Projects

Binary hero

Score points by playing the notes of a song as they scroll past

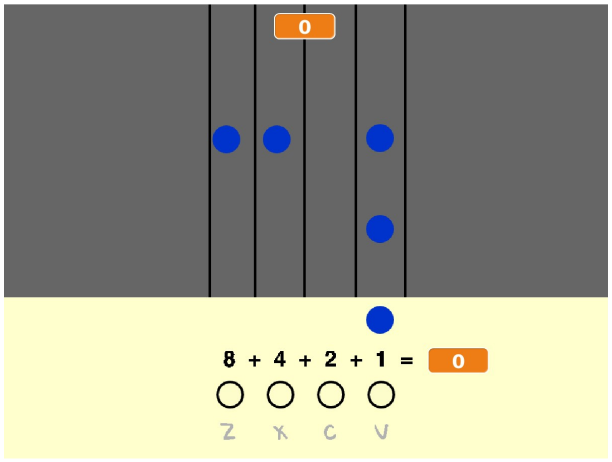
Scratch

Step 1 Introduction

In this project you will make a game in which you play the notes of a song as they scroll down the Stage.

What you will make

The notes will fall from above, and you will have to press keys to “catch” and play the notes.



What you will learn

- How to use lists to store sequences of notes and timings
- How to use custom blocks with inputs

What you will need

Hardware

- A computer capable of running Scratch 3

Software

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

Downloads

- **Offline starter project** (http://rpf.io/p/en/binary-hero-go)

Additional notes for educators

You can find the **completed project here** (http://rpf.io/p/en/binary-hero-get).

Step 2 Key presses

How many notes can you play with four keys? It might be more than you think!

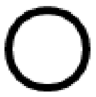
Open the 'Binary hero' Scratch starter project.

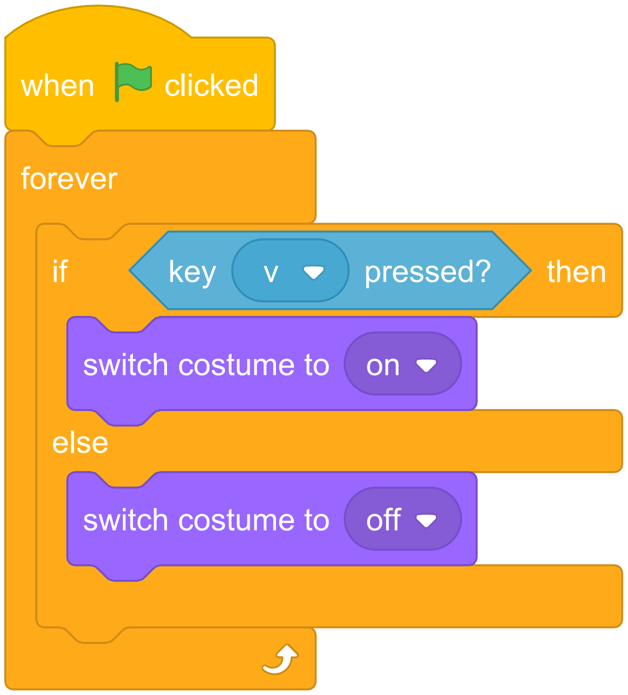
Online: open the starter project at **rpf.io/binary-hero-on** (http://rpf.io/binary-hero-on). If you have a Scratch account, you can click on **Remix** in the top right-hand corner to save a copy of the project.

Offline: open the **starter project** [(http://rpf.io/p/en/binary-hero-go)](http://rpf.io/p/en/binary-hero-go)in the offline editor. If you need to download and install the Scratch offline editor, you can find it at **rpf.io/scratchoff** [(http://rpf.io/scratchoff)](http://rpf.io/scratchoff).

Start by showing which key is being pressed.

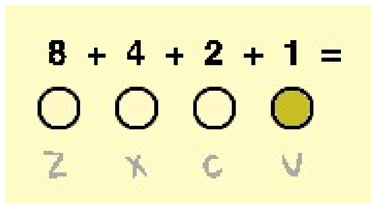
Click on the sprite called '1', and add code to change the sprite's costume if the v key is pressed.



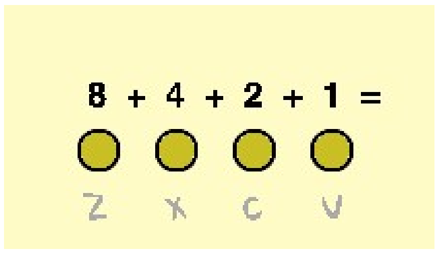


```
when flag clicked
forever
    if <key (v ▾) pressed?> then
        switch costume to (on ▾)
    else
        switch costume to (off ▾)
    end
end
```

When you test your code by pressing the **v** key, the sprite should light up.



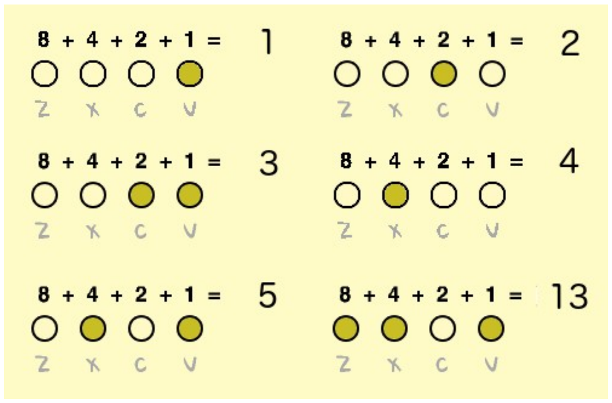
Do the same for the other three sprites so that they light up if the z, x, or c keys are pressed.



Step 3 Binary numbers

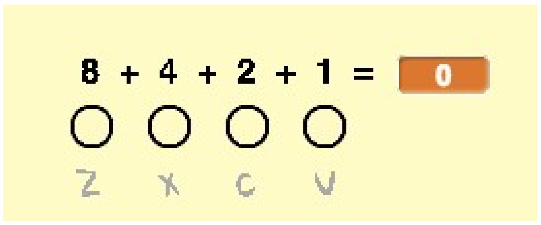
You will use different combinations of pressing the four keys to play different notes. Each of the keys is either on (pressed) or off (not pressed). This means that you can think of each combination of keys as a **binary number**.

Moving from right to left the keys double in value: 1, 2, 4, and 8. By adding up the numbers above the keys that are pressed, you can work out the value of the note.



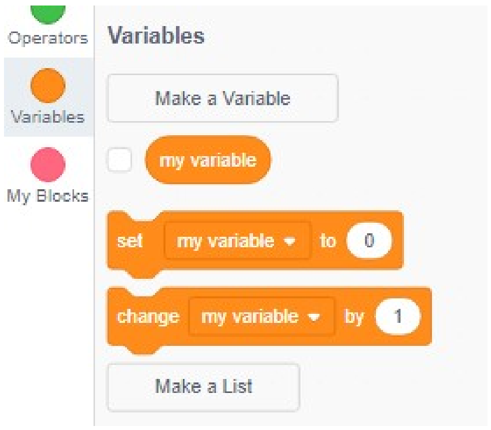
There are 2^4 = **16 combinations** of pressing the four keys. This means that you can play 15 different notes, as 0 will mean that no note plays.

Create a new variable called `note`, and drag it next to the four note sprites.

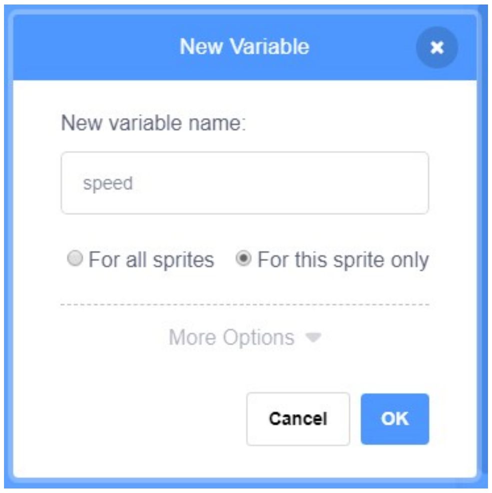


Add a variable in Scratch

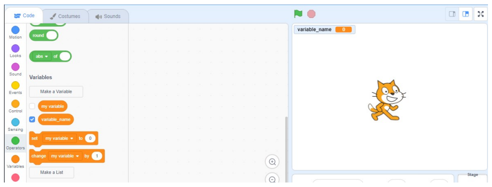
- Click on **Variables** in the Code tab, then click on **Make a Variable**.



- Type in the name of your variable. You can choose whether you would like your variable to be available to all sprites, or to only this sprite. Press **OK**.



