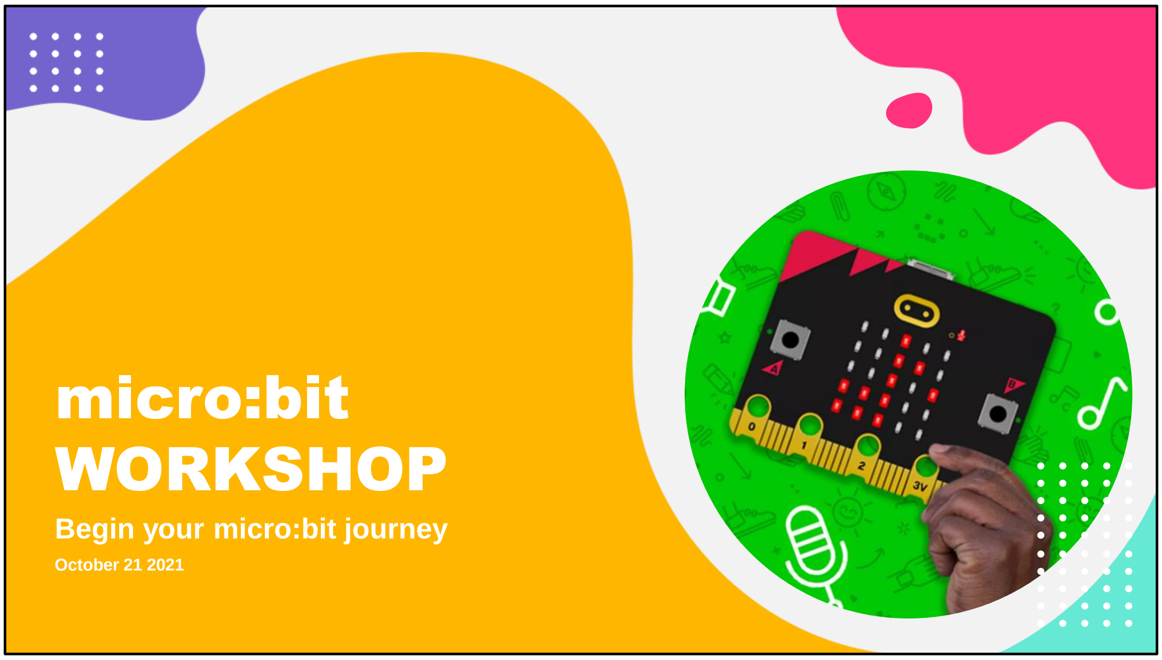


micro:bit WORKSHOP

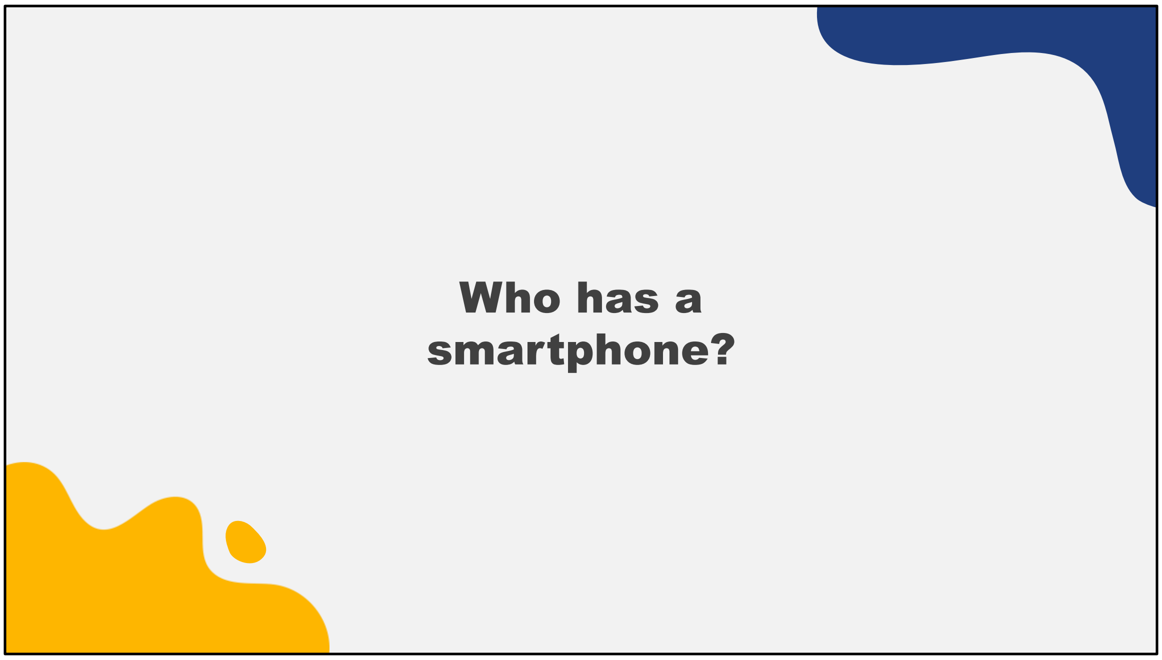
Begin your micro:bit journey

October 21 2021

Workshop Information:
Nowadays, everywhere you look are little computers that help us in our dai ly life. Ever wonder how smartphones or fitness trackers work under the ho od?
This workshop is great for peeking interest in STEM and teaching coding fu ndamentals, and we start with with drag-and-drop coding the micro:bit.
The micro:bit is a small, fully programmable computer that fits in your han ds. It has buttons, touch sensors, motion sensors and even a small screen!
During the workshop, we will build some fun and exciting apps and game s that run on the micro:bit exposing the participants to its capabilities, teac hing them programming skills, and inspiring them to take coding further.
The micro:bit is a great introduction to get a head start learning an increasi ngly important skillset.

Target audience: 9 -16 year
Participants: 8-12 (or double if they work in pairs)
Time: 2 to 3 hours (many more additional things can be made from the ma kecode website)

Material list:
For workshop giver:
- Beamer to show this presentation (notes are for presenter)
- Microbit board + usb cable
- Pc
For each participant (or per pair):
- Microbit board+ usb cable
- Pc



Who has a smartphone?

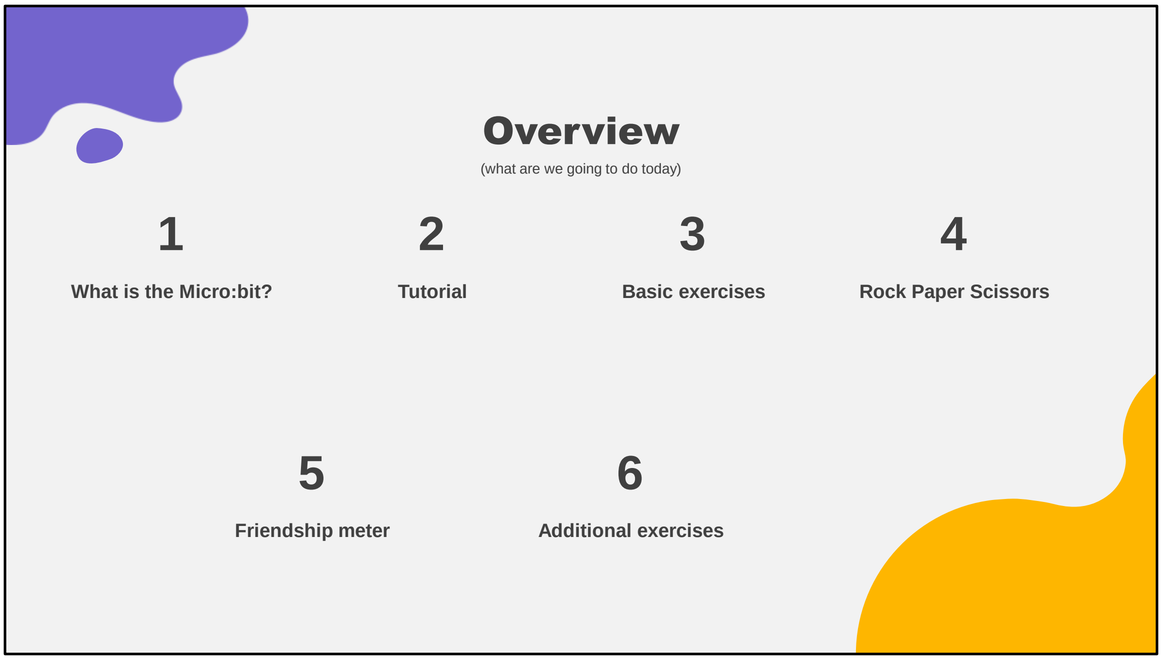
Try to start with AIDA, maybe along the lines of:
Attention:
Ask who has a smartphone? Almost everybody here has or wants a smartp
hone

Interest:
Showcase what a little wonder it is, talking to somebody at the other end o
f the world, having all books of a library in your hand, etc
Ask them if they ever had a problem with their smartphone. There will be a few who tell their issues they have or had.

Interest:
Wouldn't it be great to know a little bit how a smartphone works, so next t
ime it's easier to solve these problems?

Action:
Invite them to this workshop where they will learn the basics of coding.

Theory:
AIDA model is a traditional communication model. AIDA is an acronym for Attention, Interest, Desire, and Action. We are trying here to get them to ' buy-in' to what we have to say. So it makes sense that we look at incorpor ating or thinking about models such as AIDA to get our participants attenti on. Let's look at what AIDA is and how to use it with workshops.
Using the AIDA model will help you ensure that any kind of writing, whose purpose is to get the reader to do something, is as effective as possible. Fir st, it must grab the target audience's attention, and engage their interest. T hen it must build a desire for the product offering, before setting out how to take the action that the writer wants the audience to take.
Attention – The attention portion of the message occurs at the beginning and is designed to give users a reason to take notice. Presenting a shockin g fact or statistic that identifies a problem which can be solved by the prod uct or service is one common method of gaining attention. Other methods can include asking a thought-provoking question or using the element of s urprise. The purpose is to give the users a reason for wanting to learn mor e.
Interest – Once you've gained their attention, the next step is to maintain i nterest and to keep the recipients engaged. Explain the problem you've ide ntified in the attention step is adversely affecting how they work. A demon stration or illustration can help the users to further identify with the proble m and want to actively seek possible solutions. By personalizing the proble m, you're making it hit closer to home
Desire – In the desire stage, your objective is to show the users how to sol ve their problem. For a compliance course demonstrate the procedure or process to follow to ensure the meet the organizations standards or if it is product or service explain the benefits and demonstrate how the benefits f ulfill the need.
Action – Now that you've created the desire about the compliance issue/pr oduct/service, the final step is to persuade the users to take immediate acti on.



Overview

(what are we going to do today)

1 What is the Micro:bit?

2 Tutorial

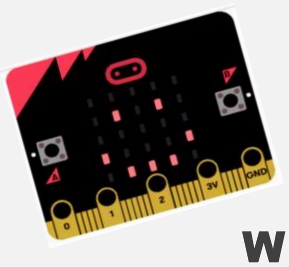
3 Basic exercises

4 Rock Paper Scissors

5 Friendship meter

6 Additional exercises

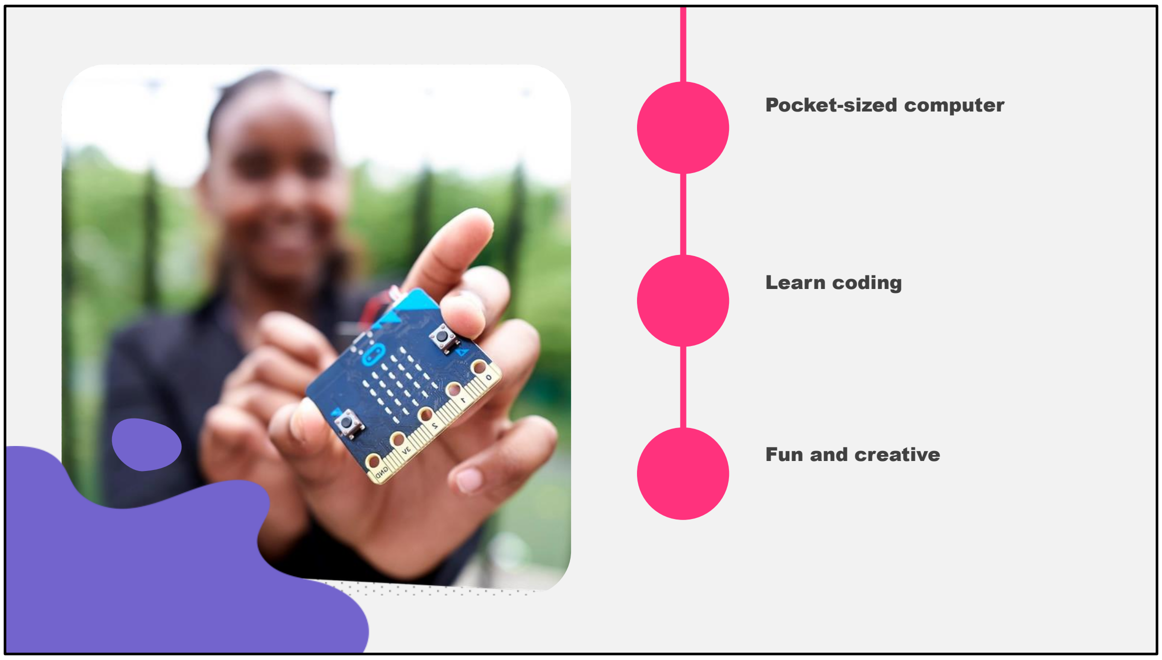
Explain in brief the different stages of the workshop.



