Computing

Lesson 1: Hello Physical World

Physical Computing

Allen Heard

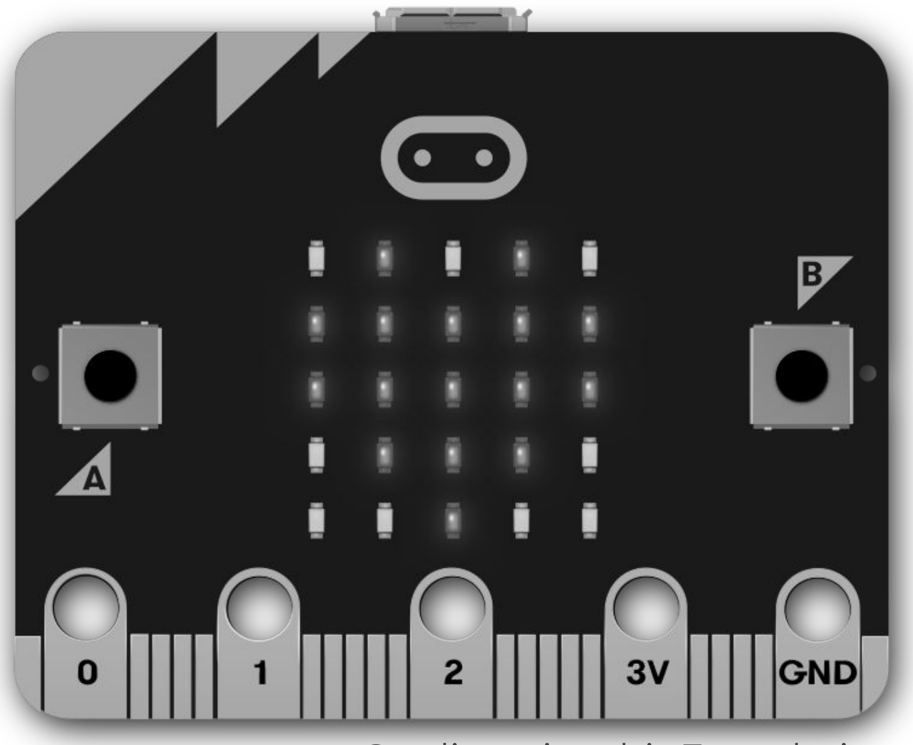
Materials from the Teach Computing Curriculum created by the National Centre for Computing Education

Task 1 - First steps - part 1

Type this program in your development environment.

```
from microbit import *
display.scroll("Hello there!")
```

Flash the program to your micro:bit to see it run.



