PowerMath Logo Tutorial

James J. Baker, Ph.D.

Engagement in learning mathematics; sound mathematics instruction in a dynamic language format that enables students to actually build and edit mathematical products; and orientation to the STEM pathway.

Working through this tutorial will introduce you to a computer language that reveals the "code" of mathematics—and that students of all ability levels find accessible and interesting. Paired with The Expert Mathematician middle school mathematics curriculum, the generative learning environment of Logo can inspire mathematics learning for all.

First Principles
Educational Solutions
Minneapolis, Mn 55408
www.expertmath.org

PowerMath Logo Tutorial

James J. Baker

Contents

A Note To Teachers

"Mathematics evolved from counting, calculation, and the systematic study of shapes and motions.... "[it] is a body of knowledge centered on such concepts as quantity, structure, space, and change. Mathematicians seek out patterns wherever they are found.*

This general description of math is easy to state, yet takes years of dedicated study to internalize mathematical thinking. How can we present it in a way that delivers the full scope of what students need for a good foundation through middle school while enticing them to really engage the media? Research shows The Expert Mathematician answers this challenge.

Computer programming, or "coding", is a way of using rules, numbers and logical symbols, together with natural language, to reason. Mathematics is a coding language that is difficult for two reasons: it is unfamiliar and demands precision. Computer languages that incorporate mathematics are used in real work to construct and test hypotheses, design plans, graphics, buildings and bridges, build analytical models, conduct the sciences and the world of health care, compose music and many other applications. Math teachers know the list of real math applications is long. But it has proven difficult to connect the excitement and rewards of real mathematical work that people do with classroom practice in a way that engages and motivates students to invest the time and effort to succeed in math.

Logo is a "kid-friendly" computer coding program, that can engage students in using a real mathematical language to compose, express and edit formal mathematical ideas. Students become accustomed to using math as a living language to create real things, like experts use math in real work. And Logo can strongly support teachers in helping students to draw out mathematical insights as they stay actively engaged in real mathematical work. A Logo driven math program can be a great catalyst for energizing a classroom learning community. Plus, Logo is a real programming language, so combined with The Expert Mathematician lesson materials, teachers can support students along a user friendly path to studying Information Technology. This means that using The Expert Mathematician program, your school will become an important contributor to the United States' STEM initiative. The 3 main features of The Expert Mathematician

instructional system are: **engagement** in learning mathematics; **sound mathematics instruction** in a dynamic language format that enables students to actually build and edit mathematical products; and **orientation** to the **STEM** pathway.

Let's get started

Java is a free programming language that works on Apple, Wintel PC's and Linux computers. Powermath Logo runs on Java.

```
To start the Logo software, open the "PowermathLogo.jar"
program.
If it doesn't open, try downloading Java from www.java.com
and try again.
If PM Logo still does not open, call the tech support
number, 612-588-9177.
```

If PM Logo has opened correctly, you will see a screen with 3 windows:

1. Turtle Graphics
2. Procedures
3. Command Center

(If the tutorial window blocks your view, click on the "Untitled" window behind it.)

The Logo Window System

Logo has 3 windows, or work areas. You actually work in two of them, while the third shows the results of your work. Some of the commands work like word processing programs. Below you will find more detailed descriptions and explanations for the window areas, but let's begin by making something.

Using the Command Line

There is one line at the very bottom of the Command Center window. You should see a blinking cursor there. This is the Command Line. Here, you can write and run individual commands as well as procedures. Try one, like this:

```
Type: forward 50
Press the Return or Enter key.
```

Note on Spacing

In written language you need spaces between "parts of speech". Spacing is likewise used to distinguish "parts of math." such as commands and inputs, as you will learn in the tutorial. If the computer gives you an **error message,** spacing is the first thing to check. Next, check spelling.