What is the Microbit?

The BBC micro:bit is a pocket-sized computer that introduces you to how s
oftware and hardware work together.
It has an LED light display, buttons, sensors and many input/output feature
s that, when programmed, let it interact with you and your world.

How computers work
The micro:bit helps you understand how computers work. When you type
on your laptop or touch the screen on your phone, you're using an **input** d
evice. Inputs allow computers to sense things happening in the real world,
so they can act on this and make something happen, usually on an **outpu
t** like a screen or headphones.
In between the input and the output, there is the **processor**. This takes info
rmation from inputs like buttons, and makes something happen on outputs
, like playing a song in your headphones.



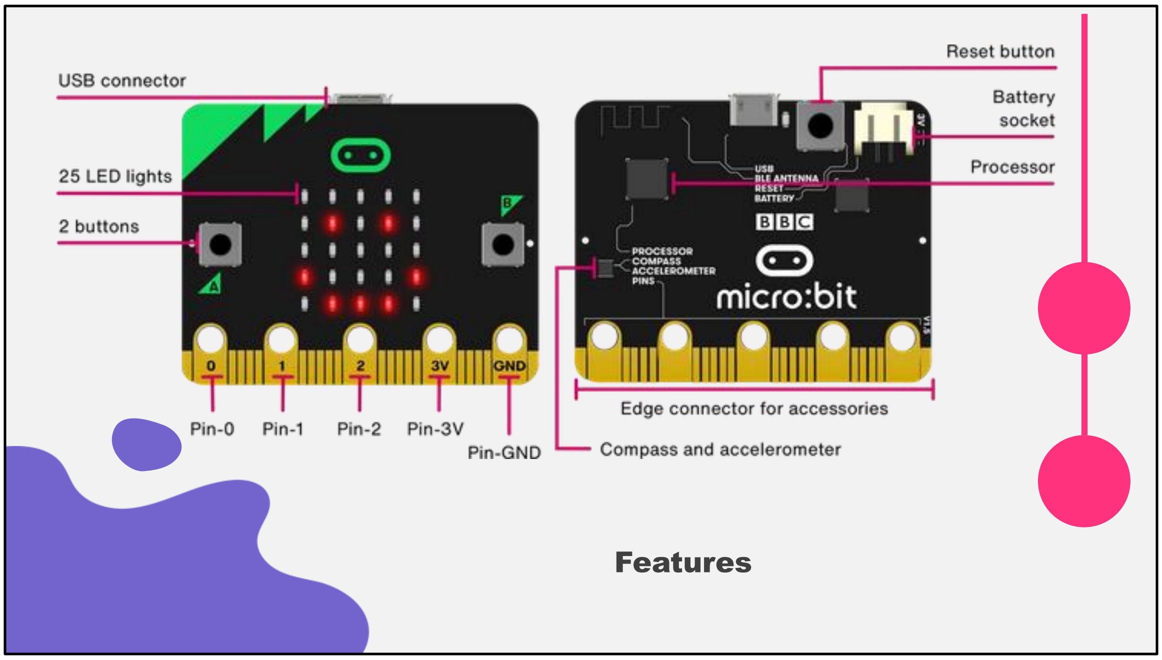
Pocket-sized computer

Learn coding

Fun and creative

The BBC micro:bit is a pocket-sized computer that introduces you to how software and hardware work together.
It has an LED light display, buttons, sensors and many input/output features that, when programmed, let it interact with you and your world.

How computers work
The micro:bit helps you understand how computers work. When you type on your laptop or touch the screen on your phone, you're using an **input** device. Inputs allow computers to sense things happening in the real world, so they can act on this and make something happen, usually on an **output** like a screen or headphones.
In between the input and the output, there is the **processor**. This takes information from inputs like buttons, and makes something happen on outputs, like playing a song in your headphones.

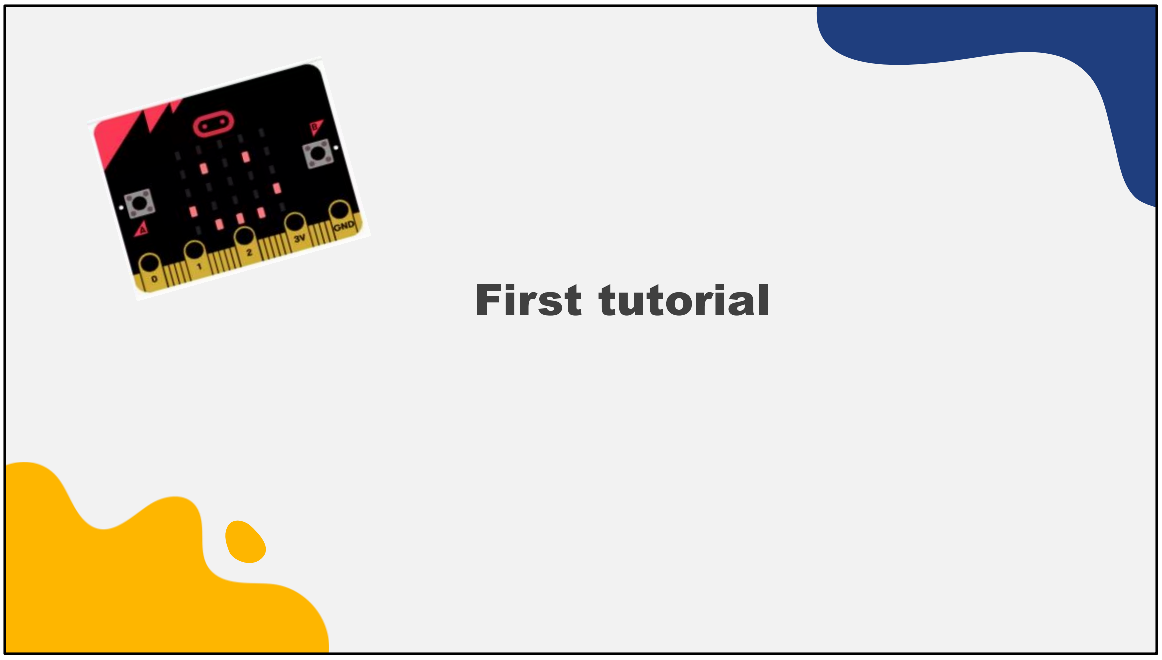


Features

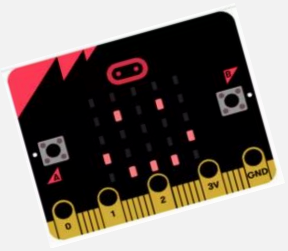
- The interface chip handles the **USB connection**, and is used for flashing new code to the micro:bit, sending and receiving serial data back and for th to your computer.
- 25 **LEDs** arranged in a 5x5 grid make up the display for showing pictures , words and numbers. They can also act as sensors, measuring how muc h light is falling on your micro:bit.
- The micro:bit has two **buttons** on the front that can be used separately or together to make things happen.
- The GPIO **pins** allow you to connect headphones, sense touch and add o ther electronics to expand the possibilities of your micro:bit.
- You can power external LEDs and other electronics using the 3 volt **pow er pin**.
- The **GND pin** is the ground or Earth pin - it's used to complete electrical circuits when you connect headphones, LEDs or external switches to your micro:bit.
- Restart your micro:bit programs with the **reset button**.
- Instead of powering your micro:bit from the USB socket, you can unplug it from your computer and use a **battery** pack instead. This is really usef ul if you want to take your micro:bit outside, wear it or play games with i

t. It can run for ages off two AAA batteries.

- The micro:bit's **processor** is its brain, fetching, decoding and carrying out your instructions. It also contains a temperature sensor so you can meas ure how warm your environment is.
- Find magnetic North or measure the strength of magnetic fields using th e micro:bit's **compass**.
- **Accelerometer** measures gestures and forces in 3 dimensions.



First tutorial



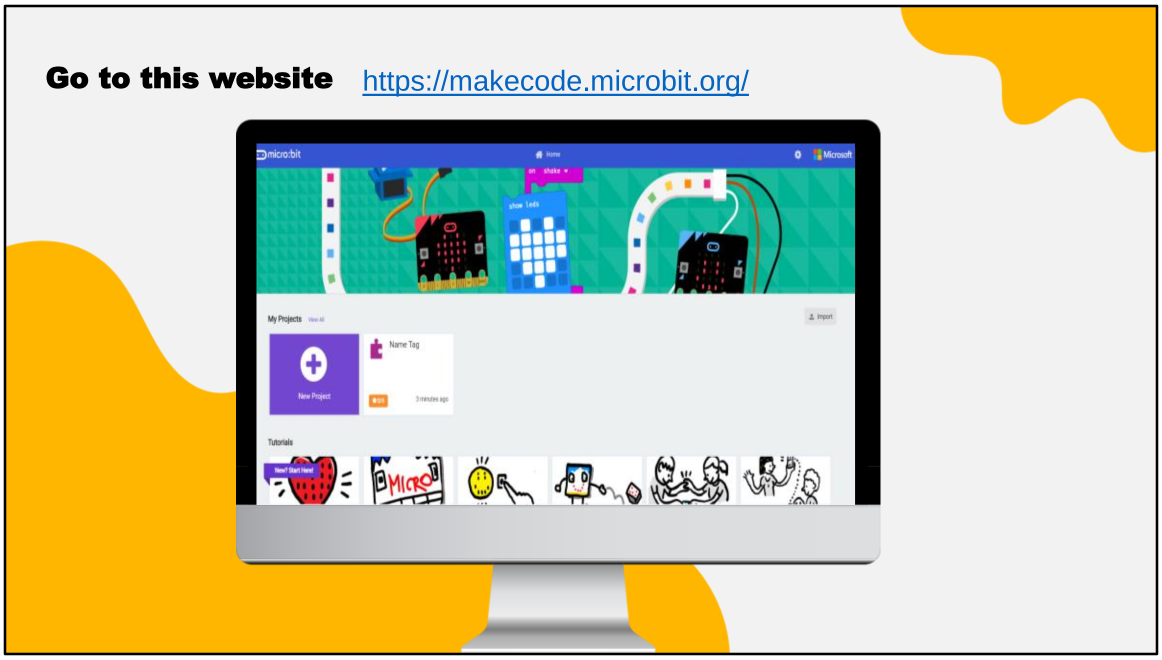
Let’s first do these steps together: program/connect/transfer
General Info:
Program
Tell your micro:bit what to do by giving it instructions. Sets of instructions f
or computers are called **programs** and are written in code.
You can program your micro:bit in the online **MakeCode block** or **Python t
ext** editors. To access an editor you need either:
A **computer** with a web browser and internet access or
A **mobile device** with the free micro:bit app for Android phones and tablet
s or iOS (iPhone and iPad) –
Then you'll need to transfer your program onto the micro:bit to make it wo
rk.

Connect
To make your micro:bit work you need to copy your code from the editor t
o the device.
Connect your micro:bit to your computer or mobile device. You will need:
Computer - use a **USB cable** to connect to your micro:bit
Mobile apps - use **Bluetooth** to connect your micro:bit to your phone or ta

blet

Transfer from a computer

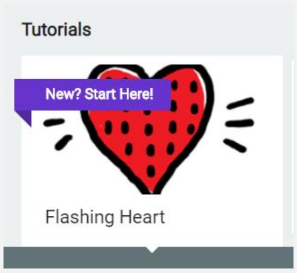
Transferring your program to your micro:bit is called **flashing** because it co
pies your program into the micro:bit's flash memory.
Your micro:bit will pause and the yellow LED on the back will blink while yo
ur program is being transferred. Once it's copied across, your program start
s running on your micro:bit.
There are two ways to transfer your program:
Drag and drop works like copying a downloaded file from your computer t
o a USB memory stick. It works on any computer.
Direct flashing sends your program directly from the code editor to your
micro:bit. It works on two popular web browsers.

