Raspberry Pi Projects

Binary Hero

Introduction

Make a game in which you play the notes of a song as they scroll down the stage.

What you will make

Click the green flag to play. Use z, x, c and v to play the correct notes when they reach the bottom of the stage. You'll score ten points for every correct note you play.

What you will learn

This project covers elements from the following strands of the Raspberry Pi Digital Making Curriculum (http://rpf.io/curriculum):

- Combine programming constructs to solve a problem. (https://www.raspberrypi.org/curriculum/programming/builder/)

What you will need

Hardware

- Computer capable of running Scratch 2.0

Software

- Scratch 2.0 (either online (https://scratch.mit.edu/projects/editor/) or offline (https://scratch.mit.edu/scratch2download/))

Binary notes

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

- Open the starter project.

I'm using Scratch online

Open the 'Binary Hero' Scratch starter project at jumpto.cc/binary-go (http://jumpto.cc/binary-go).

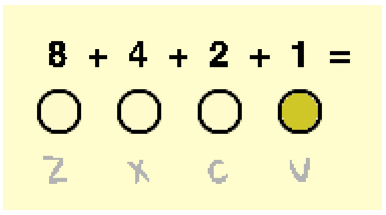
I'm using Scratch offline

Download the binary-hero.sb2 (https://s3.eu-west-2.amazonaws.com/learning-resources-production/projects/binary-hero/ad928db99a85cd729e8ab20839aecbb5640d36de/en/resources/binary-hero.sb2) Scratch starter project and open it using the offline editor.

If you have a Scratch account, you can click on **Remix** in the top right-hand corner to save a copy of the project to your account.

- Let's start by showing which keys have been pressed. Click on the sprite called '1', and add code to change its costume when the v key is pressed.

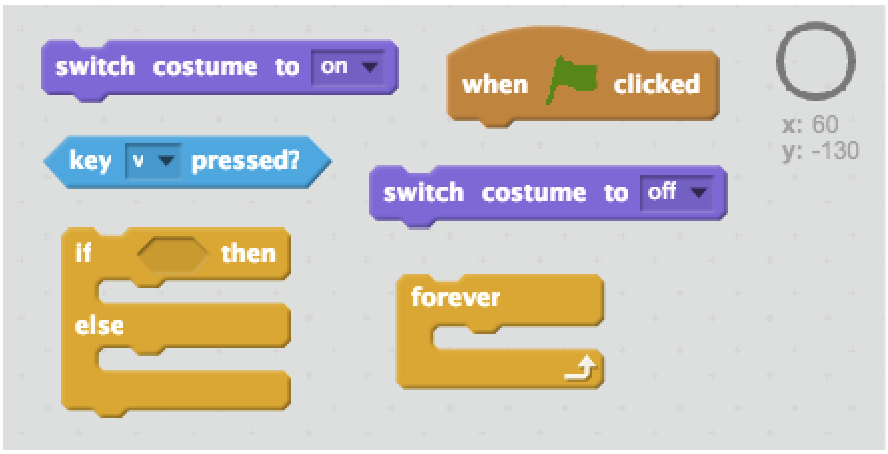
When you test your sprite by pressing the v key, the sprite should light up.



I need a hint

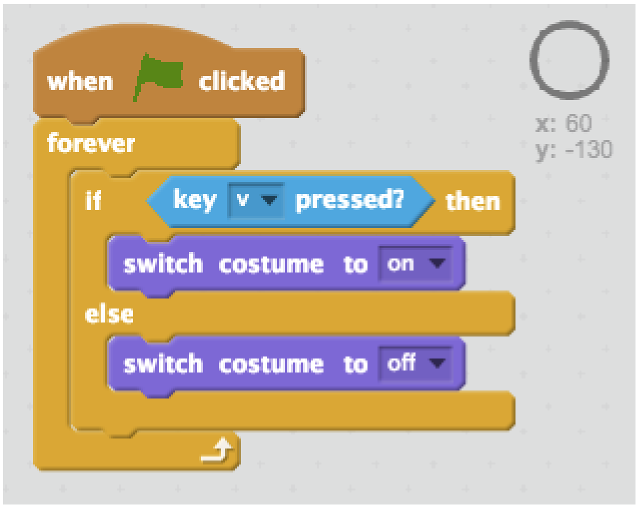
When the `flag is clicked`, your sprite should check `forever` whether the `v key is pressed`. `If` the key is pressed, the ‘on’ `costume` should be shown, `else` the ‘off’ `costume` should be shown.