You've just created a piece of code and run it in "real time." You didn't have

to give the computer any additional instructions–just hit Return or Enter and your code drew a 50 pixel line. You're working in turtle geometry mode, so the "turtle" actually drew your line. **Speed:** If your line drew so fast you didn't see movement, drag the speed slider a bit to the left.

```
Type: right 90
Press the Return or Enter key.
Type: forward 50
Press the Return or Enter Key
Type: right 90
Press the Return or Enter key
```

Can you finish drawing a square?

Errors? If you make a mistake, you can start over by Clearing Screen (Type: CS and press Return).

You have worked out the individual commands to create a square. Notice that the commands you have run accumulate above the Command Line. From there, you can copy them. Try it. Just like you would in a word processor, click and drag your mouse pointer to select those commands for the square. Use your keyboard or **Edit menu** to copy them. Next click in the Procedures window (on the right side of the Logo window system) and **paste** your commands there. Delete the question marks.

Procedures Window Preview

In the Procedures window you build and edit complete mathematical "procedures," and give them a name. A procedure is a set of commands together with math operations that can be operated as a program to create something mathematical. The result can be a description, an analysis, a puzzle, a solution to a problem.

You can operate procedures that you have "defined" in the Procedures window by typing their name, and any other inputs, on the Command Line. Examples below show you how to use "inputs". This a gentle lead-in for students to "formal" thinking and formal language—-a crucial link to understanding math that is very difficult to instill in most standard math programs. Here, they get it through guided experience.

Example 1: How To Name, Define and Run Your Procedure

Procedure: MYSQUARE

At the top of your list of commands in the Procedures window, type the words:
TO MYSQUARE
and at the bottom, or end, of the command list, type the word:
END
(case doesn't matter).
In the Procedures window, you should now have:

```
TO MYSQUARE
forward 50
right 90
forward 50
right 90
forward 50
right 90
forward 50
right 90
END
```

You have created a complete procedure in the Procedures window and given it a name. You can now run all those commands under the procedure's name. The name can be anything that's not a Logo command; here, the name use here is: MYSQUARE. Caps are used for emphasis, case doesn't matter.

A procedure will always be named. It must begin with the word, "TO," and will finish with the word "END." This form alerts the computer to run the commands between TO and END.

You will run your procedure from the Command Line, but first you have to "**define**" it in the computer's memory. Your cursor is in the Procedures window.

Hold down the **Control** key on your keyboard and press F. ("Do a Cntrl-F").

This defines your procedure and jumps the cursor to the Command Line.

NOTE: You can't just click from the Procedures window to the Command Line to run your procedure. Cntrl-F is required to define the procedure. Cntrl-F toggles back and forth between the Procedures window and the Command Line. Try it a couple of times. So if you're on the Command Line and want to edit your procedure, just Cntrl-F to the Procedures window, make your edit and Cntrl-F back to the Command Line to rerun your procedure. Those are the basics of the Logo window system.

Now you can run your procedure, but first clear the screen in the Turtle Graphics window. On the Command Line:

```
Type in: CS (Press Return or Enter).
Type in: MYSQUARE (Press Return or Enter).
The turtle should draw a square.
```

Command Center Review

The Command Center provides 2 basic functions.

First, on the Command Line, you can type and run individual commands that are independent from the Procedures window. This is convenient when you are trying out just one part of a mathematical idea (a concept with an operation).

Second, the "history" part of the Command Center–above the Command Line–shows which commands have been run. These can be copied to the Procedures window, so you can work out commands one at a time in the Command Center, then include them in a procedure you are building. Nice feature.

The Logo Turtle

The Logo icon is a "turtle," and a distance of one pixel is a "turtle step." Developmental psychologists involved in inventing Logo discovered that suggesting a "living form" (the turtle) as a mathematical object increases student interest and engagement. And by mathematically simulating how living things move in space, it permits the learner to think more naturally in mathematical terms. The irony of a speeding turtle also tends to hold attention. Thus the "turtle" icon and the name, "turtle geometry."