Credit: micro:bit Foundation

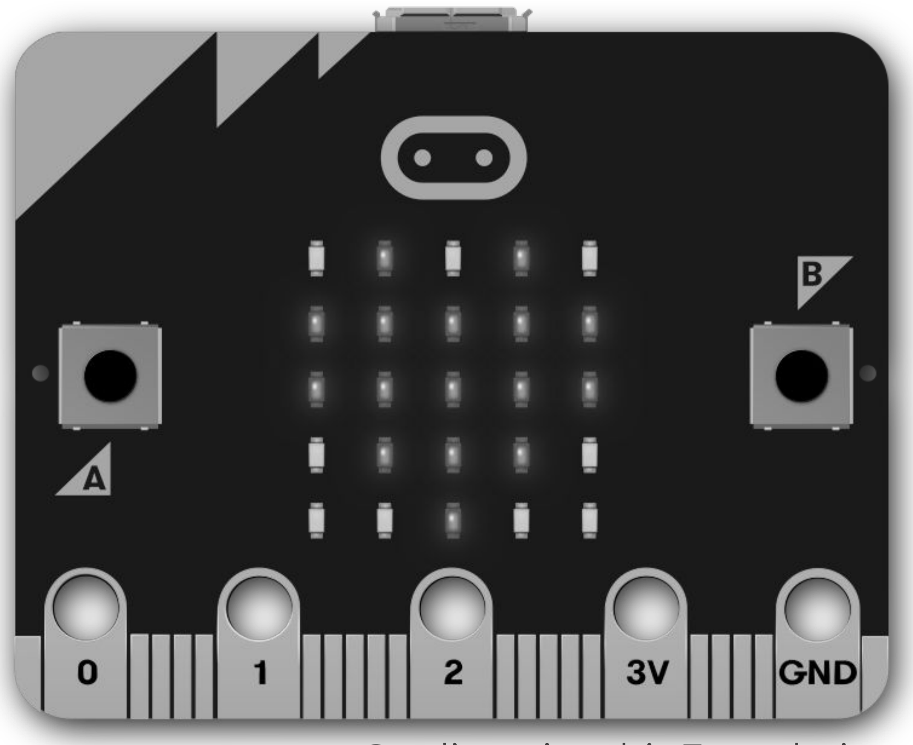
Task 1 - First steps - part 2

Change the word `scroll` to `show` in your program.

```
from microbit import *
display.show("Hello there!")
```

Flash the program to your micro:bit to see it run.

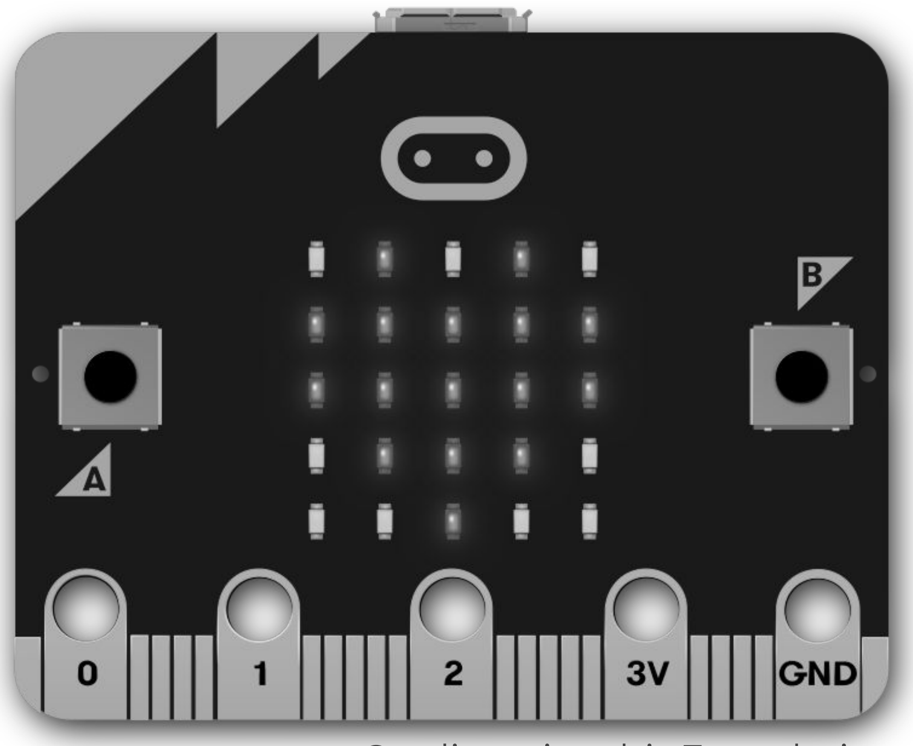
What was different?



Credit: micro:bit Foundation

Task 2: Displaying images

Using the worksheet, modify the code, replacing the text to be displayed with an image.



Credit: micro:bit Foundation

Part 1:

```
from microbit import *
display.show(Image.HEART)
```

Part 2:

Experiment with some other images, such as: `HAPPY, SAD, MEH, YES, NO, MUSIC_QUAVER`

You can find lots more, look up *micro:bit show image* on the web!

Task 3: Using the accelerometer

Fill in the missing condition in the `if`-statement. The “happy face” should appear when the accelerometer detects that the micro:bit is simply lying face up.

```
from microbit import *
if accelerometer.is_gesture("face up"):
    display.show(Image.HAPPY)
else:
    display.clear()
```

Flash the program to your micro:bit again, to see the modified version run. **You will most probably not see the “happy face” displayed.** The program is not working properly because the `if`-statement needs to be repeated, its condition checked over and over again.

Tips for resolving syntax errors

```
from microbit import *
while True:
    if accelerometer.is_gesture("face up"):
        display.show(Image.HAPPY)
    else:
        display.clear()
```

Syntax checklist

✓ **Python is case-sensitive**: Upper case and lower case characters are different.

