Project 4 - Sparkles!

For this project, we're going to write some code that interacts with some things on the micro:bit - buttons and the *accelerometer*. You know what buttons are - you press them and they do something.

But do you know what an accelerometer is?

An accelerometer is a device that measures movement. A lot of smart phones have accelerometers that can tell if the phone is moving up or down, or side to side.

The accelerometer in the micro:bit isn't as powerful as that, but it can detect some movement, especially if you shake the micro:bit.

First let's write some code. In your Mu editor, type or copy this code and then flash it to the micro:bit by clicking on the 'Flash' button.

```
from microbit import *
import random

images = [Image.HAPPY, Image.GHOST, Image.SKULL, Image.DUCK,
          Image.UMBRELLA, Image.HEART, Image.STICKFIGURE]

while True:
    x = random.randint(0, 4)
    y = random.randint(0, 4)
    brightness = random.randint(0, 9)
    display.set_pixel(x, y, brightness)
    if button_a.was_pressed():
        display.show(random.choice(images))
        sleep(1000)
    if button_b.was_pressed():
        display.scroll("Hello", delay=100)
        sleep(1000)
    if accelerometer.was_gesture('shake'):
        display.show(Image.ANGRY)
        sleep(1000)
```

One of the new things we're doing here is importing the `random` library - that tells us that we're going to be choosing some different things at random, such as integers and images.

The 'images' list is a list of possible items we can have the micro:bit display when we press one of the buttons.

The part that says "`while True:`" is important. This puts the micro:bit into an *infinite loop*.

Remember when we talked about `while` loops and how it's bad to get stuck in an infinite loop? Well this is a case where it's actually useful - it means the micro:bit is always checking for button presses, forever.

Let's look at the first thing that happens inside the `while` loop - this is where the sparkle effect is created:

```
    x = random.randint(0, 4)
    y = random.randint(0, 4)
    brightness = random.randint(0, 9)
    display.set_pixel(x, y, brightness)
```

'x' and 'y' are coordinates. If you look at the face on your micro:bit, you can see that there are tiny LEDs in the middle - 5 columns across, and 5 rows down.

To decide which LED to light up, we use the `randint()` function to pick a random number between 1 and 5 (using 0 through 4 because Python likes to start counting at 0).

'`brightness`' is another value we can set, using a random number between 1 and 10.

Finally, we call the function `display.set_pixel()`. Using the random x and y coordinates, that function selects one of the LEDs and lights it at a random brightness.

And since we're in a `while` loop, that code is loaded over and over, with different values every time. That's what makes the board 'sparkle'.

Now let's look at some of the ways we can interact with the micro:bit.

```
    if button_a.was_pressed():
        display.show(random.choice(images))
        sleep(1000)
    if button_b.was_pressed():
        display.scroll("Hello", delay=100)
        sleep(1000)
```

`button_a.was_pressed()` and `button_b.was_pressed()` are both functions that come with the microbit library.

We get to use them because we wrote `from microbit import *` at the top of our code. These functions return `True` if a button is pressed.

If button A is pressed, we choose a random image from our list and display it for a few seconds.

If button B is pressed, we scroll the word “Hello” (changing the ‘delay’ value will control how fast or slow the word scrolls).

```
    if accelerometer.was_gesture('shake'):
        display.show(Image.ANGRY)
        sleep(1000)
```

Next we check to see if the micro:bit is shaking - if it is, we show the ‘angry face’ image and pause for a few seconds.

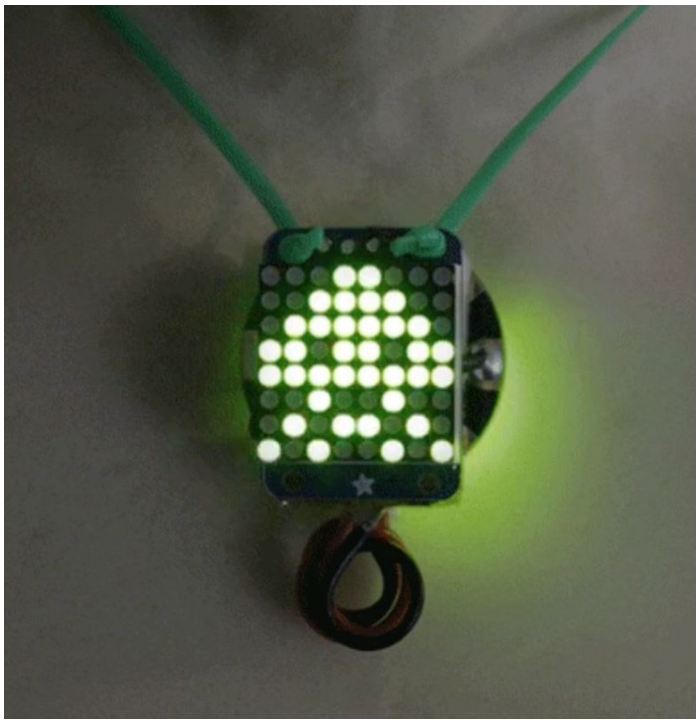
Since all this code is being executed inside a while loop, it will just keep on going forever (or until the batteries run out).

If you want to make the sparkling stop, just unplug the battery case from the back of the micro:bit board, then plug it back in when you’re ready to start again.

Now comes the fun part! Let’s string the micro:bit onto a necklace so that we can walk around showing off all that blinking!

Your finished necklace will look a lot like this.

The necklace in this picture was made using a different kind of *microcontroller* (the micro:bit is a type of microcontroller too).



Supplies you should have:

1 micro:bit
1 battery case
2 AAA batteries
1 piece of Velcro
1 piece of plastic cord

Step 1: Attach the Velcro

Peel the plastic backing off one side of the Velcro and apply the adhesive side to one side of your battery case.

Peel the plastic backing from the other side of the Velcro and *gently* apply the adhesive side to the back of the micro:bit, making sure not to cover up the holes at the bottom.

Step 2: Attach the cord

From the back of the micro:bit, slide one end of the cord through the hole marked ‘0’ and tie a small knot in the front so that the cord can’t slip back through. Do the same thing with the other end of your cord, passing it through the hole marked ‘GND’.

Step 3: Power it up!

If you haven’t already, be sure to put your batteries into the battery case. Then connect the battery pack to the micro:bit by plugging it into the port in the top lefthand corner of the board.

If you flashed your code to the micro:bit, it should start sparkling right away! Go ahead and slide it over your head, press some buttons, and shake it a little bit to see what happens. Remember: since the micro:bit will be hanging upside down, the buttons will be on opposite sides!

For more projects you can do with your micro:bit, check out these resources:

- **The MicroPython guide to BBC micro:bit**

  https://www.microbit.co.uk/python-guide

- **MicroPython/micro:bit documentation:**

  http://microbit-micropython.readthedocs.io/en/latest/

- **Micro:bit projects on Instructables:**

  http://www.instructables.com/howto/microbit/