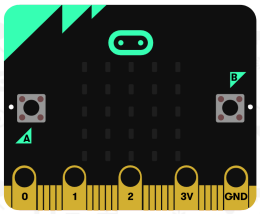
Be efficient	Grade 5 and 6

Coding Handout

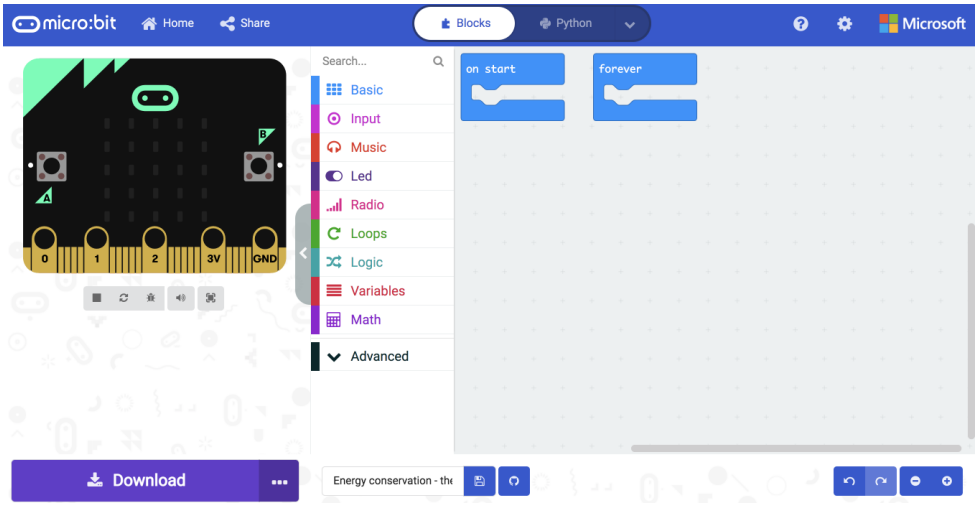
This step-by-step guide will cover how to do the activity with the on-screen simulator. Instructions on how to use the hardware if you have access will appear near the end of the document.



This is what the micro:bit simulator looks like. There are 2 buttons, A and B, an LED (Light Emitting Diode) matrix and pins to connect wires to. In some newer micro:bits, the logo also acts as a button. We can't see the sensors on the simulator right away but when we write code that uses them, they'll appear in upper left hand corner of the microcontroller. Micro:bit hardware will include a USB cable to connect to a computer and a battery clip for power.

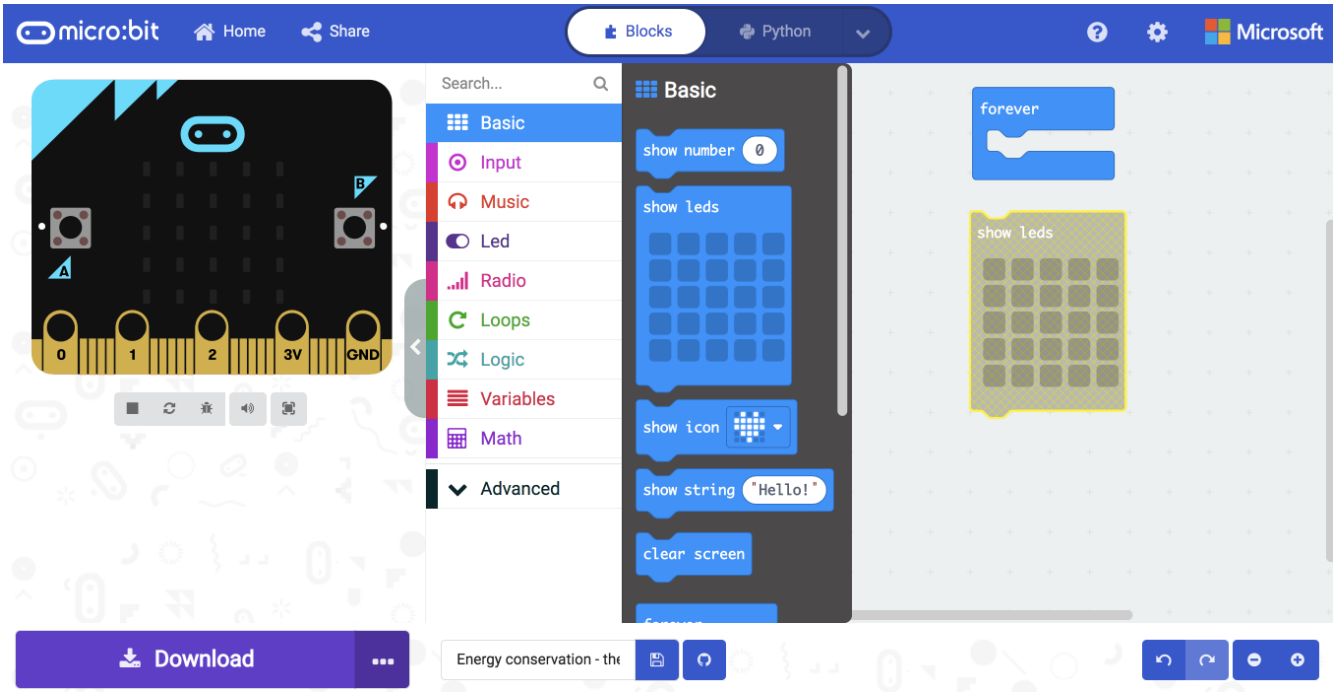
Activity 1: Use a sensor to program a more efficient light

Let's take, for example, a light that is always on. Using a light sensor in the micro:bit, we could program it to only turn on when it is dark, like a nightlight. We'll look at the code for both those cases.