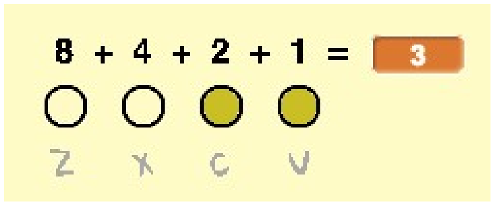
- Once you have created the variable, it will be displayed on the Stage, or you can untick the variable in the Scripts tab to hide it.



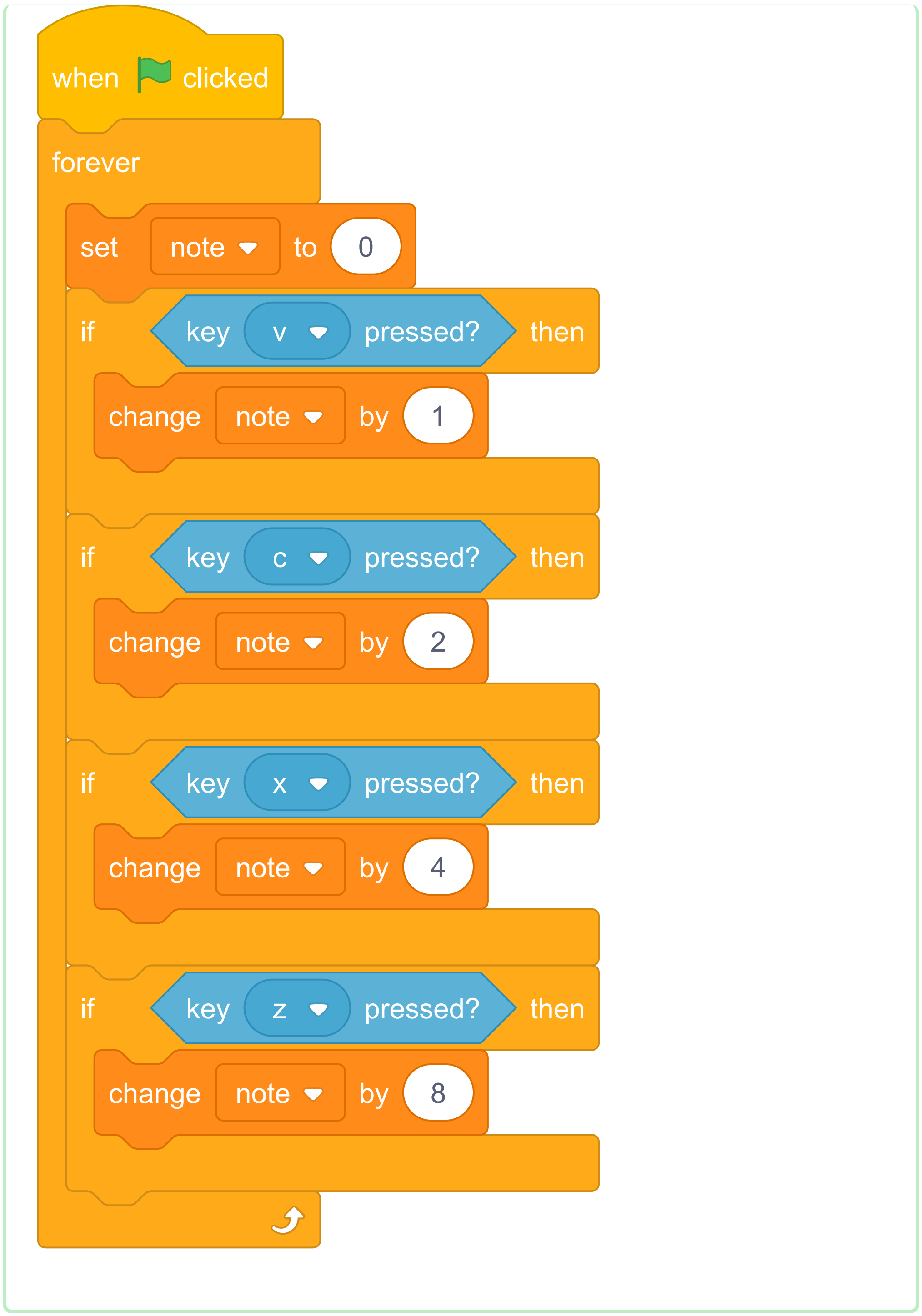
`note` will store the value of the note that should be played.

Add code to the Stage to use the combination of pressed keys to calculate the value of `note`.

For example, when c and v are pressed, the value of `note` should be 3.



This is what your code should look like:



```
when flag clicked
forever
    set note to 0
    if key v pressed? then
        change note by 1
    if key c pressed? then
        change note by 2
    if key x pressed? then
        change note by 4
    if key z pressed? then
        change note by 8
```

Step 4 Play notes

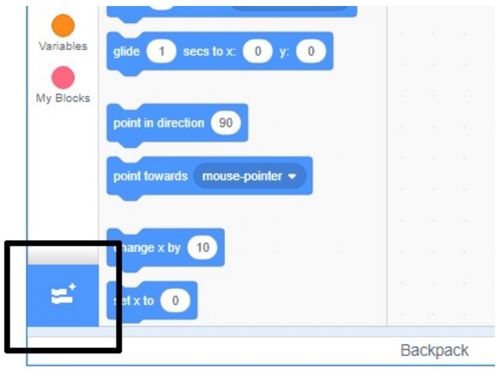
Play notes when the keys are pressed.

Add the Music extension to your project.

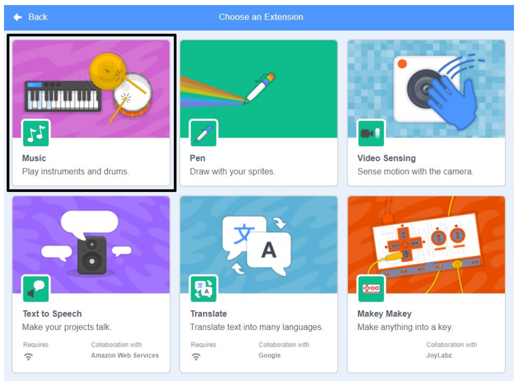
How to add the Music extension

To use the Music blocks in Scratch, you need add the **Music extension**.

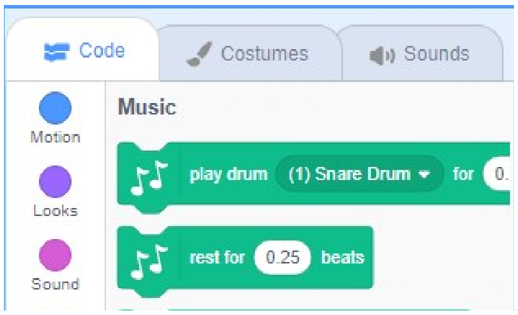
- Click on the **Add extension** button in the bottom left-hand corner.

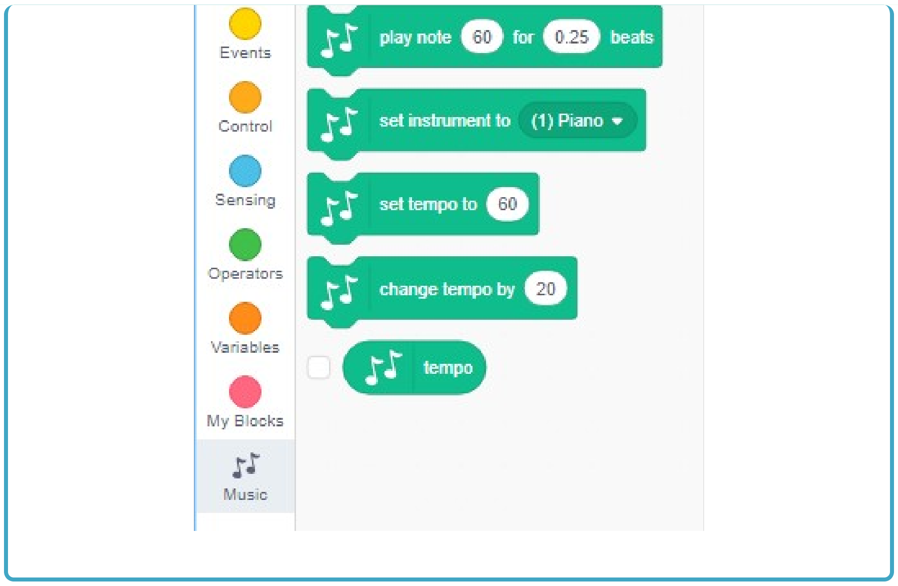


- Click on the **Music** extension to add it.

