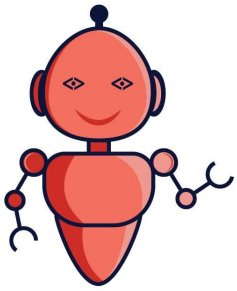
2018



KidzCanCode

IGNITE, INSPIRE, INNOVATE

www.kidzcancode.com

Higher or Lower Game

KIDZCANCODE

TREVOR WARREN

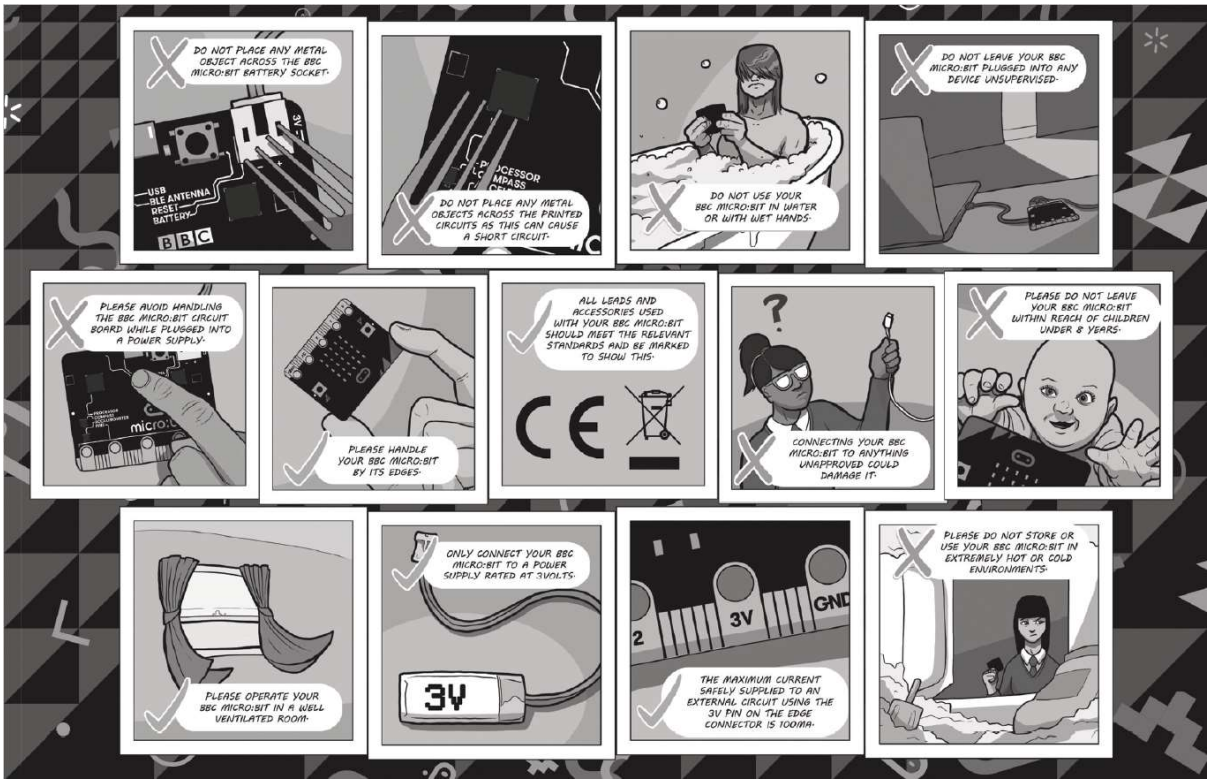
Tutorial 2.1i – Higher Or Lower Game

1. Safety Warnings

THE MICRO:BIT IS AN EXPOSED BOARD, TO BE USED WITH CARE PLEASE RETAIN THIS INFORMATION FOR FUTURE REFERENCE. You can read the detailed document at - http://microbit.org/guide/safety-advice/