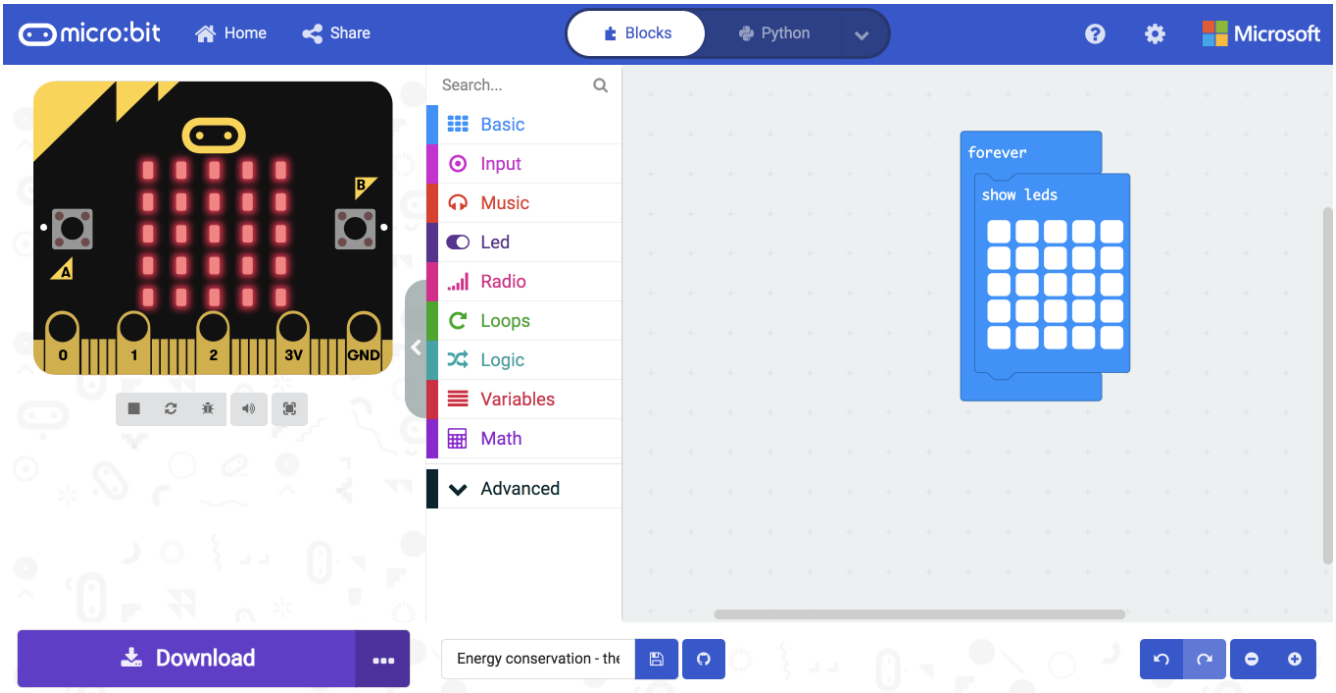


When we start in the MakeCode editor, this is what we will see. On the left, we have the simulator. Then we have code block menus where the different blocks are sorted and colour coded. On the right, we have the code area. This is where we will place the blocks of code that we want to run.

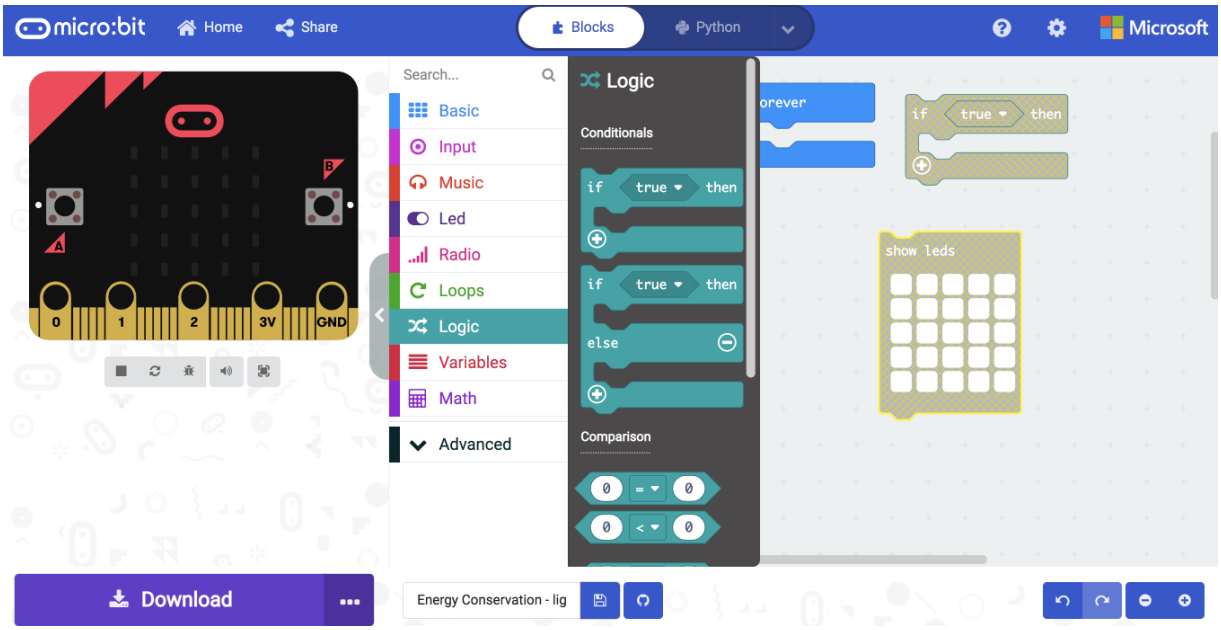
If we want to program a light that is always on, we will use the *forever* loop. Within the *Basic* code block menu, we can pull out the *show leds* block. We can now choose which leds in the matrix will be turned on. For now, we'll turn them all on.