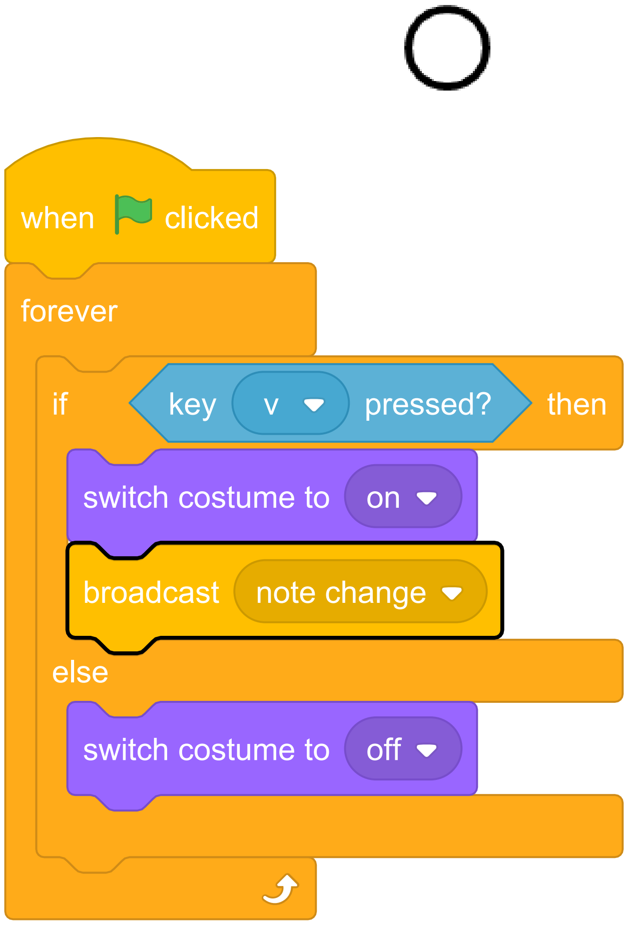

Back
Choose an Extension
Music
Play instruments and drums.
Pen
Draw with your sprites.
Video Sensing
Sense motion with the camera.
Text to Speech
Make your projects talk.
Requires
Collaboration with Amazon Web Services
Translate
Translate text into many languages.
Requires
Collaboration with Google
Makey Makey
Make anything into a key.
Collaboration with JoyLabz

- The Music section then appears at the bottom of the blocks menu.





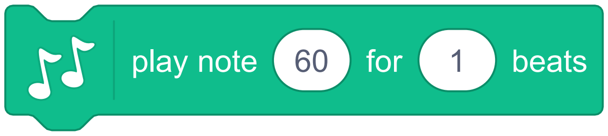
Broadcast a ‘note change’ message whenever **any of the four keys** is pressed.



```
when flag clicked
forever
    if key v pressed? then
        switch costume to on
        broadcast note change
    else
        switch costume to off
```

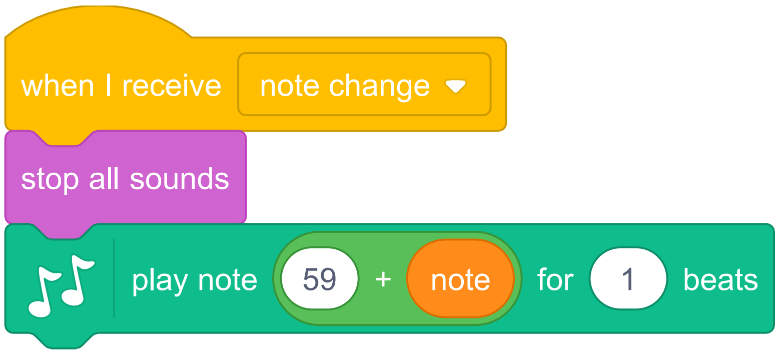
Add code to the Stage to play a note when a combination of keys is pressed.

Your notes should start at middle C, which is note 60.



```
play note (60) for (1) beats
```

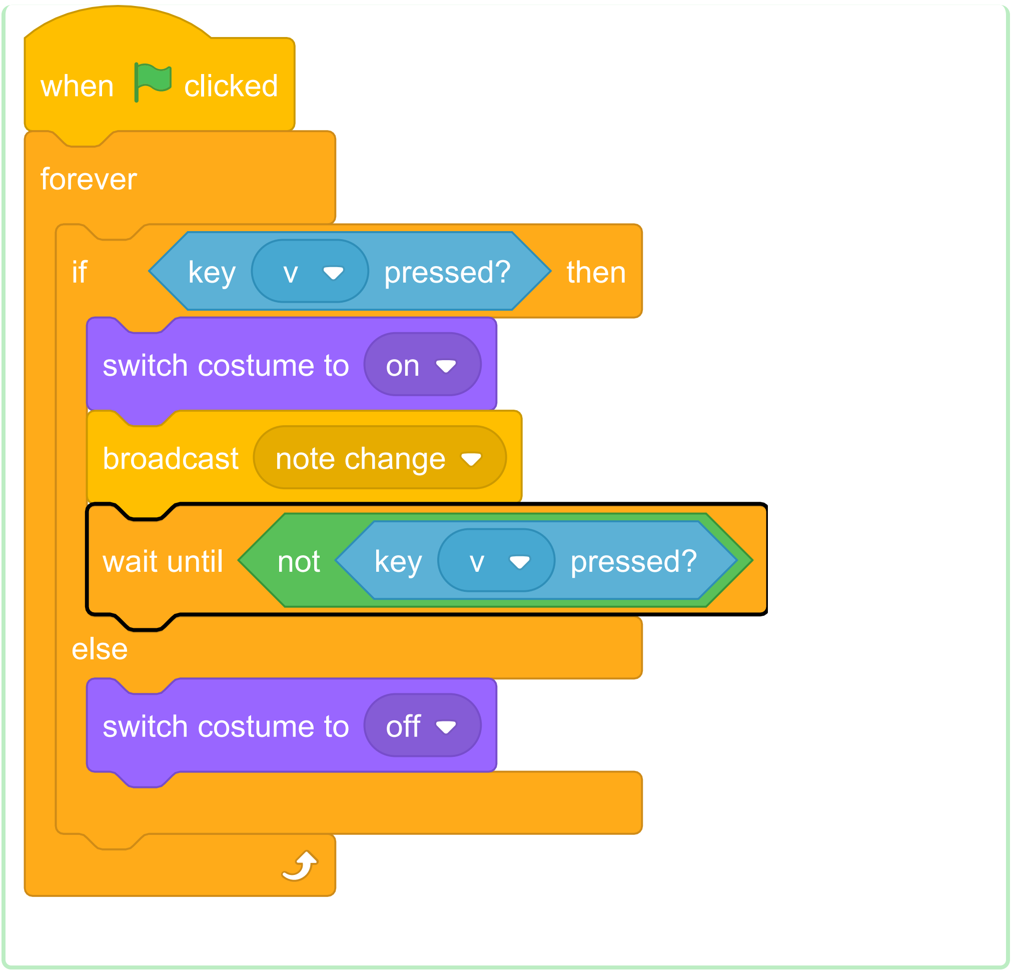
This is what your code should look like:



```
when I receive [note change v]
stop all sounds
play note ((59) + (note)) for (1) beats
```

Test your code. Can you hear that a note is repeatedly played when you hold down a key?

Add code so that the **all** the key sprites only play a note **once** when a key is held down?



```
when flag clicked
forever
    if <key (v) pressed?> then
        switch costume to (on)
        broadcast (note change)
        wait until <not <key (v) pressed?>>
    else
        switch costume to (off)
```

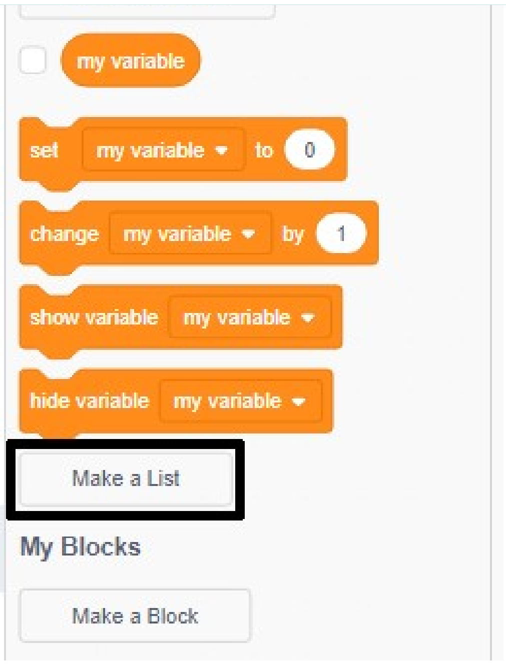
Step 5 Scrolling notes

You need to make notes scroll down the Stage so that the player knows which keys to press and when to press them.

