Microbit Artist

The Microbit has quite a large number of images predefined that can be shown on the LED. This code can be used to view them by just changing the name in the brackets.

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

```
HEART, HEART_SMALL, HAPPY, SMILE, SAD, CONFUSED, ANGRY, ASLEEP,
SURPRISED, SILLY, FABULOUS, MEH, YES, NO, CLOCK12, CLOCK11, CLOCK10,
CLOCK9, CLOCK8, CLOCK7, CLOCK6, CLOCK5, CLOCK4, CLOCK3, CLOCK2, CLOCK1
ARROW_N, ARROW_NE, ARROW_E, ARROW_SE, ARROW_S, ARROW_SW, ARROW_W,
ARROW_NW, TRIANGLE, TRIANGLE_LEFT, CHESSBOARD, DIAMOND, DIAMOND_SMALL
SQUARE, SQUARE_SMALL, RABBIT, COW, MUSIC_CROTCHET, MUSIC_QUAVER,
MUSIC_QUAVERS, PITCHFORK, XMAS, PACMAN, TARGET, TSHIRT, ROLLERSKATE
DUCK, HOUSE, TORTOISE, BUTTERFLY, STICKFIGURE, GHOST, SWORD, GIRAFFE
SKULL, UMBRELLA, SNAKE.
```

You can also generate your own images by producing a series of numbers. The code needed to show one of a boat is this:

```
from microbit import *

boat = Image("00900:"
             "09990:"
             "00900:"
             "99999:"
             "09990")

display.show(boat)
```

But what happens if you are not very good at drawing. Wouldn't it be great if the Microbit could draw its own randomly generated pictures until a combination of LED produces an image you recognise as something?

Well obviously you can.

Random Art Generator

This exercise will utilise:

- Mu Editor
- Functions
- Functions within functions
- Random
- Empty, append and accessing lists
- If/else logic control
- While True loop

Make a Start

If you haven't already done so type in the code above to make the image of the boat. Here is the code again laid the way it is normally seen.

```
from microbit import *

boat = Image("00700:" "05750:" "55755:" "90709:" "09990")

display.show(boat)
```

The first line is needed to start any Python program with the Microbit. And if you have done some Microbit coding before you are aware of this. The last line quite simply shows on the display an image, in this case, referenced as the boat. It is the second line that really has all the information that makes an image and it is this that you are going to need to change each time a new image is needed.

Random Number

So let us start this by using the same code you have above and getting the Microbit to make a random number. The new code is in blue.

```
from microbit import *
from random import randint

boat = Image("00700:" "05750:" "55755:" "90709:" "09990")

while True:
      display.show(boat)

      LED = randint(0, 9)
      print(LED)
      sleep(2000)
```

Flash this code onto your Microbit. Then click the REPL button and then restart the Microbit using the button on the back. On the REPL every two seconds a random number should show between zero and 9.

This code now runs forever, until you stop the code, generating a number. It prints the random integer (randint) number, between 0 and 9, called LED in the REPL and then waits 2 seconds before doing it again. So these are the numbers that will need to go into the list, similar to the one of the boat, but with all the correct formatting added.

The Code Didn't Work?

There should be a colon **:** after True. True is not **t**rue, it's **T**rue. **W**hile is also **w**hile. The code after the True: should be indented by one tab on the Mu editor automatically but only if you remembered to put in the colon.

The Right Numbers for an Image

You are now generating numbers that could be used for an image. The LED on the Microbit can be lit by any number from zero to nine. But just to get a nice clear image for us to recognise as maybe something, this exercise will use just a nine for a lit LED and a zero for off. It is also slightly more interesting code to learn. You can always change this code once you have finished.

The random number generator can be changed to create either a zero or a one with:

LED = randint(0, 1)

But you need a 0 and a 9 not 0 and 1. Zero is good so that can be used but everytime the code generates a one it needs to be reset to the 9, like this.

```
from microbit import *
from random import randint

boat = Image("00700:" "05750:" "55755:" "90709:" "09990")

while True:
      display.show(boat)

      LED = randint(0, 1)
      if LED == 1:
            LED = 9
      print(LED)
      sleep(2000)
```

Flash the code and run the REPL again as before. There should now only be zero and 9. The new code reads, if the value of LED is the same as 1, then do the next indented line changing 1 to 9. If it is not a 1 then ignore the following indented line(s) because it must be a zero.

The Code Didn’t Work?

There is a colon after the line if LED == 9: . There is also a double equal sign == which reads as “is it the same as?”

First Function

The code to generate the random 0 or 9 is going to be used 25 times to make every image. It makes a lot of sense to take this out of the loop and make a function that is typed once but which can be called into use 25 times per image.