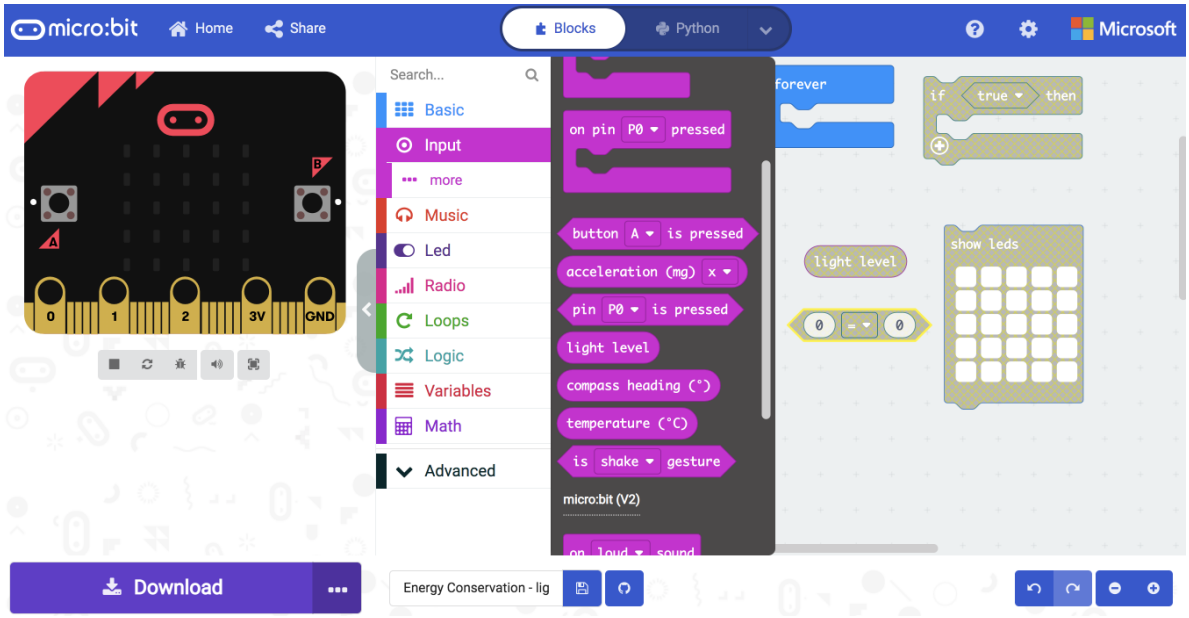
Once we slide the *show led* block into the *forever* block, the simulator will start to run the code.



If we want to change how the leds behave, we need to change their code. We will use the built-in light sensor to have the lights turn on only when it is dark. We can do this by using an 'If/then' statement. If it is dark, then turn the lights on. We can choose an *If/then* block from the *Logic* menu.

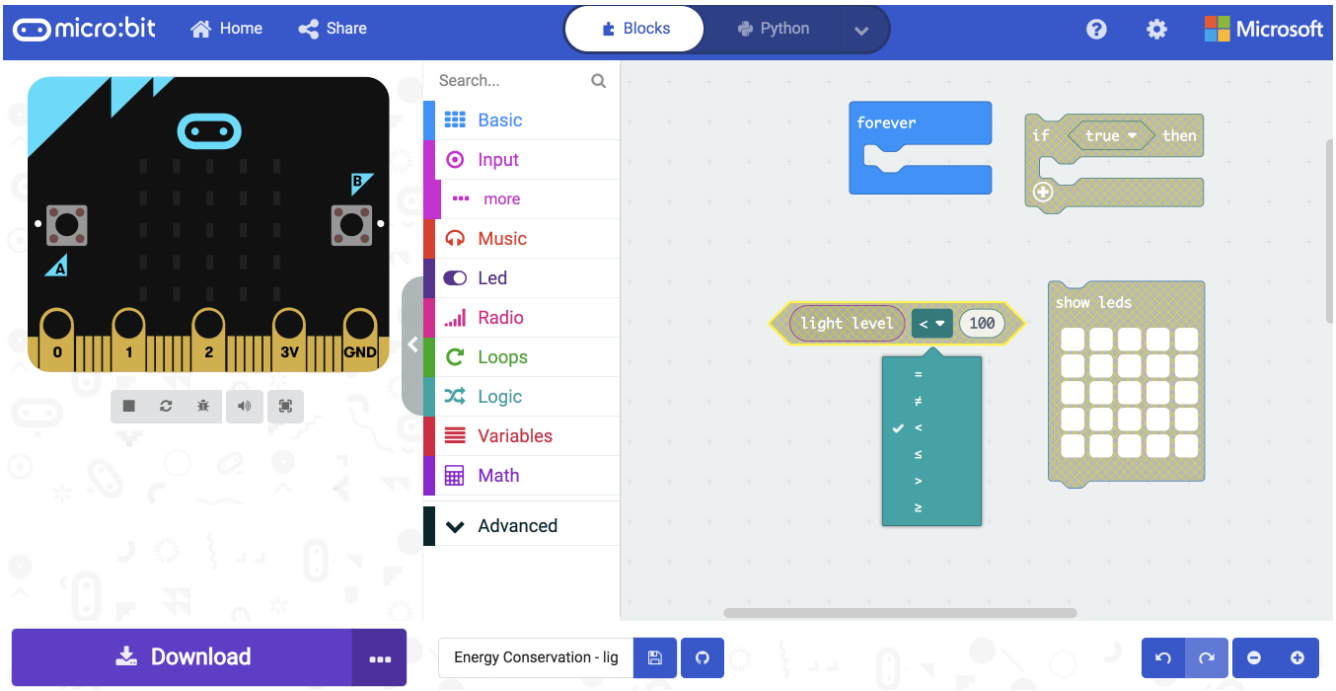


Next, we will need to set up the code to finish the If/then statement so that the micro:bit knows when it is 'dark.' The built-in light sensor will give us the input we need to be able to control the leds this way. The light sensor takes light information and then gives it a number between 0 and 255. When it reads 0, it is the darkest. When it reads 255, it is the brightest. We'll need to choose a threshold for what we consider dark to apply it to the code. Let's start with a threshold of 100 for simplicity's sake. The *light level* block is found in the *Input* menu. We'll also need a *Comparison* block from the *Logic* menu.