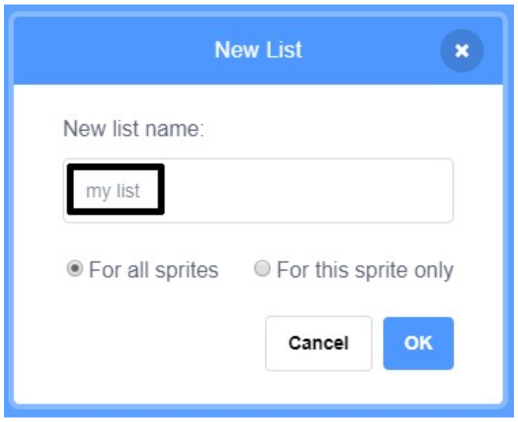
Create two lists called `notes` and `times`.

Make a list

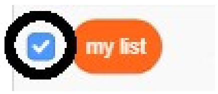
- Click on **Make a List** under **Variables**.



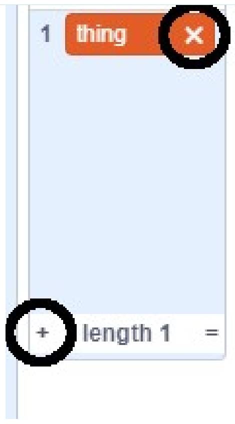
- Type in the name of your list. You can choose whether you would like your list to be available to all sprites, or to only a specific sprite. Click **OK**.



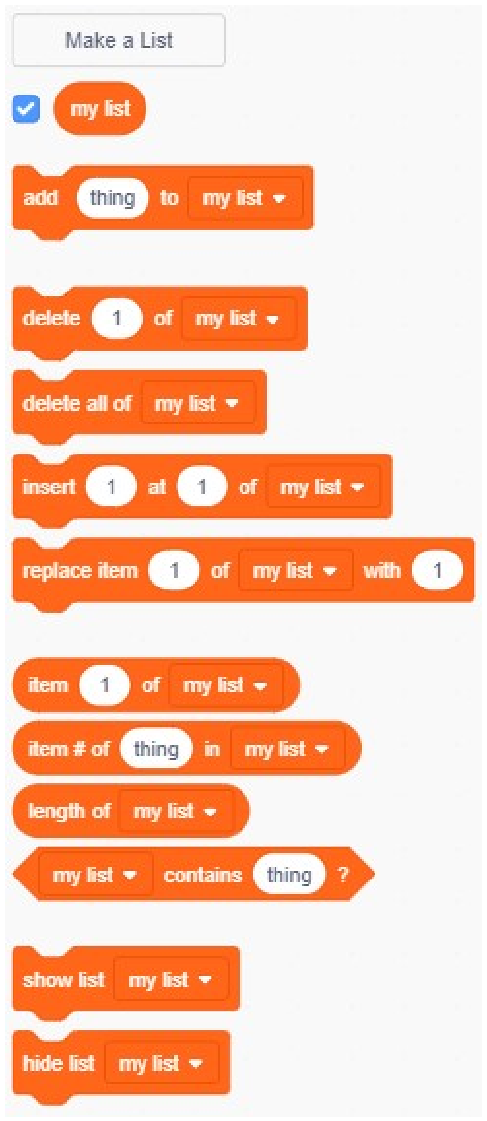
- Once you have created the list, it will be displayed on the stage, or you can untick the list in the Scripts tab to hide it.



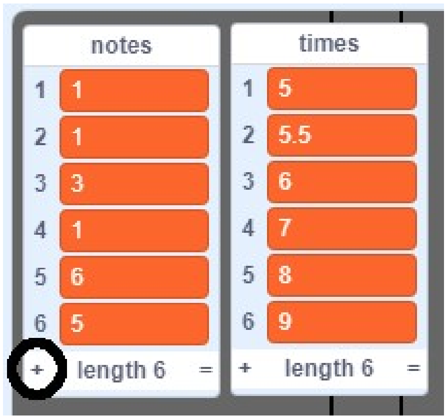
- Click the + at the bottom of the list to add items, and click the cross next to an item to delete it.



- New blocks will appear and allow you to use your new list in your project.



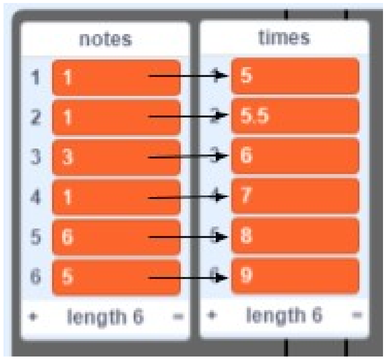
Add the following numbers to your `notes` and `times` lists.
Note: make sure to **add these exact numbers in the right order**.



	notes		times
1	1	1	5
2	1	2	5.5
3	3	3	6
4	1	4	7
5	6	5	8
6	5	6	9
+	length 6 =	+	length 6 =

Here's how songs are stored in your game:

- The `notes` list stores the notes of the song (from 1 to 15), in order
- The `times` list stores the times when the notes should be played in the song



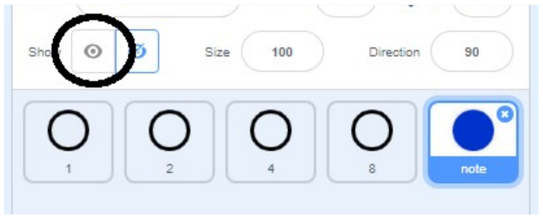
	notes		times
1	1	1	5
2	1	2	5.5
3	3	3	6
4	1	4	7
5	6	5	8
6	5	6	9
+	length 6 =	+	length 6 =

So with the two new lists:

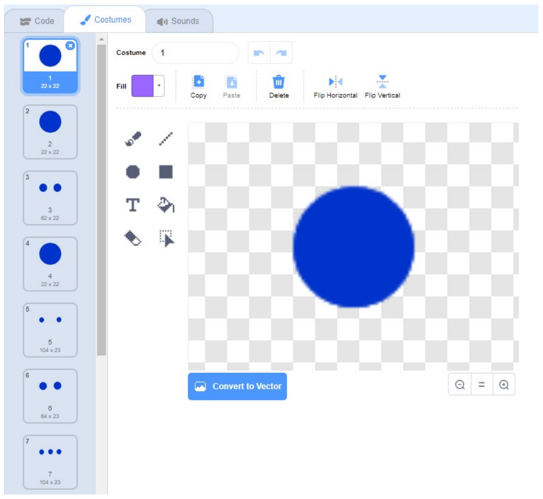
- Note 1 (middle C) should be played at 5 seconds
- Note 1 should be played again at 5.5 seconds
- Note 3 should be played at 6 seconds
- etc…

Click on the 'note' sprite and then click on **show**.

Sprite note ↔ x 22 ↕ y -130

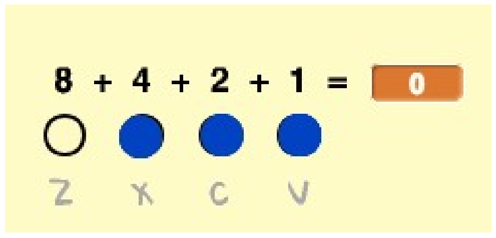


Then click on **Costumes**.

