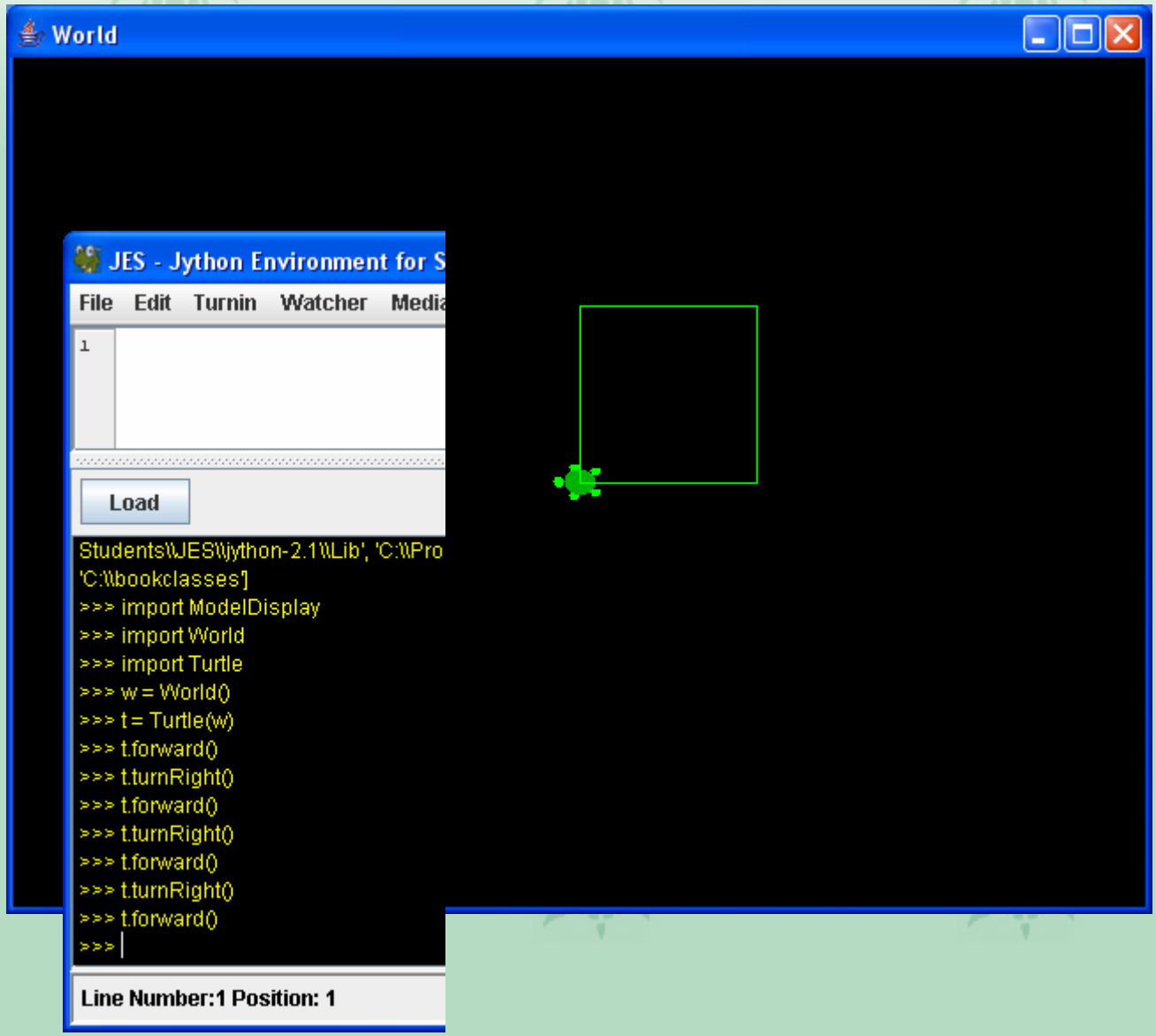
turnRight() The number represents how many degrees you want the turtle to turn. The example below will turn the turtle to the right 90 degrees.

t.turnRight()

turnLeft() This command works exactly like the RIGHT command, only it turns the turtle to the left.