a. General Safety Warnings

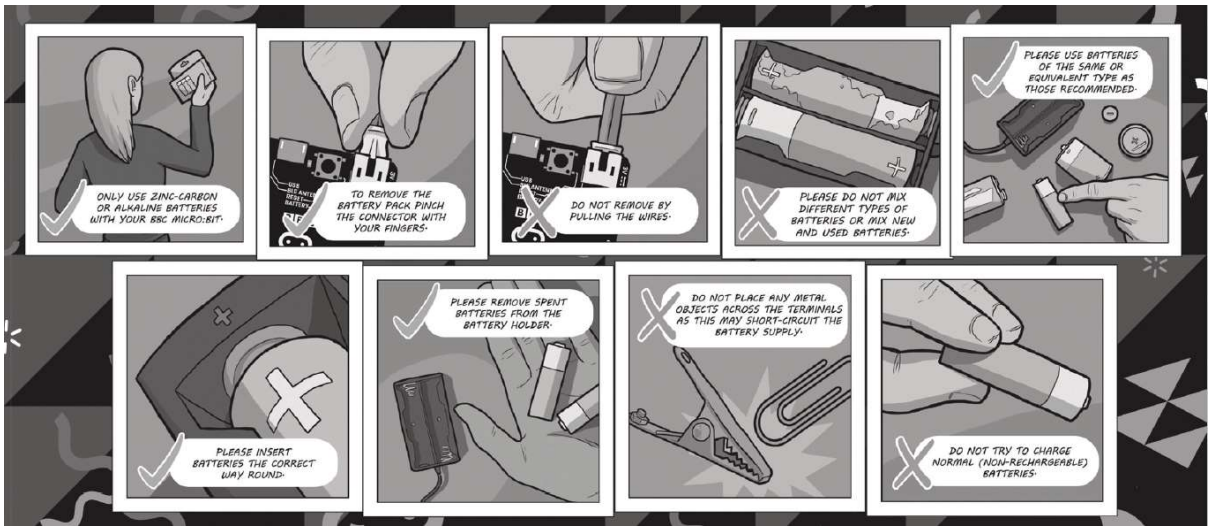
Using the BBC micro:bit is easy to use but is designed to have all the electrical parts on display. This does mean there's a small risk that the parts can be damaged and even overheat with a risk of injury but a little bit of care and caution will ensure you and your micro:bit will stay fit and healthy.



1. Always keep your BBC micro:bit in the anti-static bag when not in use. It's good practice for students to earth themselves before handling it.
2. Please handle your BBC micro:bit by its edges. This minimises the risk of damage through an electrostatic discharge.
3. Please use the battery pack and the USB lead provided to power your micro:bit. Do not use portable battery chargers or USB charging ports (often marked with a lightning bolt or 'SS'), to power your micro:bit. Using these may damage your micro:bit and stop it working properly.
4. Please avoid handling the BBC micro:bit circuit board while plugged into a power supply.
5. All peripherals (for example: USB cable, battery holder, sensors) used with your BBC micro:bit should comply with the relevant standards and should be marked accordingly.
6. Connecting your BBC micro:bit to any unapproved peripherals could damage your BBC micro:bit
7. Please do not attempt to keep using faulty micro:bits. If a school-issued micro:bit develops a fault, contact the vendor immediately.

8. The maximum current safely supplied to an external circuit using the 3V pin on the edge connector is 100mA. Please make sure this limit is not exceeded.
9. Please do not store or use your BBC micro:bit in extremely hot or cold environments.
10. Do not place any metal objects across the printed circuits on the board as this can cause a short circuit damaging your BBC micro:bit. This can cause risk of burn or fire.
11. Do not use your BBC micro:bit in water or with wet hands.
12. Do not leave your BBC micro:bit plugged into a computer or any other device unsupervised.
13. Please do not leave your BBC micro:bit within reach of children under 8 years of age.
14. Please operate your BBC micro:bit in a well ventilated room To remove the battery pack, pinch the connector with your fingers. Do not remove by pulling the wires.

b. Battery Warnings



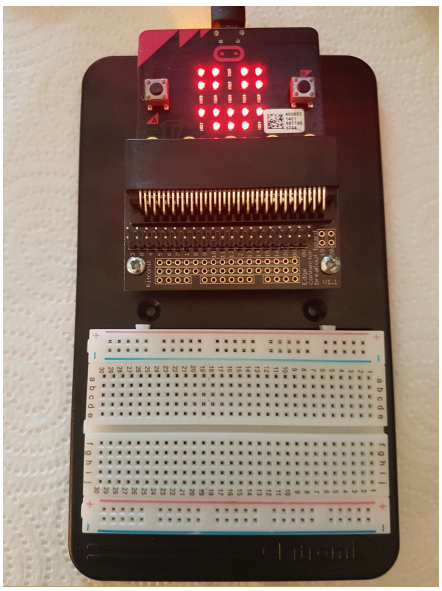
1. Do not try to charge normal (non-rechargeable) batteries
2. Please do not mix different types of batteries or mix new and used batteries.
3. Please use batteries of the same or equivalent type as those recommended.
4. Please insert batteries the correct way round (with the correct polarity).
5. Please remove spent batteries from the battery holder.
6. Do not short-circuit the battery supply terminals, for example by placing a metal object across the terminals.
7. Only use Zinc or Alkaline batteries with your BBC micro:bit.
8. Please do not use rechargeable batteries

2. Pre-requisites

If you have questions with the assembly of the micro:bit, edge connector breakout board, mounting board and the breadboard please drop us a note at help@kidzcancode.com. The edge connector board, mounting board and the breadboard are part of the Kitronix Inventors kit which needs to be purchased separately.