Here are the code blocks you’ll need:



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

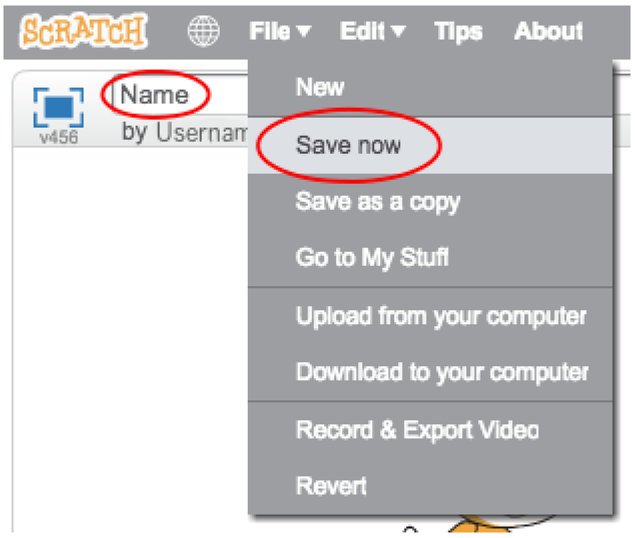
This is what your code should look like:



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

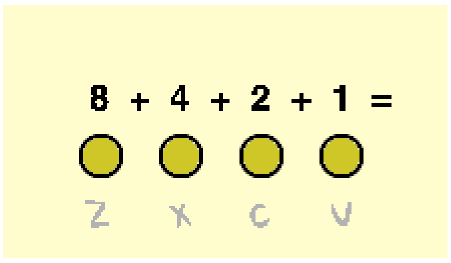
Saving a Scratch project

- Give your program a name by typing into the text box in the top-left corner.
- You can click **File** and then **Save now** to save your project.



- **Note:** if you're using Scratch online but don't have a Scratch account, you can save a copy of your project by clicking **Download to your computer** instead.

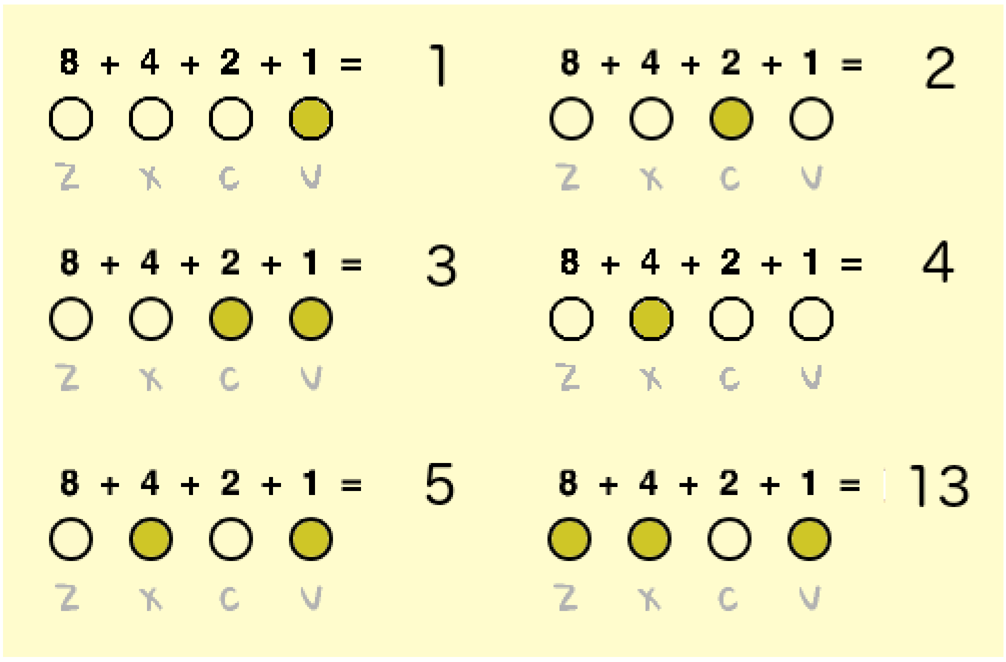
- Do the same for the other three sprites, so that they light up when the `c`, `x`, and `z` keys are pressed.