To draw a SQUARE:

```
t.forward(100)
t.turn(90)
t.forward(100)
t.turn(90)
t.forward(100)
t.turn(90)
t.forward(100)
```



IMPORTANT

When you are typing commands into the input window, you might notice that sometimes there are messages in the commander window that you didn't type in.

These are called ERROR MESSAGES.

When you give Logo input that it doesn't understand, this is how it tells you about what it didn't get. Here are a few examples of Logo ERROR MESSAGES:

Your Input: t.forward(10 20)
Logo's Error Message: "Your code contains at least one syntax error"

Your Input: t.back
Logo's Error Message: "Your code contains at least one syntax error"

Commands for Pen

penUp() This command will cause the turtle to pick up its “pen” so you can move the turtle without drawing a line.

penDown() This command is used to put the “pen” back down so you can draw again.

setPenColor(Color color) Use this command to set the color of the pen the turtle is drawing with. When inserting a color, you MUST type it this way:
java.awt.Color.<your color>

t.setPenColor(java.awt.Color.**blue**)

setPenWidth(number) Use this command to set the width of the pen line the turtle Draws by inserting a number

A Few More Commands…

setShellColor(color) – sets the shell of the turtle to the specified color

- Ex: t.setShellColor(java.awt.Color.blue)

setBodyColor(color) – sets the body of the turtle to the specified color; it also sets the color of your pen by default

- Ex: t.setBodyColor(java.awt.Color.green)

setHeight(number) – sets the height of your turtle

setWidth(number) – sets the width of your turtle

turnToFace(Turtle) – makes the turtle face another turtle (which you specify) on the screen.

updateDisplay() – updates the screen

clearPath() This command will erase the lines that the turtle has drawn, but will not move it back to the center of the canvas

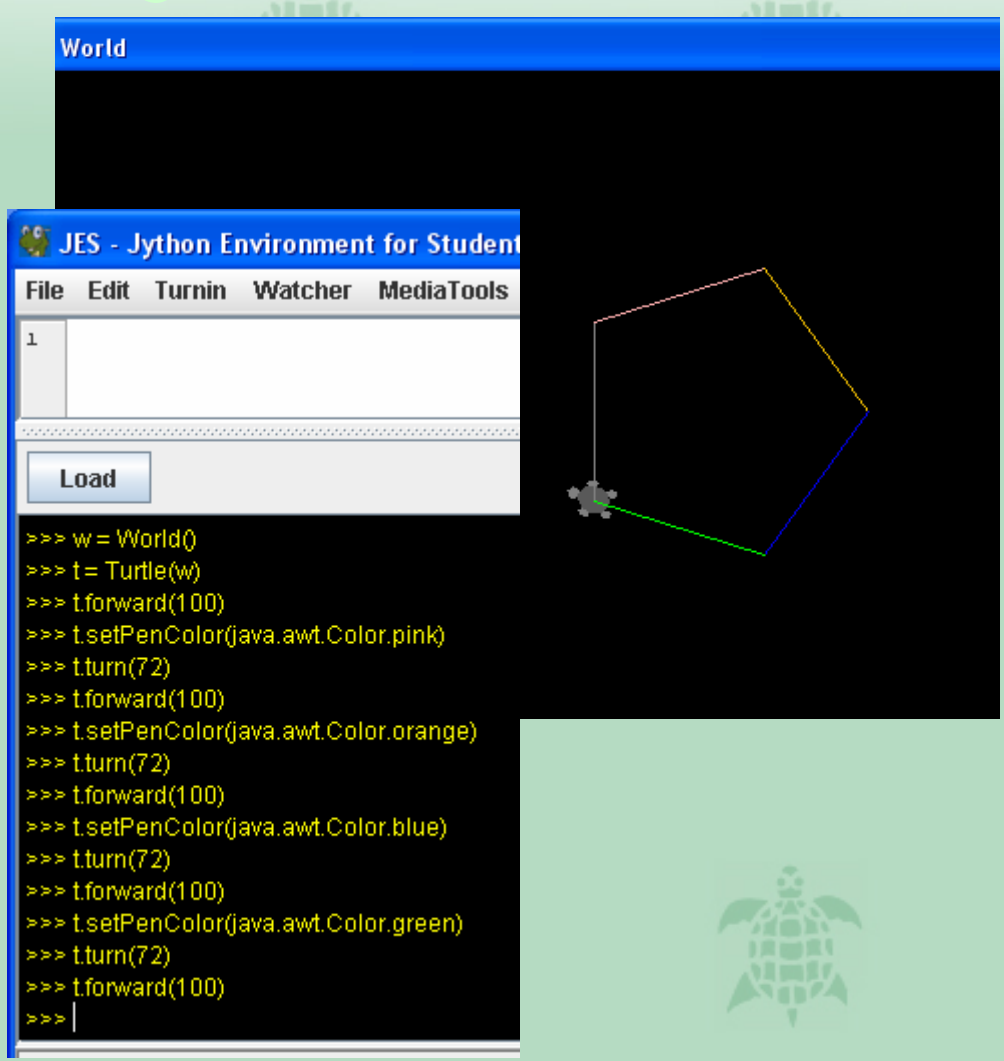
moveTo(x, y) This command will move the turtle to the coordinates you specify (you must think of the screen as a coordinate plane)