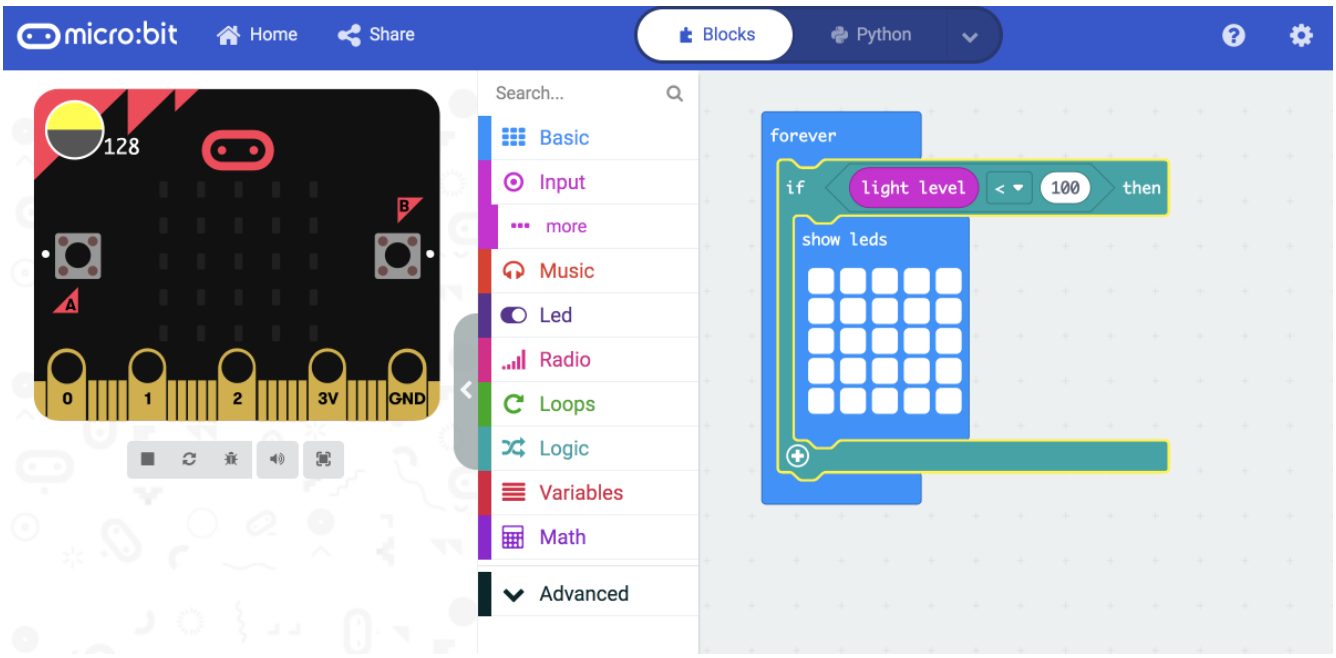


These are all the code blocks we will need. We'll just have to connect them together appropriately. The *light level* block will need to go into the *Comparison* block. Here is where we add our threshold light

level value for when the leds will turn on. If 0 is dark and we want our light to turn on when it is darker than 100, we will need to adjust our comparison symbol to reflect this.



Once our comparison is set, we can place it into the If statement. We will also place the *show leds* block into the 'then' portion of the block. Then we place the *If/Then* block into the *forever* loop. Once we complete that step, the simulator will start to run the code.