Binary numbers

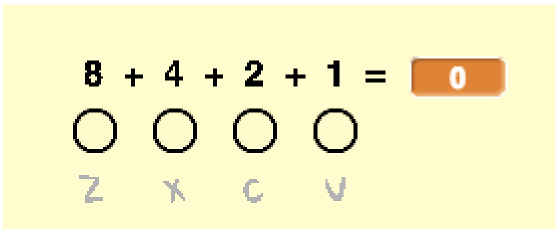
In this project you'll be using combinations of the four different keys to play different notes. You can think of each of the keys as either on (pressed) or off (not pressed). This means that you can think of each combination of keys as a **binary number**.

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



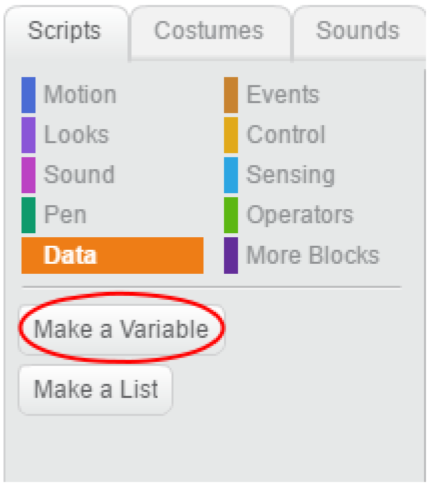
There are 2^4 = **16 combinations** that can be made with the four keys. This means that we can play 15 different notes, as `0` will mean that no note is played.

- Create a new variable called `note`, and drag it next to the 4 note sprites.