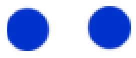


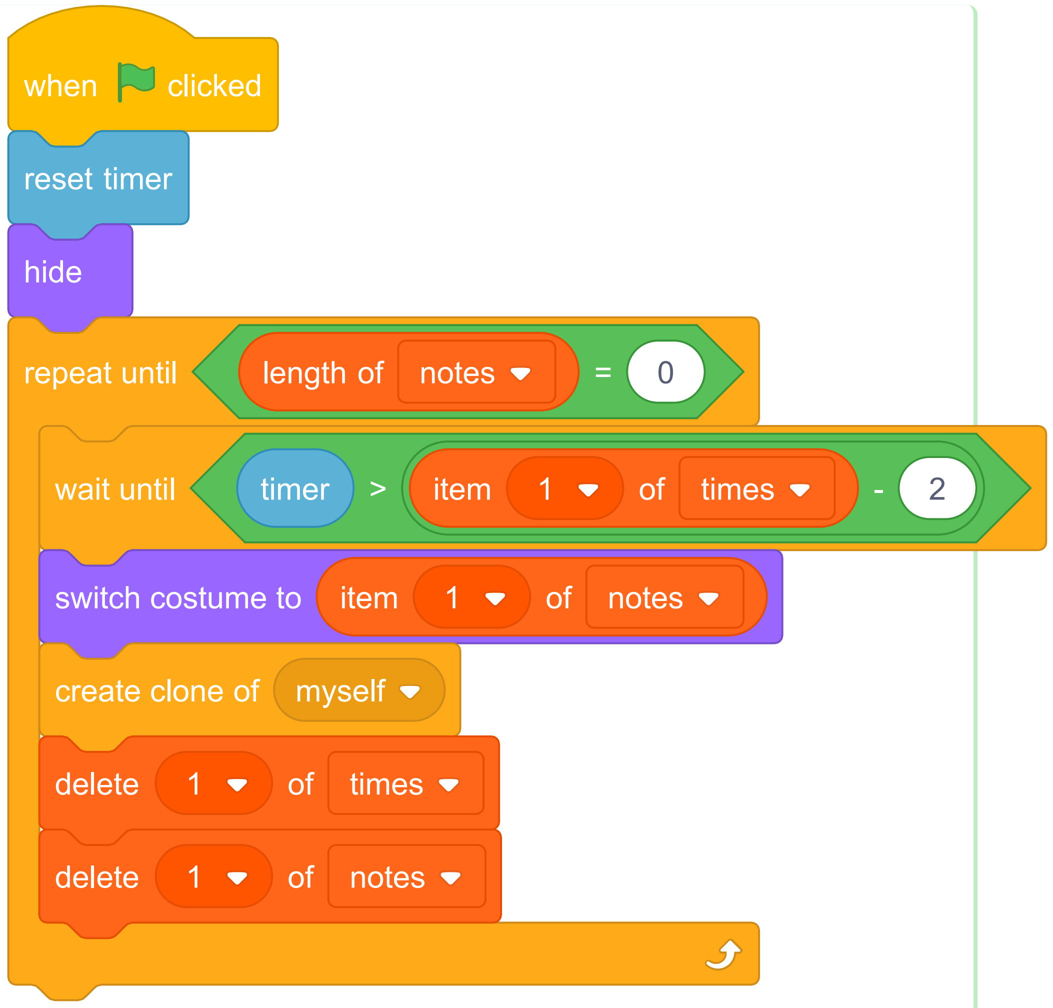
You should see that the 'note' sprite has 15 different costume, one for each different note from 1 to 15.

Add code to create a 'note' sprite clone for every note stored in `notes`. Each clone should be created at the correct time stored in `times`. Each clone should be created two seconds before its note needs to be played. This gives the clone two seconds to move down the screen. You'll create the code to move your clones in a little bit!



This is what your code should look like:

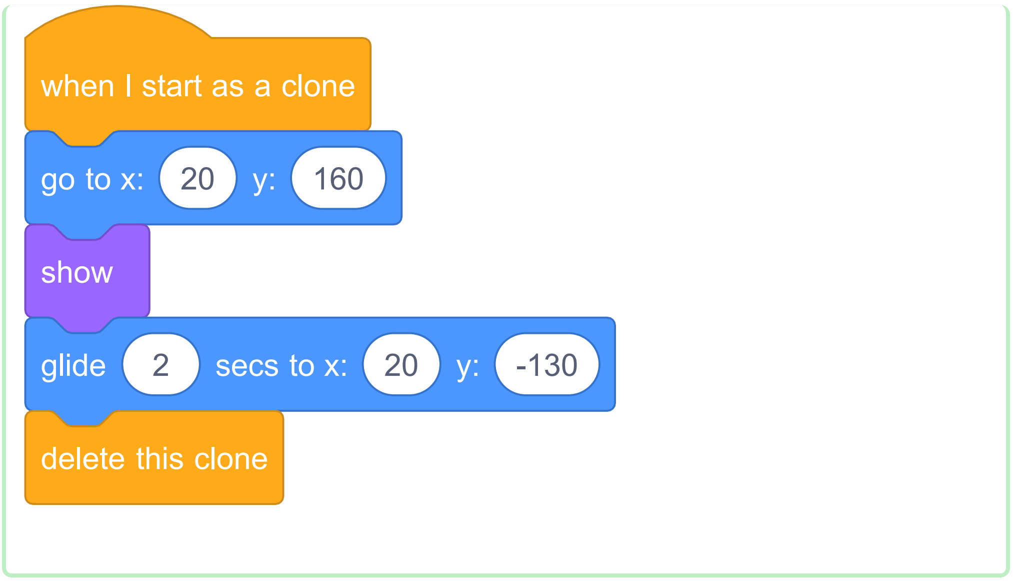




```
when green flag clicked
reset timer
hide
repeat until <(length of [notes v]) = (0)>
    wait until <(timer) > ((item (1 v) of [times v]) - (2))>
    switch costume to (item (1 v) of [notes v])
    create clone of (myself v)
    delete (1 v) of [times v]
    delete (1 v) of [notes v]
end
```

When you test your code now, nothing seems to happen, because the 'note' sprite is hidden. If you show (or don't hide) the sprite, then you should see clones being created on top of each other.

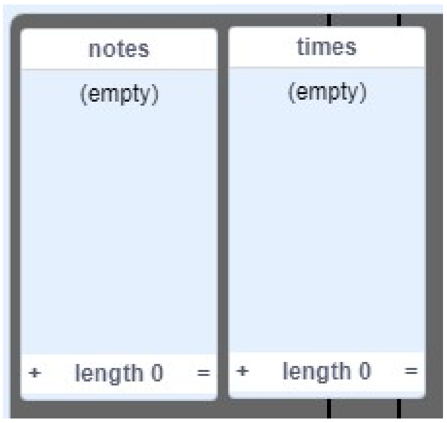
Add code to make each 'note' clone glide from the top to the bottom of the Stage before being deleted.



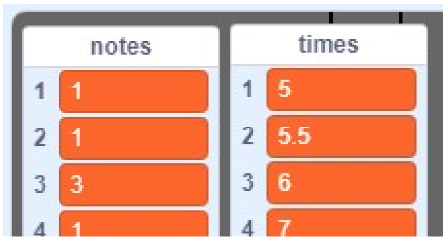
```
when I start as a clone
go to x: (20) y: (160)
show
glide (2) secs to x: (20) y: (-130)
delete this clone
```

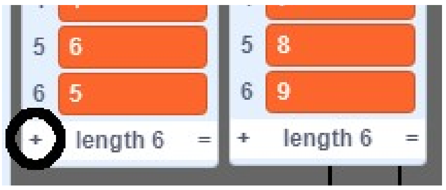
Step 6 Store your song

At the moment, notes are removed from the lists after being played, so you're left with empty lists:



You're now going to add code to store songs in your project, so that you don't have to add to your lists each time.



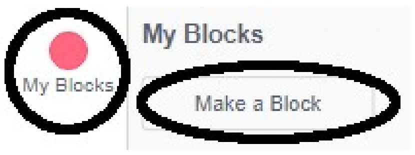


Make a new block called `load 'happy birthday'` that clears both the `notes` and `times` lists, and then adds the correct numbers back into both lists.

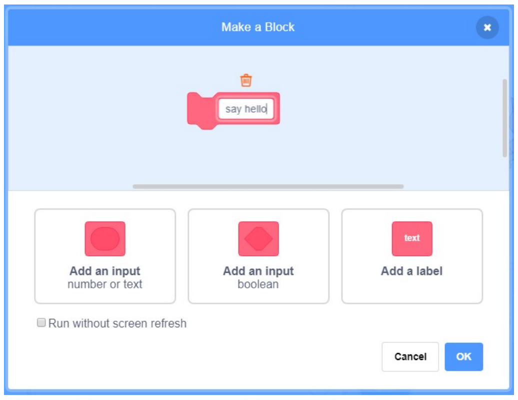
Making a block

Making a block

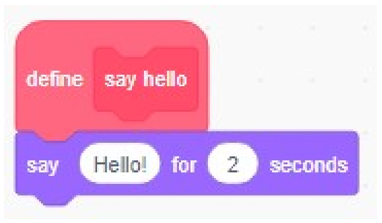
- Click on **My Blocks**, and then click **Make a Block**.



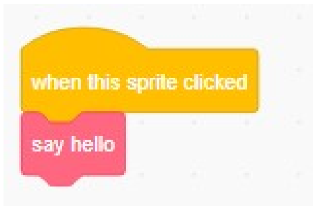
- Give your new block a name and then click **OK**.



- You will see a new `define` block. Attach code to this block.



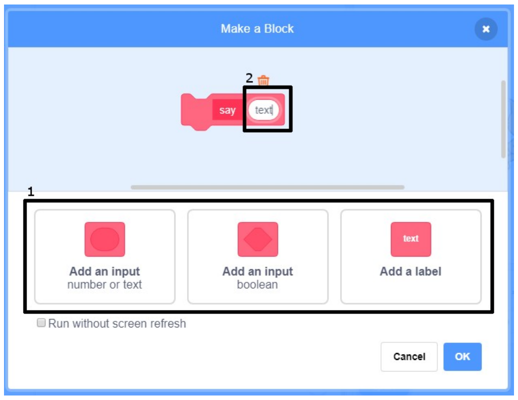
- You can then use your new block just like any normal block.



