BBC Micro: bits – Rock, Paper, Scissors

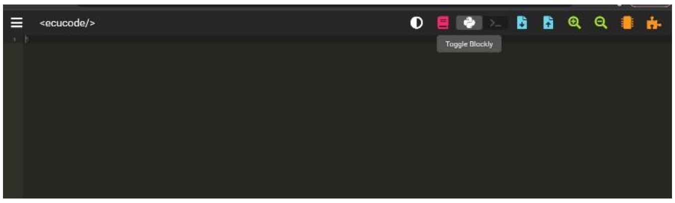
ECU Makerspace Group

Written by Julia Szymanski

Getting started

Let's do a practice run just using the Micro: bit LED's. Once you have access to the computers, follow through this guide. If your computer is not on the makerspace website then use this link to gain access. **Please use Google Chrome to access this website (makerspace.ecu.edu.au)**

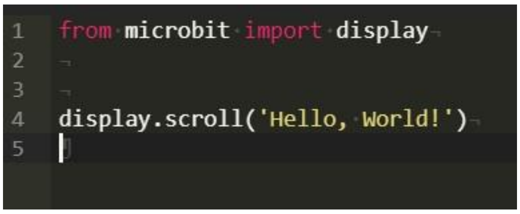
Click on **'Toggle Blockly'** to switch to Python code.



Task 1 – Print your name!:

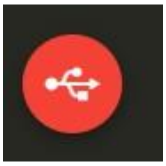
Step One – Print your name:

Click on to the first line and import 'display' from 'microbit', then display a message using display.scroll(" ").

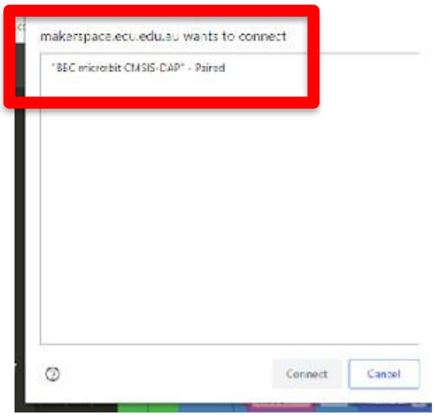


```
from microbit import display


display.scroll('Hello, World!')

```

Step Two - Now that you have the scroll message on the page, write your name and upload it to the Micro: Bit. Do this by following these instructions:



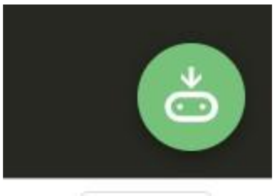
Click on the red button then a box comes up inviting you to pair the micro: bit. Click on the "BBC Micro: bit CMSIS-DAP"

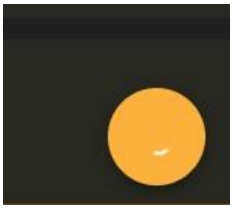


makerspace.ecu.edu.au wants to connect

"BBC micro:bit CMSIS-DAP" - Paired

Once this has been completed click on the green button to download. An orange button will appear as it is downloading.

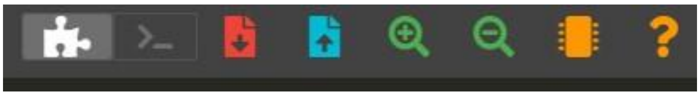




Congratulations! Remember to delete this scroll message before starting the next section.

Troubleshooting:

Updating the firmware on the Micro: bit

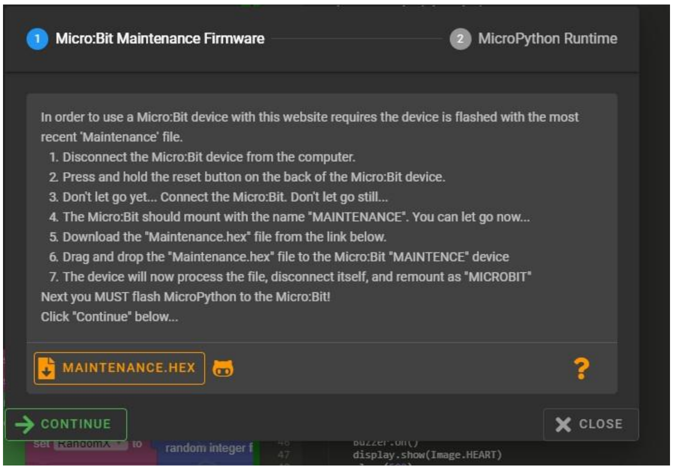


Click on the orange shape which appears 'update micro: bit'. Follow set instructions on the **Maintenance Firmware** guide.