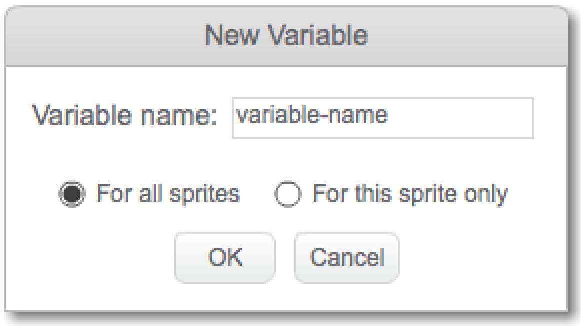


Add a variable in Scratch

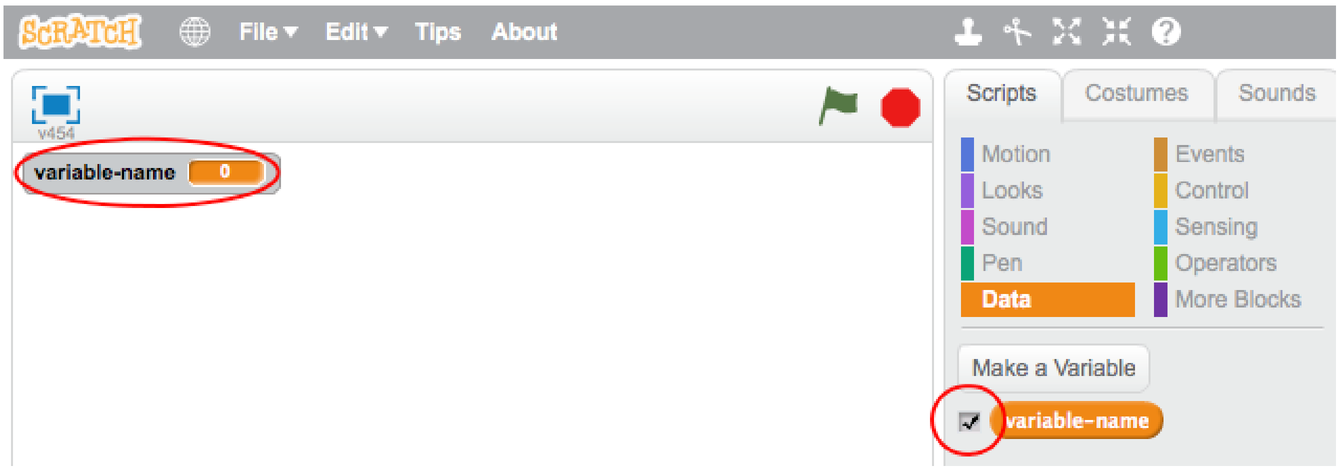
- Click on “Data” in the Scripts tab, then click on “Make a Variable”



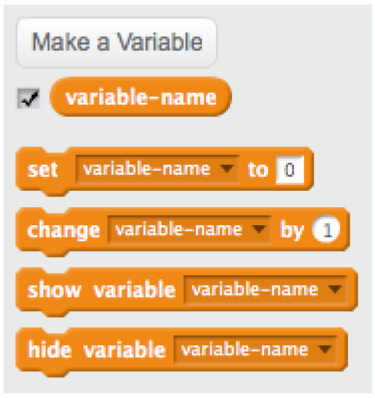
- Type in the name of your variable. You can choose whether you would like your variable to be available to all sprites, or to only this sprite. Press OK.



- 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.

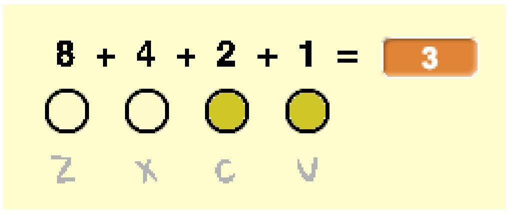


- New blocks will appear and allow you to change the value of the variable.


Make a Variable
variable-name
set variable-name to 0
change variable-name by 1
show variable variable-name
hide variable variable-name

- Add code to the Stage to use the keys pressed to calculate the value of the note to be played.