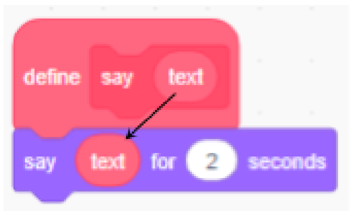
- The code attached to your new `define` block is run whenever the block is used.

Making a block with parameters

- You can also create blocks that have ‘gaps’ for adding data. These ‘gaps’ are called ‘parameters’. To add parameters, first make a new block, and then click on the the options below to choose the type of data you want to add. Then give your data a name, and click **OK**.

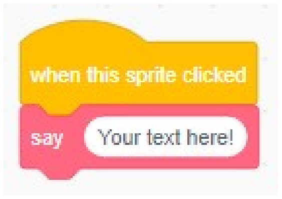


- You will see a new `define` block as usual, except that this one contains the data gap you added and which you gave a name.



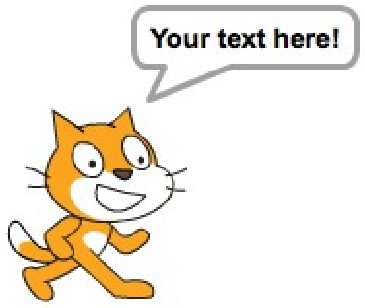
```
define say (text)
say (text) for (2) seconds
```

- You can then use your new block, filling in data in the gap.

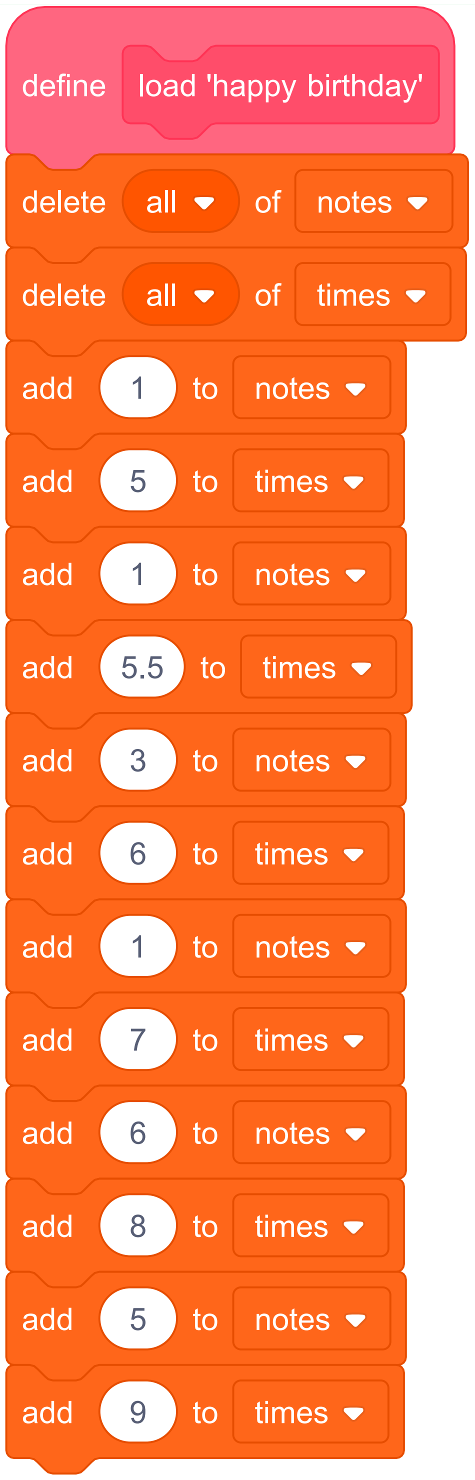


```
when this sprite clicked
say (Your text here!)
```

- As usual, the code attached to your new `define` block is run whenever the block is used.

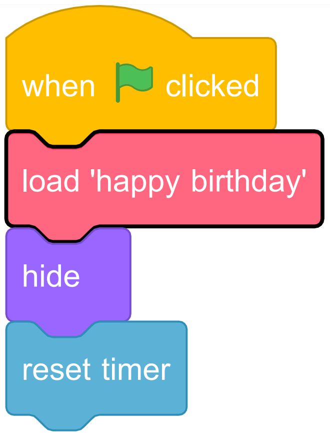


This is what your code should look like:



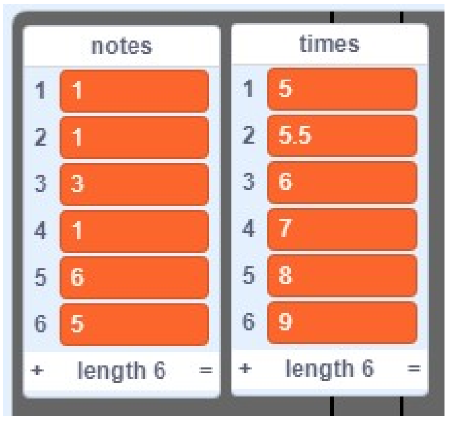
```
define load 'happy birthday'
delete all of notes
delete all of times
add 1 to notes
add 5 to times
add 1 to notes
add 5.5 to times
add 3 to notes
add 6 to times
add 1 to notes
add 7 to times
add 6 to notes
add 8 to times
add 5 to notes
add 9 to times
```

Test your new block by running it at the start of your project.



```
when 🏁 clicked
load 'happy birthday'
hide
reset timer
```

Each of your lists should now contain six numbers.



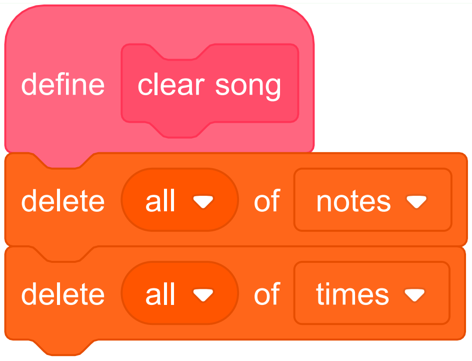
	notes		times
1	1	1	5
2	1	2	5.5
3	3	3	6
4	1	4	7
5	6	5	8
6	5	6	9
+	length 6 =	+	length 6 =

Step 7 More custom blocks

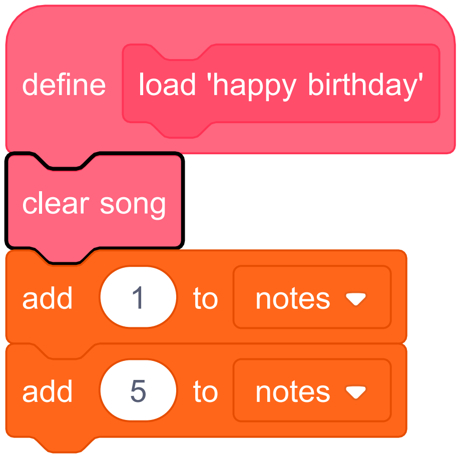
The newest section of code is difficult to read, so you’re going to use more custom blocks to make it simpler.

Make another block called `clear song` that deletes all items from both lists. Use this block before adding numbers back intothe lists.

● ●



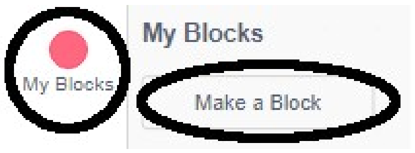
When you test your code, it should work just as it did before.



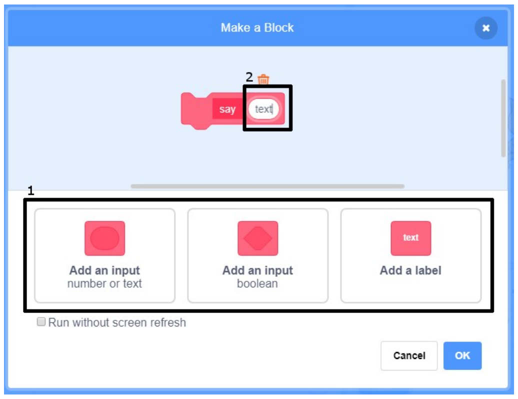
So that your code is even easier to read, make another block that allows you to specify a note to be played and a time to play the note at.

Making a block with parameters

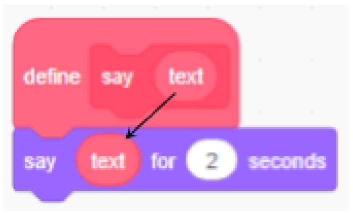
- Click on **My Blocks**, and then click **Make a Block**.