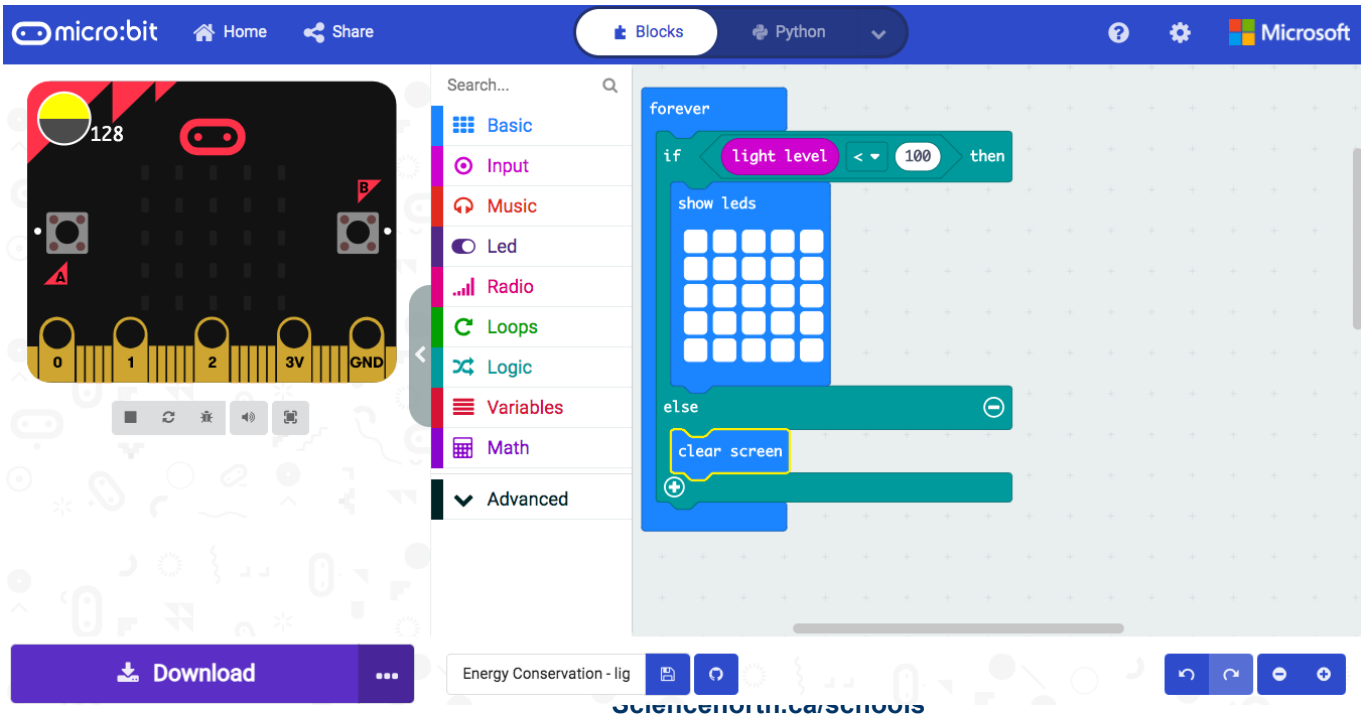
We can use our cursor to slide the level on the light sensor up or down to see how the leds behave when we change the light level. What happens when we cross the threshold? In this case, the way the code is written, once we cross below the 100 mark, the leds will turn on and they'll stay on regardless of what happens with the light level after that. We didn't give any instruction to turn off.

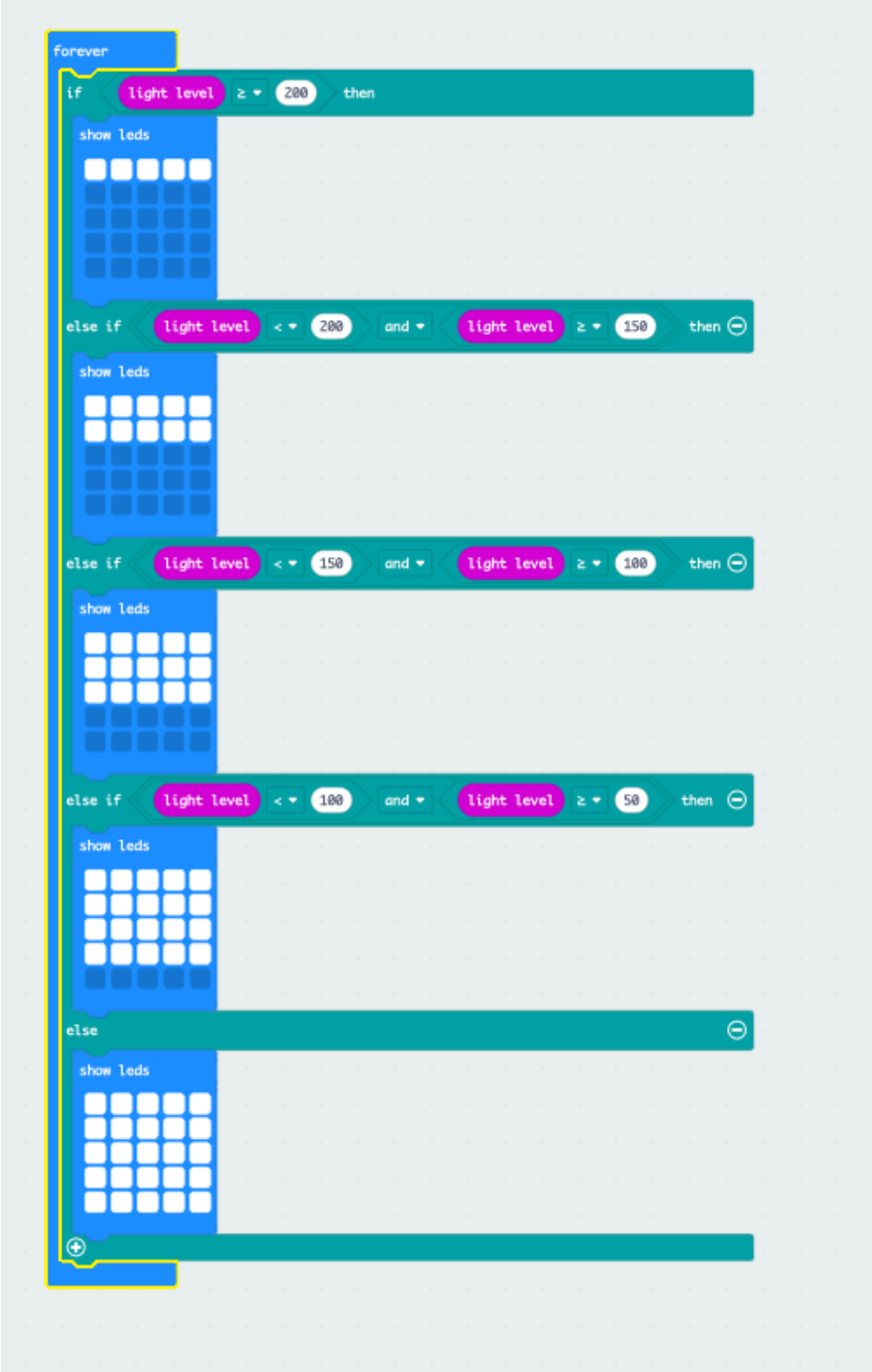
We can add an *Else* portion to our *If/Then* statement so that if the light level is above 100, the leds turn off. Here we can revisit the idea of efficient code as we could write 2 separate *If/Then* statements but we could also, more efficiently, use the *Else* on the first statement instead. By clicking the plus symbol in the bottom left corner of the *If/Then* statement, we add an *Else* line. We will code the leds to turn off when the light level is 100 or higher by adding the *clear screen* block from the *Basic* menu.

After adding the *Else*, the leds will turn off when the light level is 100 or more.

Challenge students to think of other conditions under which they might want their lights to work to increase their efficiency. How can they alter the code to satisfy different conditions?

Examples of other conditions might include:
A light that only goes on when a button is pressed if it is dark enough.
A light that gets brighter as it gets darker. (Code shown on next page)
A light that only goes on when the object is in a certain position.

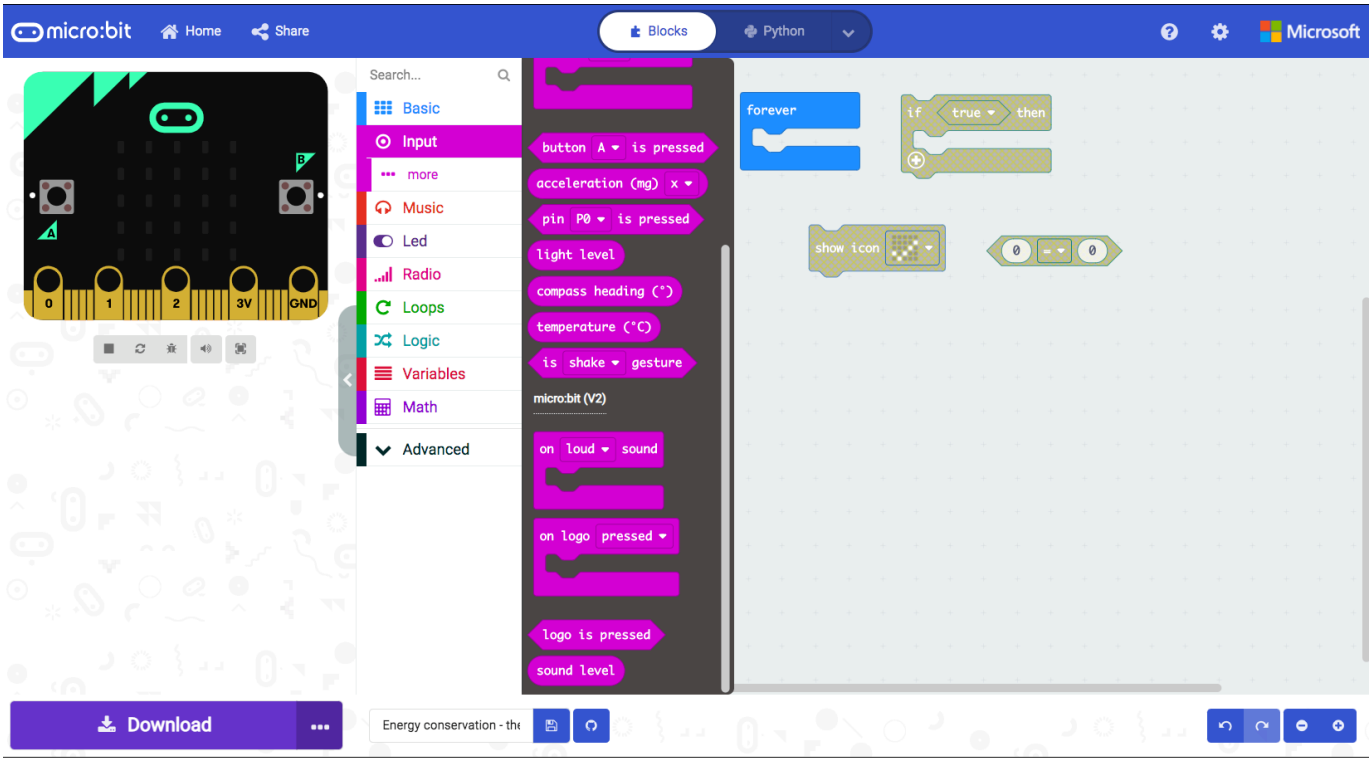



forever
if light level ≥ 200 then
show leds
else if light level < 200 and light level ≥ 150 then
show leds
else if light level < 150 and light level ≥ 100 then
show leds
else if light level < 100 and light level ≥ 50 then
show leds
else
show leds

