A Taste of Coding

As you can see, knowing how to code can be quite a useful skill. (According to the New Yorker)

So …what is it like to write a program?

Recall from the first week of the class that a computer an only
1. Input
2. Output
3. Do symbolic processing (including arithmetic)
4. Access memory and
5. Control the order of the operations that it performs

We will write some programs using Scratch, then with the Python programming language. Presumably you have already downloaded Python and installed it on your computer.
Let’s get a “taste” of programming using Scratch.
Bring up the browser and go to: www.scratch.mit.edu

And click on create

We can write programs – called scripts by chaining together blocks that perform various actions.

**Input:** get input from the environment
- Sensing
- Events

**Output:** make something happen in the environment
- Looks
- Sound
- Motion
- Pen (used in conjunction with Motion)

**Memory:** save a value for later use
- Answer (Sensing)
- Data

**Processing:** do arithmetic and symbolic processing (+, -, concatenate …)
- Operators

**Control:** Control the order of the operations, do conditional processing (if), repetition (loops) …
- Control
Let’s start with OUTPUT. It is traditional when learning a new programming language. The first program we usually write send greetings to the world. We have our program print:

Hello world!

Scratch has some of the output commands in the Looks tab because the output changes how the “world” that your program creates looks. The other output commands are in Motion, Sound, and Pen.

Here is the program.

It’s output is

Hello world!
**Input**: get input from ("sensing") the environment. Click on “Sensing”.

If you click on “Sensing” in the “Scripts” tab, you get a list of all the things that you can “sense”.

You can ask for any single input value with this block: (This is also an example of “output” because it prints the “prompt” “What’s your name?”.)

![Ask block example](image)

and the answer is stored in the variable `answer`

This is an example of the use of “**memory**”.
So, if you click on the ask block above, when you type “Bob” in the input bar and press the enter key, you see

![Input operation example](image)

You can also get input from “**Events**”. Note: here too you are getting “input: about the state of the environment”.

![Event list example](image)
**Processing:** Lets you do various arithmetic operations on numbers, and “join” (combining) “string” values.
Example: Write a Scratch program to ask the user for their name and greet them by name.

Problem: Modify the program so that we get an exclamation point after the name:

Answer:
Problem: Write a program to ask the user their age in years and have the program tell them their age in minutes. Don’t worry about leap years.

Answer:
Problem: write a program to ask the user for the temperature in Fahrenheit and print out the resulting temperature in centigrade. The formula is:

\[
T(\degree C) = (T(\degree F) - 32) \times \frac{5}{9}
\]
or
\[
T(\degree C) = \frac{(T(\degree F) - 32)}{\frac{9}{5}}
\]
or
\[
T(\degree C) = \frac{(T(\degree F) - 32)}{1.8}
\]

**Example**

Convert 68 degrees Fahrenheit to degrees Celsius:

\[
T(\degree C) = (68 \degree F - 32) \times \frac{5}{9} = 20 \degree C
\]
Let’s draw some pictures! For this we will use the blocks from “Pen” and “Motion”.

Problem: Have the sprite draw a square with each side equal to 100 steps. Do this with nine blocks. One from Pen and eight from Motion.

Answer:
Control: All the programs that we have written so far executed in a straight line, block after block. We can alter the order by using some of the blocks from the “Control” group.

Problem: Modify the last problem so that you use one block from “Control”, one block from “Pen” and two blocks from “Motion”.

Answer:
You can clear the canvas and reset the sprite like this:

Problem: Have the sprite draw a triangle with each side of length 100. It should look like this:

Answer:

Problem: Have the sprite draw a triangle with each side of length 100. It should look like this:

Answer:
Problem: Have the sprite draw a hexagon with each side of length 100. It should look like this:

Answer:

Problem: Have the sprite draw a circle. It should look like this:

Answer:

Problem: Have the sprite draw a 5 pointed star. It should look like this:

Answer:
In the examples above, when you asked for input, Scratch deposited the value in “answer”. What if we need more than one input? How do we save the previous input so that we can get another and then use both?

The answer is that Scratch provides us with variables that we can use to save the values in RAM. We get the variables from the **Data** tab.

![Data tab](image)

When we click on Make a Variable we get a dialog box. Each time we want a new variable we just click on Make a Variable again.

![New Variable dialog](image)

A variable creates a place in RAM in which we can a value.

As an example let’s create three variables and call them x, y and sum. We will get each value using “ask”. The entered number will automatically be stored in “answer”.

Next, transfer the values that are in answer to x and y by using the “set” block.

Then add x and y and store the value in sum.

Finally, print the sum.

The program looks like this. Create and run it and make sure that it works.
We can use loops to generalize the programs above. Here is the name program you wrote above, but in a loop.

**How do we tell the program to break out of the loop and stop?**

We can use an ask block and an if block like this”
**Here is something really cool:** we can make new blocks by combining existing ones. In programming languages this is called defining **functions**. This like making up new commands and extending Scratch’s original vocabulary!

Here is how it works.

**Click on “More Blocks”**.

![Image of More Blocks]

When we click on “Make a Block” we get to name it:

![Image of Make a Block]

And the when we use the name our block will be executed. Had we called our new block **square** we could define it with the blocks on the left and when we click the new block name on the right the sprite will draw a square.

![Image of Square Block]
We can even pass “parameters” into our block so that it modifies its behavior based on the value that we pass in.
For example, our square above will always be 100X100. We can make a more flexible square that lets us specify the length of a side. When we define our new block, we click on “Options” and choose “Add number input:”

Here is one possible block for a square with a parameter to allow for a variable length side:
Earlier, we saw how to write a program to “reset” the sprite back to the middle.

We can a new block to do this.

And now we have new command, “reset” that we can incorporate into our programs.