✓ **Indentation matters**: spaces before a statement mean that it belongs inside a nested block.

✓ **Strings** (text literals) need to be enclosed in quotation marks.

Task 3: Using the accelerometer

Extend your program by nesting the statements into a `while` loop (make sure that the statements are indented). This will repeat the nested statements **forever** (since the while condition is always True).

```
from microbit import *
while True:
    if accelerometer.is_gesture("face up"):
        display.show(Image.HAPPY)
    else:
        display.clear()
```

Flash the program to your micro:bit again, to see the modified version run. Tilt, rotate, and shake the micro:bit. The “happy face” will only appear when the micro:bit is lying face up.

MicroPython cheat sheets

The following slides contain further reference information for using Python on the micro:bit. They include short explanations, brief notes, syntax, and selected examples for you to try.

- Display
- Display as a light sensor
- Buttons
- Accelerometer

The micro:bit module

The `microbit` module contains everything your programs will need to interact with the micro:bit hardware.

Start all your micro:bit programs with the line below. All the examples here assume this has been done.

Example:

```
from microbit import *
```

Display

The display on the micro:bit is a 5×5 LED matrix, represented by the `display` object.

Show or scroll a string or numerical value on the display, one character/digit at a time. With show, the value can also be an image or a list of images (to be animated).

Displaying values and images

Syntax:

```
display.show(value)
display.scroll(value)
```

Display - examples

```
display.show("Hello there!")
sleep(2000)
display.show(Image.HAPPY)
```

Display some text and one of the built-in images.

```
display.show(Image.ALL_CLOCKS,
             delay=1000, loop=True)
```

Loop over the list of built-in images Image.ALL_CLOCKS. Keyword arguments control the animation.

```
pattern = Image("01234:"
                "12345:"
                "23456:"
                "34567:"
                "45678")
display.show(pattern)
```

Create and display a custom image called pattern, by specifying the brightness of individual LEDs.

Display

Clearing the display

Syntax:

```
display.clear()
```

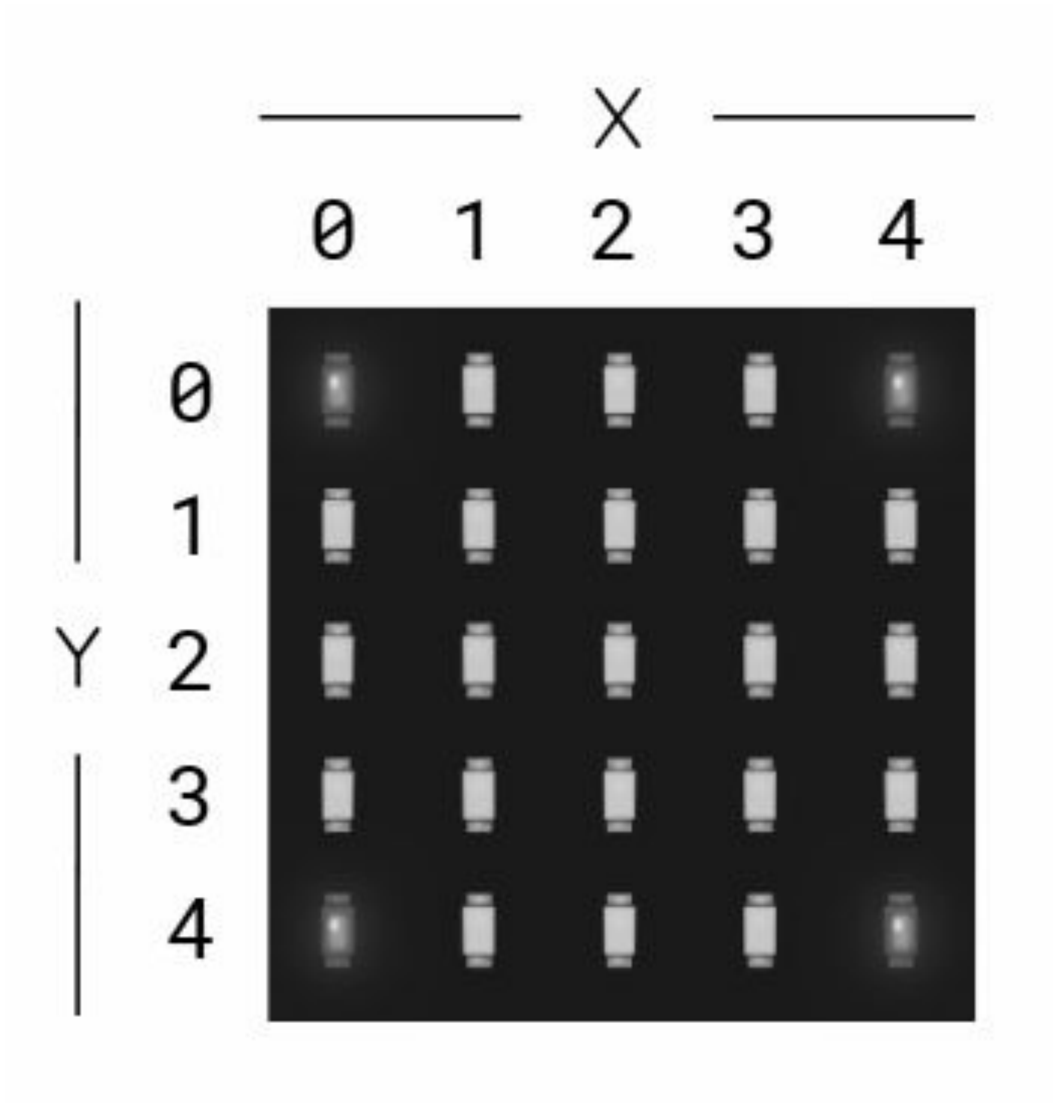
Turns all LEDs off.