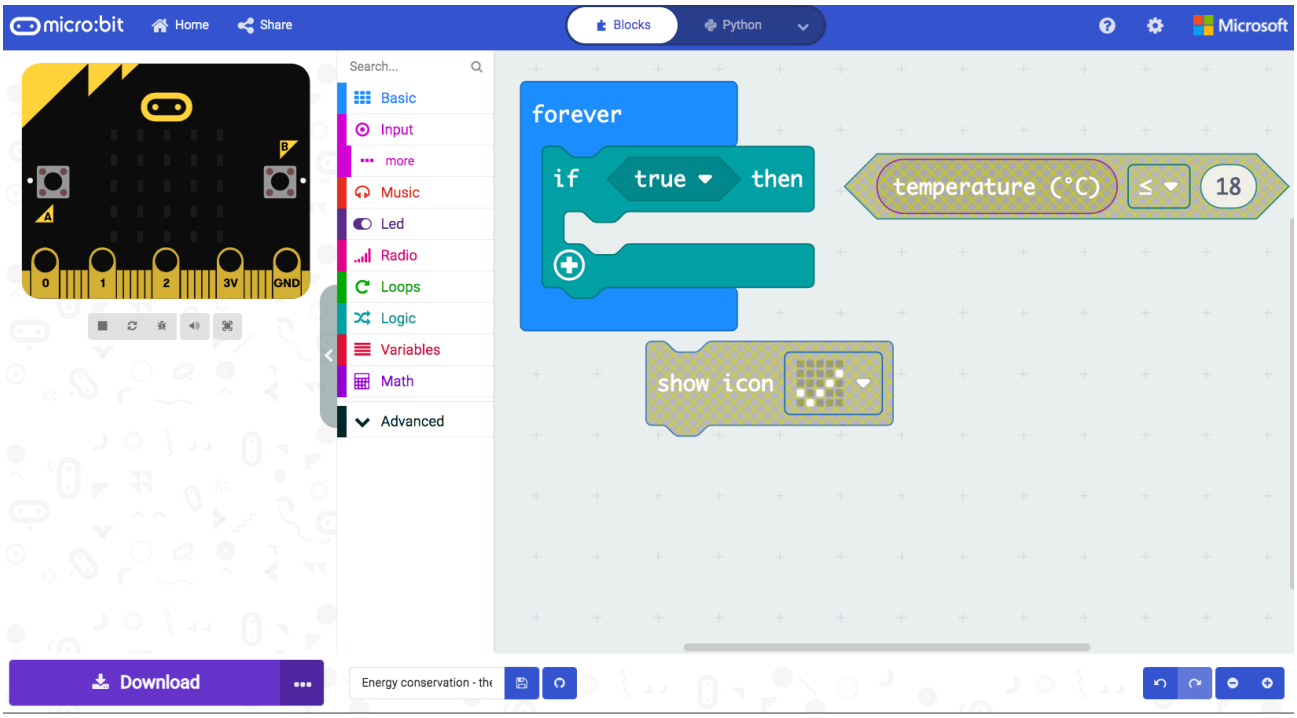
Another thing that can help our homes be more efficient are programmable thermostats. A programmable thermostat can help us set a point to turn on and off, so it isn't always running.

Since the micro:bit doesn't include a heater, we will choose a light output as a stand-in for a heater function. Rather than turn on the heat, we'll turn on a few lights in the led panel on the micro:bit.

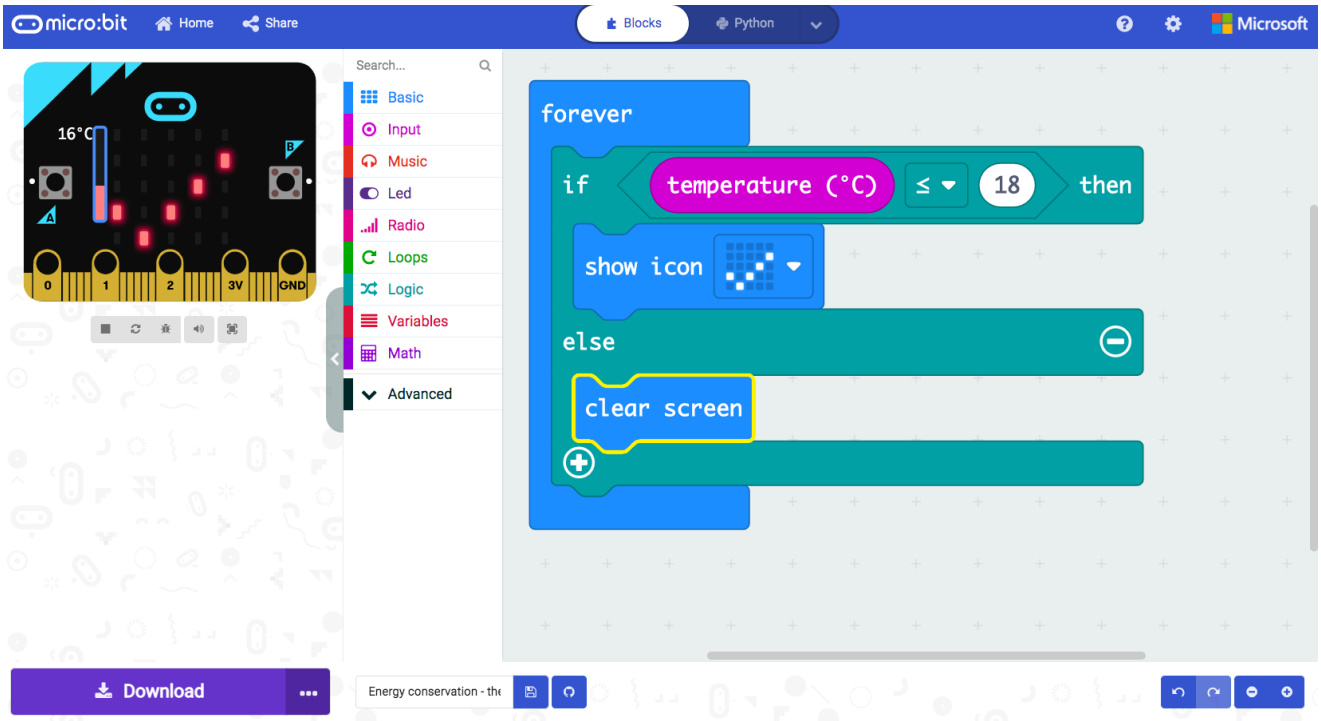
The first program will be a simple thermostat that controls only a heater. This will be very similar to the light that only goes on when it is dark. We will now use a temperature sensor block rather than a light sensor. The *temperature* block can be found in the *Input* menu. Below we see the blocks that we will be using for this code. We have a *forever*, and *If/then*, *comparison* and *show icon* and will be selecting the *temperature* block.