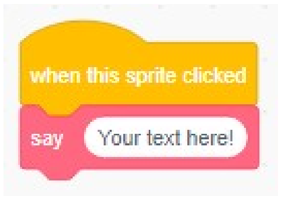
- You can create blocks that have 'gaps' to add data. These 'gaps' are called **parameters**. To add parameters, click on the options below to choose the type of data you want to add. Then give your data a name, and click **OK**.



- You can then define your new block, and use the data by dragging the circular blocks to use them in your code.



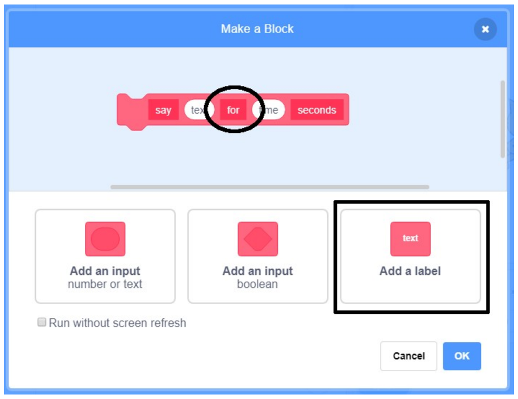
- Now you add data as parameters into the gaps of your new block.



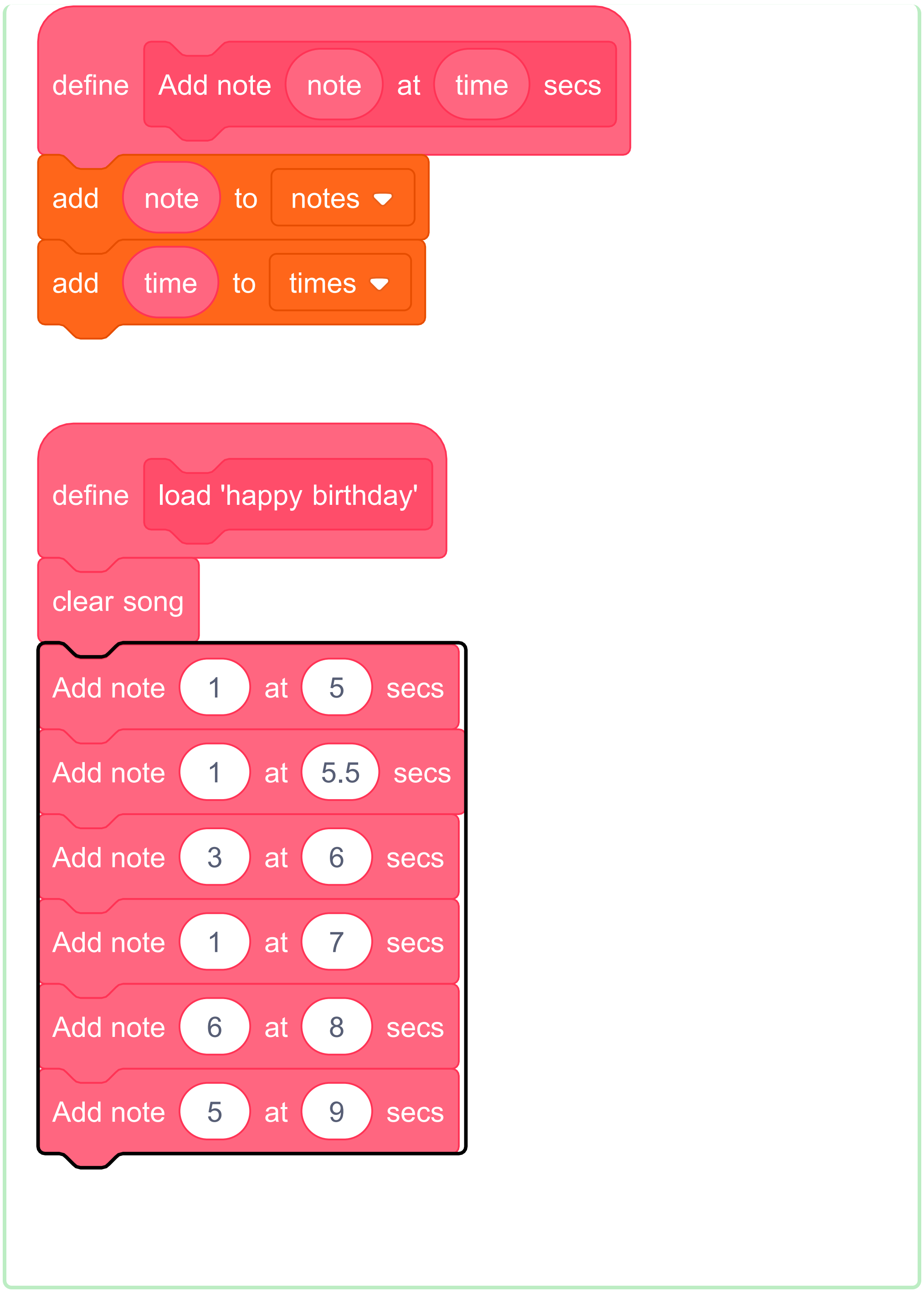
- Use the new `define` block with the gaps you have filled in by attaching code to it and adding it to your script.

Your text here!

- If you want to add some text between parameters, you can add label text:



This is what your code should look like:



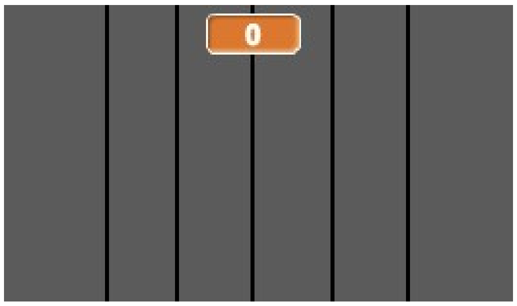
```
define Add note (note) at (time) secs
add (note) to [notes v]
add (time) to [times v]

define load 'happy birthday'
clear song
Add note (1) at (5) secs
Add note (1) at (5.5) secs
Add note (3) at (6) secs
Add note (1) at (7) secs
Add note (6) at (8) secs
Add note (5) at (9) secs
```

Step 8 Keep a score

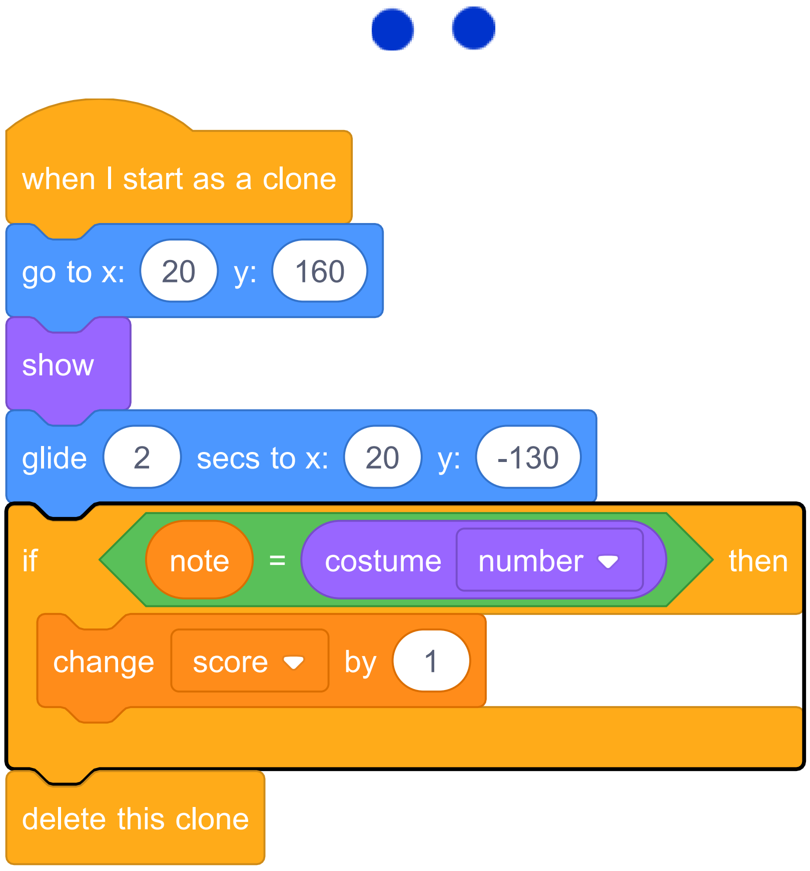
Improve your game by giving the player points for playing the correct note.

Create a new variable called `score`, and place it at the top of your Stage.