Controlling individual LEDs

Syntax:

```
display.set_pixel(x,y,value)
```

Set the brightness of the LED at column x and row y to value, which has to be an integer between 0 and 9.



Display - examples

```
display.set_pixel(2, 2, 9)
```

Light up the central LED at (2,2).

```
for x in range(5):
  for y in range(5):
    display.set_pixel(x, y, x+y+1)
```

Iterate over all x and y coordinates. The brightness of each LED is set according to the sum of its coordinates.

Display (as a light sensor)

The display on the micro:bit can also be used to sense the amount of light falling on it.

Input from the light sensor
Syntax:

```
display.read_light_level()
```

Returns an integer (0-255) representing the light level.
an integer between 0 and 9.

Example:

```
light = display.read_light_level()
```

Buttons

The micro:bit offers two buttons, represented by the `button_a` and `button_b` objects.

Detecting button presses

Syntax:

```
button_a.is_pressed()
button_b.is_pressed()
```

Returns `True` if the specified button is currently being held down, and `False` otherwise.

Example:

```
while True:
  if button_a.is_pressed():
    display.show(Image.ARROW_W)
  elif button_b.is_pressed():
    display.show(Image.ARROW_E)
  else:
    display.show(Image.DIAMOND_SMALL)
```

Display an image depending on the button that is currently being pressed.

Detecting button presses - continued

Syntax:

```
button_a.was_pressed()
button_b.was_pressed()
```

Returns `True` or `False` to indicate if the button was pressed since the last time this method was called.

Calling this method will clear the press state, so that the button must be pressed again before this method will return `True` again.

Detecting button presses - continued

Example

```
counter = 0
while counter < 10:
  display.show(counter)
  if button_b.was_pressed():
    counter = counter + 1
```

Display a counter and increment it by 1 whenever button B is pressed. Stop the process when the counter reaches 10.

```
display.show(Image.ARROW_W)
while not button_a.was_pressed():
    pass
display.show(Image.HAPPY)
```

Wait until button A is pressed (`pass` is a statement that does nothing).

Accelerometer

The accelerometer in the micro:bit is able to detect gestures and acceleration in 3 axes.

Detecting gestures

Syntax:

```
accelerometer.is_gesture(name)
accelerometer.was_gesture(name)
```

These methods return `True` or `False` to indicate if the named gesture is currently active or if it was active since the last call.

These are the possible gesture name, represented as strings: `"up", "down", "left", "right", "face up", "face down", "freefall", "3g", "6g", "8g", "shake".`

Accelerometer

Example

```
while True:
  if
accelerometer.was_gesture("up"):
    display.show(Image.HAPPY)
    sleep(1000)
    display.clear()
```

Display an image for a second whenever an ‘up’ gesture is detected.