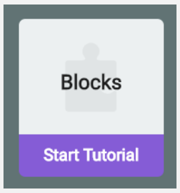


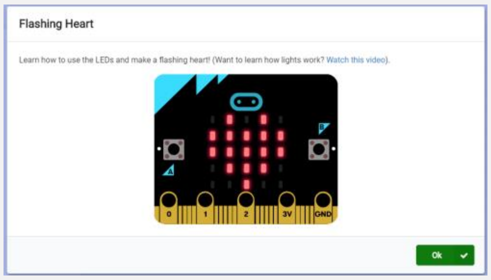
Go to this website https://makecode.microbit.org/

Program
Go to https://makecode.microbit.org/ with Chrome browser and you should see this screen
We will make a very small program first, that will display a heart on your m icrobit!

https://makecode.microbit.org/



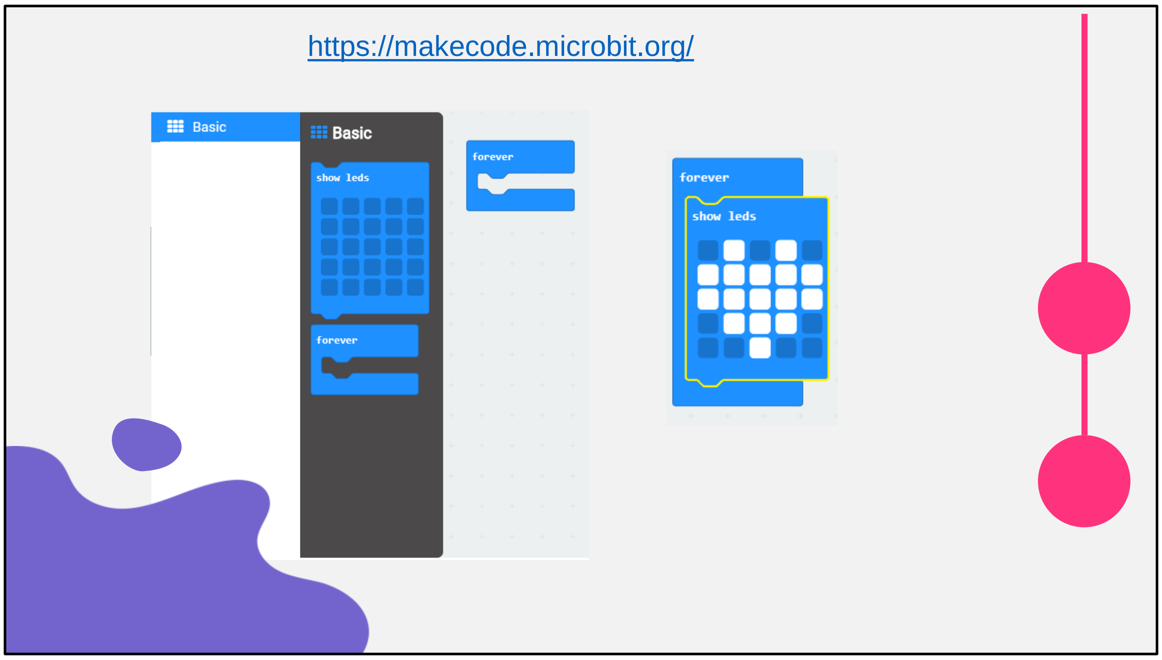




Learn how to use the LEDs and make a flashing heart! (Want to learn how lights work? Watch this video).

Under tutorials, click on the *Flashing Heart*, next choose '*Blocks*'
You will see the flashing heart we will make now. Click *Ok*.

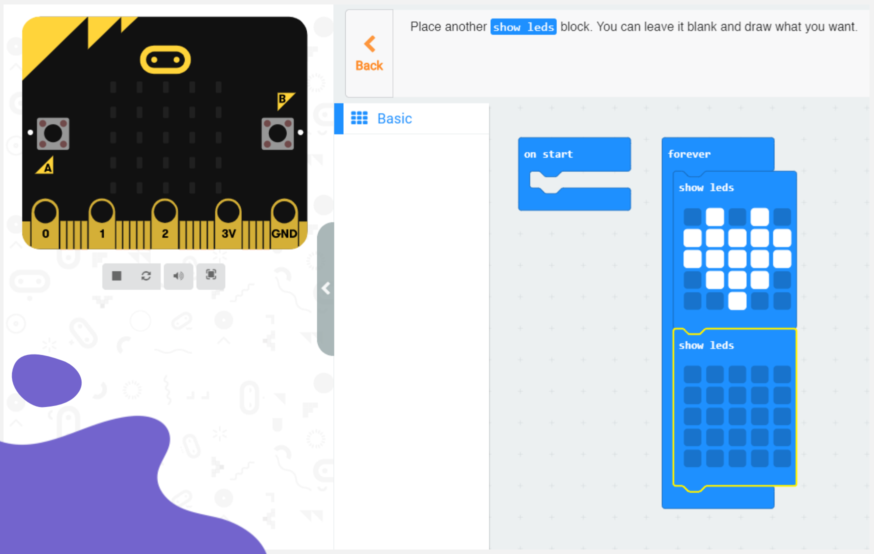
https://makecode.microbit.org/



Follow the on screen instructions:

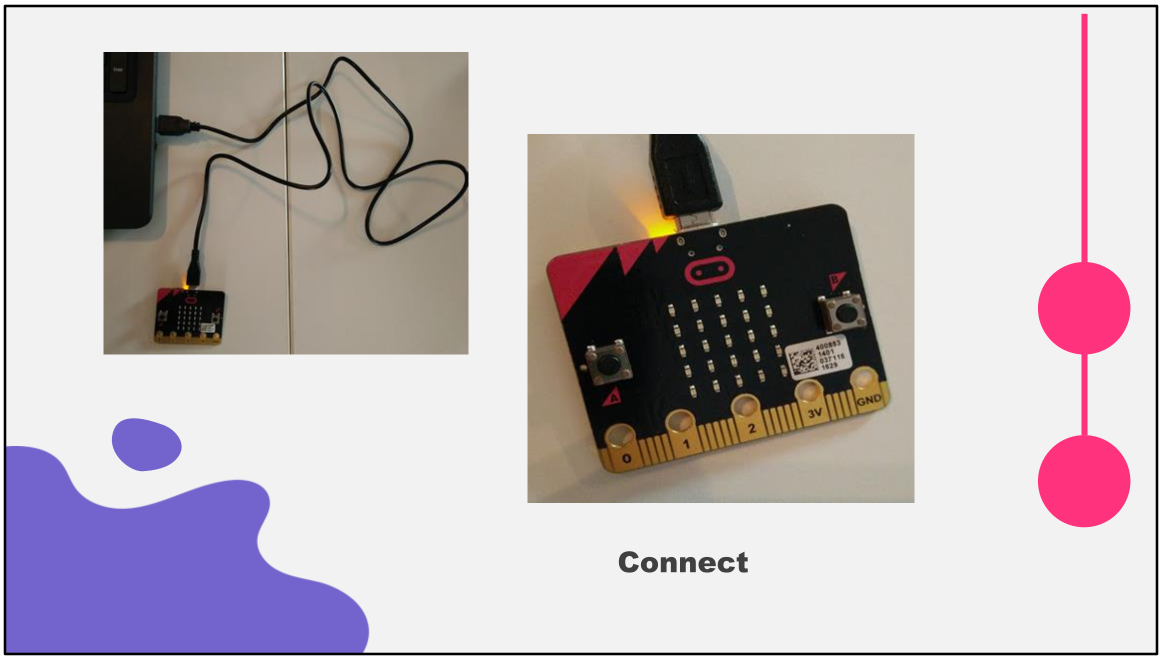
Place the '*show leds*' block in the 'forever' block and draw a heart.
Click Next.

https://makecode.microbit.org/


Place another show leds block. You can leave it blank and draw what you want.

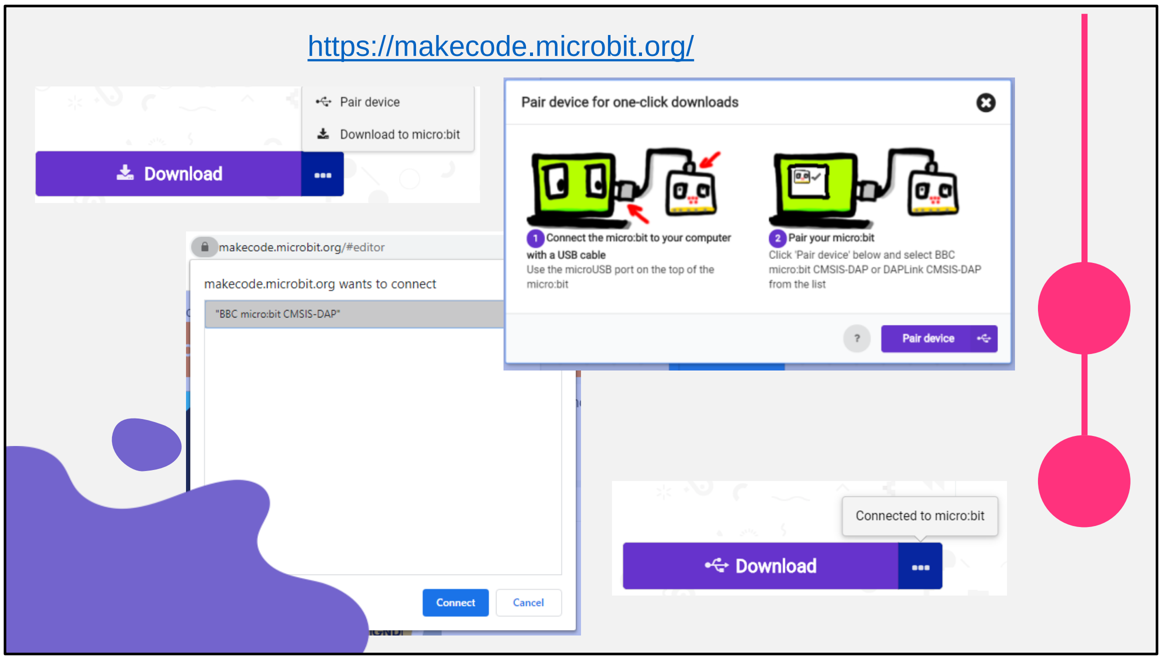
Follow the on screen instructions:

Place another '*show leds*' block that you leave blank.
In the upper left corner, you can see a simulation of what your microbit will do.
Congratulations, you coded your first program!
We want to transfer this program to your physical microbit, but first we ne ed to connect it.



