math.code Worksheets

Contents

Convert Between U.S. Measurements (liquids)

While this activity can be done in any programming language, the write up below refers to specific things in the Scratch programming environment. If using something other than Scratch, adjust your work accordingly.

You are going to write some code to convert between ounces, cups, pints, quarts, and gallons.

For additional information on variables and doing calculations in Scratch, see the video clip below or the many other ones available.

https://www.youtube.com/watch?v=MT5u5_iFaJA

Create variables for ounces, cups, pints, quarts, and gallons. Each of the variables should be visible in Scratch.

Using information from the video clip mentioned above, remove the current image and create a button called "Gallons To". In the script area, drag over the Event "when this sprite is clicked" then do the necessary calculations to convert the number of gallons to the corresponding number of quarts, number of pints, number of cups, and number of ounces.

A very important part of coding is to test out your code to make sure it runs both without any errors and also that it produces the correct results. In order to test this out, you need a way to change the Gallons variable to the slider type which allows the variable to be changed using the slider. Double click on the variable and each time you do it will cycle between the default display, just the value, and the slider display. You should hand calculate some of the conversion values then check your code to make sure the answers match. Check things like 1 gallon, 5 gallons, and 10 gallons.

If you have a Scratch account or if you are using the iPad app, be sure to save your work as you go. Give your project a meaningful name.

Next, add a new button alongside the "Gallons To" button called "Cups To". As you have probably guessed, when this sprite is clicked, it should take the number of Cups and calculate the appropriate number of ounces, pints, quarts, and gallons. You will need to change the Cups variable into a slider. Be sure to test with a small value like 1 and a larger value like 20 and other values to make sure all of your calculations are correct.

If time permits, add in buttons for Ounces To, Pints To, and Quarts To.

Challenge: Start a new project where the value of cups can be changed using the slider then when the button is clicked the number of cups are converted to the maximum number of gallons in that many cups, the remaining cups are converted to the maximum number of quarts, the remaining cups from that converted to the maximum number of pints, and the remaining value in the cups spot. So for instance, 79 cups would convert to 4 gallons, 3 quarts, 1 pint, and 1 cup.

Pattern Generator

Write some code that will generate a pattern. Your pattern generator will create squares with sides of length 5. You will have 11 squares across and 8 squares down.

Once completed, turn a copy of your code into your teacher and if time proceed to Challenge 1.

Note: Use a different code file/project for each challenge so you always have a working copy the last completed part of this activity. You can use the last completed project as the starting point for the next project.

Challenge 1: Ask the person running your program to type in the length of the square. Use that length instead of 5 to generate the pattern. Once completed, turn a copy of your code into your teacher and if time permits, proceed to Challenge 2.

Challenge 2: Ask the person running your program to type in the number of squares across and the number of squares down. Use those values to generate the pattern. Once completed, turn a copy of your code into your teacher and if time permits, proceed to Challenge 3.

Challenge 3: Ask the person running your program how many sides to the shape to draw. Instead of a 4 sided square, draw the shape based on the number of sides. Once completed, turn a copy of your code into your teacher and if time permits, proceed to Challenge 4.

Challenge 4: Make enhancements of your choosing to make the pattern generator program better.

f(x) – Function Exploration

For this activity, you are going to write some code to graph the results of a function.

Your code is going to take a given function such as …

$f(x) = x + 5$

… and create a graph of that function.

Start by writing some code to create the x and y axis extending from left to right in the middle of the screen and top to bottom in the middle of the screen.

Then your code should go through all of the integers starting on the left side of your graph to the right side of your graph. Those integers are the value for x. You will place each of those values into the given function and use the resulting value to place a dot on the graph. For example, when the x value is 25, the result will be 30 and you will place a point at (25, 30) on your graph.

Caution: Your code should check for division by zero which will give an error. Avoid your program ending in error.

After getting that to work, answer the following questions and change your code accordingly to answer those questions.

1. Make note of what your graph looks like then change your function to … $f(x) = x + 30$
    a. How does your graph change?
    b. If you change to $f(x) = x – 100$, how does your graph change?

2. Go back to the original function (top of the page) then compare that with the function … $f(x) = 8x + 10$
    a. How does your graph change?
    b. Compare the current graph with … $f(x) = -6x – 24$ … what changes do you observe?

3. Graph the function … $f(x) = ½ x^2 – 2x + 10$
    a. What happens when you change the ½ from a fraction to a whole number (try smaller and bigger whole numbers when exploring this)?
    b. What happens when you change the -2x to +5x? While exploring this, try different size negative and positive numbers (smaller and larger).
    c. What happens when you change the +10 to a negative number? While exploring this, try different size negative and positive numbers (smaller and larger).

4. Graph the function … $f(x) = 2/x + 5$ … what do you observe about this graph?

Create a Fractal (and break it up a little)

Fractals are figures created using a similar pattern at a progressively smaller scale.

For this activity you are going to submit a document with your observations along with your Python code file.

Spend some time doing researching how fractals can be used and also looking at some different fractal images. In your document (and in your own words) describe what you found along with the URL to one article you found helpful and the name of the three fractals you think look the coolest.

For the remainder of this activity, we are going to create the Koch fractal in Python and explore a little bit of how recursion works. Go to the following link and read the top part of the page and read through all of section 18.1.

http://openbookproject.net/thinkcs/python/english3e/recursion.html

Next, start a new Python code file and type in (do not copy/paste but type it in) the 16 lines of code right under the images of fractals. You can try running this but since it is just a function that never gets called, it will not run anything.

This code uses the Python turtle to do the drawing so at the top of your code file (above the koch function), you will need the line …

```
import turtle
```

Below the koch function, you will need to define a turtle (you can use the variable name of your choosing) which I will call my turtle george …

```
george = turtle
```

Then I need to call the function giving it the turtle, the order, and the size. That would look like …

```
koch(george, 1, 100)
```

Run it! This draws the order 1 fractal of length 100. Change the order from 1 to 0 and run it again. What do you get? Run this changing to the order of 2 then 3 then 4. While trying some of those things out also adjust to different sizes (i.e. change the 100 to bigger and smaller numbers).

During each run, observe the differences and similarities in the generated image. Record a few of your observations. Look at the code again and try to follow how the code is working.

In the line "def koch", change that to "def koch_version1" then type in the 7 lines of code for a reduced code version of the koch function. Run the updated code for the different orders (i.e. 0, 1, 2, 3, 4) and make sure it still produces the same results. Look at the revised code and how it works. Below the 7 line version of the koch function is some code where the author wrote separate functions for each order. You do not need to run this but observe how it works.

Challenge 1: The fractal starts in the middle of the screen and fills to the right and up. Add code necessary to start further to the left and down so the fractal is centered more. For an additional challenge, use the size and order to calculate an optimum starting position.

For the remainder of the activity, you are going to write 3 different functions (each their own name but not koch) based on either koch or koch_version1. Each function should have a comment line at the beginning of the function describing what was changed in the code. Alter the order 0 part of the code in one of the functions and the non-zero part in the other two. Your code changes should result in a different looking fractal each time. The

koch is called four different times in the non-zero section of the code. Alter that number of calls as well as the base shape. Sketch out what you think it should look like then try to make it look that way.

Hint: Each function name will be different than koch so you will need to change the references to the function you are working on. Also, you will need to call the new function name in the main routine (i.e. call that not koch). Not changing those references will produce some interesting results.