Random

Sometimes you want to leave things to chance, or mix it up a little: you want the device to act randomly.

MicroPython comes with a `random` module to make it easy to introduce chance and a little chaos into your code. For example, here's how to scroll a random name across the display:

```
from microbit import *
import random

names = ["Mary", "Yolanda", "Damien", "Alia", "Kushal", "Mei Xiu", "Zoltan" ]

display.scroll(random.choice(names))
```

The list (`names`) contains seven names defined as strings of characters. The final line is *nested* (the "onion" effect introduced earlier): the `random.choice` method takes the `names` list as an argument and returns an item chosen at random. This item (the randomly chosen name) is the argument for `display.scroll`.

Can you modify the list to include your own set of names?

Random Numbers

Random numbers are very useful. They're common in games. Why else do we have dice?

MicroPython comes with several useful random number methods. Here's how to make a simple dice:

```
from microbit import *
import random

display.show(str(random.randint(1, 6)))
```

Every time the device is reset it displays a number between 1 and 6. You're starting to get familiar with *nesting*, so it's important to note that `random.randint` returns a whole number between the two arguments, inclusive (a whole number is also called an integer - hence the name of the method). Notice that because `display.show` expects a character then we use the `str` function to turn the numeric value into a character (we turn, for example, `6` into `"6"`).

If you know you'll always want a number between `0` and `N` then use the `random.randrange` method. If you give it a single argument it'll return random integers up to, but not including, the value of the argument `N` (this is different to the behaviour of `random.randint`).

Sometimes you need numbers with a decimal point in them. These are called *floating point* numbers and it's possible to generate such a number with the `random.random` method. This only returns values between `0.0` and `1.0` inclusive. If you need larger random floating point numbers add the results of `random.randrange` and `random.random` like this:

```
from microbit import *
import random

answer = random.randrange(100) + random.random()
display.scroll(str(answer))
```

Seeds of Chaos

The random number generators used by computers are not truly random. They just give random like results given a starting *seed* value. The seed is often generated from random-ish values such as the current time and/or readings from sensors such as the thermometers built into chips.

Sometimes you want to have repeatable random-ish behaviour: a source of randomness that is reproducible. It's like saying that you need the same five random values each time you throw a dice.

This is easy to achieve by setting the *seed* value. Given a known seed the random number generator will create the same set of random numbers. The seed is set with `random.seed` and any whole number (integer). This version of the dice program always produces the same results:

```
from microbit import *
import random

random.seed(1337)
while True:
    if button_a.was_pressed():
        display.show(str(random.randint(1, 6)))
```

Can you work out why this program needs us to press button A instead of reset the device as in the first dice example..?