Connect

Connect
To make your micro:bit work you need to connect a micro **USB cable** to yo ur micro:bit
This will supply power to function and will make it possible to transfer your program to the board.
When you have connected it, the yellow light (led) should be on (on the ba ckside)


https://makecode.microbit.org/

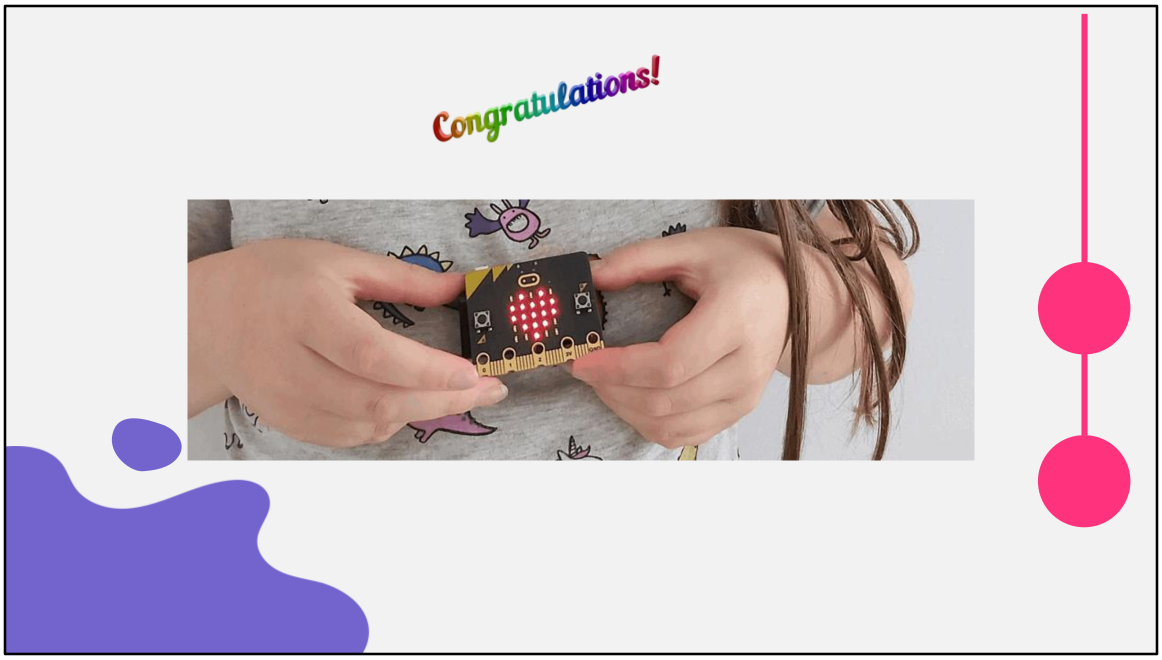
Pair device for one-click downloads

1 Connect the micro:bit to your computer with a USB cable
Use the microUSB port on the top of the micro:bit

2 Pair your micro:bit
Click 'Pair device' below and select BBC micro:bit CMSIS-DAP or DAPLink CMSIS-DAP from the list

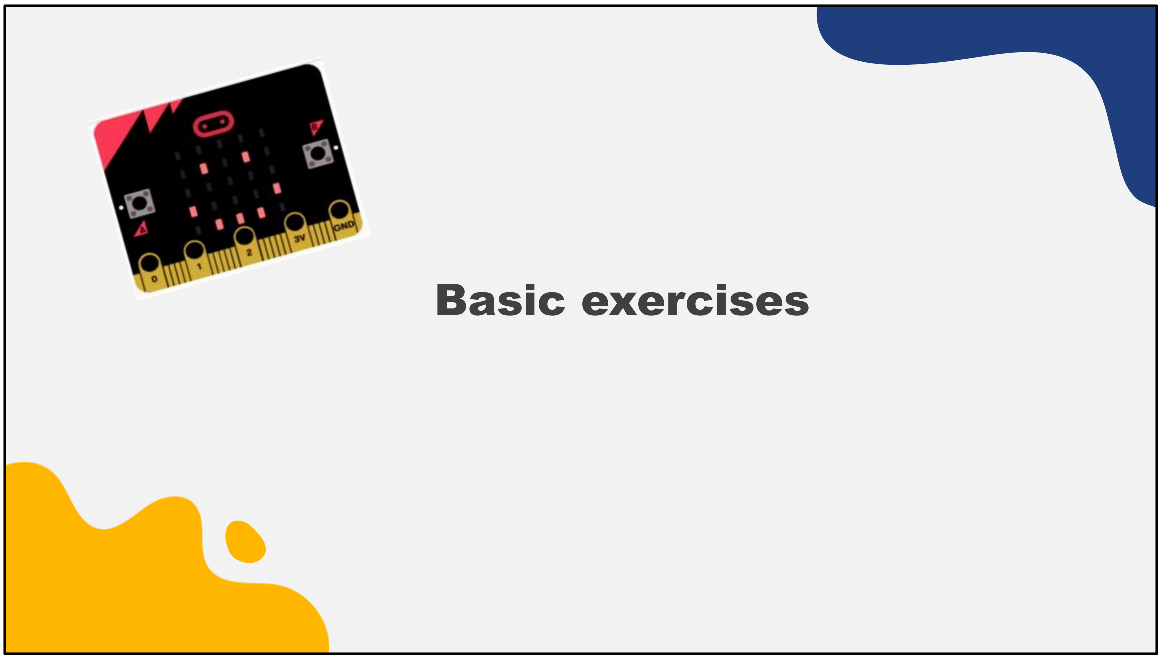
makecode.microbit.org/#editor
makecode.microbit.org wants to connect
"BBC micro:bit CMSIS-DAP"

Click on the … next to Download, choose '*Pair Device*'
Choose '*Pair Device*' in the next screen.
Select BBC micro:bit CMSIS-DAP from the list and click Connect.
When you hover above the …, it should say 'Connected to micro:bit.
Click '*Download*' and …



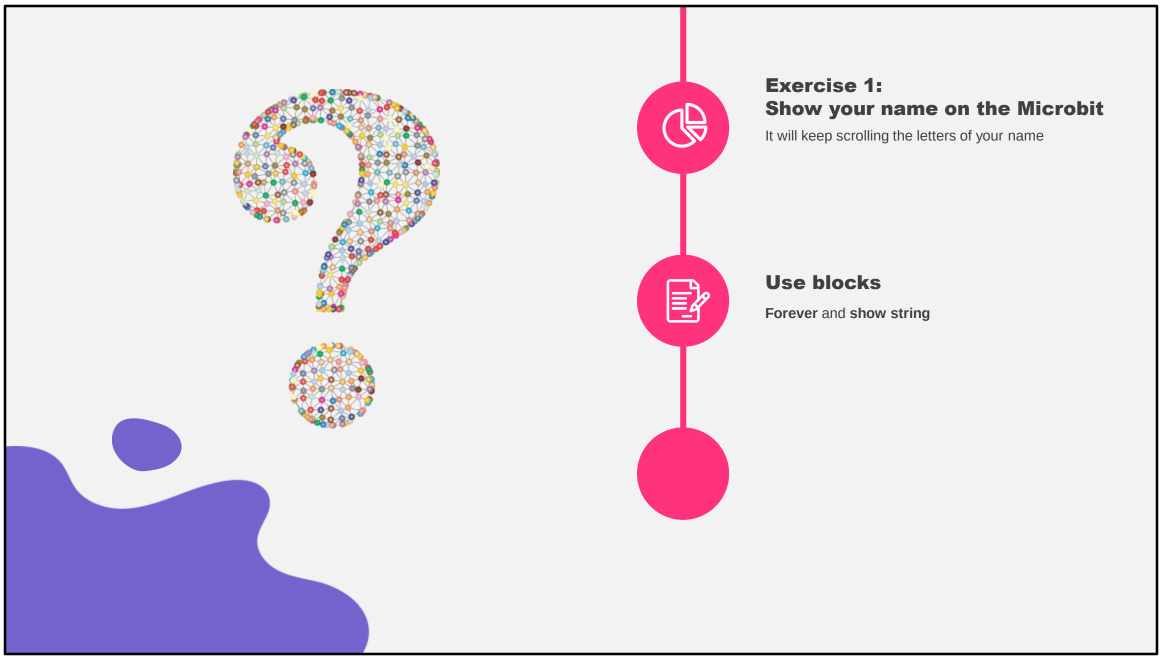
Click '*Download*' and ...
Congratulations , you programmed your microbit with your own program!.

If you think the participants can without much 'handholding' you can let th
em the next basic exercises.
If they need more step by step help, you could let them follow more tutori
als on the makecode website first.



Basic exercises

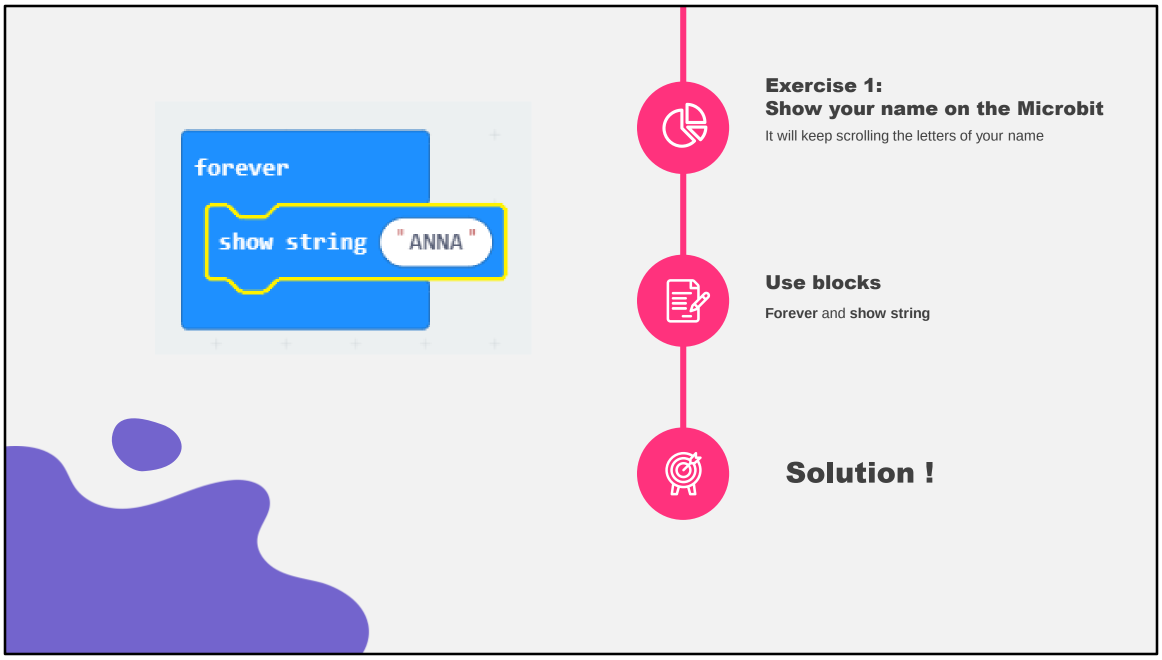
Now that you know the basics, let's try some other programs.
Some exercises to get to know the different code blocks
Learn how to program using 'if –then', conditions and loops
We start simple and gradually increase complexity



Exercise 1:
Show your name on the Microbit
It will keep scrolling the letters of your name

Use blocks
Forever and **show string**

In Makecode:
Place the "*show string*" block in the "*forever*" block to repeat it. Change the text to your name.



Exercise 1:
Show your name on the Microbit

It will keep scrolling the letters of your name

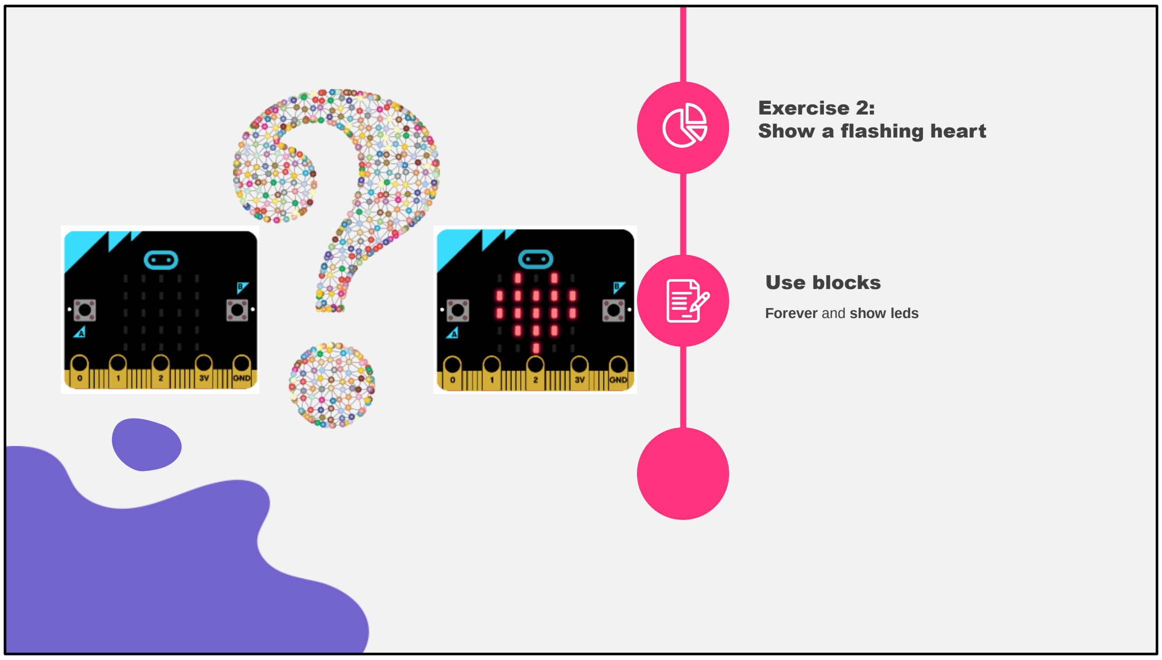
Use blocks

Forever and **show string**

Solution !

```
forever
    show string "ANNA"
```

In Makecode:
Place the "show string" block in the "forever" block to repeat it. Change the text to your name.

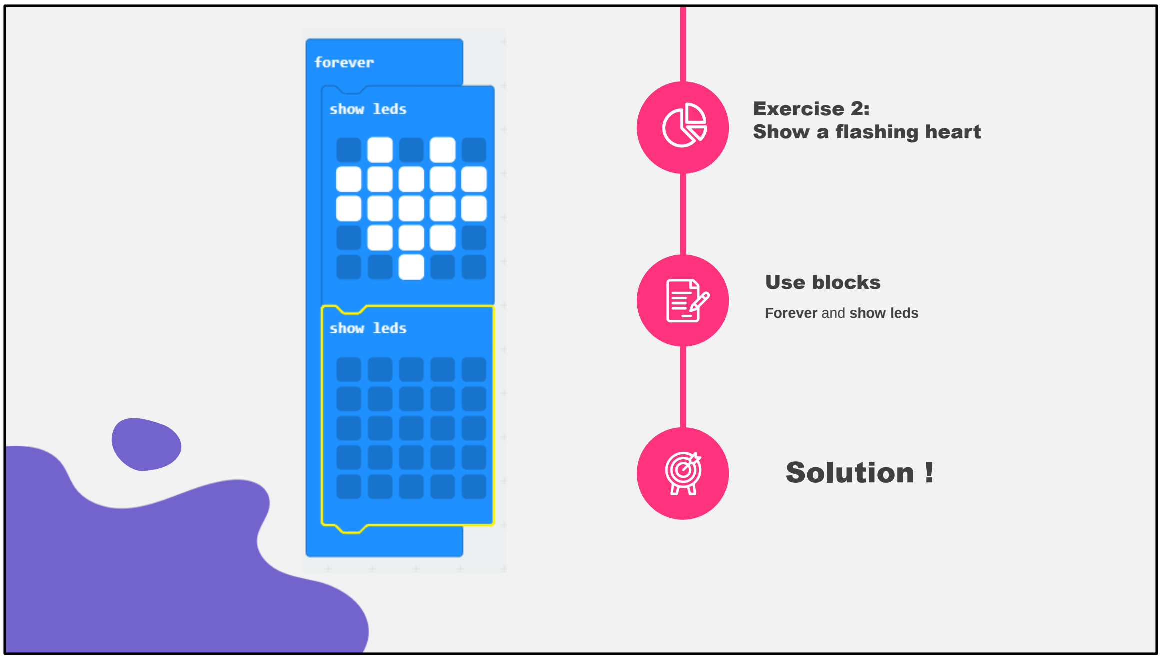


Exercise 2:
Show a flashing heart

Use blocks

Forever and **show leds**

In Makecode:
Place the "*show leds*" block in the "*forever*" block and draw a heart
Place another "*show leds*" block in the "*forever*" and leave blank



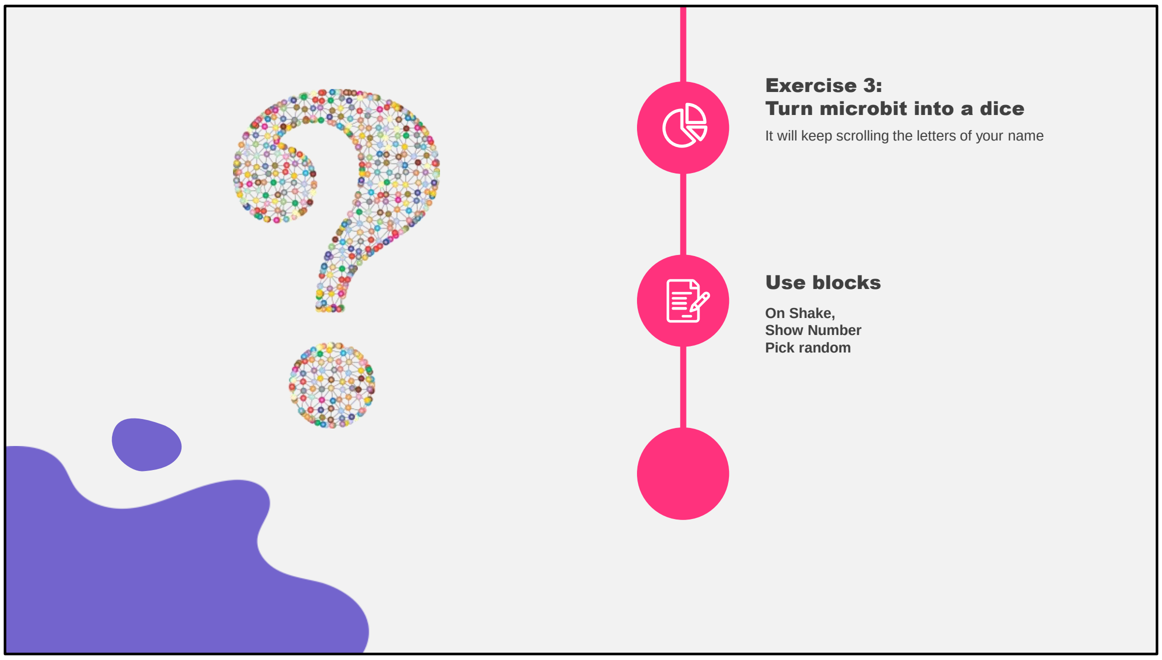
Exercise 2:
Show a flashing heart

Use blocks

Forever and **show leds**

Solution !

In Makecode:
Place the "*show leds*" block in the "*forever*" block and draw a heart
Place another "*show leds*" block in the "*forever*" and leave blank



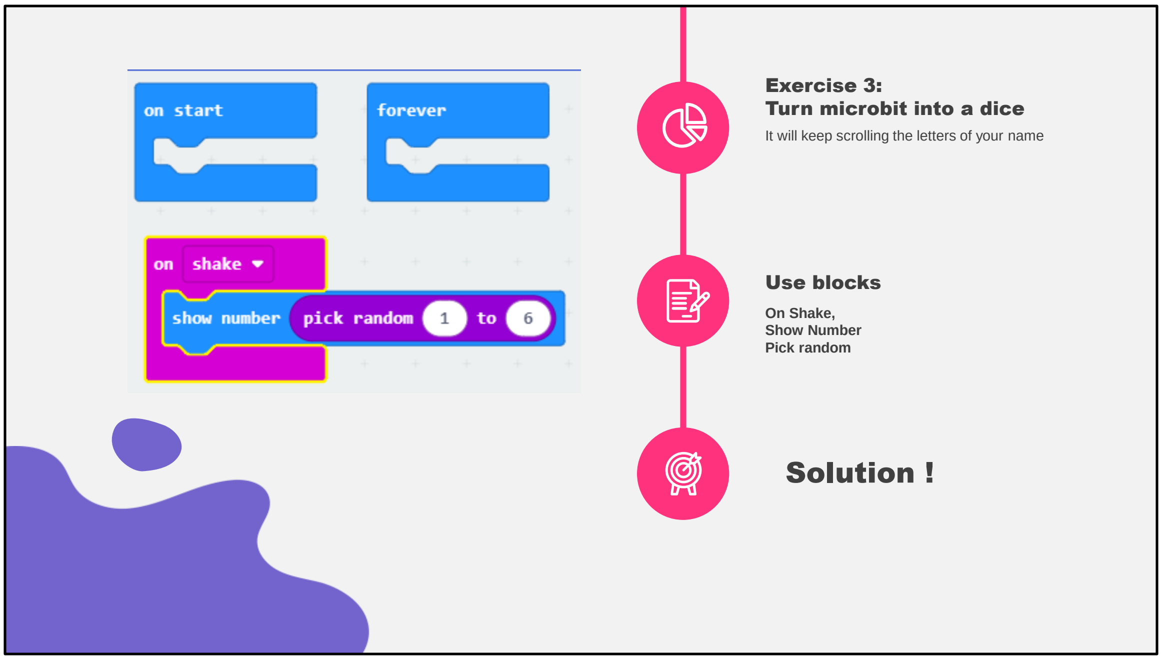
Exercise 3:
Turn microbit into a dice

It will keep scrolling the letters of your name

Use blocks

On Shake,
Show Number
Pick random

In Makecode:
Let them try using those blocks.



```
on start

forever

on shake ▾
    show number pick random 1 to 6
```

Exercise 3:
Turn microbit into a dice
It will keep scrolling the letters of your name

Use blocks
On Shake,
Show Number
Pick random

Solution !

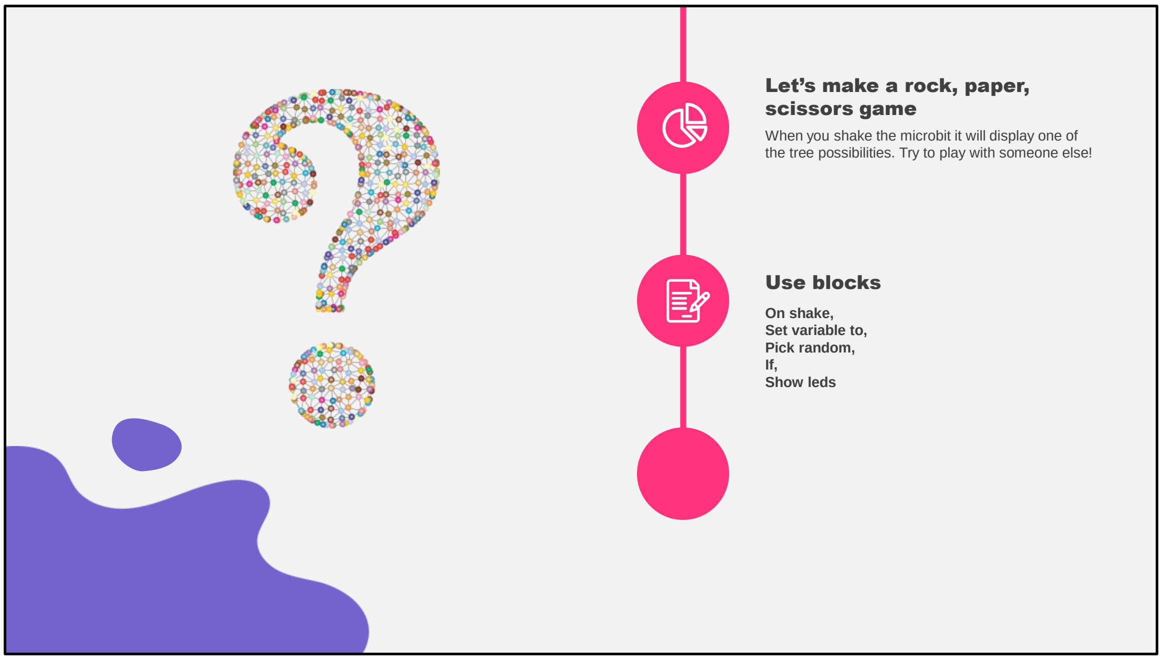
In Makecode:
Place the blocks as shown. Click download when finished to run on the mic robit.

Rock Paper Scissors

We continue with a more advanced example where we use conditionals ad if-then-else clauses

Add **On Shake** block
Add a variable called '**hand**' and place the '**set hand to**' block in the shake event
Add '**pick random**' block (1 to 3) and store it in variable hand
Place an '**if**' block and check whether hand = 1. Add a '**show leds**' block th en to show paper
Do the same for the rock (=2) and scissors (=else).

Alternatively, let them follow the same titled tutorial on the makecode web site.



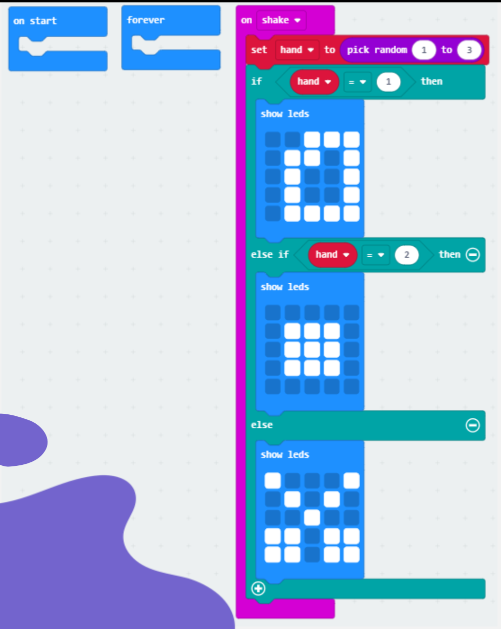
Let's make a rock, paper, scissors game

When you shake the microbit it will display one of the tree possibilities. Try to play with someone else!

Use blocks

On shake,
Set variable to,
Pick random,
If,
Show leds

Add **On Shake** block
Add a variable called '**hand**' and place the '**set hand to**' block in the shake event
Add '**pick random**' block (1 to 3) and store it in variable hand
Place an '**if**' block and check whether hand = 1. Add a '**show leds**' block then to show paper
Do the same for the rock (=2) and scissors (=else).



```
on start

forever

on shake
    set hand to pick random 1 to 3
    if hand = 1 then
        show leds
            . . # # #
            . # # . #
            . # . . #
            . # . . #
            . # # # #
    else if hand = 2 then
        show leds
            . . . . .
            . # # # .
            . # # # .
            . # # # .
            . . . . .
    else
        show leds
            # . . . #
            . # . # .
            . . # . .
            # # . # #
            # # . # #
```

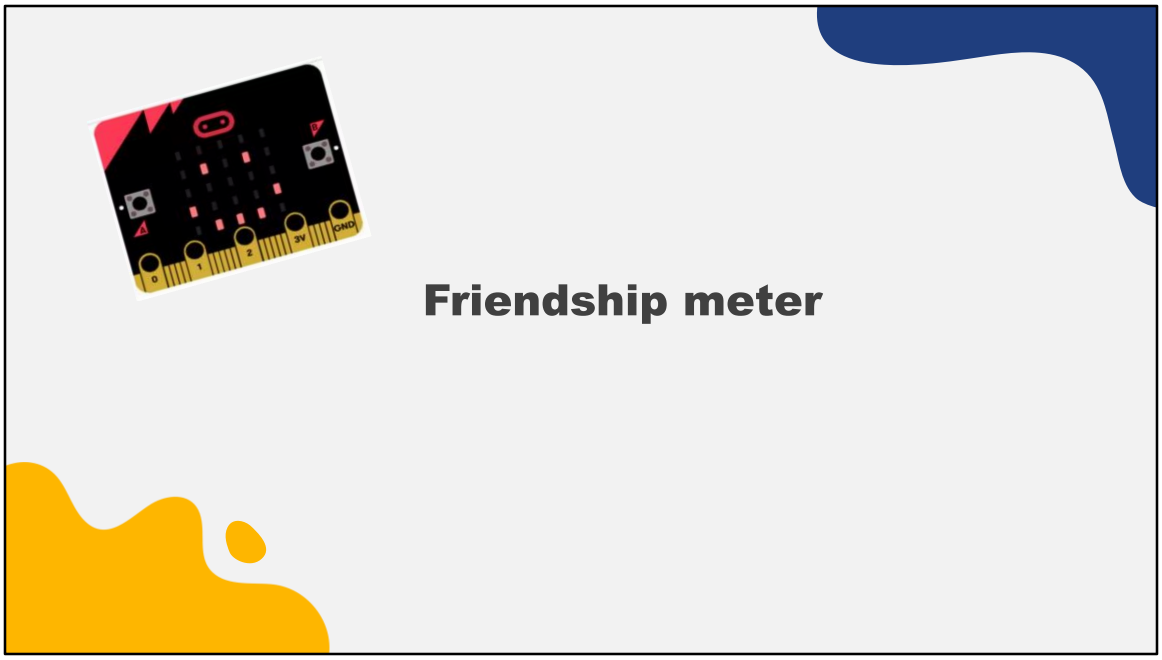
Let's make a rock, paper, scissors game

When you shake the microbit it will display one of the tree possibilities. Try to play with someone else!

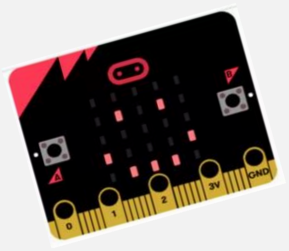
Use blocks

On shake,
Set variable to,
Pick random,
If,
Show leds

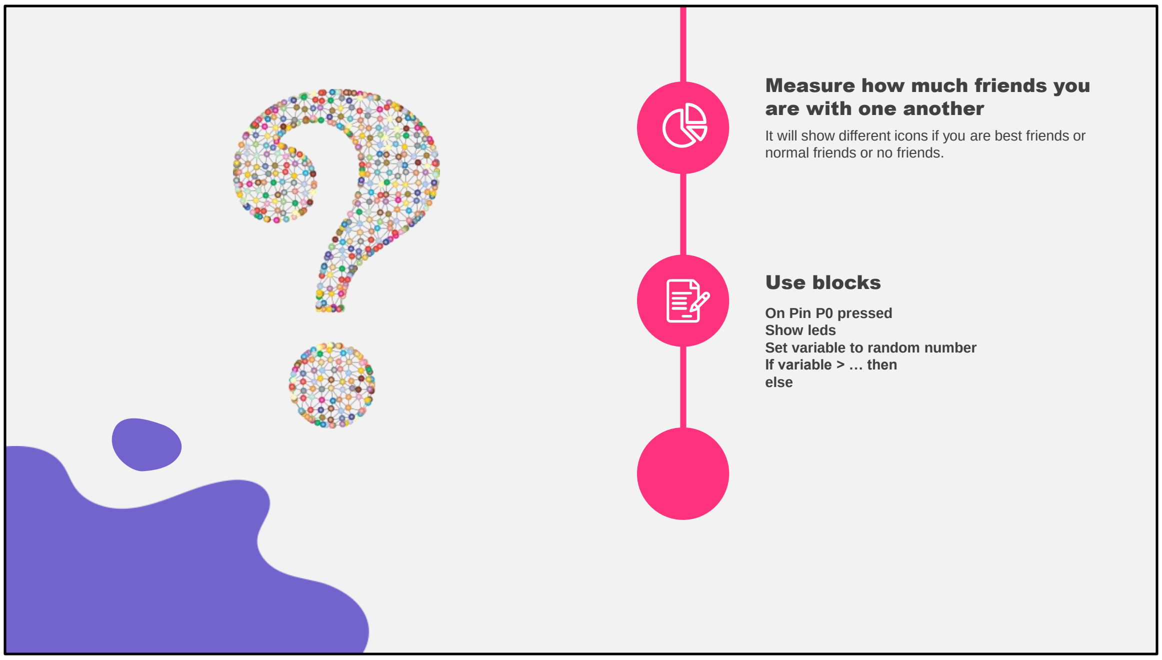
Solution !



Friendship meter



Now we will make a friendship meter. The point is that we will measure ho
w much you are friends with another person!
For this, both persons need to hold hands, while one holds the GND pin a
nd the other person holds Pin 0



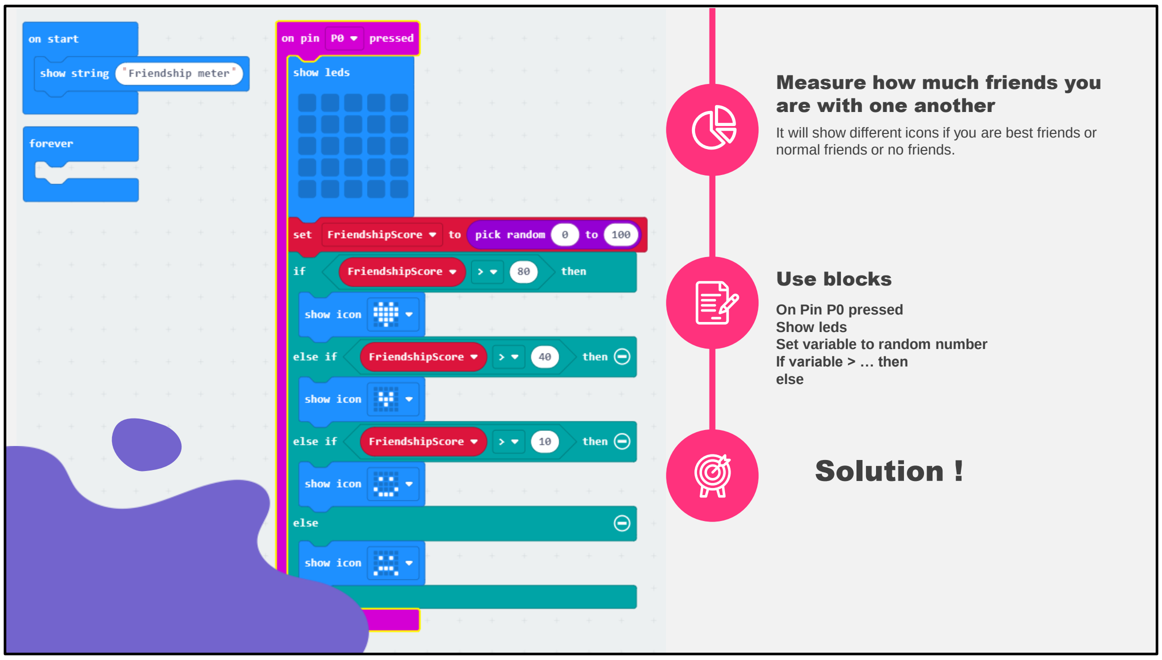
Measure how much friends you are with one another

It will show different icons if you are best friends or normal friends or no friends.

Use blocks

On Pin P0 pressed
Show leds
Set variable to random number
If variable > … then
else

Explanation:
When one person holds the GND pin and the other the Pin 0 and they join hands:
we have a trigger: when Pin P0 is pressed.
At that moment the screen is temporarily blanked and a random number between 0 and 100 is generated. We store that random number in a variable that we create
If that number is bigger than 80, both participants are best friends.
If that number is bigger than 40, both participants are good friends.
If that number is bigger than 10, both participants are friendly but not good friends.
Else, both participants don't like each other.
For each of these options the icon is shown.



```
on start
    show string "Friendship meter"

forever

on pin P0 pressed
    show leds
    set FriendshipScore to pick random 0 to 100
    if FriendshipScore > 80 then
        show icon
    else if FriendshipScore > 40 then
        show icon
    else if FriendshipScore > 10 then
        show icon
    else
        show icon
```

Measure how much friends you are with one another

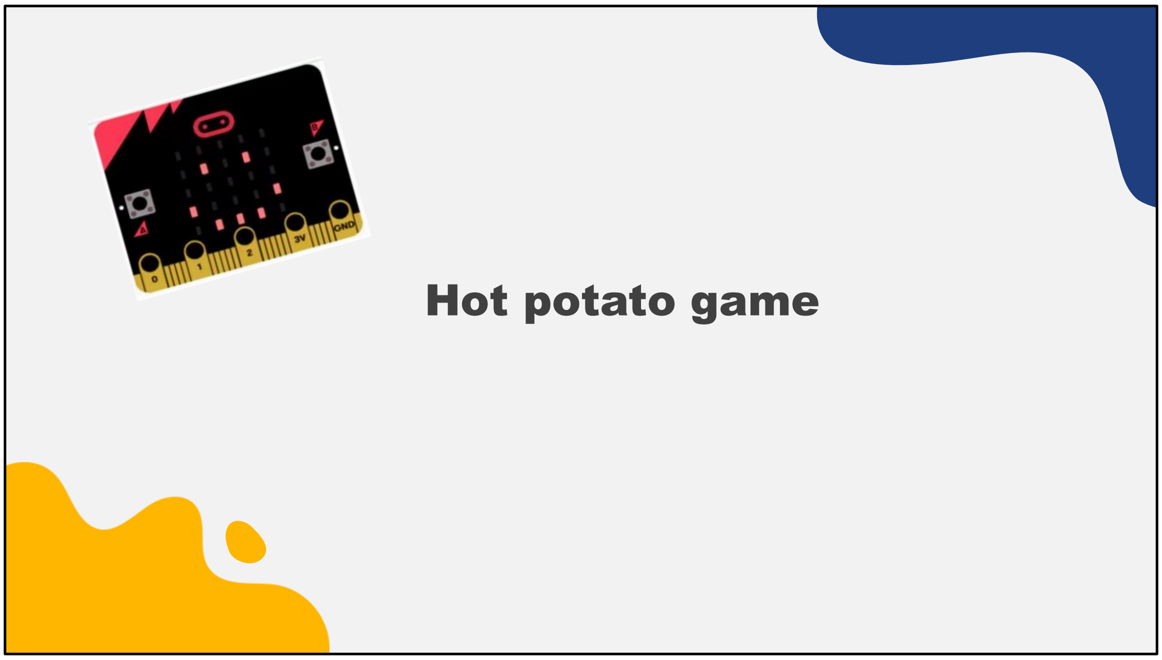
It will show different icons if you are best friends or normal friends or no friends.

Use blocks

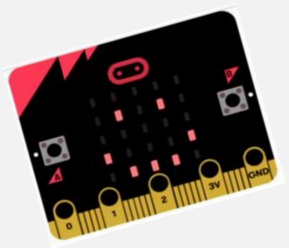
On Pin P0 pressed
Show leds
Set variable to random number
If variable > … then
else

Solution !

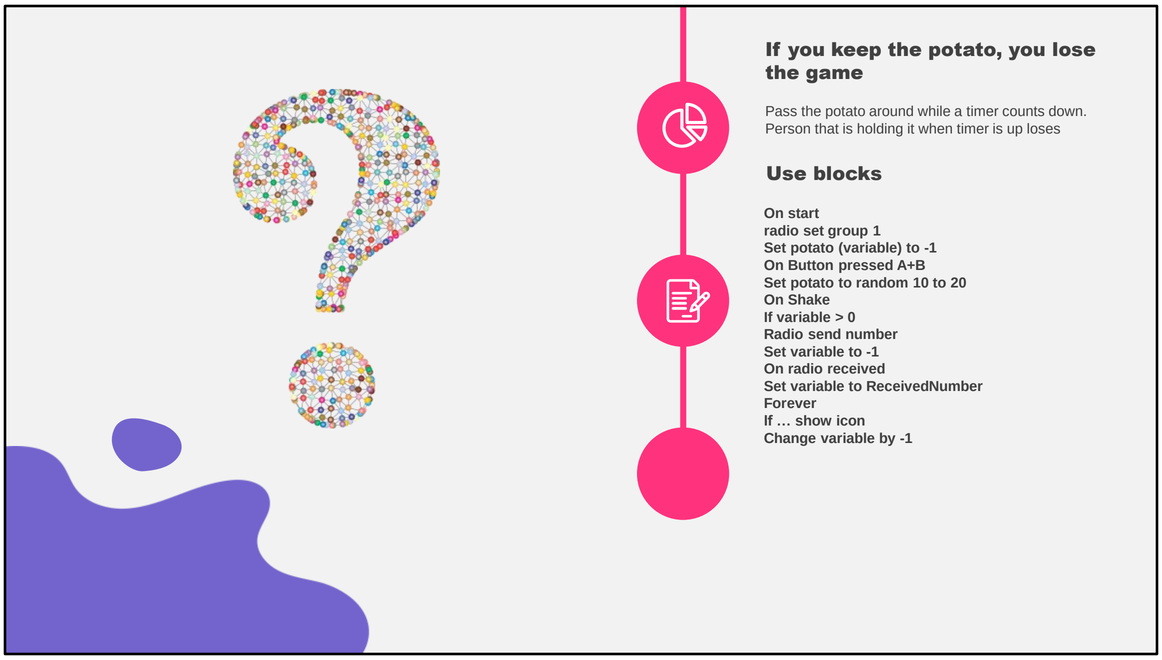
Explanation:
When one person holds the GND pin and the other the Pin 0 and they join hands:
we have a trigger: when Pin P0 is pressed.
At that moment the screen is temporarily blanked and a random number between 0 and 100 is generated. We store that random number in a variable that we create
If that number is bigger than 80, both participants are best friends.
If that number is bigger than 40, both participants are good friends.
If that number is bigger than 10, both participants are friendly but not good friends.
Else, both participants don’t like each other.
For each of these options the icon is shown.



Hot potato game



Now we will make a game where we pass around a virtual potato. There is a timer counting down and if you have the potato when timer runs out yo u lose!
We will be using the radio function to communicate between microbit boar ds.
we need to **model** the clock as **a number** being tossed around with the po tato
To keep track of things, let's have a variable called **potato**:
If the value of **potato** is positive, the player has the potato and the **potat o** variable represents the **remaining time**
if value of **potato** reaches 0, the game is over
if the value of **potato** is negative, this means that the player doesn't have t he potato in their hand
This is how will the user play the game:
press the A+B button to start the game and send the first potato
when a potato is received, the screen displays some image
when the player shakes the microbit, they send the potato to other players



If you keep the potato, you lose the game

Pass the potato around while a timer counts down.
Person that is holding it when timer is up loses

Use blocks

On start
radio set group 1
Set potato (variable) to -1
On Button pressed A+B
Set potato to random 10 to 20
On Shake
If variable > 0
Radio send number
Set variable to -1
On radio received
Set variable to ReceivedNumber
Forever
If … show icon
Change variable by -1

Explanation:
Initialization
Let's start by creating the potato variable and initializing it to -1 in on start. Remember, a negative potato value means you **don't** have the potato. We use radio set group to make sure players receive the messages.

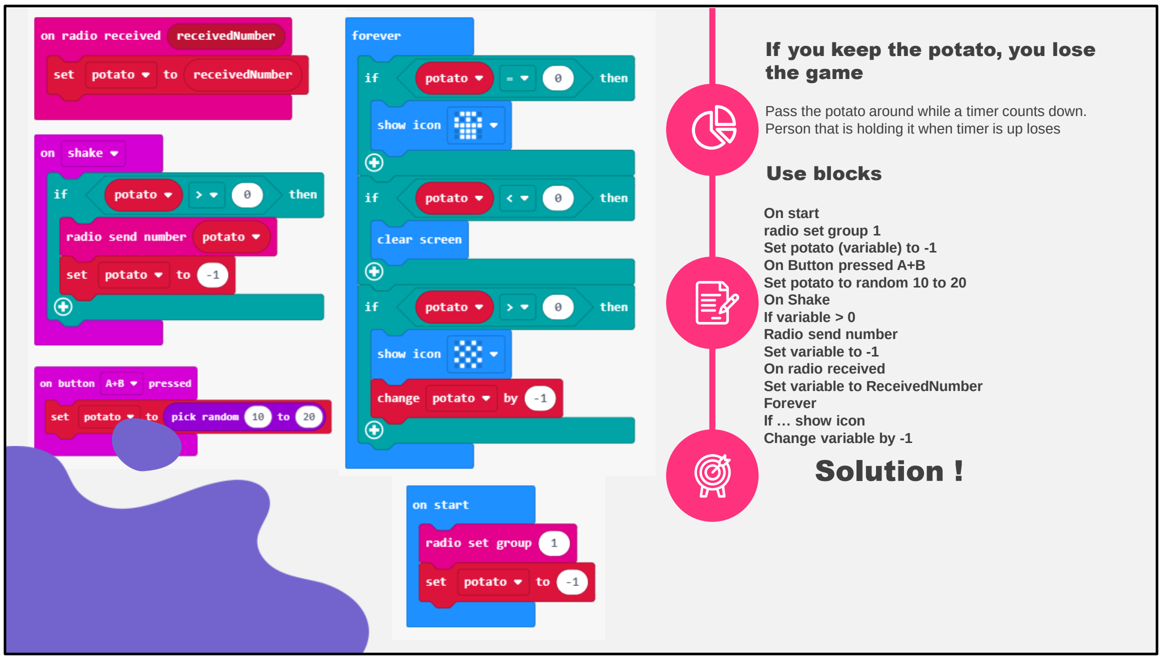
Starting the game
To start the game, we respond to the A+B button press and assign a positi ve number to the **potato** variable. To make the game less predictable, we u se the pick random block to generate a value between 10 and 20.

Sending the potato
Sending the potato is done by shaking the micro:bit. If the **potato** variable is positive, we have the potato and we can send it. After sending it, we set the **potato** variable to -1 since we don't have it anymore.

Receiving the potato
Receiving the potato is done in the on received number block. The **receive dNumber** represents the potato and is stored in the **potato** variable.

Ticking the clock

Making the clock tick down is done with a forever loop.
If the **potato** is equal to 0 (potato == 0), KABOOM! you lose!
If the **potato** variable is negative (potato < 0), we don’t have the potato so we clear the screen.
If the **potato** variable is positive (potato > 0), we display a potato image an d decrease the variable by 1.



```
on radio received receivedNumber
  set potato to receivedNumber

on shake
  if potato > 0 then
    radio send number potato
    set potato to -1

on button A+B pressed
  set potato to pick random 10 to 20

forever
  if potato = 0 then
    show icon
  if potato < 0 then
    clear screen
  if potato > 0 then
    show icon
    change potato by -1

on start
  radio set group 1
  set potato to -1
```

If you keep the potato, you lose the game

Pass the potato around while a timer counts down.
Person that is holding it when timer is up loses

Use blocks

On start
radio set group 1
Set potato (variable) to -1
On Button pressed A+B
Set potato to random 10 to 20
On Shake
If variable > 0
Radio send number
Set variable to -1
On radio received
Set variable to ReceivedNumber
Forever
If … show icon
Change variable by -1

Solution !

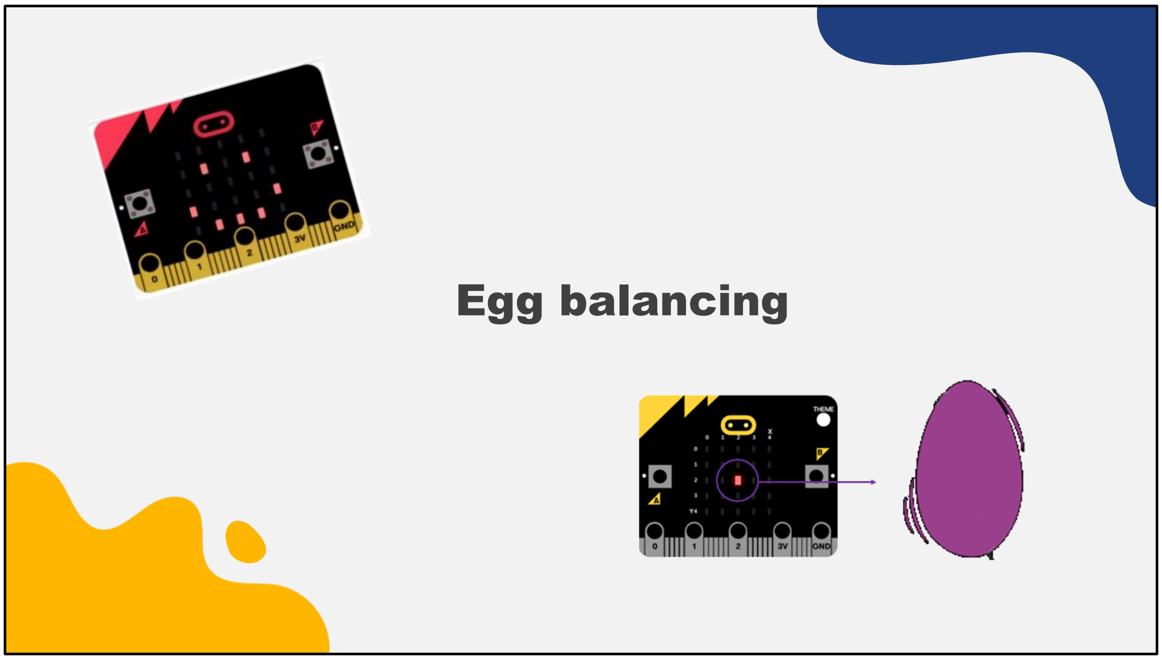
Explanation:
Initialization
Let's start by creating the potato variable and initializing it to -1 in on start. Remember, a negative potato value means you **don't** have the potato. We use radio set group to make sure players receive the messages.

Starting the game
To start the game, we respond to the A+B button press and assign a positi ve number to the **potato** variable. To make the game less predictable, we u se the pick random block to generate a value between 10 and 20.

Sending the potato
Sending the potato is done by shaking the micro:bit. If the **potato** variable is positive, we have the potato and we can send it. After sending it, we set the **potato** variable to -1 since we don't have it anymore.