BBC Micro: bits – Rock, Paper, Scissors

ECU Makerspace Group

Written by Julia Szymanski



In order to use a Micro:Bit device with this website requires the device is flashed with the most recent 'Maintenance' file.

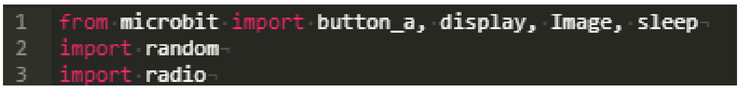
1. Disconnect the Micro:Bit device from the computer.
2. Press and hold the reset button on the back of the Micro:Bit device.
3. Don't let go yet... Connect the Micro:Bit. Don't let go still...
4. The Micro:Bit should mount with the name "MAINTENANCE". You can let go now...
5. Download the "Maintenance.hex" file from the link below.
6. Drag and drop the "Maintenance.hex" file to the Micro:Bit "MAINTENCE" device
7. The device will now process the file, disconnect itself, and remount as "MICROBIT"

Next you MUST flash MicroPython to the Micro:Bit!
Click "Continue" below...

If the micro: bit appears not to be responding. Unplug the micro: bit, press the reset button at the back of the micro: bit and repeat this process. If the micro: bit continues to fail, swap it for another one!

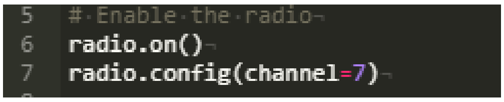
Now let's learn how to code our own Rock, Paper, Scissors and see our Microbits compete!

First, we need to import a few libraries so that we can use the button on the Microbit, display an image and get the Microbit to create a random number for us (for our Rock, Paper, Scissors)



```python
from microbit import button_a, display, Image, sleep
import random
import radio
```

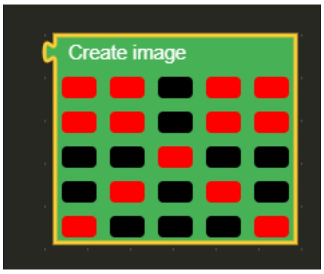
In the first step, we need to switch on the radio and set a communication channel, so that our two Microbits can talk to each other, but not to other Microbits around.



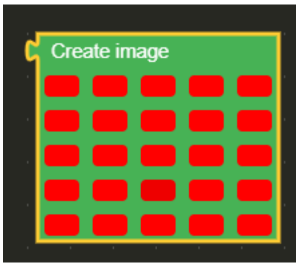
```python
# Enable the radio
radio.on()
radio.config(channel=7)
```

In our game, we need three variable – one for rock, one for paper and one for scissors. They will have a different image each:

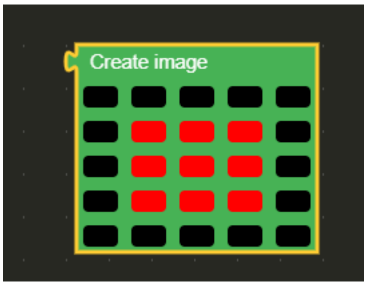
Scissors:



Paper:



Rock:



BBC Micro: bits – Rock, Paper, Scissors

ECU Makerspace Group

Written by Julia Szymanski

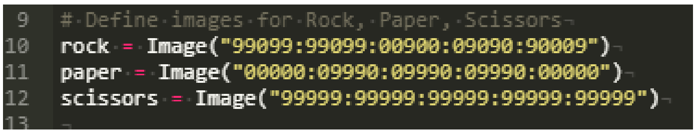
Remember, the Microbit has got 5 LED’s in each row, and 5 LED’s in each column.
"99099:99099:00900:09090:90009"

ROW1 ROW2 ROW3 ROW4 ROW5 This means that for Row 1 and 2, the brightness of LED 1,2,4 and 5 is set to 9 (the maximum brightness).

