Make Your Own Wearables Workshop

CIRCUITS AND CODE
Today's Project

https://www.youtube.com/watch?v=gWZi71JkPAA
What is in our circuit?

- Battery supplies electricity
- Pressure sensor detects a step
- Microcontroller tells the lights to turn on when a step is detected
Your brain is like a microcontroller—that’s where all the processing happens!

Your hands, ears, eyes, nose and tongue are all sensors.

Your nerves are like wires that send messages from your sensors to your brain.

You get energy from food, like the circuit gets energy from a battery.

You can move and talk based on your sensory input and what your brain tells you to do!
What is computer programming?

- A program is step-by-step instructions for a computer.
- Each instruction is processed one at a time, *exactly* as written.
- For the Firewalker circuit, code instructions are:
  - Light up when a step is detected by the pressure sensor.
“If” Statements

“If” statements allow your program to ignore lines of a code (or instructions) if a certain condition isn’t met.

Your brain uses “if” statements every day!

You wake up and check the temperature outside.

If it’s freezing (less than 32 degrees Fahrenheit)...

Then you wear a hat and gloves that day. (Otherwise you skip the hat and gloves)

Example Code:

```
int temperature = checkTemperature();
if (temperature < 32) {
    wearHatAndGloves();
}
```

© Getty Images. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.
A “loop” allows your program (or instructions) to repeat until a certain condition is no longer true.

Let's say...
You want to print every integer value starting from 0 that is less than 8.

You know how to use variables, functions, and “if” statements, so your code would probably look something like this.

```
int i = 0;
if (i < 8) {
    print(i);
}
i = i + 1;
if (i < 8) {
    print(i);
}
i = i + 1;
if (i < 8) {
    print(i);
}
```

Here's an example!
```
for (int i = 0; i < 8; i = i + 1) {
    print(i);
}
```

1. This "for" loop creates a variable named "i" and gives it an initial value of 0.
2. Then it checks if the value of "i" is less than 8.
3. If it is, it prints the value of "i", and then adds 1 to the value of "i".
4. It repeats steps 2 and 3 until the value of "i" is no longer less than 8, then it goes to the next part of the program.
Let’s program a “robot” to build a PBJ Sandwich
Variables

Use **variables** to give names to the values in your program.

The values might change, but the names will stay the same.

You can name variables whatever you want! Give them good, descriptive names and your program will be easy to read and understand.

![Image of a speed limit sign with an algebra problem]

© source unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see [http://ocw.mit.edu/help/faq-fair-use/](http://ocw.mit.edu/help/faq-fair-use/).

Here's an example!

```
int i = 0;
```

This line of code creates a variable named `i`. 

© source unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see [http://ocw.mit.edu/help/faq-fair-use/](http://ocw.mit.edu/help/faq-fair-use/).
**Functions** are like variables, but they give names to whole sections of code instead of just one value.

---

**Remember this from math class?**

Functions in programming can take variables or values as inputs, and return different values as outputs, just like functions in math!

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y = f(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>$f(-2) = 3^{-2+1} - 2 = 3^{-1} - 2 = \frac{1}{3} - 2 = -\frac{5}{3}$</td>
</tr>
<tr>
<td>-1</td>
<td>$f(-1) = 3^{-1+1} - 2 = 3^0 - 2 = 1 - 2 = -1$</td>
</tr>
<tr>
<td>0</td>
<td>$f(0) = 3^0 + 2 = 3^1 - 2 = 1$</td>
</tr>
<tr>
<td>1</td>
<td>$f(1) = 3^1 + 1 = 3^2 - 2 = 9 - 2 = 7$</td>
</tr>
<tr>
<td>2</td>
<td>$f(2) = 3^{2+1} = 3^3 - 2 = 27 - 2 = 25$</td>
</tr>
</tbody>
</table>

---

**Here's an example!**

```c
int timesTwo(int x) {
    return x*2;
}
```

You (or someone else) has to define the function before you can use it. This function—named "timesTwo"—takes a variable named "x" as input, multiplies it by 2, and returns the value as output.

Then you can use the function in other parts of your program, like this!

```c
int x = 3;
int y = timesTwo(x);
```