Working Smarter, Not Harder

Computers are great time savers when anything needs to be done over and over. What would be a quicker way to make the square? We could copy and paste the first line 3 more times. That would save time, but still wouldn't be very efficient.

A better way is this: we can tell the computer to REPEAT one set of commands instead of writing the set 4 times. Notice the word "times." The REPEAT command is like a multiplier. REPEAT multiplies, by the number the user tells it to, all the commands in a "list" which is inside square brackets [ ].

Example 2 helps you learn how REPEAT works.

EXAMPLE 2: "Multiplying" lists with a REPEAT command.

```
First, clear the screen with a CS command (on the Command
Line).
Then, again on the Command Line, Type: REPEAT 4 [FD 50 RT
90]
Press Return or Enter. If it ran too fast, drag the speed
slider a bit to the left.
```

Using REPEAT is more efficient and it lets students see the logical relationships between the terms, but even using REPEAT, should we have to type all those commands every time we want to make a square?

A review. You've done this above with MYSQUARE. In the **Procedures window**, you can group your commands under a name. Then you only have to type the name of the procedure to operate all the commands at once. Computers are designed to save time!

EXAMPLE 3: Naming a REPEAT procedure in the Procedures window.

```
Click in the Procedures window.

Type the following:

TO SQUARE
REPEAT 4 [FD 50 RT 90]
END
```

NOTE: As noted above, all named procedures you program must begin with the word TO (and a name) and must be typed in the Procedures window. Type END on a separate line after the last command.

"Define" your procedure: Hold down the CONTROL KEY and PRESS F. ("Do a CNTRL-F"). You can just click back to the Procedures window to edit your work; but then you must do a CNTRL-F back to the Command Line to run it.

```
CNTRL-F to the Command Line.
Type- CS (and Return or Enter).
Type- SQUARE (and Return or Enter)
```

Students will save their work, like this: Create a new folder In a separate place from Powermath Logo where other students cannot tamper with it.

Then from the **File** menu, choose **Save.** Type in **Square** for the name and save it in your folder.

Skill Games and Teacher Files

Look at the top of the Turtle Graphics screen. You should see:

File Edit Skill Games Teacher Files

Students use the Skill Games. Teachers use the Teacher Files.
The Teacher Files are password protected. The password is: **tfss123**

Teacher files are, for the most part, copies of procedures students make in the curriculum. These come in especially handy if a student loses his or her work. The teacher can then quickly restore the lost work using the Teacher Files. To see how this works, click on Teacher Files and unlock them with the password, **tfss123.**

Then scroll through the list of procedures until you find the SQUARE.logo procedure. Open it. This is the completed student SQUARE procedure. As you can see, it contains a variable so different sized squares can be drawn. You will add a variable like this to your SQUARE procedure later in this tutorial. Just leave it for now and let's look at the Student Games.

Skill Games provide calculation practice appropriate to the level being learned and at carefully spaced intervals.

Choose a skill game:

```
From the Skill Games menu, choose ADD.GAME.
```

Skill games are programmed in Logo and you can see all their code when they load.

Run your skill game. The cursor should be blinking in the Command Line.

```
Type in: ADD.GAME Press Return or Enter.
```

You will get a problem to solve.

Type your answer on the Command Line. Press Return or Enter. The computer will tell you if it's right!

If it is right, you will get another problem to solve.
If it is not right, the computer will tell you to "Try again."
You will lose a point for a wrong answer and gain 10 points for a correct answer. When you get 100 points, the game will conclude. You will have to type in ADD.GAME again to get a new game. (You can also use the "Go" and "Stop" buttons.)

Using Variables

A SQUARE is a polygon. You can modify your SQUARE procedure to create any polygon. To do this, you use a VARIABLE. A variable is an attribute (feature) of a mathematical object that can take on a range of values/numbers we call INPUTs."

In the following example, ANGLE is an attribute that can be *varied* with a range of inputs, so we will call it a variable. Instead of the constant, "RT 90", in the repeat command, we now have "RT :ANGLE."

