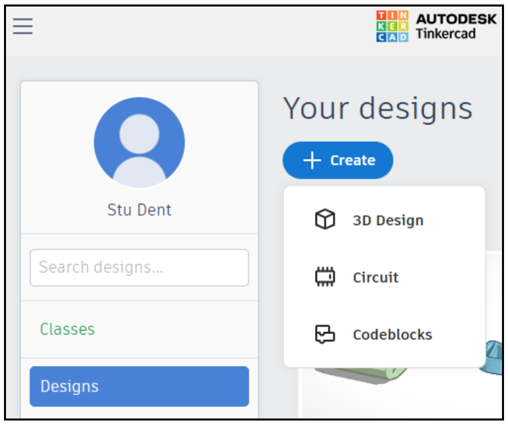
Let's start a NEW circuit design

Just as we did previously, go ahead and create a new circuit project:

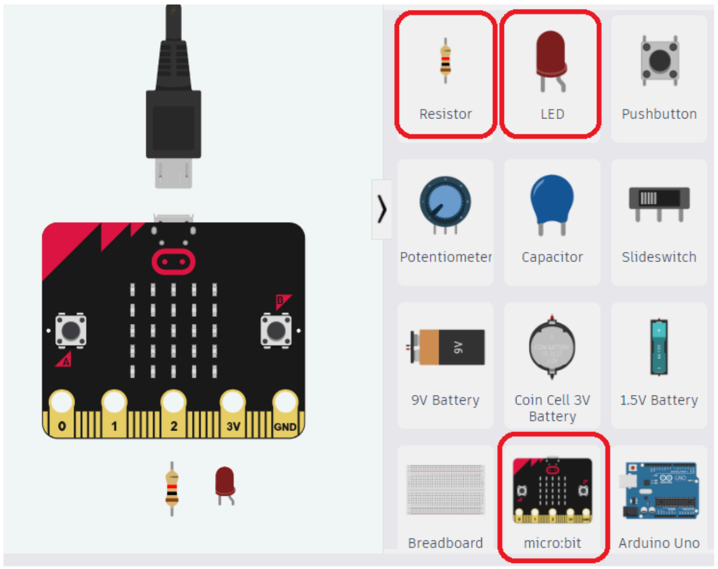
Click on the "Designs" tab if it isn't already selected (on the left)

Click on the "+Create" button and Select "Circuit"



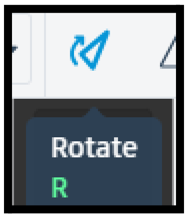
Find the **micro:bit** and drag it to the workspace.

Now, drag 1 **resistor** and 1 **LED** over to the work space.



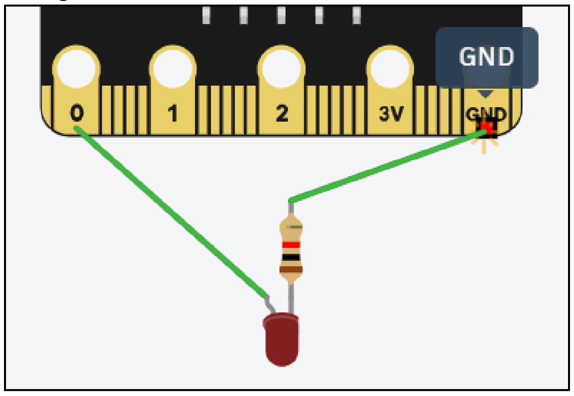
Click on the "Code" button, then click on "Blocks" and select "Blocks and Text" so you can see both the Blockly code language, and the Python code language side-by-side.

Select the virtual LED by clicking on it once.



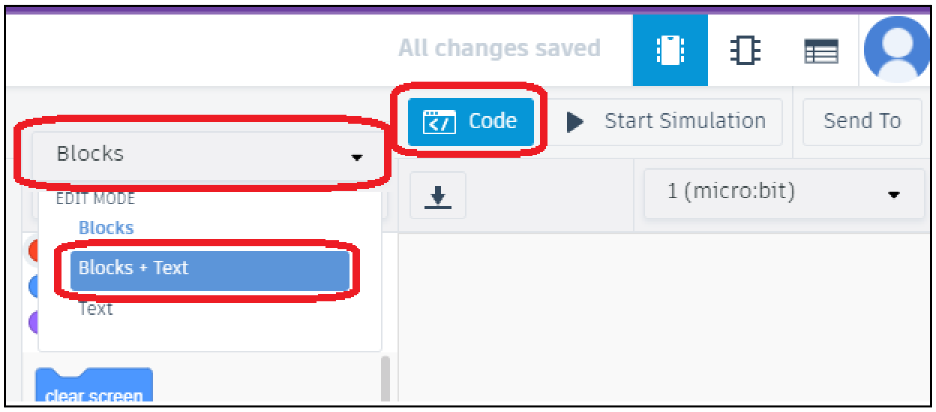
Use the "rotate" tool at the top of the screen to turn the LED 180° so the leads are facing up.