Clearing The Screen

- When you want to clear the lines on your screen, use clearPath(), and then use updateDisplay().
- If you want to start complete from scratch, you must exit your world, and make a new world (and then new turtle).

An Example using setPenColor:

```
t.forward(100)
t.setPenColor(java.awt.Color.pink)
t.turn(72)
t.forward(100)
t.setPenColor(java.awt.Color.orange)
t.turn(72)
t.forward(100)
t.setPenColor(java.awt.Color.blue)
t.turn(72)
t.forward(100)
t.setPenColor(java.awt.Color.green)
t.turn(72)
t.forward(100)
```



Because the turtle turned right 72 degrees 5 times, it made a PENTAGON

Programs in LOGO

You can teach Logo how to draw squares or circles or other shapes by writing a PROGRAM.

Every time you want to write a program it must be in this form:

```
def <name of your program>(t):
    <command 1>
    <command 2>
    …
```

Make sure you indent!

You don't have to worry about the details of this, it simply makes it so that the program will be done on the turtle you created

For example:

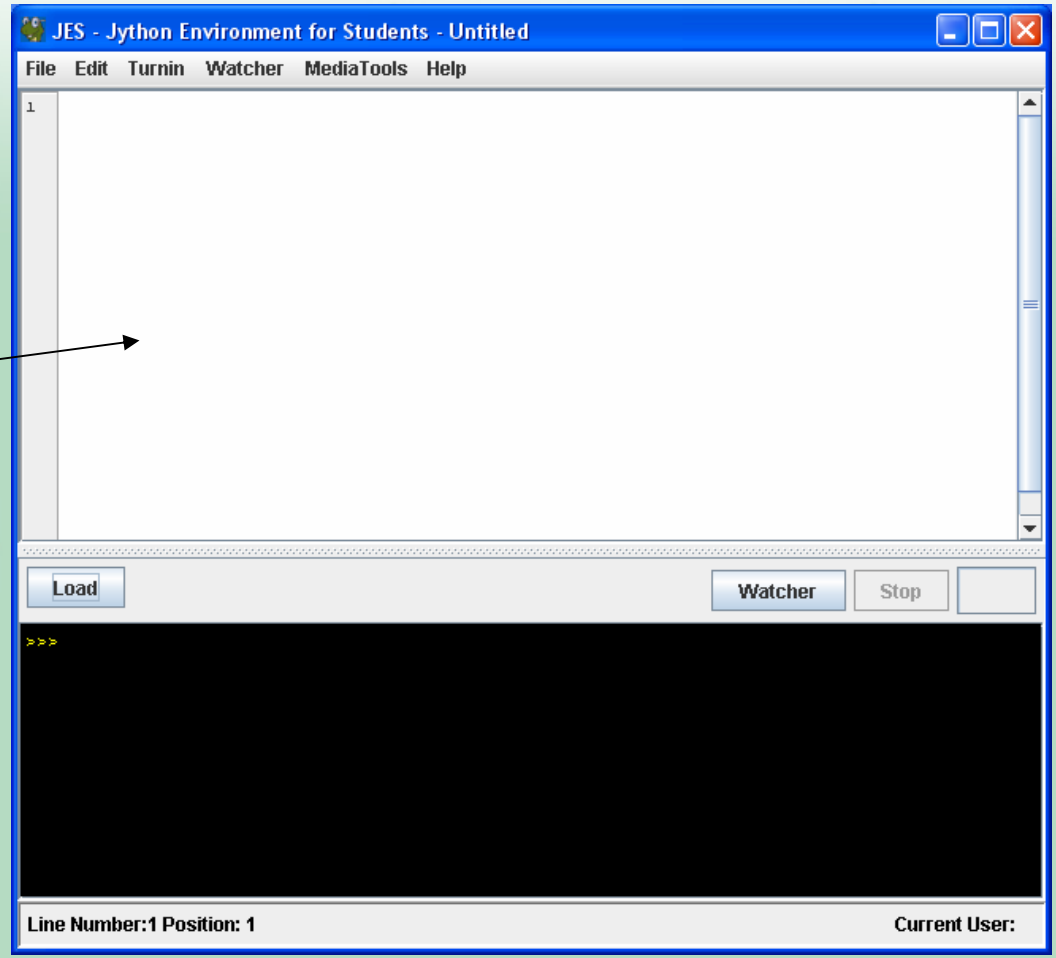
```
def moveTurnMove(t):
    t.forward()
    t.turnRight()
    t.forward()
```

name of the program

the commands (indented!)

NOTE:

You want to write your programs in the top white screen, NOT the bottom black screen.



Writing Simple Programs

```
def square(t):
    t.forward(100)
    t.turnRight()
    t.forward(100)
    t.turnRight()
    t.forward(100)
    t.turnRight()
    t.forward(100)
    t.turnRight()
```

Can you tell what this will draw???

Using Loops

You can get your turtle to do one (or several) things repeatedly, *without typing them again and again* using **loops.**

The commands that will be repeated (or "looped") are indented after the line where you type "while"

This example will move the turtle forward 100 spaces, 4 times, using a **while** loop. In all, the turtle will move forward 400 spaces.

```
def square(t):
    x = 4
    while x > 0:
        t.forward(100)
        t.turnRight()
        x = x-1
```

Can you figure out what this code is doing step by step?

Try These Questions:

1. Can you make a square yourself? How could you make a bigger square? Smaller square? How could you make a square with a different color?