To be able to perform this tutorial you will need the following components –

1. Parts required –
    a. 1 x BBC Micro:bit
    b. 1 x Mounting Plate
    c. 1 x Edge connector breakout board
    d. 1 x Bread board
2. Assembly required –
    a. Bread board mounted on top of the mounting plate
    b. BBC Micro:bit inserted into the Edge Connector breakout board



Before proceeding please check your setup and confirm that all the required parts are configured as demonstrated in the above picture.

3. Learning Objectives

The objectives of this tutorial are to introduce the student to the following concepts –

- Logic of If-Then-Else statements
- Non repeating code
- Using custom variables and assigning them during certain times in the program execution

The BBC micro:bit is a powerful little computer. Through programming these games kids explore more advanced computer science concepts. Along the way kids are encouraged to share, create and extend the games using their own imagination and creativity.

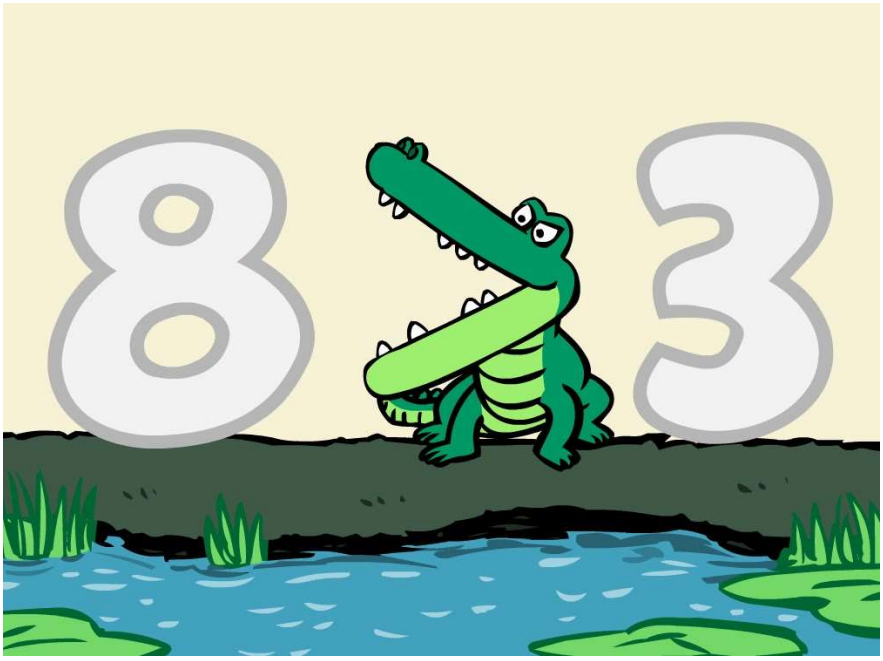
This tutorial builds upon concepts introduced in previous tutorials so please make sure you have covered the previous tutorials before you dive into this one. So overall this tutorial intends to build upon concepts learnt in previous tutorials while exploring new concepts.

In future tutorials we will continue to build upon the concepts learned here and will build more complex interactive games using the functionality provided by the micro:bit.

4. Activity

a. Activity

This activity involves designing a game of higher or lower. The game involves the person having 3 lives and being given a random guess number of 0 – 100 that they will have to use for the entire game. There is also a hidden number that the user can't see and must guess. If they guess correctly, their score will go up by one and if not, the lives they have will go down by one. The main challenge here is using logic statements and checking if the number is higher or lower.



The activity is designed to have additional challenges which allow the developer to keep pushing the boundaries. The main challenges in this activity involve –

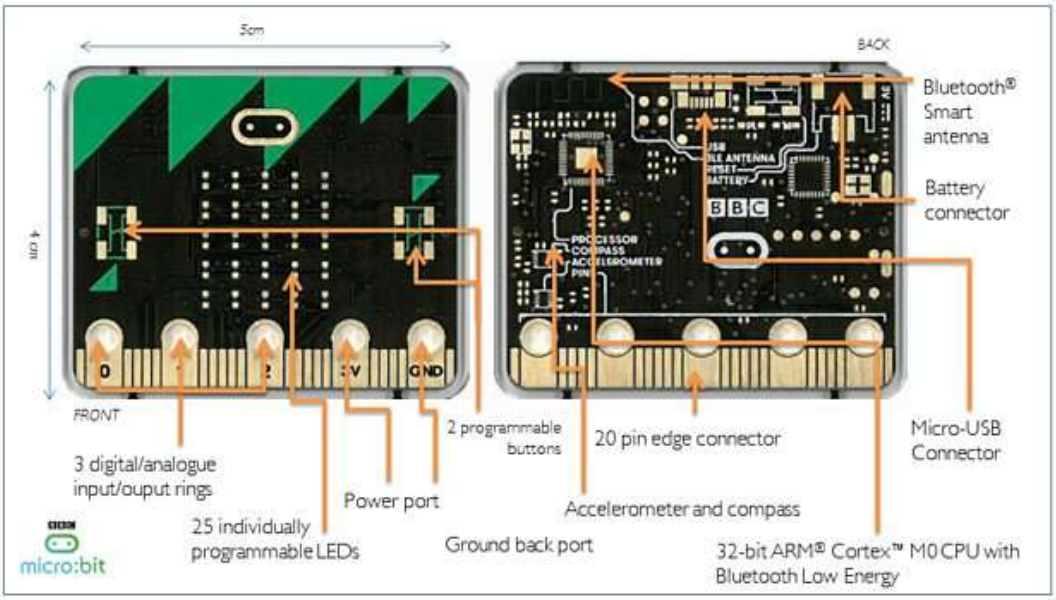
- Working with logic statements
- Using custom variables
- Making the code clean by using DRY and custom functions

b. How Does It Work

This section talks about the BBC micro:bit and what it's made up of. If you have already read through this section then feel free to skip directly to the next section. The BBC micro:bit is a powerful little board and has various types of sensors on board. Here's what makes up the BBC micro:bit.

1. Size: approx. 5cm x 4cm.
2. Weight: 8g.
3. Processor: 32-bit ARM Cortex M0 CPU.
4. Bluetooth Low Energy.
5. Digital Compass.
6. Accelerometer.
7. Micro-USB controller.
8. 5x5 LED matrix with 25 red LEDs.
9. Pins for connecting external sensors, LED's, etc.

Here's what the micro:bit looks like –



Front of the board (left hand side)

1. Button A (left button with edge connector at the bottom) – labelled A on the board
2. Button B (right button with edge connector at the bottom) – labelled B on the board
3. P0 (left large pin (crocodile clip port) with edge connector at the bottom) - labelled 0 on the board
4. P1 (middle large pin (crocodile clip port) with edge connector at the bottom) - labelled 1 on the board
5. P2 (right large pin (crocodile clip port) with edge connector at the bottom) - labelled 2 on the board
6. +3V - labelled 3V on the board. This is 3V PWR OUT
7. GND
8. P3 – P22 pins from left to right with edge connector at the bottom. Referred to as Pins when referencing that part of the board. Text will talk about 'pins' when referring to

individual connections or the general way of connecting to the board – not labelled on the front of the board
9. LED matrix referred as the 'screen' - not labelled on the board
10. LED coordinates starting at 0,0 top left corner and ending at 4,4 at the bottom corner - not labelled on the board

The order of the large pins as follows: P0 P1 P2 3V GND labelled 0, 1, 2, 3V GND on the board

Rear of the board (Right hand side)

1. 1. USB Plug (Micro-USB plug) – labelled USB on the board
2. Button R (reset button) – labelled Reset on the board
3. Status LED – not labelled on the board
4. Battery socket – labelled Battery on the board

Other components on the board include

1. Accelerometer
2. Compass
3. Bluetooth Smart Technology Antenna
4. AAA Battery Holder - not labelled on the board
5. Processor (Cortex M0)

The BBC micro:bit is programmable in a few different languages. You can write code for the micro:bit using the Makecode block coding interface, Javascript, Python of even in C. Most of our tutorials will cover the use of the Makecode block coding interface built by Microsoft for the micro:bit.

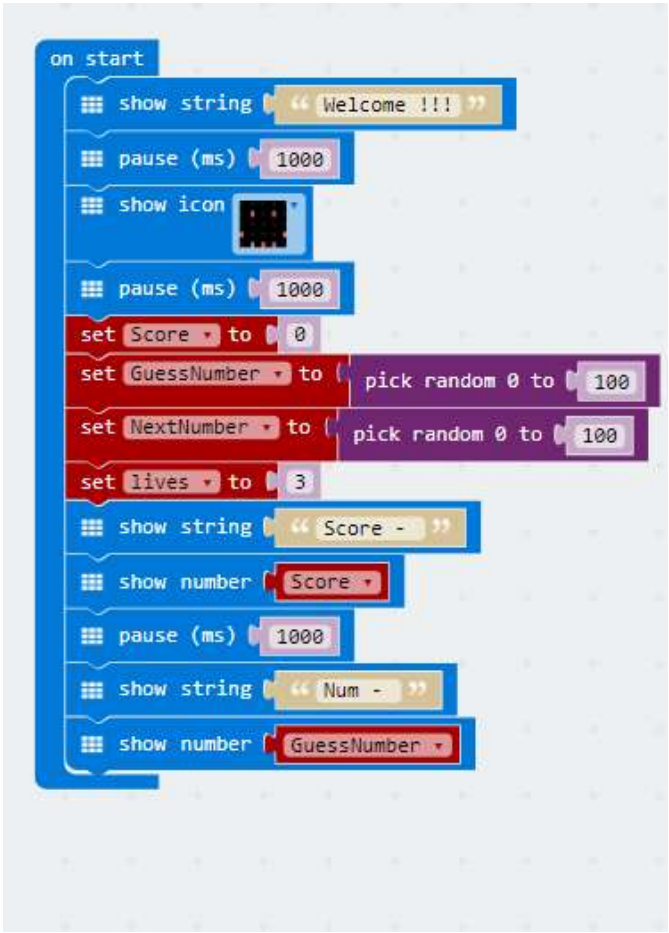
5. Let's write some code

It's time to write some code and get going with coding our game. So let's head over to the micro:bit block code editor page (https://makecode.microbit.org/) and get coding!!!

a. On Start

In the following section we will dive into the code you will put together for this tutorial. The code is split into various different sections just to make things a bit easier to comprehend. The first code block below covers off concepts that you would have covered in previous tutorials so while we cover the relevant blocks, we will not dive into a lot of depth here.

In the first code block, we use the "on start" block provided by the micro:bit which is run only "once" during a given program. You can only trigger the "on start" block once again when you hit the re-set button or pull the power plug and reboot the board. So please do keep that in mind with regards to the "on start" block of code. You want to put stuff into the "on-start" code block that you want run at the start of the program.



a) The "on start" block of code simply shows a string 'Welcome !!!'
b) We then pause briefly to ensure that the reader has been able to see the text being displayed
c) We then show a pre-defined icon
d) We again pause briefly for a second
e) We use the "set" commands to initialize a series of variables required

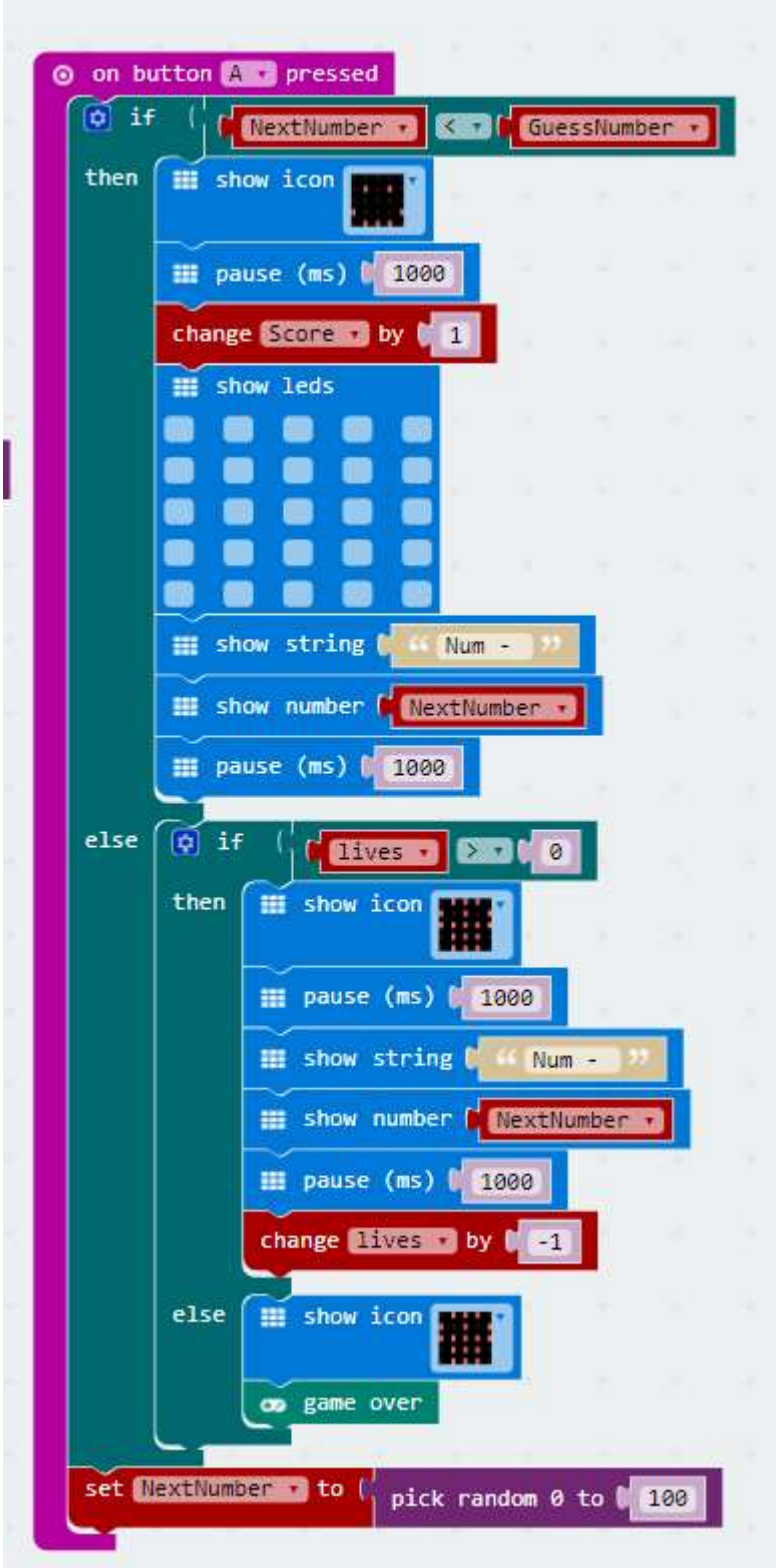
f) We now show a string that reads 'Score –' and then show the number of the custom variable 'Score'
g) We do the same thing as above but for the randomly generated number for the person playing to guess higher or lower on.

This brings the on-start block of code to an end. Feel free to dive in and customize the "on-start" code block with additional code that you want to put in. Please note that the addition of the "pause" commands (similar to our use of the wait commands in scratch) is intended to inject wait and slow down processing so that the flow of the program makes sense to the human being. To the computer, not having the wait command just lets it breeze through all the code one instructions after another.

b. On Button A Pressed

Let's now put together the listener for when the Button A is pressed. This will check if the hidden number is **less** than the 'guess' number shown to the player. Let's break down what is happening in this block of code step by step.

Tutorial 2.1i – Higher Or Lower Game



a) We use a If-Then-Else logic statement to check if the hidden number is **less** than the 'guess' number shown to the player

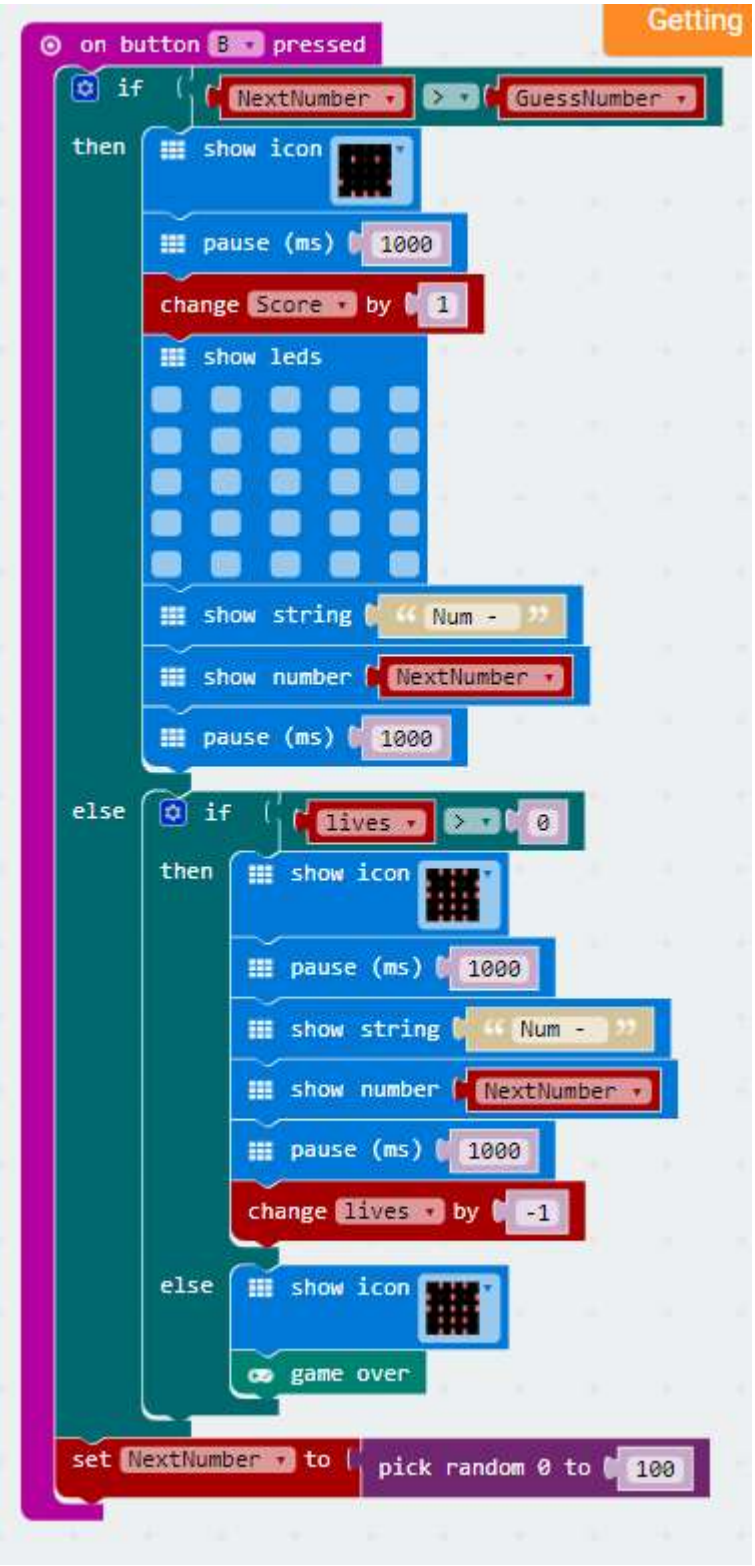
b) If this is true (then the player is correct), we show a happy icon to indicate the guess is correct
    a. We then pause for 1 second. (1 Second = 1000)
    b. We increase the player score by 1 and then clear the screen by showing no LED's
    c. The player is shown the number 'NextNumber' which is the hidden number.
    d. Skip to the bottom and the 'NextNumber' is now set to a new random number for the next round to begin!
c) If this is false (then the player guessed wrong), we use another If-Then-Else to check if the player has any lives left, otherwise show an icon and present game over to the player.
    a. Otherwise, show an icon and pause for 1 second.
    b. We show the hidden number that the player guessed wrong
    c. We decrease the number of lives the player has by 1
    d. Set the 'NextNumber' to a new random number

Now that we have put together the main functionality of the game, the next part is practically identical

c. On Button B Pressed

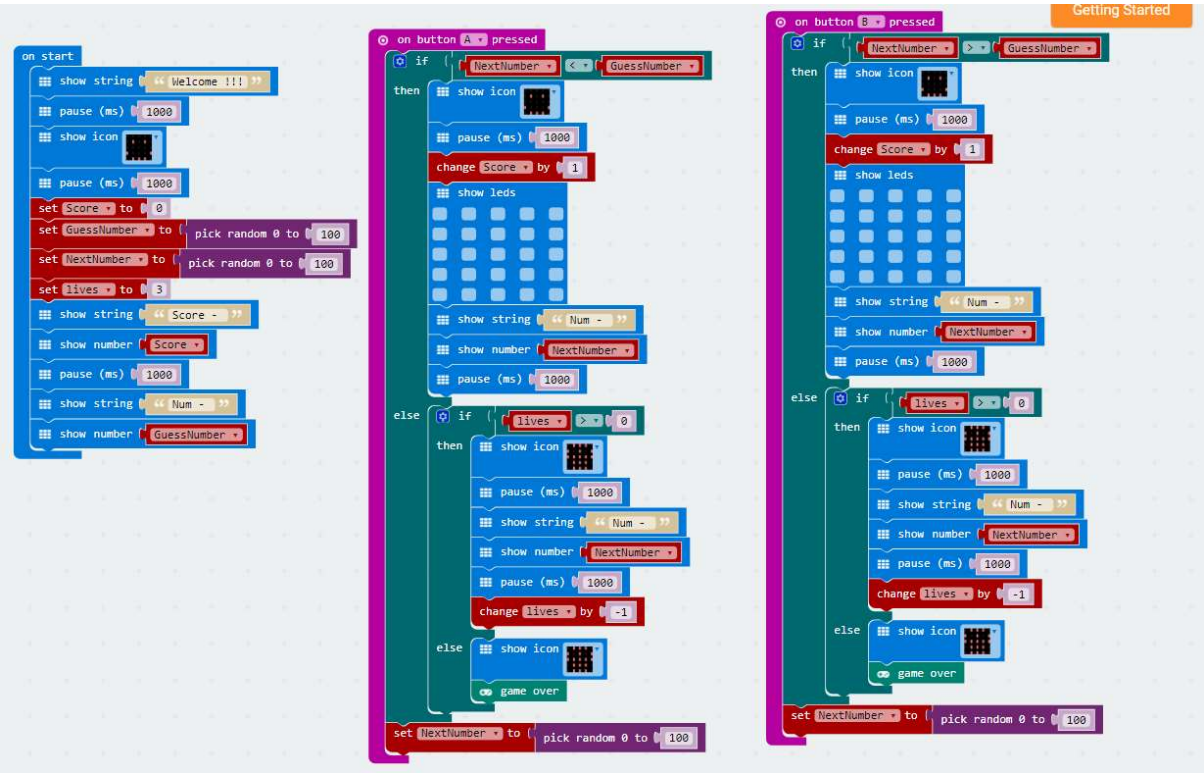
This code block runs when the B button is pressed. Now, here the same thing happens as above when the A button is pressed but one thing is different. Can you see what it is? Here we are checking if the hidden (nextNumber) number is **greater** than the 'guessNumber' that is set at the beginning of the game. The rest of the code is the same as above. However, note that we are repeating code here a lot. Is there something you could do to make it so we don't have to repeat this code?

Tutorial 2.1i – Higher Or Lower Game



The next section provides a combined view of the all the code used for the game.

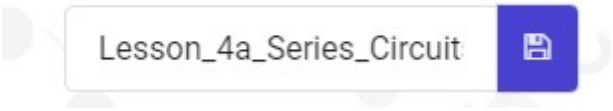
Tutorial 2.1i – Higher Or Lower Game



You are encouraged to make changes, improvise and customize the game using your own ideas.

6. Downloading Your Code To The micro:bit

Once you have completed the program, enter a name for your program using the option provided below i.e. in the text box adjacent to the small save button.



Now that you have given your program a name and saved it you can download it your micro:bit. But before we do that let's confirm what drive your micro:bit shows up as. On most machines the micro:bit will show up as an additional USB drive. So head over into windows explorer and confirm what drive name (D:, E:, F:. G:, etc.) the micro:bit shows up as. You need to absolutely be sure what drive the micro:bit shows up as. Once you've confirmed what drive the micro:bit shows up as on your machine you can select the right drive when downloading the code to the micro:bit. If in doubt please ask the volunteer/mentor/session facilitator helping out.

To download the newly written code to the micro:bit, hit the download button shown above. You should now see a dialog box open up and you will be asked to save the file somewhere on your machine. Please choose the drive your micro:bit shows up as i.e. D: or E: or F: or whatever it shows up as on your machine. A sign of success is when you see the lights on the rear (orange) lighting up in quick succession suggesting that the code is being written to the micro:bit. On completion the micro:bit reboots and you should now see the code in action on the micro:bit.

If hitting the download button shown above does not open up a dialog box asking you to save to the micro:bit please save the file (you will have a <filename.hex> file) to your desktop. Then open up windows explorer and drag that file onto the drive which is your micro:bit. A sign of success is when you see the lights on the rear (orange) lighting up in quick succession suggesting that the code is being written to the micro:bit. On completion the micro:bit reboots and you should now see the code in action on the micro:bit.

Please feel free to customize the code blocks, have a play. Add your own custom code and re-download the code to the micro:bit. Give yourself a tap on the back, you've just completed your first circuit!!!!!

7. Challenges

Well done for completing the tutorial. There's a lot of ground we have covered in this tutorial so please feel free to make notes, come back to the tutorial at some point down the line and ask your learning facilitator any questions or doubts you might have on the concepts covered this far.

Let us now stretch it a bit further -

- When A + B is pressed, reset the entire game and start the game over.
- An important part of coding is to be DRY. This means 'Don't repeat yourself' it is important to try to avoid repeating code as much as possible. For this, creating custom functions comes in handy as you can call them in multiple places without re writing the whole code out every time. The challenge here is to redo the code so you use custom functions for whenever there is repeated code.
- Make it so that after each correct guess, not only is the hidden number shown but also the new user score.