With real LEDs, the positive lead (wire) is longer than the negative lead, but virtually, the positive lead is indicated by a slight bend.



Connect the positive lead of the LED to Pin0 on the microbit by clicking on the tip of the positive lead and dragging a "wire" to the 0 on the micro:bit. Then, click to lock it in place.

Next, drag the resistor so it connects to the negative lead on the LED. Click on the other side of the resistor and drag a 'wire' to the GND pin on the micro:bit. Now you have a completed circuit. We are ready to code!



As we did before, click on the "Code" button, then click on "Blocks" and select "Blocks and Text" so you can see both the Blockly code language, and the Python code language side-by-side.

FIRST:

We are going to write some simple code to blink the led on for 1 second, off for one second.

Let's think the logic through… (this is a practice most coders do!)

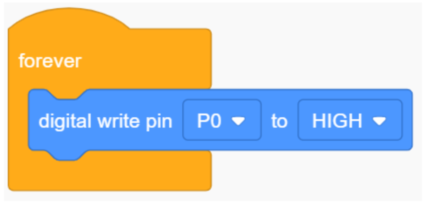
You will need a line of code to turn the LED on
You will need a line of code to pause the code from running the next line for 1 second WHY?
You will need a line of code to turn the LED off
You will need a line of code to pause the code from running the next line for 1 second
You will need to put all of the code in a loop to repeat forever (or for as long as there is power to the micro:bit)

IMPORTANT information to understand and remember:
Your code MUST identify the pins being used, as well as what they are being used for (input/read, output/write, digital or analog, etc.)

Here are some important terms to know as well:
Digital - it is binary. Meaning only two options. The code should indicate that the power to the pin will either be on or off, ones or zeros.
Analog - indicates a range of values - so, when talking about an LED for instance, rather than it simply being on or off, it could be anywhere from off to dim to bright to blinding… If it were a speaker, it could be a range of nothing, quiet, audible to ear-piercingly loud.

For this first experiment we will simply be turning the LED on and off, so what code is called for? Digital or Analog?

Click on the "Output" tab/menu and scroll down to find the "digital write pin P0 to HIGH" code block and place it in the "forever" block.



We have wired our LED to Pin 0 on the micro:bit so we need to make sure the code block is also set to P0. If we were to connect our LED to Pin 1 or 2 on the micro:bit, we would need to click on the drop-down and select the corresponding pin.

Notice that the code is set to "High" - or in digital terms that would be on. Remember - digital is all or nothing, on or off, 0 or 1, High or Low. In other words, it is currently set to have full power going to P0.

Now, you can get another "digital write pin P0 to HIGH" block, or simply right-click and duplicate it. Whichever you prefer. Place that under the first and since this line of code will turn the LED off, can you guess what to change?

Start the simulation and observe. Is the LED blinking on and off?

Here's the thing, computers tend to process code so fast the human eye can't detect a change so we will need to add a wait block to give our eyes time to see the switch. Get a wait 1 second block and place it in between the two blocks. Since this is a loop repeating 'forever', we will also need a wait block at the end, otherwise, the code goes from the low to high code so fast as the loop starts over, we'll miss it!

Start the simulation again and observe. Is the LED blinking on and off now?

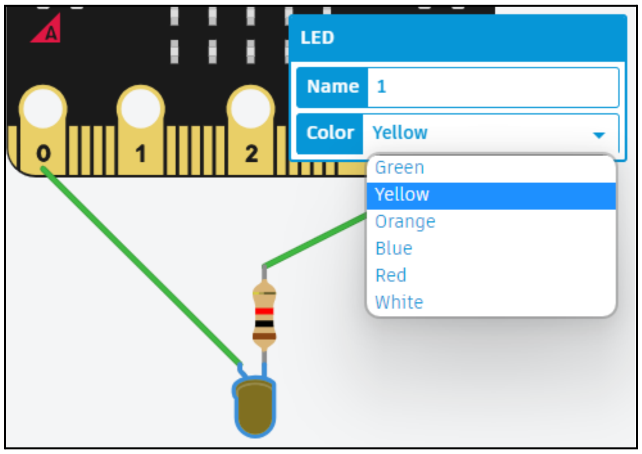
When pros work with electricity, especially when there are many components and the wires begin to look like a plate of spaghetti,they will often use different colored wires to help them keep track of what is connected.

They know that even with really low voltage, A 'short circuit' can be dangerous. Put simply, when the electrons are moving along a conductive material, they can generate heat if they are not controlled. We control the electricity by providing resistance. This can be longer wires or components like resistors. A "short" circuit means the electricity found a much shorter way to complete the circuit than intended. When this happens, the heat generated can melt the insulation on a wire and even start a fire if there are flammable materials nearby.