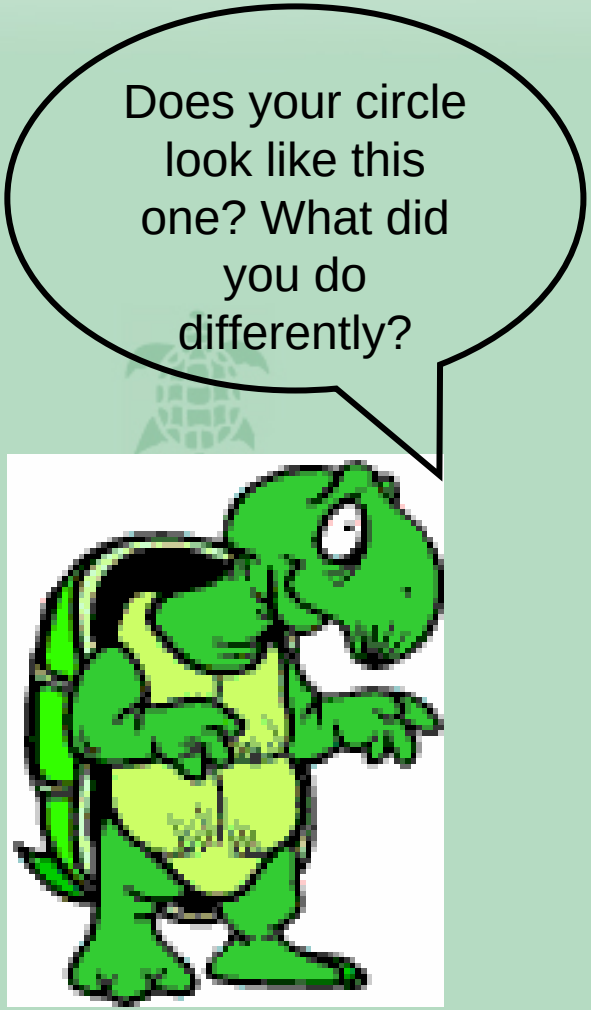
2. Can you figure out what this command will draw before you try it?

```
def shape(t):
    x = 3
    while x > 0:
        t.forward(100)
        t.turn(120)
        x = x-1
```

Now type it in and see if you were correct.

3. Here's a difficult one: Try to make a circle using a while loop.

Answer: Here's one way to make a CIRCLE



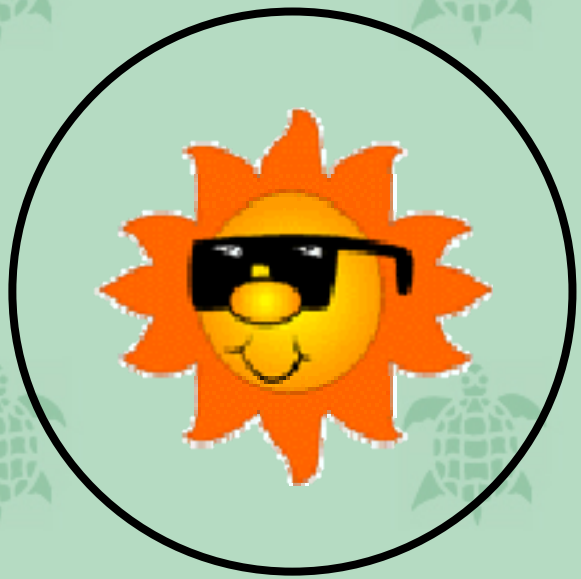

```
def circle(t):
    x = 36
    while x > 0:
        t.forward(10)
        t.turn(10)
        x = x-1
```

Try playing around with the numbers inside forward() and turn() to see the different kinds of circular shapes you can make!

Your First LOGO Program

Try making a program that will draw a circle.

Remember to use **def <name of program>(t):** and remember to use proper **indentation**!



After writing the program, test it out by typing in the program name.

CIRCLE

Save it as **<name of program>.py**

Click LOAD.

Test it out (in the black screen) by typing:

```
circle(t)
```

and hitting enter.

A Pentagon

Try to write a program that will draw a purple pentagon:

Here are some methods that might be helpful:

setPenColor(java.awt.Color.purple)

forward(75)

turn(72)

Saving your Programs

Since you will probably want to use the programs you wrote today some other time, you should SAVE them.

To do this, go to the "FILE" menu at the top of your screen and select "SAVE AS". Remember to save it like this! **<name of program>.py**

Click LOAD

Type in the BLACK SCREEN

<name of your program>
ex:
shape(t)

Here's a few more interesting (and more complicated) things Logo can do: