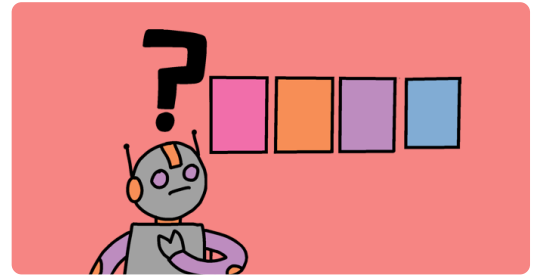![Raspberry Pi logo] **Projects**

Memory

Create a game in which you memorise and repeat random colour sequences

Step 1 Introduction

In this project, you will create a memory game in which you have to memorise and repeat a random sequence of colours!

What you will make

(i) What you will learn

- How to add sound to your Scratch project
- How to create and use lists to store data
- How to create and use custom blocks for repeating code

(i) What you will need

Hardware

- A computer capable of running Scratch 3

Software

Scratch 3 (either **online (https://rpf.io/scratchon)** or **offline (https://rpf.io/scratchoff)**)

Step 2 Create a colour sequence

First create a character that can display a random sequence of colours.

Open a new Scratch project.

Online: open a new online Scratch project at **rpf.io/scratch-new (https://rpf.io/scratch-new)**.

Offline: open a new project in the offline editor.

If you need to download and install the Scratch offline editor, you can find it at **rpf.io/scratchoff (https://rpf.io/scratchoff)**.

Choose a character sprite and a backdrop. You could use the ballerina, but your character doesn't have to be a person, they only need to be able to show different colours.
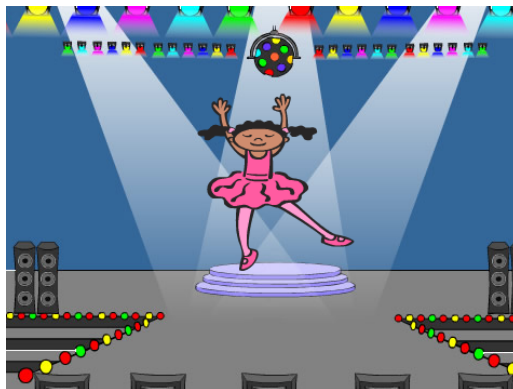

- Your game should use a different number to represent each colour:

  - 1 = red
  - 2 = blue
  - 3 = green
  - 4 = yellow

Give your character four costumes that have different colours, one costumes for each of the four colours shown above. Make sure that your coloured costumes are in the same order as the list above.
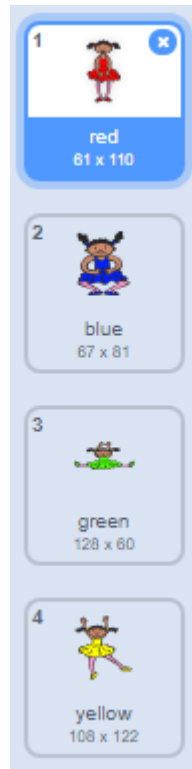
If you want, you can use the **color a shape** tool to fill parts of the costume with a different colour.
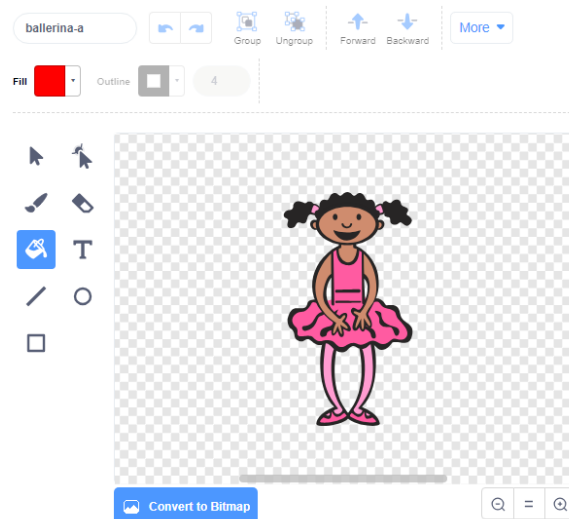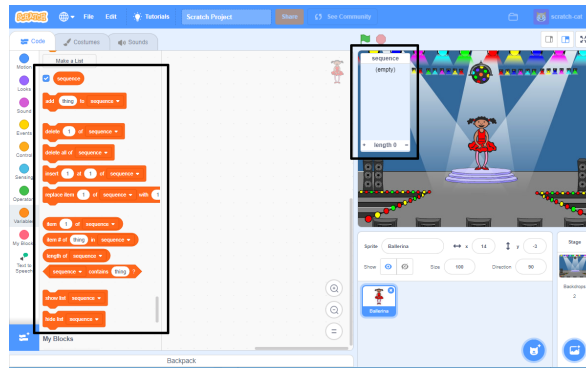

Next, add a list for storing the random sequence of colours that the player has to remember.

Create a list called sequence. Only the character sprite needs to see this list, so you can select **For this sprite only** when you create the list.

You should now see lots of new code blocks for using lists. The empty list should be visible in the top left-hand corner of the Stage.

Each colour has a different number, so you can choose a random colour by randomly choosing a number and adding it to the list.

Add this code to the character sprite to choose a random number and add it to sequence:

Test your code. Check that, each time you click the flag, a random number between 1 and 4 gets added to the list.

Can you add code to your program to generate five random numbers at once?

This is what your code should look like:

```
when 🚩 clicked
delete  all ▼  of  sequence ▼
repeat  5
    add  pick random  1  to  4   to  sequence ▼
```
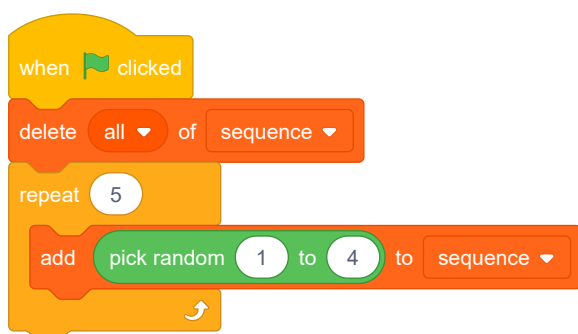
Each time a number gets added to the list, the character should change its costume so the costume's colour matches the number. Put these blocks into your code immediately below where a random number is added to sequence:

```
switch costume to  item ( length of  sequence ▼ ) of  sequence ▼
wait  1  seconds
```
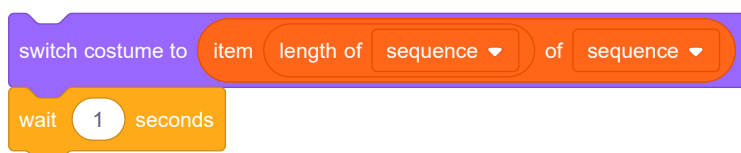
Step 3   Add sound

Test your project a few times. Do you notice that sometimes the same number is chosen twice (or more) in a row, which makes the sequence harder to memorise? ☑

Can you make a drum sound play each time the character sprite changes costume? And how about a different drum sound for each colour?

Add the Music extension to your project so you can use the `play drum` block. ☑

The code that plays the drum is **very** similar to the code that changes the character's costume. ☑

Here is how your finished code should look:


```
when ⚑ clicked
delete  all ▾  of  sequence ▾
repeat  5
    add  pick random  1  to  4  to  sequence ▾
    ♪♫ play drum  item  length of  sequence ▾  of  sequence ▾  for  0.25  beats
    switch costume to  item  length of  sequence ▾  of  sequence ▾
    wait  1  seconds
```
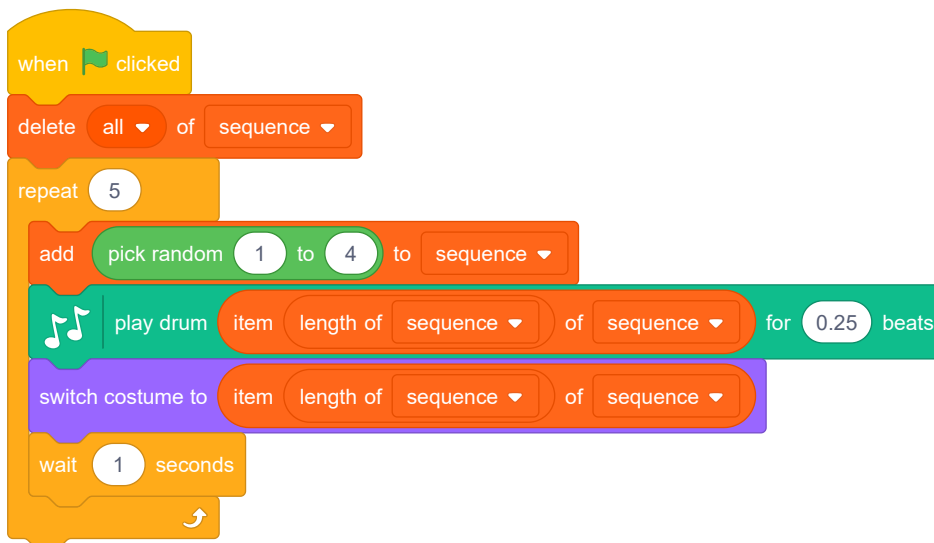
Step 4   Repeat the sequence

Now you're going to add four buttons the player has to press to repeat the colour sequence.

Add four new sprites to your project to represent the four buttons.

- Edit the new sprites' costumes so that there is one sprite in each of the four colours
- Put the sprites in the same order on the stage as the costumes: red, blue, green, yellow
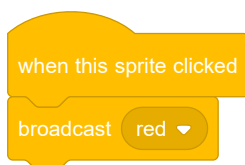

Add code to the red sprite so that, when the sprite is clicked, it **broadcasts** a 'red' message to the character sprite:


```
when this sprite clicked
broadcast  red ▼
```

A **broadcast** is like a message announced over a loudspeaker, which you can for example hear in schools or supermarkets. All of the sprites can hear the **broadcast**, but only the sprite whose job it is to respond will do something.

Add similar code to the blue, green, and yellow sprites to make them **broadcast** messages about their own colour.

Do you remember that the **broadcast** is like a loudspeaker message? You will add code to make it the character sprite's job to respond to the **broadcast** messages.

When your character sprite receives the message **red**, the code should check whether the number **1** is at the start of the **sequence** list (which means that **red** is the next colour in the sequence).

If **1** is at the start of the list, the code should remove the number from the list, because the player remembered the correct colour. Otherwise it's game over, and the code needs to **stop all** to end the game.

```
when I receive [ red ▼ ]
if < item (1 ▼) of [ sequence ▼ ] = (1) > then
    delete (1 ▼) of [ sequence ▼ ]
else
    say [Game over!] for (1) seconds
    stop [ all ▼ ]
```

Add to the code you just wrote so that a drum beat also plays when the character sprite receives the correct **broadcast**.

Here is the code you will need to add:

```
when I receive [ red ▼ ]
if < item (1 ▼) of [ sequence ▼ ] = (1) > then
    play drum (1) Snare Drum ▼ for (0.25) beats
    delete (1 ▼) of [ sequence ▼ ]
else
    say [Game over!] for (1) seconds
    stop [ all ▼ ]
```

Duplicate the code you used to make your character sprite respond to the message `red`. Change the duplicated code so that it sends the message `blue`.

When the sprite responds to the message `blue`, which bit of code should stay the same, and which bit should change? Remember that each colour has a corresponding number.

Change the character sprite's code so that the character responds correctly to the `blue` message.

Here is how your code should look for the `blue` broadcast.

```
when I receive [blue ▼]
if < item (1 ▼) of [sequence ▼] = (2) > then
    play drum [(2) Bass Drum ▼] for (0.25) beats
    delete (1 ▼) of [sequence ▼]
else
    say [Game over!] for (1) seconds
    stop [all ▼]
```
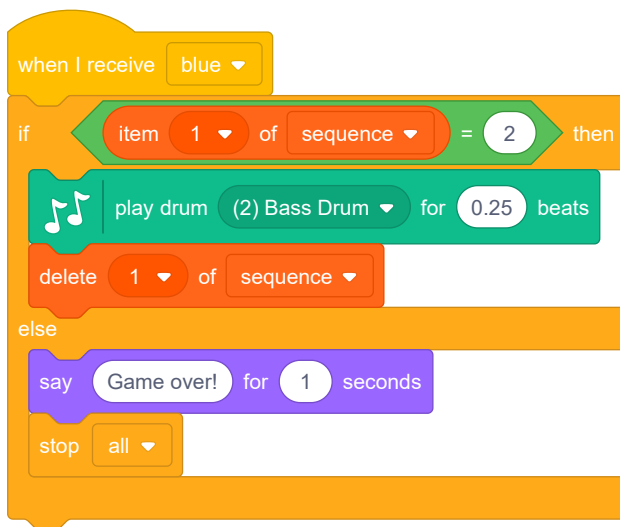
Duplicate the code again twice (for the green and yellow buttons), and change the necessary parts so that the character responds correctly to the new `broadcasts`.

Remember to test the code! Can you memorise a sequence of five colours? Is the sequence different each time?

When the player repeats the whole colour sequence correctly, the `sequence` list emtpy and the player wins. If you want, you can also display some flashing lights as a reward once the `sequence` list is empty.

Add this code to the end of your character's `when flag clicked` script:

```
wait until < length of ( sequence ▾ ) = ( 0 ) >
broadcast ( won ▾ ) and wait
```

Switch to the Stage, and import the `drum machine` sound or another sound you like.

Add this code to play a sound and make the backdrop change colour when the player wins.

```
when I receive ( won ▾ )
start sound ( drum machine ▾ )
repeat ( 50 )
    change ( color ▾ ) effect by ( 25 )
    wait ( 0.1 ) seconds
clear graphic effects
```

Step 5   Multiple levels

So far, the player only has to remember a sequence of five colours. Improve your game by adding a score, and adding code so that as the player scores points, the game moves to the next level and the colour sequence to remember becomes longer.

Create a new variable called `score`. ✓

Based on the `score`, the game will decide on the length of the colour sequence. Start with a score (and a sequence length) of 3.

Add a block at the start of your character's `when flag clicked` code to set the `score` to 3. ✓

Instead of always creating a sequence of five colours, you now want the `score` to determine the sequence length.

Change the character's `repeat` loop (for creating the colour sequence) to repeat `score` times: ✓
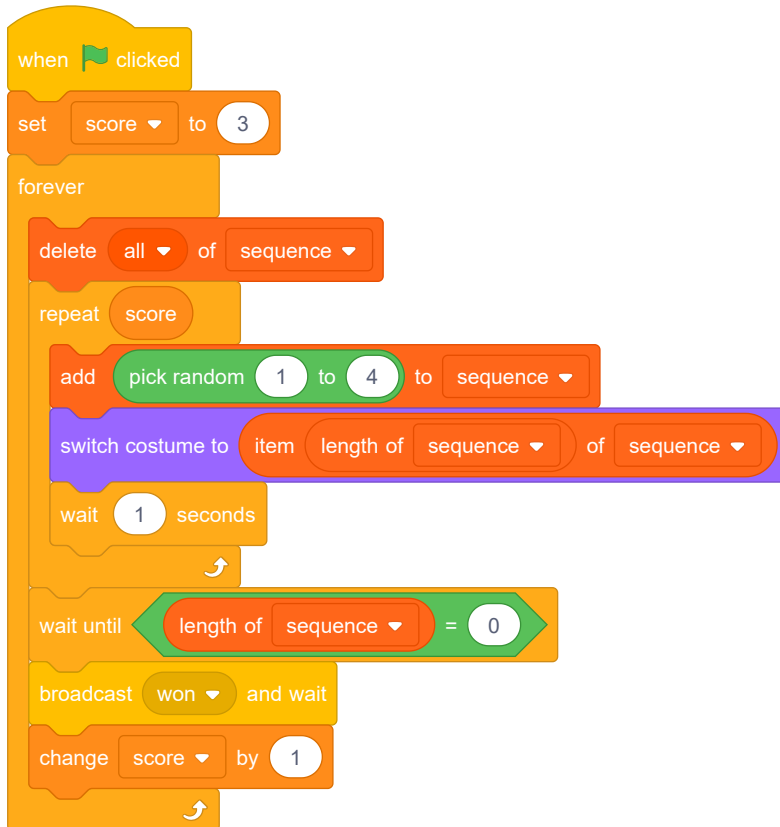


```
repeat  score
      ↻
```
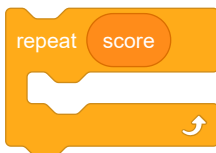
If the player repeats the correct sequence, you should add 1 to `score`, and doing so increases the length of the next sequence. Add the following block to the character's code **at the point you know the sequence is correct**: ✓


```
change  score ▾  by  1
```

You know the sequence is correct at the point when the game `broadcasts` the 'win' message.

Finally, add a **forever** loop around the code that generates the sequence, so that the game creates a new colour sequence for each level. This is how your character's code might look:


```
when 🏳 clicked
set score to 3
forever
    delete all of sequence
    repeat score
        add (pick random 1 to 4) to sequence
        switch costume to (item (length of sequence) of sequence)
        wait 1 seconds
    wait until <(length of sequence) = 0>
    broadcast won and wait
    change score by 1
```

Get your friends to test out your game. Remember to hide the **sequence** list before they play it!

Step 6   High score

Now save the high score so that you can play against your friends.

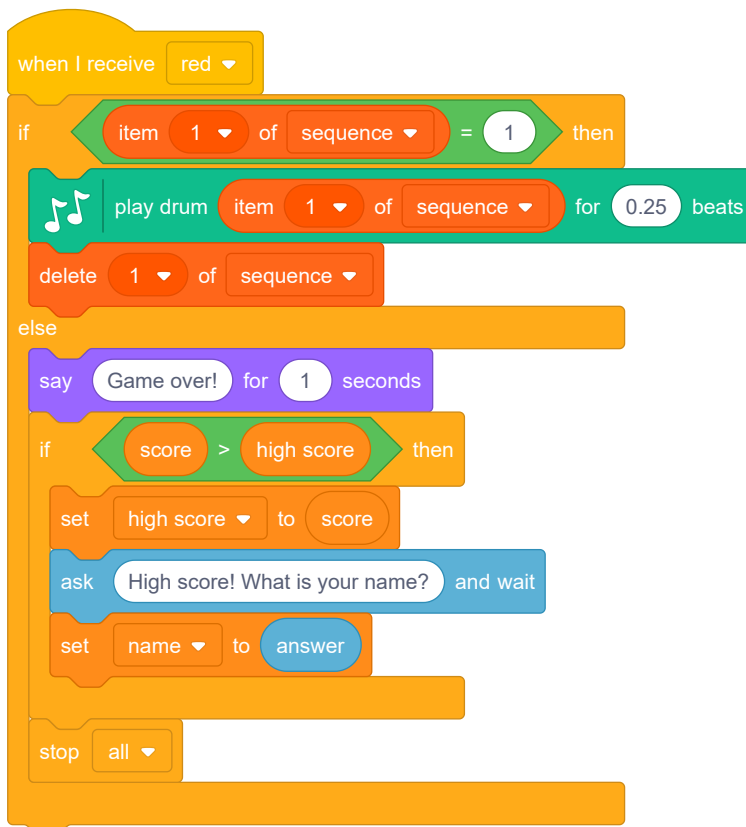> Add two new variables called `high score` and `name` to your project.

When the game ends because the player gets the sequence wrong, the game should check whether the score is higher than the current high score. If it is, the game should save the score as the high score, and also store the name of the player.

> Add code to your character sprite to store the `high score`. Also ask for the player's name, and store it in the `name` variable.
>
> Here's how your code for when the red button is pressed should look:


```
when I receive red
if < item 1 of sequence = 1 > then
    play drum item 1 of sequence for 0.25 beats
    delete 1 of sequence
else
    say Game over! for 1 seconds
    if < score > high score > then
        set high score to score
        ask High score! What is your name? and wait
        set name to answer
    stop all
```

You need to add this new code to the character sprite for the other three colours too!

Can you see that the 'Game over' code for each of the four colours is exactly the same?

```
say  Game over!  for  1  seconds
if  score > high score  then
    set  high score ▼  to  score
    ask  High score! What is your name?  and wait
    set  name ▼  to  answer
stop  all ▼
```
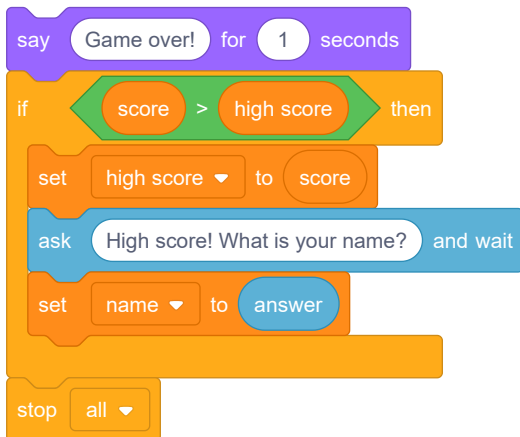
If you need to change any of the 'Game over' code, for example to add a sound or change the 'Game over' message, you have to change it four times. That's annoying and wastes a lot of time.

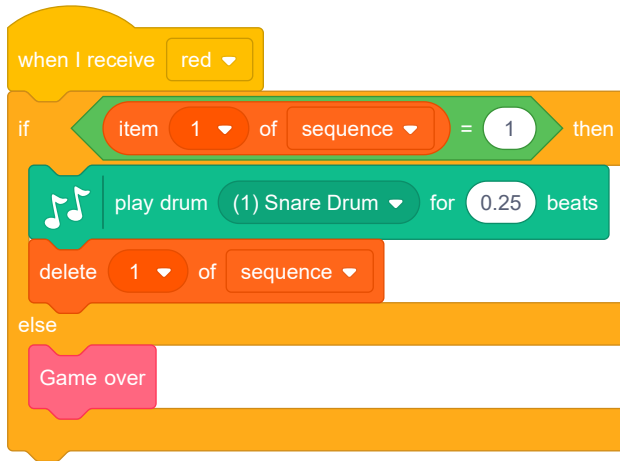Instead, you can define your own code block, and use it anywhere in your project.

Click on **My blocks**, and then on **Make a Block**. Call this new block **Game over**.

Add the code from the **else** block connected to the **red** broadcast to the **Game over** block so that it looks like this:


```
define  Game over
say  Game over!  for  1  seconds
if  score > high score  then
    set  high score ▼  to  score
    ask  High score! What is your name?  and wait
    set  name ▼  to  answer
stop  all ▼
```

Now remove the code that's in the `else` block connected to the `red` broadcast, and add in the `Game over` block instead:


```
when I receive  red ▾
if  ( item ( 1 ▾ ) of  sequence ▾ ) = ( 1 )  then
    play drum  (1) Snare Drum ▾  for  0.25  beats
    delete ( 1 ▾ ) of  sequence ▾
else
    Game over
```
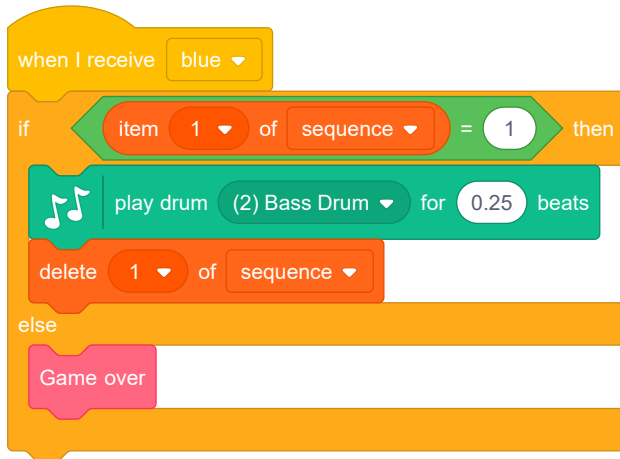
Test your new block by playing the game and clicking the red button at the wrong point in the colour sequence.

Your new `Game over` block is a **function**, a little script that you can use anywhere you like in your code by adding the `Game over` block in.

Also replace the code in the `else` block connected to the `broadcasts` for the other colours with your new `Game over` block. Here is what the code for the `blue` message should look like


```
when I receive  blue ▼

if        item  1 ▼  of  sequence ▼  =  1    then
    🎵  play drum  (2) Bass Drum ▼  for  0.25  beats
    delete  1 ▼  of  sequence ▼
else
    Game over
```
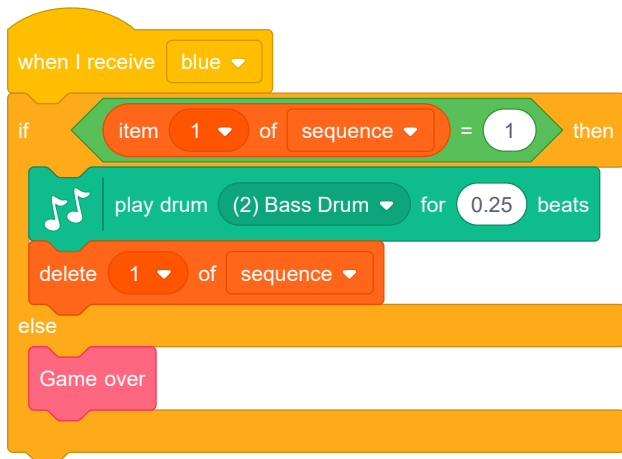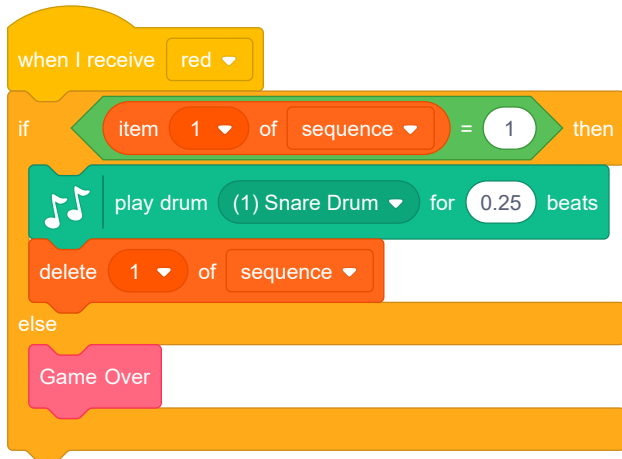

Now add a sound that plays when the wrong button is pressed. You only need to add this code once in the `Game over` block that you made, and not four separate times!


```
define  Game over

start sound  Cough1 ▼
say  Game over!  for  1  seconds
if    score  >  high score    then
    play sound  trumpet1 ▼
    set  high score ▼  to  score
    ask  High score! What is your name?  and wait
    set  name ▼  to  answer

stop  all ▼
```

Step 7   Challenge: improve your game

Make more blocks

Do you see any other code that is the same for all four buttons?

```
when I receive red ▼
if  item 1 ▼ of sequence ▼ = 1  then
    ♫ play drum (1) Snare Drum ▼ for 0.25 beats
    delete 1 ▼ of sequence ▼
else
    Game Over
```

```
when I receive blue ▼
if  item 1 ▼ of sequence ▼ = 1  then
    ♫ play drum (2) Bass Drum ▼ for 0.25 beats
    delete 1 ▼ of sequence ▼
else
    Game over
```

Can you make another custom block that all buttons can use?

Another costume

Can you see that your game starts with your character showing one of the four colours, and that the character always displays the last colour in the sequence while the player is repeating the colour sequence?

Can you add another plain white costume to your character, and add code so that the character displays this costume at the start of the game and while the player is repeating the sequence?

Difficulty level

Can you allow your player to choose between playing the game in 'easy mode' (using just the red and blue colours) and 'normal mode' (which uses all four colours)?

If you want, you can even add a 'hard' mode, which makes use of a fifth drum!

Step 8   What next?

Try out the next project, **Dodgeball (https://projects.raspberrypi.org/en/projects/dodgeball?utm_source=pathway&utm_medium=whatnext&utm_campaign=projects)**, where you will make a game in which you have to avoid balls while moving from platform to platform.

dodgeball game being played