day82 Arduino coding setup
Monday 4/15/24

Get checked off for your superblink program before starting today's work.

To begin today's assignment, open a new program. Save it as day85_coding and save it into the folder you made on the desktop with your name. Add the following line to your setup() function:

```
Serial.begin(9600);
```

Add the following line to your loop() function:

```
Serial.println("Hello!");
delay(5000);
```

Save and then upload your program, then press the icon or open Serial Monitor from the Tools menu. The Arduino is now talking back to your PC on the USB cable. You should see "Hello!" appear once every five seconds in the Serial Monitor window.).

As we discussed on day 1, every Arduino program has two required parts: **setup** and **loop**. Both are actually functions, that is, blocks of code that get run. The "void" before each indicates that they don't return any data to whomever is calling them (and we don't have to worry about who is calling them, they get called automatically). When an Arduino powers up, resets, or gets a new program uploaded, the Arduino executes the code above the setup() function once, then the setup() function is executed once, then the loop() function is run over and over forever.

Over the next three pages I cover some important programming tips and commands. Please read through this slowly, perhaps trying things out in the little program you've already written for today. The next three pages are NOT today's assignment, but you need to read them to be able to do today's assignment. Just work your way through this material. When done reading the material, delete the Hello line and do today's assignment in this file.

Before we get to the electronics, I want us all to get familiar with programming in Processing. It has several concepts that should seem very familiar, plus some that are new. Here are some highlights:

**For Loops**: Arduino "for" loops look like this:

```
for (int i = 0; i<25; i++) {
   Serial.println(i);
}
```

This prints the numbers 0 through 24, each on its own line. Technically this loop says we're going to use a variable i, it will be an integer, it starts at zero, and the loop will repeat as long as i is less than 25, and to increase i by one each time through (that is the "i++" part). This would be like the following in Python:

```
for i in range(25):
      print(i)
```

The three important parts of this loop are:

| | |
|---|---|
| int i = 0 | This declares an integer variable i  and sets  it to zero. This variable is only used here inside this for loop. |
| i<25 | This tells Arduino to loop as long as i is less than 25 |
| i++ | This says "increment i by 1" which means "add 1 to i" each time through the loop. |

(continued on next page)

## If Statements

As we saw on the first day, basic Arduino if statements look like this:

```
if (i==40) {

}

or

if (i>1000) {

}
```

In if statements we have to use the double equals signs (==) if you want to check if something is equal to another thing, just like in Python.

You can also do if/else statements. They look like this:

```
if (i<12) {
   Serial.println("Less than 12");
}
else {
   Serial.println("Not less than 12");
}
```

You can even do if/else if/else statements (like our if/elif/else statements in Python). Remember, you can always look up things in the Arduino Reference under the Help menu.

## Declarations

One big difference between Arduino and Python is that in Arduino sketches you have to declare your variables before you use them. You also have to say what type of variable something is at the start (and you can't change it later.) If you want to be able to use a variable all over the program, declare it before the setup() function like we did with the "int x = 1;" line from our superblink program.

## Printing to the Serial Monitor

You have two basic calls to print using Arduino:

```
Serial.println();
Serial.print();
```

The first prints whatever is in the parentheses and then goes to a new line. The second prints but does NOT go to a new line. So, the following:

```
Serial.print("It hasn't rained");
Serial.println(" in quite a while!");
```

prints this:

```
It hasn't rained in quite a while!
```

If they were both "Serial.println()" lines you would instead get:

```
It hasn't rained
 in quite a while!
```

Here are two more examples. This loop:

```
for (int i = 0;i<5;i++)
{
  Serial.print(i);
}
```

generates the following output:

```
01234
```

While this loop:

```
for (int i = 0;i<5;i++)
{
  Serial.println(i);
}
```

generates the following output:

```
0
1
2
3
4
```

If you want to print a blank line use the call "Serial.println();" with nothing in the parentheses, or use the "\n" characters sequence as in this call: "Serial.println("Part 1: \n\nMr. Hays");" which prints prints "Part 1" then two blank lines, then prints my name. You can also use "\t" for tab to indent something.

Arduino Serial.print() calls only accept one argument per line. So if you want to print multiple things, put them all in one print call if possible, or use multiple print calls. You can't print (x, y, z), you have to use three different lines to do this.

So do this:

```
Serial.println("Go Analy! 2024");
```

or

```
Serial.print("Go");
Serial.print(" Analy! ");
Serial.println("2024");
```

not

```
Serial.println("Go","Analy!","2024");  //you can't use commas in a print call
```

or

```
Serial.println("Go"+"Analy!"+"2024");  //you can't use plus signs with text here
```

When you tell the Arduino to print things back on the computer you will notice the very small tx and rx LEDs on the board (near the pin 13 LED) blink on and off. Those show that the Arduino is sending or receiving data.

Today's assignment is in a separate PDF.