The *temperature* block says (°C) so it will measure in degrees Celsius. Students will have to consider their temperature threshold. How does this affect their heater usage? What are strategies they might use to be able to comfortably keep a lower set point?

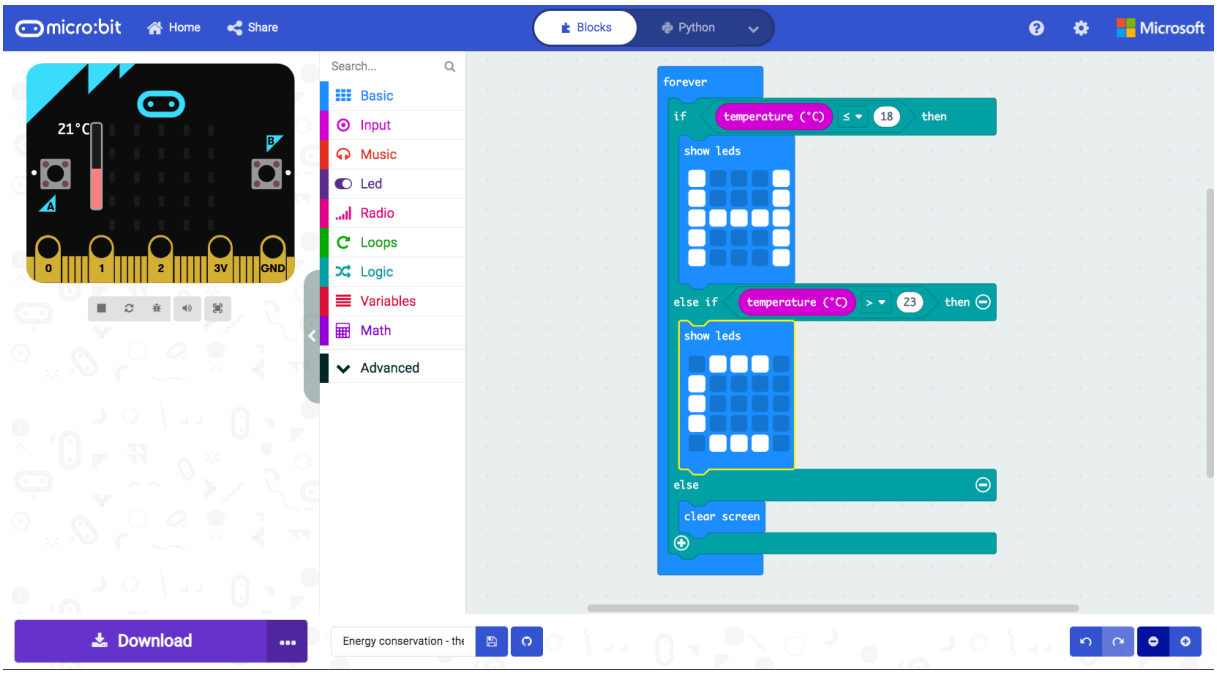


We have set the range to temperature less than or equal to 18 °C. That means if the temperature sensor reads below 18 or below, the heater will turn on. We'll also need to include a shut off. We'll use an *Else* so that if our temperature sensor reads above 18 so 19 or more, that the heater will turn off.

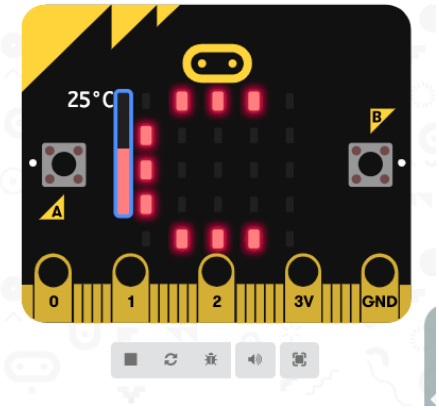


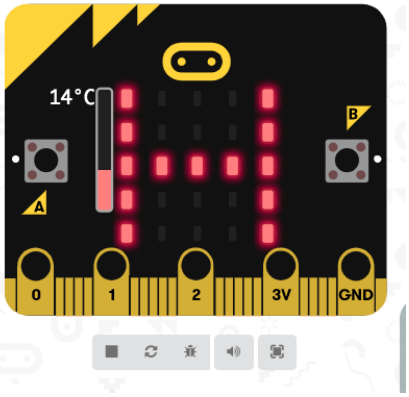
The next edit of the program will consider both a heater and an air conditioner. What range of temperatures will be comfortable?

We'll use the leds to write an H to indicate the heat is on. We'll also use the leds to say C for when the AC (Air Conditioner) is on. We have chosen a range of temperatures and placed them in the code.



Use the temperature sensor slider to see how your thermostat controls the house.





Things to consider and encourage students to think about during this activity: Would this be an efficient thermostat? Would we have the heat and AC working at the same time? Could we have different code for different times of day or different times of year? How do we keep these devices from doing more work than they have to? Foregoing air conditioning can be a great way to conserve energy and not all homes have air conditioners, what are other ways we can keep

homes cool? Have students think about conditions or sensors that we could use that would increase efficiency. If time allows, have them program these conditions.

**Notes on using micro:bit hardware: When students have finished writing their code, they can click the download button in the bottom left of the screen. This saves a .hex file onto the computer. Students must use the USB connector to plug the micro:bit into the computer and then drag that .hex file into the MICROBIT folder that will appear. When the micro:bit is turned on, it should run the code that was downloaded. This is great exploration especially for use with the light sensor code. It is easy to vary the amount of light on a desk by using objects or your hands to create shadows. It is more challenging to vary the temperature the same way, but it is still a valuable exercise to interact with a physical interface in addition to the virtual one.