For Row 3, only LED 3 is set to a brightness of 9, all other LED’s in this row are dark. For Row 4,LED’s 2 and 4 are lit up. For Row 5, only LED’s 1 and 5 light up. Together, this resembles an image of scissors.

Can you do the same for rock and paper?

Once we have each of them set up, we assign the images to variables so we can refer to them later.

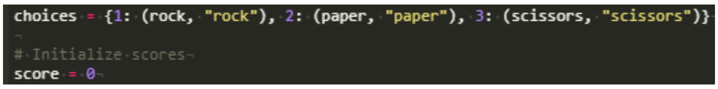


```
# Define images for Rock, Paper, Scissors
rock = Image("99099:99099:00900:09090:90009")
paper = Image("00000:09990:09990:09990:00000")
scissors = Image("99999:99999:99999:99999:99999")

```

We then need to create a variable that holds both the assigned numbers (1 for rock, 2 for paper and 3 for scissors) as well as the variable names and a string.

The variable choices will keep track of this.

We also need to create a variable to keep track of the score. As we start with a score of 0, we set score to 0.



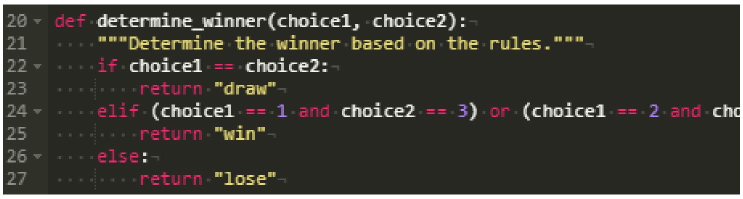
```
choices = {1: (rock, "rock"), 2: (paper, "paper"), 3: (scissors, "scissors")}

# Initialize scores
score = 0
```

So that we can play by the Rock Paper Scissor rules, we need to create a logic that checks who won. One Microbit will be assigned choice1 and the other one the variable choice2. This way we can compare the outputs and let the program decide who won the game.

We can create a function to take care of this. We will name the function determine_winner() and the input parameters for the function are choice1 and choice2, so either rock, paper or scissors for both of the Microbits.

If both Microbit have the same output, no one wins.

If one Microbits outputs rock and the other scissors, or one Microbit outputs paper and the other one outputs rock, or one microbit outputs scissors and the other one paper, the first microbit wins and we want the program to return “win” or “lose” for either of the microbits.



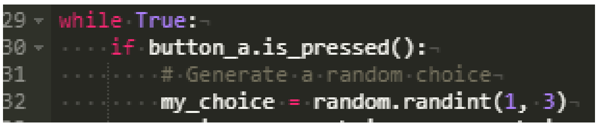
```python
def determine_winner(choice1, choice2):
    """Determine the winner based on the rules."""
    if choice1 == choice2:
        return "draw"
    elif (choice1 == 1 and choice2 == 3) or (choice1 == 2 and cho
        return "win"
    else:
        return "lose"
```

We want to start the game if button A is pressed, so we need to set up w While True loop that continues after the button has been pressed.

All of the remaining logic for our game will happen within the While True loop.

First, we need to check whether button A has been pressed.

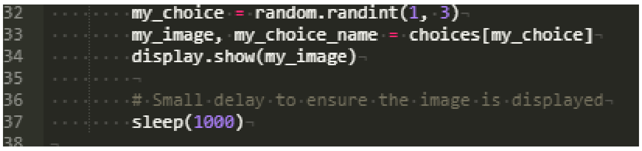
If button A has been pressed, we let the program randomly select a choice between 1 and 3 (either rock, paper or scissors)



```python
while True:
    if button_a.is_pressed():
        # Generate a random choice
        my_choice = random.randint(1, 3)
```

In the next step, we want our Microbit to show the corresponding image for the random choice that it selected.