NOTE: If you use a variable with commands in your procedure, you need to show the variable name on the TITLE LINE of your procedure. Indicate that a word in the title is a variable name by use of a : (colon) at the beginning of the variable name–such as :ANGLE, as shown in the next example.

The variable name can be any word or letter that is not a computer command. Commands in this tutorial are in upper case, but Logo isn't case sensitive so you can use either.

NOTE: When you add a new procedure to the Procedures window, any other procedures already there can remain there, or you can Select and Delete them. It's best, though, to begin with a new Logo "document."

To open a new document, choose File>New.

EXAMPLE 4. From Squares to Other Polygons.

Click to the Procedures window and type the following (either single or double spacing will work):

```
TO POLYGON :ANGLE
REPEAT 4 [FD 50 RT :ANGLE]
END
```

CNTRL-F to the Command Line.
Clear your draw screen (workspace) by typing CS and Return.

Type: POLYGON (in upper or lower case) and Return.

Error. "NOT ENOUGH INPUTS".

For now, if you use a variable instead of a constant in a Logo command, the computer does not assign a value to the variable; the user must do that. From the Command Line, if you try to operate a procedure with a variable in it but forget to assign a value ("input"), you'll see the error message, "Not enough inputs".

Would you like to create a hexagon? CNTRL-F to the Command Line. Retype POLYGON, then leave a space and type in 60. The computer knows to put the number 60 into the variable.

Your command should look like:

POLYGON 60 (Press Return or Enter).

Do you see that something else needs to be changed to get a complete HEXAGON? The list of commands in the [ ] brackets makes a line and an angle. How many of these do you need to complete a HEXAGON? Try changing the 4 to a 5.

CNTRL-F to the Command Line.

Type POLYGON 60 and press Return or Enter. Did you get a correct hexagon? If not, what needs to be different? Edit the change and try again.

Making "Functions" Do The Work

PROBLEM: When you run your POLYGON procedure you can change the :ANGLE variable with any input (value) you want. But a polygon has both angles *and sides.* Should you have to go back into the procedure to change the number of REPEATs for the correct number of sides every time you change the ANGLE value to make a different polygon? Why do that when you can teach the computer to change the number of REPEATs *as a function* of the size of the ANGLE?

How do you do that? A "function" in arithmetic relates one term (variable or constant) to another. We want the number of times a command list [ ] REPEATs to be a *function* of the size of ANGLE. The angles of any regular polygon add up to 360 degrees, and the math parts (terms) that make the polygon in the list [ ] are a line for the sides, and a number of degrees for the angles [FD 50 RT :ANGLE]. You can use a second VARIABLE in the REPEAT command to determine the correct number of REPEATs (for the polygon you want to make) from the *size of the angle*. It sounds more complicated than it is.

> Click to the Procedures window and take the 6 out of your REPEAT command. The "6" was a constant, a fixed number ("input", in our new vocabulary).
> In place of the 6, type 360/:angle . Your REPEAT command should look like: REPEAT 360/:angle . You've just made the input for REPEAT *a function* of your input for :ANGLE.

Any input for :ANGLE will now "break down," i.e.,"divide," the total of something (here, 360) to the number of parts we need for a hexagon (360/60 = __). Fill in the blank.

NOTE: You can define any angle with whole numbers and decimals, but the 360/:angle function will only work with angle sizes that are factors of 360. Examples are 1,2,3,4,6,9,10,12 and doubles and triples of some of those integers. Logo will throw an error message that an "INTEGER IS EXPECTED" if other angle inputs are used with the Repeat 360/:angle function.