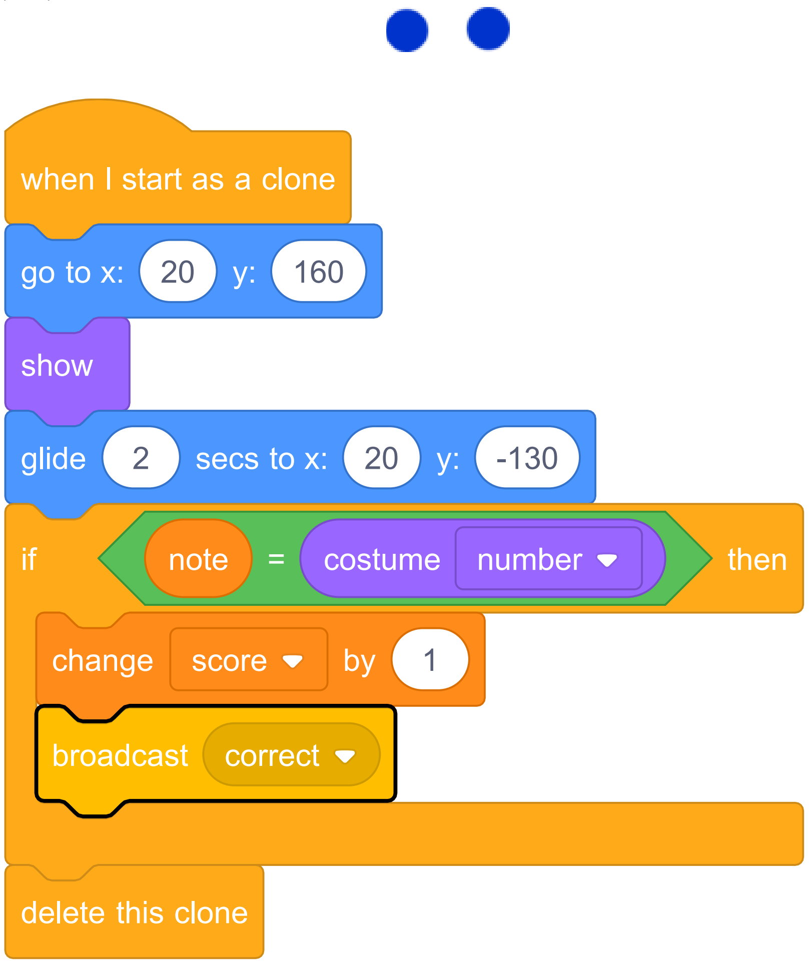
Add to `score` whenever the player plays the correct note at the correct time. Remember to set `score` to 0 at the start of the game.

This is what your code should look like:

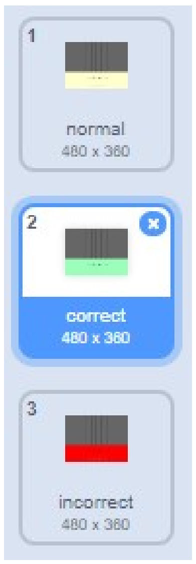


```
when I start as a clone
go to x: (20) y: (160)
show
glide (2) secs to x: (20) y: (-130)
if <(note) = (costume [number v])> then
    change [score v] by (1)
end
delete this clone
```

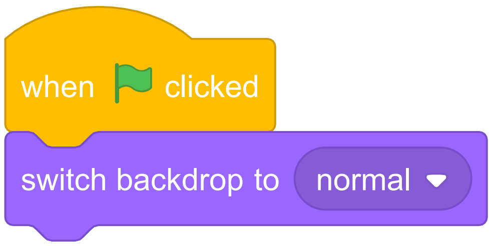
Broadcast a message called ‘correct’ when the correct note is played.



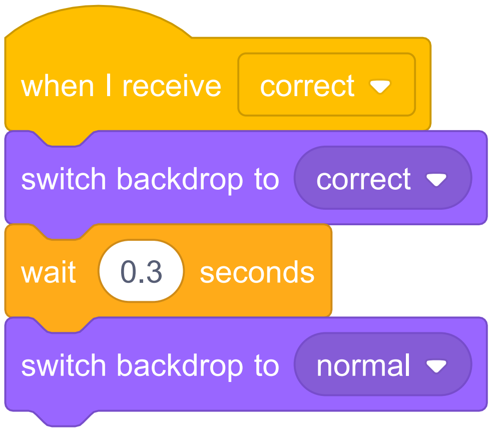
```
when I start as a clone
go to x: (20) y: (160)
show
glide (2) secs to x: (20) y: (-130)
if <(note) = (costume [number ▾])> then
    change [score ▾] by (1)
    broadcast [correct ▾]
end
delete this clone
```



Add code to your Stage to briefly change the backdrop when the player plays the correct note. The project already contains a second backdrop for this.



```
when green flag clicked
switch backdrop to (normal v)
```



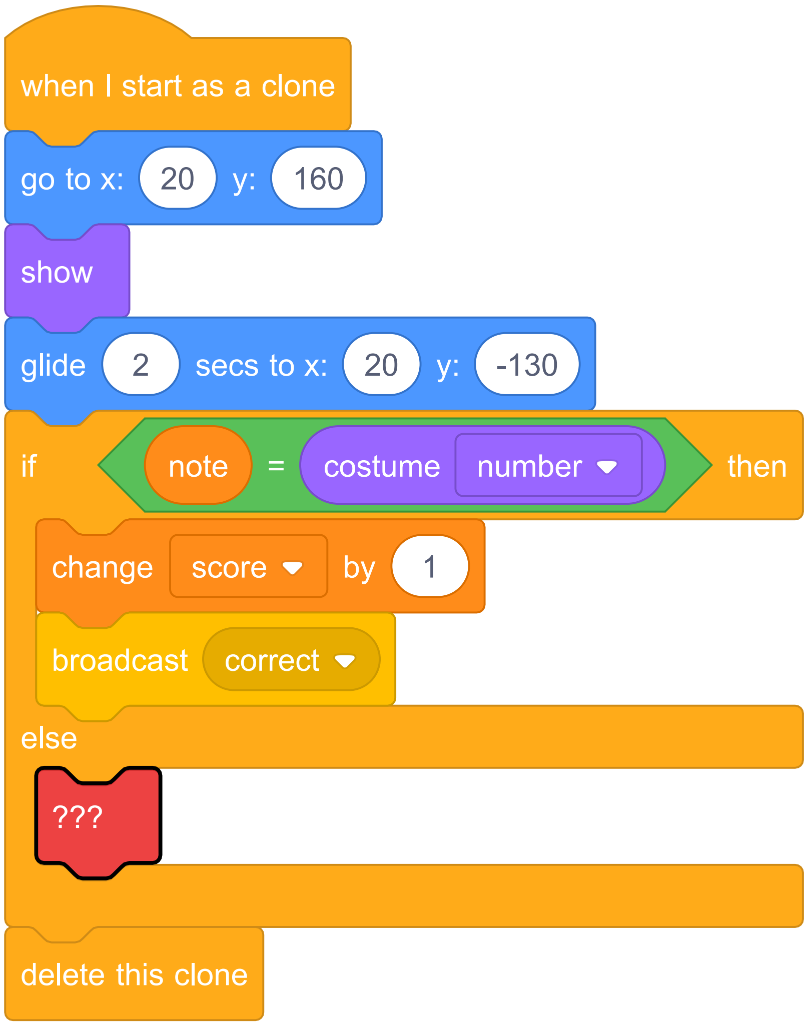
```
when I receive [correct v]
switch backdrop to (correct v)
wait (0.3) seconds
switch backdrop to (normal v)
```

Challenge!

Challenge: take it further

Your game is done now, but there are a few things you can do to make it even better if you want to!

For example, can you add code to change how the Stage looks if the correct note is not played?



```
when I start as a clone
go to x: (20) y: (160)
show
glide (2) secs to x: (20) y: (-130)
if <(note) = (costume [number v])> then
    change [score v] by (1)
    broadcast [correct v]
else
    ???
end
delete this clone
```

To do this, you need to add code that's very similar to the code that changes the backdrop when the correct note is played. The project contains another backdrop you can use.

Step 9 What next?

Try these other projects to build you knowledge of other programming languages.

- **About me**
  [(https://projects.raspberrypi.org/en/projects/about-me?](https://projects.raspberrypi.org/en/projects/about-me?)

utm_source=pathway&utm_medium=whatnext&utm_campaign=projects)
is a great introduction to Python.

- **Happy birthday** (https://projects.raspberrypi.org/en/projects/happy-birthday?utm_source=pathway&utm_medium=whatnext&utm_campaign=projects) introduces HTML and CSS.