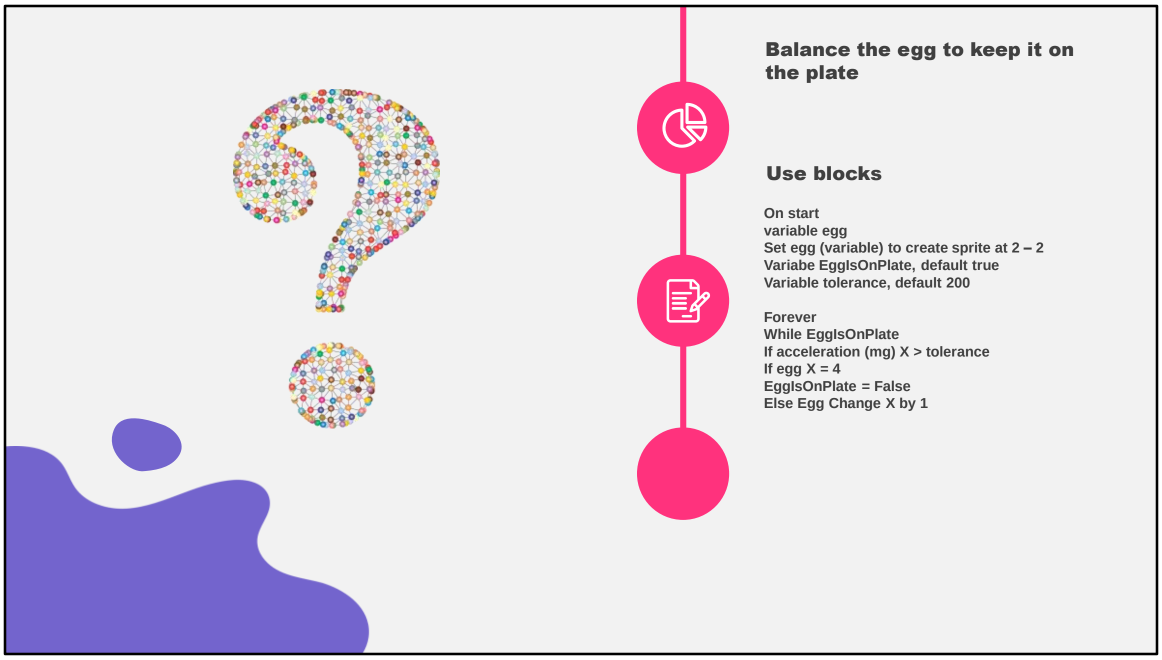
Receiving the potato
Receiving the potato is done in the on received number block. The **receive dNumber** represents the potato and is stored in the **potato** variable.

Ticking the clock
Making the clock tick down is done with a forever loop.
If the **potato** is equal to 0 (potato == 0), KABOOM! you lose!
If the **potato** variable is negative (potato < 0), we don't have the potato so we clear the screen.
If the **potato** variable is positive (potato > 0), we display a potato image an d decrease the variable by 1.



Egg balancing

Imagine you have an egg that is in the middle of a plate that you hold, try not to let the egg roll of!
The plate is the 5x5 led matrix and the egg the dot in the middle.
We will use the accelerometer sensor how the microbit board is leaning (backward/forward/to the left/right). If it is, the egg will move in that direction.
When the egg falls off the game ends.



Balance the egg to keep it on the plate

Use blocks

```
On start
variable egg
Set egg (variable) to create sprite at 2 – 2
Variabe EggIsOnPlate, default true
Variable tolerance, default 200

Forever
While EggIsOnPlate
If acceleration (mg) X > tolerance
If egg X = 4
EggIsOnPlate = False
Else Egg Change X by 1
```

Explanation:

Initialization

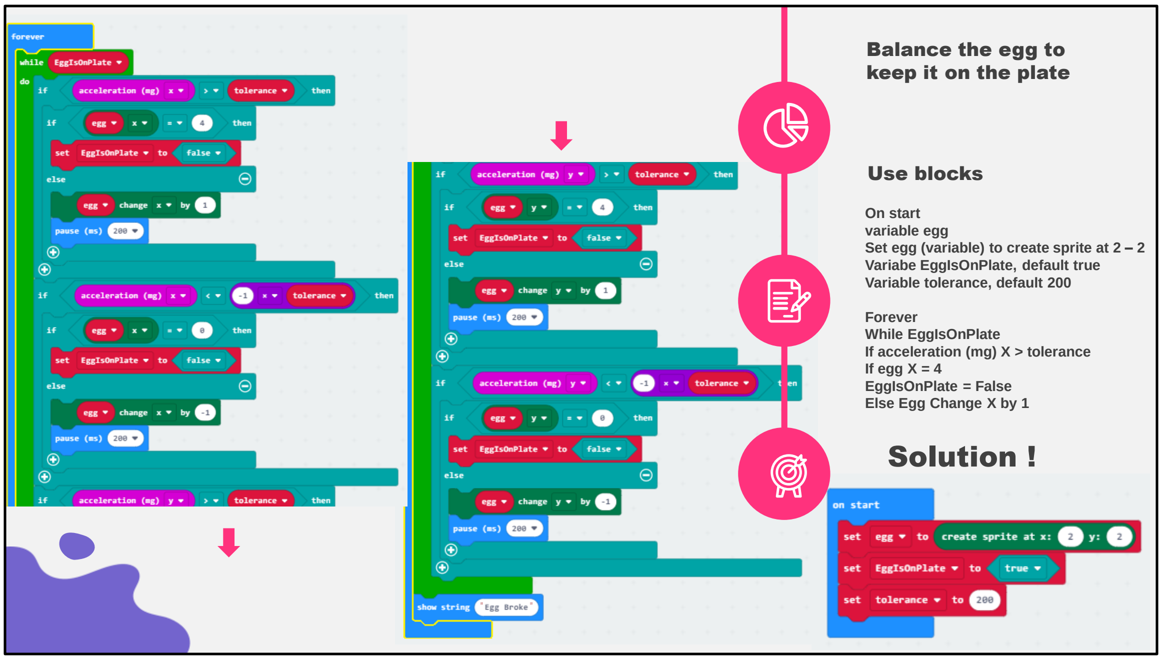
Create egg variable as a sprite with an x and y position
Create variable to flag when egg fell off the plate
Create variable for the tolerance or difficulty

In Forever loop

If you hold the plate completely level, the acceleration (mg) sensor should be close to zero
We check for each of the 4 directions whether the readout of this sensor is above the threshold of tolerance.
If that is the case, check if egg falls off plate, if not move the egg

Extra task

Make the game gradually harder by lowering the tolerance every 5 seconds



Balance the egg to keep it on the plate

Use blocks

On start
variable egg
Set egg (variable) to create sprite at 2 – 2
Variabe EggIsOnPlate, default true
Variable tolerance, default 200

Forever
While EggIsOnPlate
If acceleration (mg) X > tolerance
If egg X = 4
EggIsOnPlate = False
Else Egg Change X by 1

Solution !

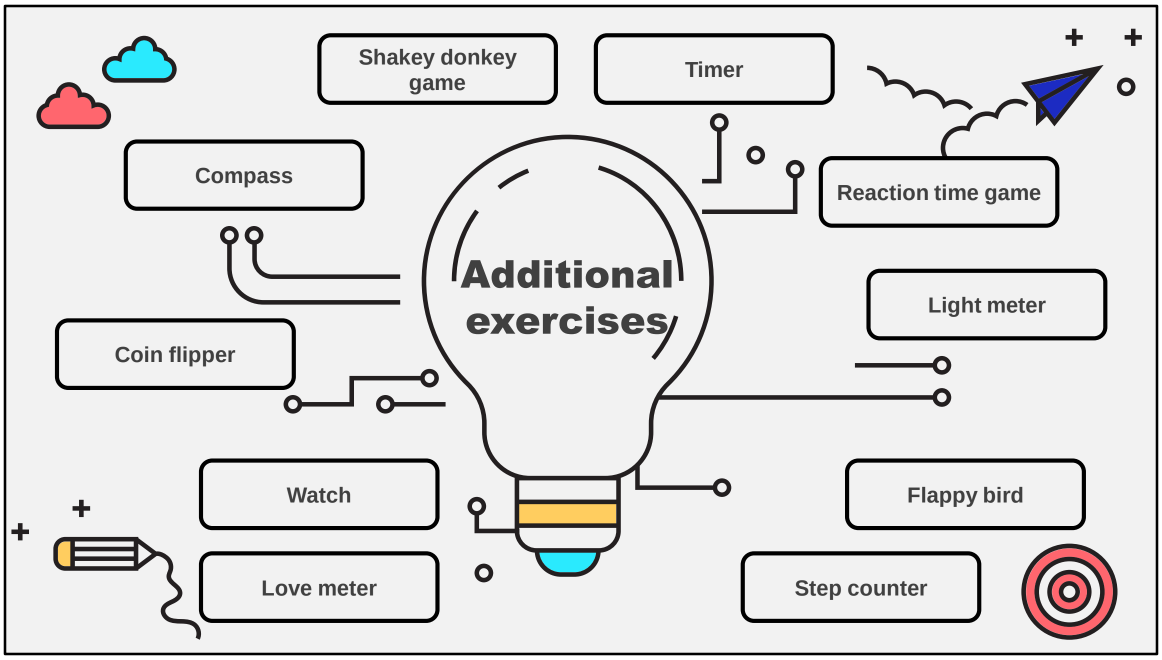
```
forever
  while EggIsOnPlate
  do
    if acceleration (mg) x > tolerance then
      if egg x = 4 then
        set EggIsOnPlate to false
      else
        egg change x by 1
        pause (ms) 200
    if acceleration (mg) x < -1 × tolerance then
      if egg x = 0 then
        set EggIsOnPlate to false
      else
        egg change x by -1
        pause (ms) 200
    if acceleration (mg) y > tolerance then
      if egg y = 4 then
        set EggIsOnPlate to false
      else
        egg change y by 1
        pause (ms) 200
    if acceleration (mg) y < -1 × tolerance then
      if egg y = 0 then
        set EggIsOnPlate to false
      else
        egg change y by -1
        pause (ms) 200
  show string "Egg Broke"
```

```
on start
  set egg to create sprite at x: 2 y: 2
  set EggIsOnPlate to true
  set tolerance to 200
```

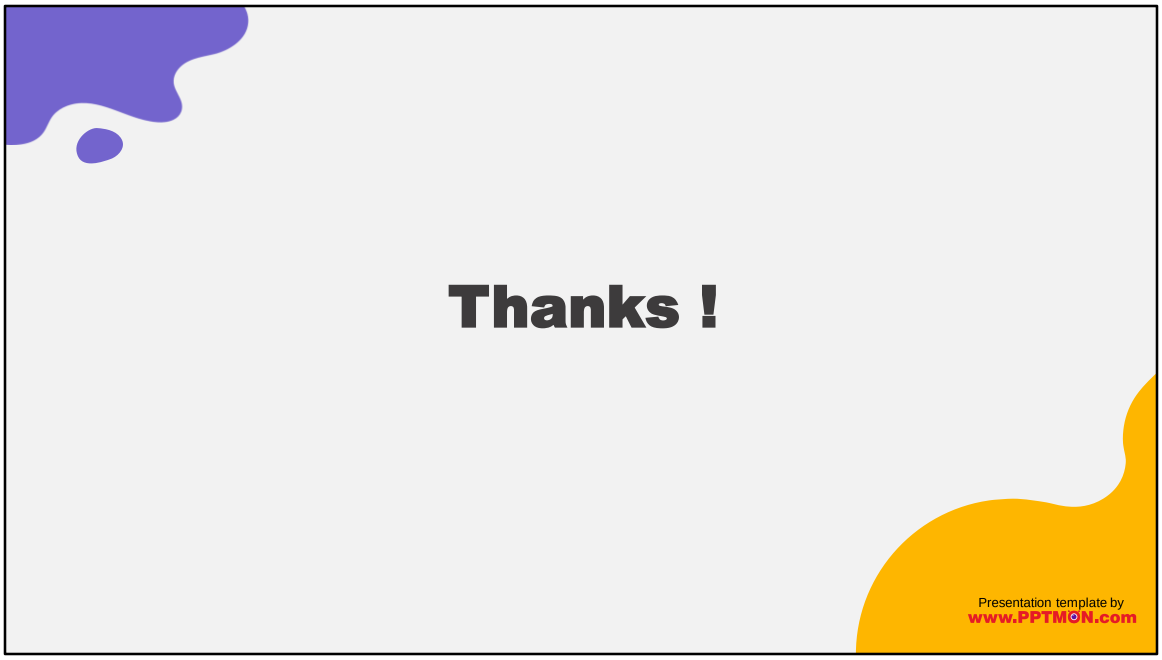
Explanation:
Initialization
Create egg variable as a sprite with an x and y position
Create variable to flag when egg fell off the plate
Create variable for the tolerance or difficulty

In Forever loop
If you hold the plate completely level, the acceleration (mg) sensor should be close to zero
We check for each of the 4 directions whether the readout of this sensor is above the threshold of tolerance.
If that is the case, check if egg falls off plate, if not move the egg

Extra task
Make the game gradually harder by lowering the tolerance every 5 seconds



Many more examples can be found on the makecode website.
They can be done as tutorials, or serve as inspiration to try it themselves or create something new!
The possibilities are only limited by the imagination.



Thanks !

Presentation template by
www.PPTMON.com

Conclude by saying they learned:

- The basic skill set to start using the microbot
- Coding fundamentals
- If they want to continue after the workshop, how to follow tutorials and make exercises from the makecode website
- How to think creatively and solve problems