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



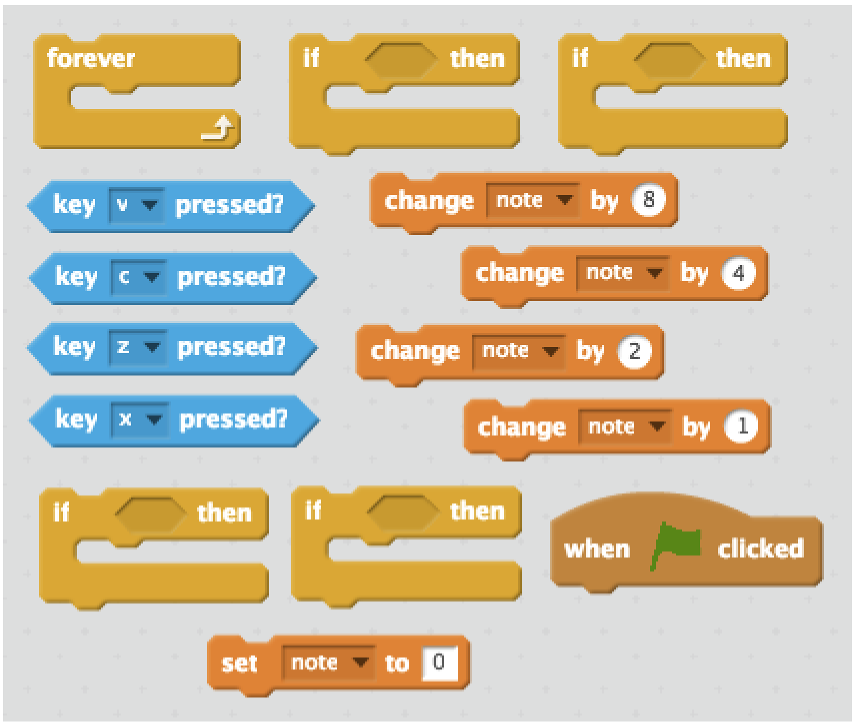
I need a hint

When the `flag is clicked`, the `note` variable should be `set` to `0`.

- `If` the `v key is pressed`, the `note` should be `changed by 1`
- `If` the `c key is pressed`, the `note` should be `changed by 2`
- `If` the `x key is pressed`, the `note` should be `changed by 4`
- `If` the `z key is pressed`, the `note` should be `changed by 8`

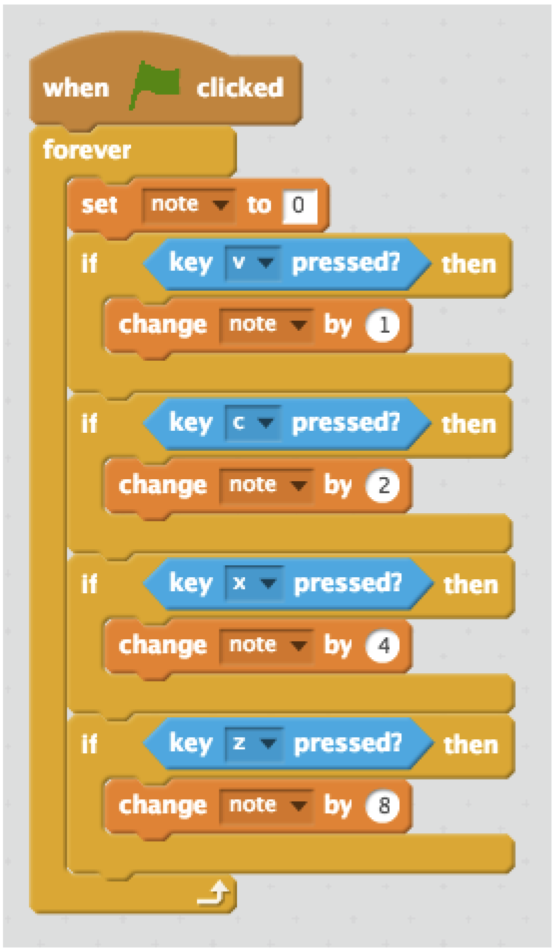
All of this code should be repeated `forever`.

Here are the code blocks you'll need:



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

This is what your code should look like:

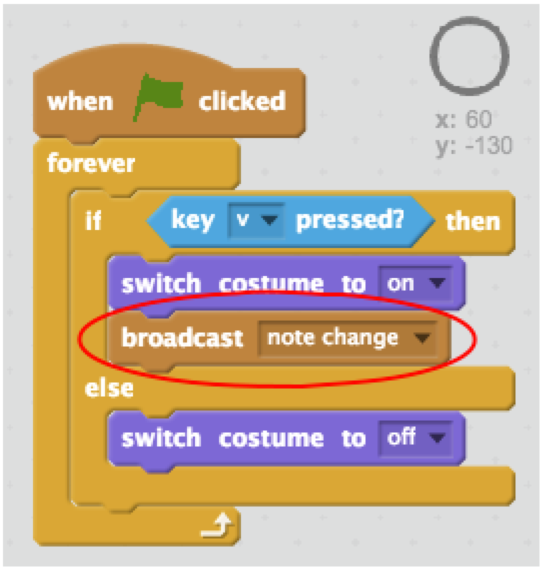


```
when flag clicked
forever
    set note to 0
    if <key v pressed?> then
        change note by 1
    if <key c pressed?> then
        change note by 2
    if <key x pressed?> then
        change note by 4
    if <key z pressed?> then
        change note by 8
```

Playing notes

Let’s play notes when keys are pressed.