We assign the image and choice name to the selected choice and get the Microbit to display the corresponding image. Make sure that the image doesn't disappear straight away by setting a sleep timer. The image will be shown for as many milliseconds as the sleep timer is set to. In this example we will use 1000 ms.



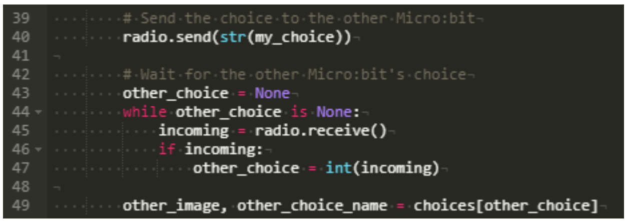
```python
        my_choice = random.randint(1, 3)
        my_image, my_choice_name = choices[my_choice]
        display.show(my_image)

        # Small delay to ensure the image is displayed
        sleep(1000)
```

Well done, you have completed the first part!

In the next part we need to implement some functionality so that the two Microbits compare the random choices (rock, paper or scissors) and decide who wins.

For this, we need to send the random choice from each Microbit to the other one, while we wait for the choice from the other Microbit to be sent.

We also need to create two variables named other_image and other_choice to keep track of the image and random choice from the other Microbit.



```python
        # Send the choice to the other Micro:bit
        radio.send(str(my_choice))

        # Wait for the other Micro:bit's choice
        other_choice = None
        while other_choice is None:
            incoming = radio.receive()
            if incoming:
                other_choice = int(incoming)

        other_image, other_choice_name = choices[other_choice]
```

BBC Micro: bits – Rock, Paper, Scissors

ECU Makerspace Group

Written by Julia Szymanski

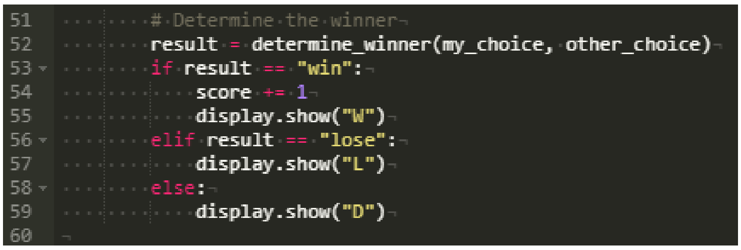
Now it is time to see what the result is!

For this, we create a result variable that will hold the result and we call the determine_winner() function that we created earlier, with two input parameters: my_choice and other_choice so we can compare the random choices from each Microbit.

If a Microbit wins, we want to increase the score by 1 and show “W” on the winning microbit.

If a Microbit loses, we want to show “L” on the losing Microbit.

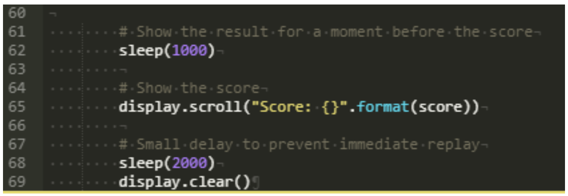
If neither of them win, we show “D” for draw.



```
        # Determine the winner
        result = determine_winner(my_choice, other_choice)
        if result == "win":
            score += 1
            display.show("W")
        elif result == "lose":
            display.show("L")
        else:
            display.show("D")

```

Lets also include a sleep timer so that can see the result for 1000 ms before it disappears.

All we need to do now is to format the score to ensure it can be displayed and make sure that the display gets cleared after each round of Rock, Paper, Scissors.



```

        # Show the result for a moment before the score
        sleep(1000)

        # Show the score
        display.scroll("Score: {}".format(score))

        # Small delay to prevent immediate replay
        sleep(2000)
        display.clear()
```

Well done! You can now play Rock Paper Scissors!

Challenge: Can you adjust your code so you can play with your neighbour?