Change your code to read as below. There are two versions of the function. They do exactly the same thing. One might be easier to understand. Can you see how the first version is shorter but still works? Try it out by removing the # before the print and placing a # before the other print.

This will work exactly as before when run, nothing new except that now there is a function defined (def) called LED_value. This is “called” into action inside the print in the loop at the bottom. The return sends either the 0 back to be printed or the 9.

```
from microbit import *
from random import randint

boat = Image("00700:" "05750:" "55755:" "90709:" "09990")

def LED_value():
    LED = randint(0, 1)
    if LED == 1:    # if LED is 0 the next two lines are ignored
        return 9    # this returns a 9 if a 1 is generated
    return LED      # this returns the 0 unchanged to be printed

def LED_state_value_2():
    LED = randint(0, 1)
    if LED == 1:    # if LED is 0 the next two lines are ignored
        LED = 9
        return LED  # this returns a 9 if a 1 is generated
    else:           # else LED is 0 then do the next line
        return LED  # this returns the 0 to be printed

while True:
    display.show(boat)

    print(LED_value())
    #print(LED_value_version_2())
    sleep(2000)
```

A Second Function and Lists

Now we have a neat way of making the 0 and 9 for the images. The next step is to make a series of five values for a line of LED.

Firstly clean up the code removing the second function above so that you only have this:

```
from microbit import *
from random import randint

boat = Image("00700:" "05750:" "55755:" "90709:" "09990")

def LED_value():
    LED = randint(0, 1)
    if LED == 1:    # if LED is 0 the next two lines are ignored
        return 9    # this returns a 9 if a 1 is generated
    return LED      # this returns the 0 unchanged to be printed

while True:
    display.show(boat)

    print(LED_value())
    sleep(2000)
```

Flash the code and open the REPL but **do not restart the Microbit.** The next piece will use the REPL to code live in Python.

The new function is going to build up a list of five values to make up a line of LED values. So this is a practice on using lists. Here create an empty list called row.

In the REPL type this after the >>>

```
row = []
row.append(0)
row
```

The REPL should produce:

```
[0]
```

This shows that the empty list called Line now has a zero in it. Now append it four more times.

```
row.append[0]
row.append[9]
row.append[9]
row.append[0]
row
```

And the result should be:

```
[0, 0, 9, 9, 0]
```

This is the beginning of a line of values for the image. You will need five rows per image. But the format is incorrect. It needs to be “0, 0, 9, 9, 0:”

To do this we create a new empty string variable (string_value) and add each value to it but as a string not an integer. That is the integer is changed into text. The dots will add automatically. Keep entering until they stop.

```
string_value = “”
for value in row:
... string_value = string_value + str(value)
...
...
...
row
[0, 0, 9, 9, 0]
string_value
“00990”
```

This shows that the row has been converted to string_value. No longer numbers of a string of text. This is shown by the “ ”. The very last piece of the puzzle is to add a colon (:) at the end of the first four sets of values. Here are the boat image values to remind you.

```
boat = Image("00700:" "05750:" "55755:" "90709:" "09990")
```

Type this into the REPL:

```
row_value = string_value + “:”
row_value
“00990:”
```

You should now see how each line is built up. If this is done five times and then the result is displayed a randomly created image will show.

Turn off the REPL and return to the code.

Final Code

The final code is here. Flash to the Microbit, restart using the REPL and as each image is shown the actual values will be printed. The best ones can then be written down and kept for later use.

```
from microbit import *
from random import randint

# obtain pixel value for an on 9 or off 0
def LED_value():
    LED = randint(0, 1)
    if LED == 1:  # if it is 1 change to 9 (a bright LED)
        return 9
    return LED  # if it is zero then leave as is

# build a number list from values from above
def image_values():
    line = []                   # create an empty list
    for onoff in range(0, 5):   # for each of the 5 LED
        LED = LED_value()       # get either a zero or 9
        line.append(LED)        # build the row of 0/9
    return line                 # send line list back

# building a line of text from image_values above
def build_row():
    line = image_values()       # obtain the line list above
    row = ""                    # create an empty string
    for LED in line:            # for each 0/9 convert to string
        row = row + str(LED)    # then add to row string
    return row                  # send string list back

while True:
    rowA = build_row() + ":"  # build first line
    rowB = build_row() + ":"  # build second line
    rowC = build_row() + ":"  # build third line
    rowD = build_row() + ":"  # build fourth line
    rowE = build_row()        # build fifth line without :

    all_rows = (rowA) + (rowB) + (rowC) + (rowD) + (rowE)

    # display the produced image
    random_image = Image(all_rows)
    display.show(random_image)
    print(all_rows)
    sleep(1500)
```