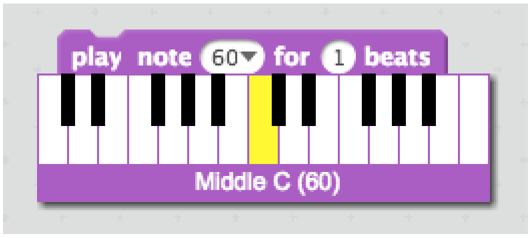
- Broadcast a ‘note change’ message whenever `each of the four keys` is pressed.



```
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.

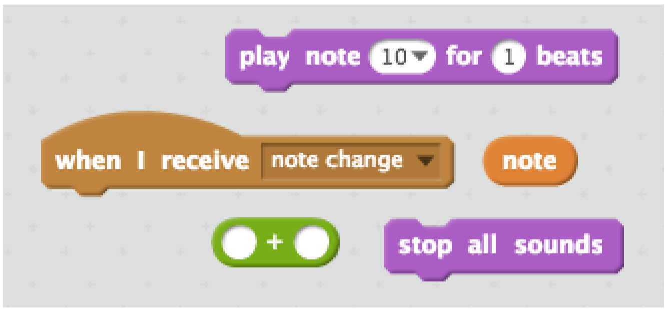


I need a hint

When your stage `receives` the ‘change note’ message, it should `stop all sounds` before `playing a note` based on the value of your `note` variabe.

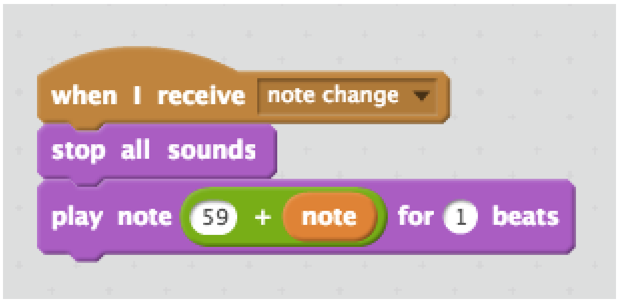
- When your `note` variable is `1`, note 60 should be played
- When your `note` variable is `2`, note 61 should be played
- When your `note` variable is `3`, note 62 should be played
- etc…

Here are the code blocks you'll need:



```
play note (10 v) for (1) beats
when I receive [note change v]
(note)
(() + ())
stop all sounds
```

This is what your code should look like:



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

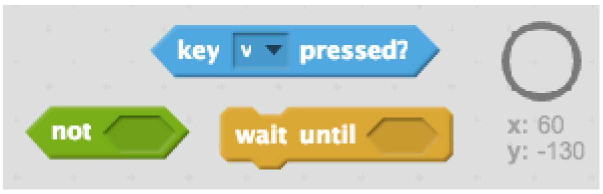
- Test your code. You'll notice that the note is repeatedly played when a key is held down.

- Can you add code so that the key sprites only play a note `once` when a key is held down?

I need a hint

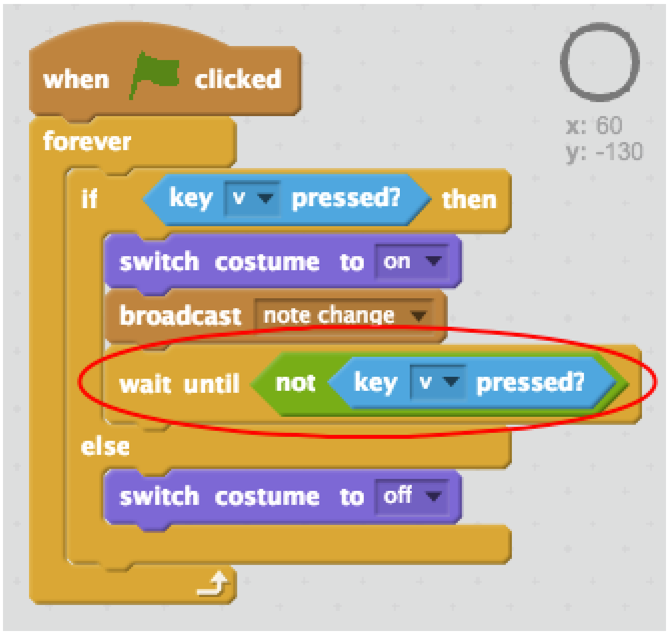
When each of the `z`, `x`, `c` and `v` keys is pressed, your code should `wait until` the `key is not pressed` before continuing.

Here are the code blocks you'll need:



```
key [v v] pressed?
not < >
wait until < >
```

This is what your code should look like:



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

Scrolling notes

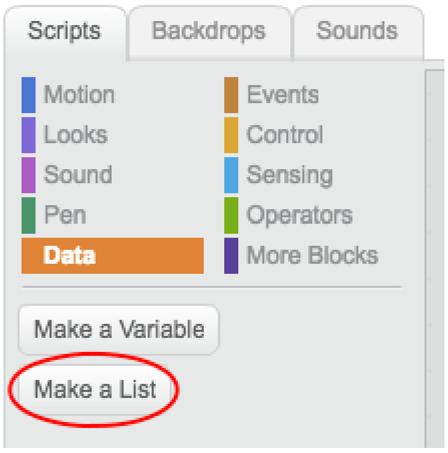
Scroll notes down the stage so that the player knows which keys to press.

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

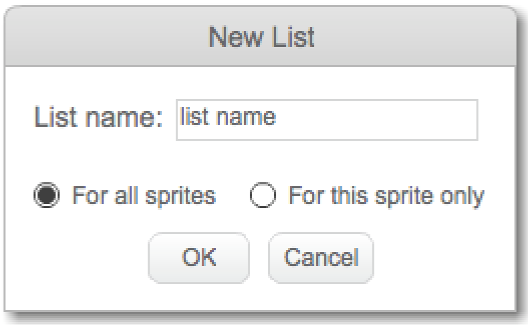
Make a list

Make a list

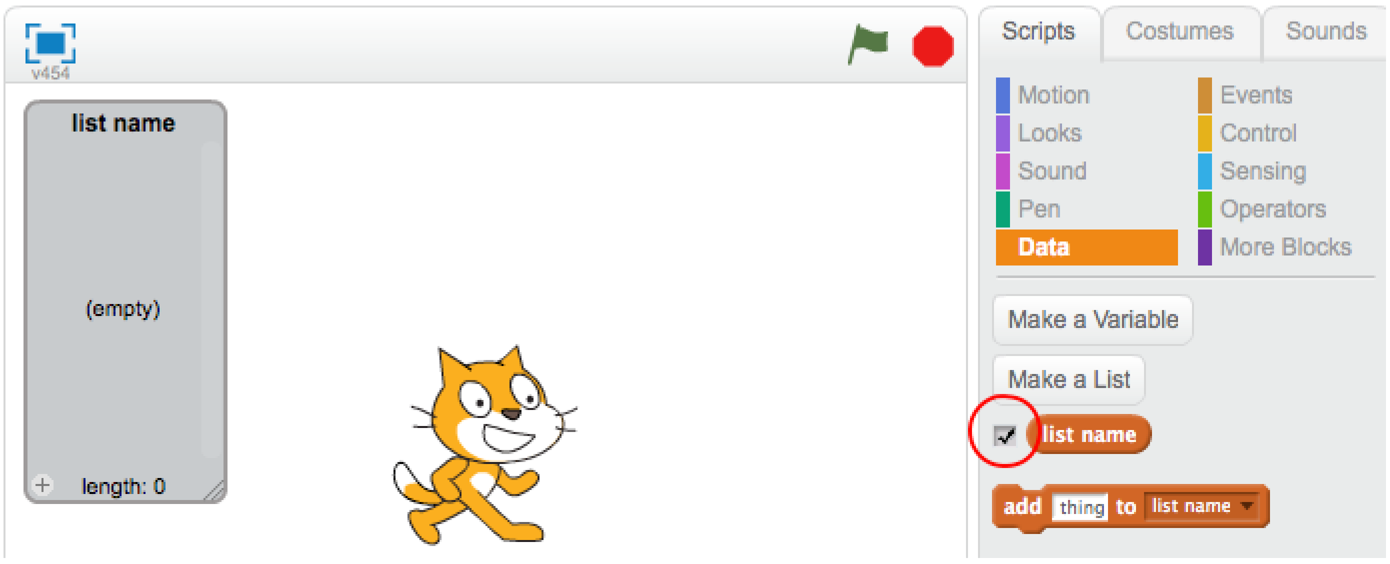
- Click on **Data** in the **Scripts** tab, then click on **Make a List**.



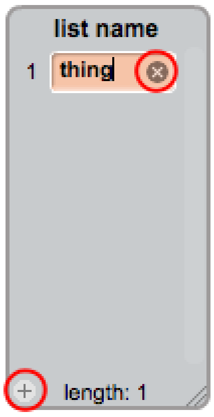