A powerful feature of Logo is everything instantly operates so students can see the fine details of their logical edits at work. And, in keeping with modern learning theory, you want students to discuss and explain their

investigations to/with each other and to/with the teacher! This step supports reorganization of schemata so that the explained insights are more likely to be remembered--and used creatively (and analogously) in another construction.

```
Do a CNTRL-F to the Command Line and type in HEXAGON 60.
```

Can you see how Logo can help simplify teaching concepts and number logic that many students find so difficult to grasp? When students begin teaching the computer to write, operate and revise mathematical statements like this, they tend to get more interested in why the logic works. Of course, teachers help by prompting and extending investigations–as well as leading students toward correct understanding.

EXAMPLE 5. Three steps to a diagonal function.

Suppose we wanted to make a square, bisected diagonally. We know that all sides and angles of a square are equal. A diagonal exactly bisects a square into two congruent triangles. If every angle of the square is 90 degrees, how many degrees might be in an angle that exactly cuts a square in half, diagonally? If you're not sure, experiment.

```
CNTRL-F to the Command Line and run SQUARE.

Now, type ST and Return.
```

Next, use the RT (right turn) command to test an angle that will head diagonally to the opposite corner, and an FD (Forward) with some number to test the distance across to the opposite corner (length of your diagonal). It's easiest to see the correct length of "diagonal" if you hide the turtle (HT). Multiple short FD commands can help you pace it off more accurately.

Line's too long? To ERASE the line, type PE (Pen Erase), then HOME and return. You can also BK (back up) some number with ERASE on, to delete part of a line segment).

NOTE: After using PE (Pen Erase), Type PD (Pen Down). Then try again.

Error: No line. If you move the turtle and no line draws, the pen is on ERASE or PEN UP, just type PD in the Command Line and return, then try again.

Investigations are often messy in real life. If you end up with a mess, type CS (clear screen) on the Command Line and start over. Use the arrow key to move the cursor up a couple of lines in the Command Line to SQUARE and run it again—there's no need to retype a command. CLEAN will also erase all lines, but leave the turtle cursor at its last point.

For your size 50 square, you have discovered that the corner-to-corner length of the diagonal is about 70 turtle steps. Insert a simple FD command in SQUARE to automatically draw the diagonal next time you run it. (Tip: When you insert a command, it will run in a logical order from left to right, top to bottom—after everything that comes before it, and before everything that comes after it. So *where* you insert your edits make a difference.)

In this example, we'll call the variable "size" (As noted above, the variable's name can be any word except for a computer command, although it's a good idea to use names for variables that suggest what the variable does).

In the Procedures window, type the following:

```
TO SQ :SIZE
REPEAT 4 [FD :SIZE RT 90]
END

CNTRL-F to the Command Line.
First, CLEAR SCREEN. Type- CS and return.
```

Run SQ. If you get an error message, you forgot an input for :SIZE, or have a typing error in the procedure. Type SQ, a space, and some number smaller than 100. Clear Screen (CS-return).

So we can now create a square of any size with very little effort. Make a size 70 square. Now put in the diagonal. Hmmm. Will it be the same size as it was in the size 50 square?

Type:

```
TO SQ :SIZE
REPEAT 4 [FD :SIZE RT 90]
RT 45
FD 70
END
```

Try it. So, just how long should it be? You'll need to pace it off like you did with the size 50 square. If you overshoot the opposite corner, you can erase: Type PE and HOME-- or BACK (BK) and the last number you tried (for example, PE BK 16). AFTER USING PE, Type Pen Down (PD) before next forward command.

If you have the TEM General Math manual, complete the table on page 85, lesson 2.

(This is not essential to complete your tutorial.) If you have not yet purchased The Expert Mathematician lesson series, divide the length of your square's diagonal by the length of your square's side. You can let Logo do it for you (although we encourage students to calculate by hand

until they understand the logic of number operations).

On the Command Line, type:

```
print 100/70
```

You can see that Logo knows precision to 4 decimal places! You can round it off to 1.4 for these exercises or use all 4 decimal places.

Now you have a relationship between the length of the square's side and the length of its diagonal. At least we have a hypothesized relationship; we can call it a ratio. But does it hold for every size square? Test it by trying different size squares:

```
TO SQ :SIZE
REPEAT 4 [FD :SIZE RT 90]
RT 45
FD :size * 1.4
END
```

For your next exercise, try going back to POLYGON and modifying it with two variables; one for size, one for number of angles. On the Command Line, you might want to try incorporating POLYGON as a term in the list of a REPEAT command. You could make multiple polygons, each one set off from the other a certain number of degrees.

Example 6: The internal (or "local") variable.

Suppose now we wanted to be able to calculate the diagonal for any square, then output the result for use in another part of the program. For this, we will need to make an "internal" variable, which is quite easy using the "Make" command. (These are used frequently in statistics and "sheet" type programs to combine two or more variables to make a third one). The computer will recognize Make as your command. The name of your new variable is shown with a **single quote**, as in "name. In this example, we are creating "diag from an existing variable, :size multiplied times 1.4.

```
TO SQ :SIZE
REPEAT 4 [FD :SIZE RT 90]
RT 45
Make "diag :size * 1.4
print :diag
END
```

Notice that the value printed in the workspace (upper left), but the computer won't do anything else with that value unless you tell it to. In the following edit, you are prompted to include a command that will move the turtle FORWARD a distance defined by your diag variable. Recall, to apply a variable name to an action, you need to precede it with a colon.

```
TO SQ :SIZE
REPEAT 4 [FD :SIZE RT 90]
RT 45
Make "diag :size * 1.4
```

In this line, type a command that will draw your diagonal line using :diag, whose value will be an output from the Make "diag command (then remove this whole instruction).

```
pr :diag
END
```

Notice an internally produced variable doesn't need to be identified on the title line of your procedure. To repeat, a **quote** precedes the variable name when you Make it; then, when you use it, a colon precedes the variable name.

Finally, we can "embed" variables in sentences that will calculate and print a result right within the sentence (abbreviated, "se" or you can spell it out).

For example, try this:

```
TO SQ :SIZE
REPEAT 4 [FD :SIZE RT 90]
Make "diag :size * 1.4
RT 45
FD :diag
pr (se [The diagonal of this square is] :diag [turtle steps
long.])
END
```

EXAMPLE 7. Using multiple variables; and calculating and applying a compound expression.

You may want to calculate the diagonal more accurately. You might recognize the diagonal as a common hypotenuse of two triangles. If so, then its length can be calculated using the Pythagorean Theorem. (This by the way is well developed in The Expert Mathematician's prealgebra volume.) All you need to do is Make a variable that calculates the square root of the sum of the squares of two sides of the quadrangle. It sounds more complicated than it is.

Here, we will work with a rectangle; you can add a second variable :SIZE2 for a second side. It looks something like the SQ edit below (recall that you can name a variable almost anything, but its name can't have spaces). This one is offered as a challenge. There are lots of hints and it uses much of what we've already explored.

CHALLENGE:

Add a second :Size variable on the title line, as suggested above (remember the colon and that the variable names need a space between them, but not within them). You'll need to include it in the REPEAT command list. Add into the list any other commands you think you'll need. For Pythagoras ("Mr. P"), we need two dimensions, as in a rectangle.

You have changed the variable names on the title line, so they need to be changed in the body of the procedure as well. The square root formula is done for you, but "diag" is still in there.

Finished, this procedure will 1) calculate Mr P's formula, 2) print it, 3) apply it with a FD command to actually draw the diagonal, exactly, to 4 decimal places (the precision is automatic), and 4) incorporate the variable in the descriptive sentence in place of "diag". You might as well hide the turtle, (HT) as a final step before your END command.

Of course, when you've got all that working, edit in some other variations to...investigate. TEM lessons teach you how to use other commands for multiple variations, including adding color.
For illustration purposes, we add a little trigonometry into this last example.

Acrtan is just one of the many trig functions Logo knows. This lets us calculate the correct angle for the diagonal of any rectangle. Will it also work for a square? Conjecture. Then try and confirm (or refute).

There are omissions in this procedure. Try it. Read the challenge paragraph above. Fix any bugs you can identify, then try it again. Read error messages in the Command Center for suggested fixes. Note the question marks; something needs to replace them.

```
TO SQ :SIZE1 ?????
REPEAT ? [FD :SIZE1 rt 90 :size2]
rt 90 - arctan (:size1 / :size2)
Make "diag (sqrt (:SIZE1^2 + :size??? ))
pr (se [The diagonal of this square is] :diag [turtle steps
long.])
END
```

EXAMPLE 8. Here's another application of the arctan command and some utility commands.

Logo allows "ST" for showturtle, and "HT" for hide turtle.
Wait units are in milliseconds. The "pacing" slider under the workspace can also be used, but "hard coding" turtle speed teaches a little more coding.

Colors range from 0 - 7. Colors can be changed from within a procedure, or from the Command Line where the color command must be alone on the line. As always, use a Return or Enter to activate it.

```
to tri :x :y
showturtle
setc 1
lt 90 fd :x rt 90 fd :y
rt 180 - arctan (:x / :y)
make "diag sqrt (:x^2 + :y^2)
fd :diag
wait 2000 hideturtle
end
```

Note below that pinwheel "calls" tri, so tri needs to be in the Procedures window when you run pinwheel.

Supply a constant for the ? in the repeat command and inputs (on the Command Line) for :x and :y variables.

```
to pinwheel :x :y
repeat ? [tri :x :y]
end
```

Can you define a circle? Use a repeat command. It will be round but not perfectly smooth. There will be a relationship between the number of repeats and the size of each increment. Return to your POLYGON procedure for insight.

Skill Games

We believe in the importance of being able to mentally manipulate quantities, however defined, according to logical rules for specific problem solving applications, is an essential part of learning math.

You will notice that we try to avoid densely pedantic wording like that in all lesson materials. What we mean, in other words, is students should know how to calculate, by hand. Skill Games, prompted throughout TEM lessons, foster those skills. We recommend 10 minutes of calculation practice daily before students begin working on conceptual development lessons.

Students should record their work on paper, write out—and do—the problem, and show solution strategies. No calculators. If students are working in dyads, which we strongly suggest (see the Teachers Manual), each student should do their own set of daily calculations, though they can help each other. When tested, they should be individually accountable.

Annotate A Lesson

The Expert Mathematician lessons provide sound instruction in mathematical concepts and calculation practice. As you are becoming familiar with a lesson before giving it to students, we suggest you pencil in some "deepening prompts" that coax students to elaborate lessons beyond what is given. These will lead students to extend their understanding through critical analysis and experimentation. Then, students should get accustomed to writing up their final lesson in the Turtle's workspace.

Simply by clicking there, they can write text. We suggest posting a class "rubric" for guidance on write-ups. Examples are given in the Structuring Learning file on the CD.

TEM Beyond the Classroom

If students have a computer at home, or can use one outside of class at school, they can work on lessons remotely. Lessons can be written up and emailed to the teacher or a workmate for collaborative work. See suggestions at Save Graphics.

Save Graphics

Peer learning outside of class on assignments or on extracurricular projects is a great way to get students into mathematical habits of mind. Extra credit might be given for work turned in. But if more than one student works on a project should they all get the credit? Should they each get partial credit? How can you know who did what?

The **Save Graphics** command saves the turtle's workspace as a jpeg graphic. The jpeg file can be opened in any graphics editor, attached to an email, dropped into word processing file, etc, though it's no longer an editable file so it can't be opened again in Logo or changed. One advantage of this feature is to enable multiple students to collaborate on a creative work.

Example: Janelle creates something, signs it, Saves as Graphic, emails it to a peer to show what she's done–including a provocative question or two (she makes up based on teacher suggestions or previous lessons). Mike gets the email, replicates the first part (he won't be able to see Janelle's Logo procedure(s) unless she copies them into her email), so he needs to figure out the code.

Mike codes his replies to Janelle's questions, writes it up (staying within rubric guidelines), signs his name, Saves as Graphic, adds a question or two of his own in his email or right in the workspace before saving, and sends it back to Janelle.

A similar "team game" could be set up within a school, with students working at different computer sites around the building—or around the world. There are lots of possibilities.

Anything students do in Logo will increase their ease with using formal language and critical thinking with mathematical ideas. It would seem to make sense that connecting kids via "technology tools to think formally with" can help build interest and socially catalyzed fine grained analytical thinking with peers that can lead to pursuit of higher mathematics, science and engineering. Interest and accessibility motivate students along this path. Logo activities can provide both.

Facilitating Learning

Teachers find a balance between "telling" how something works, and coaching students to discover the good explanation among themselves. The balance can vary with individual students, but always attempting to get them to stretch for the understanding themselves as much as possible. This is a habit of mind that is crucial to successful problem solving. Persistence will pay off, but we need to be sensitive to their tolerance for frustration and, as necessary, issue "connecting conceptual bits" they will need to reach the new understanding. Helping students get it on their own fosters deeper learning--and ultimately more satisfaction and self-efficacy—confidence that "I can do this." This stimulates higher motivation and positive attitudes about math. Our data show that.

Some students instantly grasp how the syntax works. Others need to be coached more to grasp it. Wait time between prompts supports logical leaps. Allowing students to regulate their own pacing (within realistic constraints) gives them a chance to make connections and realize the value of their effort.

The following beginning exercise is an example of how this might sound:

"Suppose now, that we would like to make different size squares *without having to edit* the procedure to change the input each time." Pause. Ask: "Do you get what I mean?" Ask students to restate the problem. In this way, you help them to construct a blueprint.

Continuing, you might say: "To change the square's size, what term (variable) in the square's [list] needs a different number (input)?" Point to variables you're referring to. (Assess: Ask student to restate/rephrase.) Ask the student to describe the properties of a square, just to reinforce the concept and prompt an insight.

"The angle? If we change the input for the angle, will we still have a square? If you're not sure, try it and see." What other variable is there in a square that will affect the square's size? There are only two and we've decided we can't change the input for ANGLE and still have a square (ensure that students can explain this clearly before stating that conclusiion). Then, "What is the one other variable of the square (the list in brackets) that you can change with a different number to make a different size square? Wait.

As soon as students have an idea to test, leave them to test it, while assuring them you will check back in a couple of minutes. Facilitating and coaching as students use generative tools that are engaging pays off, even if sometimes it seems as though just telling them the answer would keep things going a lot faster. It will. But if real understanding, confidence and interest are the goals, teachers are at their best when coaching and facilitating learning.

A Final Reminder

Many students enjoy creating geometric shapes, as they should because geometry is the mathematical specification of space and shape—the makeup of the world we live in. The socratic teacher can convert what might seem like mathematical "doodlings" in Logo to spontaneous formalized investigations. But, primarily, Logo is about learning to problem solve with math, to acquire habits of thinking by analogy, reasoning inductively and deductively, working with others in learning mathematical terms, expressions, and principles, and using that knowledge to set up, test, confirm—and defend—problem solutions. In short, students are learning to think and act "formally." As explained in the Teachers Manual, these habits are reinforced by teachers and peers. Over time, disciplined, formal thinking evolves. And, as already noted, these are the new basic skills of mind that employers of career track jobs tend to value.

We have only scratched the surface in this tutorial. Please read through the Teachers Manual for more tips and aids for getting the most productivity out of PowerMath Logo and The Expert Mathematician instructional system. The teaching tips do not all need to be applied for The Expert Mathematician to be effective; but gradually including them can further enhance learning. Each teacher will find the right balance. Training is available for those desiring professional development.

Reference

* http://en.wikpedia.org/wiki/Mathematics