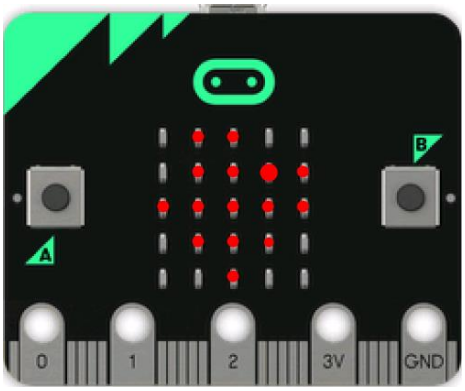
micro:bit **The Light Sensor**

In this lesson, you will monitor the light sensor on the micro:bit, collect the data in a python list and store the data in a TI-84 Plus CE list for further analysis using the calculator.

1. The micro:bit can read the ambient light level using the display LEDs. Yes, the display LEDs can also be used as an input device!



For more information on the micro:bit light level see:

[Sensing changes in light on the micro:bit : Help & Support]

2. In a new program (LITE) add the usual imports and the **while** loop.

   This is a good place to use the MBSTART template file suggested in an earlier activity.

3. Add another **import** statement at the top to access the micro:bit **display** menu:

   from mb_disp import *

   In the **while** loop body, write the assignment statement:

   b = display.read_light_level()

   Type the variable **b** and then get **= .read_light_level()** from **[math] Display…**

   Add a print statement to see what the function produces:

   print(“b = “, b)

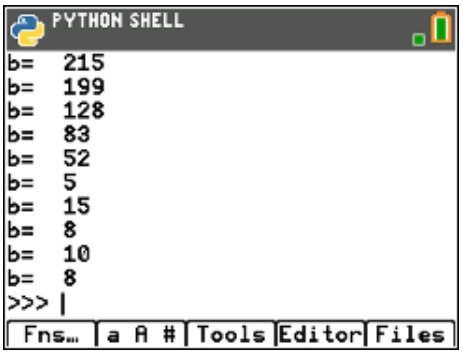
   Remember that these last two statements are *indented* so that they are both in the **while** loop body.

4. **<Run>** the program and point the display side of the micro:bit at a light

source. It does not matter what is showing on the display. Move the micro:bit toward and away from the light and observe the changing values on the TI-84 screen. You should see values varying between 0 and 255.

Note: if you do not see changing values as in this image, add a **sleep(100)** *statement to the loop to slow things down.*

As you probably expect, the further from the light source, the lower the light level value. Now you will add code to the program to collect the light level data. Then, using the calculator, you can create a scatter plot of light vs. time.



Press **[clear]** to end the program and then go back to the **<Editor>**.

5. Create two empty lists before your **while** loop:

   times = [ ]
   brites = [ ]

   Find the square brackets on the keypad, on **<a A #>**, on **<Fns…> List** or on **[list]**. You can use shorter variable names to save on typing (like **ts** and **bs**).

   Also before the **while** loop, add a statement to start a 'time' counter variable (**t**) at 0:

   t = 0

   Avoid using the word 'time' as a variable because there is a time module. It is a good practice to pluralize list names because they contain many values.

6. In the loop body, after the **print** statement, add a statement to increase the timer variable **t**. We will use a one second time interval between light readings, so use:

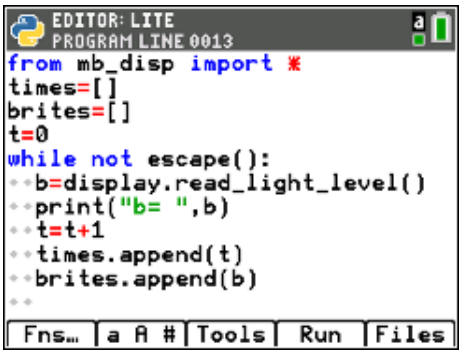
   t = t + 1

   Note: in Python you can also write this statement as: **t+=1**

7. Add the values of **t** and **b** to their respective lists using the statements:
   - **times.append(t)**
   - **brites.append(b)**

   .**append( )** is found on **<Fns…> List** and is pasted after typing the variable name.

   These statements add the current b (brightness) value and t (time) value to the end of the lists.



8. To control the timing of the sampling, add:
   - **sleep(1000)**

   after the two .**append** statements. This pauses data collection for one second between each sample.

   sleep( ) is included in the **microbit** module and is modified to use milliseconds.

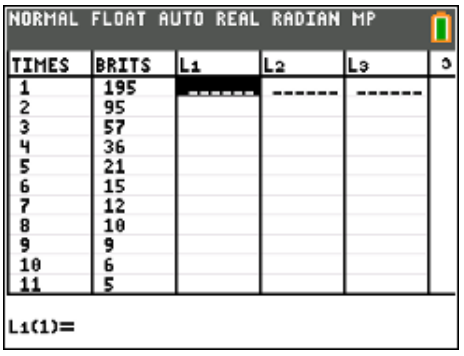
9. After the **while** loop ends, store the two Python lists into TI-84 Plus CE lists using **store_list( )** found on **[math] ti_system…**.
   The TI-84 list names (the arguments in **"QUOTES"**) must be 5 UPPERcase letters or less. We use BRITS and TIMES for those lists.
   The second argument of **store_list(,)** is the Python list variable to store.

   You can also use **"1"** for the built-n list L_1 and **"2"** for list L_2.

10. **<Run>** the program. Start with the micro:bit close to your light source. An exposed light bulb or a smartphone flashlight work well. Slowly but *steadily* move the micro:bit away from the light at a constant rate until the brightness reading is less than 10.

    Press **[clear]** to end the program.

    Repeat the process until you feel that you may have 'good' data. Displaying the data on the calculator screen using **print( )** or **disp_at( )** may be helpful.
    The sample data in this image of the TI-84 Plus CE Stat Editor shows some collected data in the lists TIMES and BRITS.



11. Quit the Python App and set up and view a scatter plot of the data (TIMES, BRITS).
Then use your calculator to find a function that 'best' fits your data. What physics principle is controlling the data?