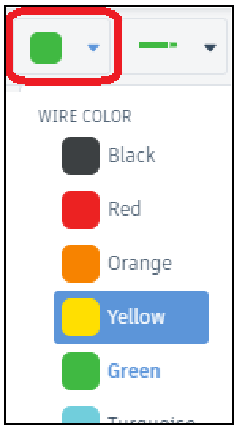
So, this is why many pros who work with small electronics use some standards - such as red wires for positive and black wires for negative. Or yellow for signal wires. Not only does it make it safer, using 'standards' also can make it more efficient to make repairs or adjustments to our circuits and devices. Another standard you may find helpful is when you are using LEDs with clear covers - unless it is on, you can't tell what color it is so often pros will use wire that is the same color as the LED.

Now, on your virtual LED circuit, let's change the color of the LED, and then change the positive wire color to match, and the negative/GND wire to black!

Click on the LED to select it. Next, in the blue box that pops up, click on the color and select a new color for your LED.



Click on the positive wire going to pin0 to select it. Then, click on the color selector at the top of the screen.

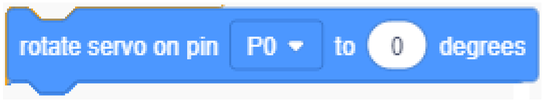


Click on the negative/GND wire and change that to black.

How about we add a little servo action? A servo motor is a geared motor that can be programmed to move to precise locations marked by degrees. Most are 180° position servos, but some are continuous rotation or full 360°.

Find and drag over a servo motor. (you can use the search tool if that is faster)
Hover your mouse over the pin holes at the top of the servo (where the wires connect) and the labels for each hole will show. Power (positive), Ground and Signal. Use the standard color wires to connect it to your micro:bit.
Power - connect this to the 3v (three volt) pin on the micro:bit
Ground - connect it to the GND on the micro:bit - the LED can share that pin.
Signal - you can connect it to either Pin 1 or Pin 2 - just make sure when you code, you use the corresponding Pin!

We need to decide what event we want to control our servo.
Based on a button pressed?
On some sensor input (temperature, tilt, light, etc.)
When the micro:bit first starts up?



```
rotate servo on pin  P0 ▾  to  0  degrees
```

Next, What position between 0 and 180 do we want it to go to?

If we move it to one position, does it need to move back or to another position?

Experiment and see what you come up with. For mine, I coded it so when button A is pressed, it moves to the 180° position and the LED turns on. And if button B is pressed, it moves to the 0° position and the LED blinks 3 times at 1 second intervals.

Here is my code:

```
# Python code
#
```

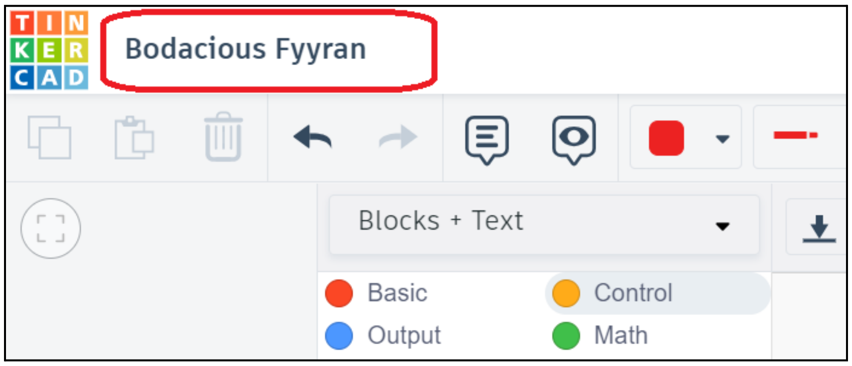
```
def on_forever():
  pass
basic.forever(on_forever)

def on_button_pressed_a():
  pins.servo_write_pin(AnalogPin.P2, 180)
  pins.digital_write_pin(DigitalPin.P0, 1)
input.on_button_pressed(Button.A, on_button_pressed_a)

def on_button_pressed_b():
  pins.servo_write_pin(AnalogPin.P2, 0)
  for index in range(5):
    pins.digital_write_pin(DigitalPin.P0, 1)
    basic.pause(250)
    pins.digital_write_pin(DigitalPin.P0, 0)
    basic.pause(250)
input.on_button_pressed(Button.B, on_button_pressed_b)
```

Are you ready to try it with the actual micro:bit now?

OK! first, click on the wacky name on the top, left of your screen and let's re-name this experiment we've been working on..



After you give it a new name, click on the "Tinkercad" logo right next to it. TinkerCad automatically saves your projects.