- Type in the name of your list. You can choose whether you would like your list to be available to all sprites, or to only a specific sprite. Press **OK**.



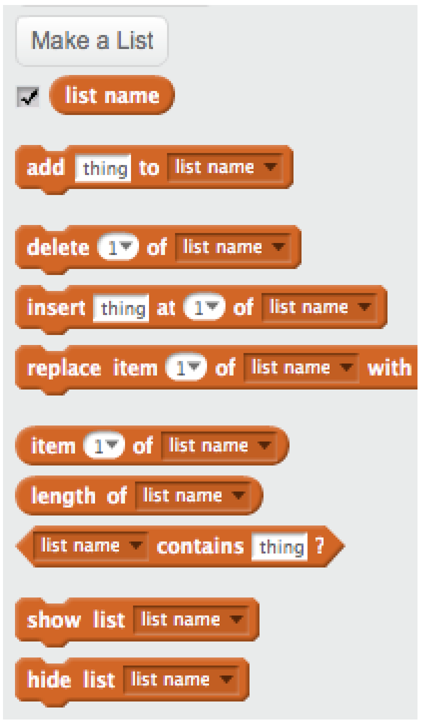
- Once you have created the list, it will be displayed on the stage, or you can untick the list in the **Scripts** tab to hide it.



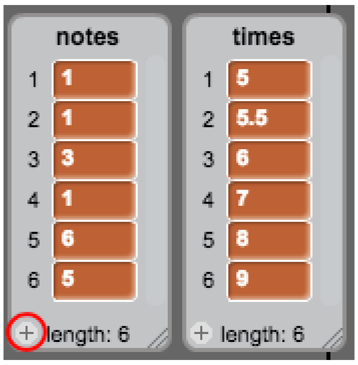
- Click the **+** at the bottom of the list to add items, and click the cross next to an item to delete it.



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



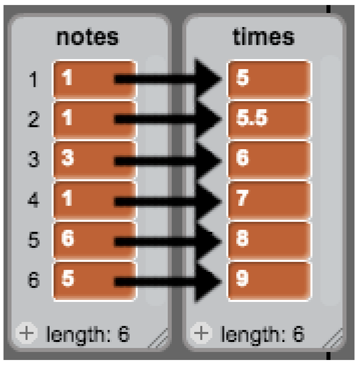
- Add the following numbers to your `notes` and `times` lists. Note: make sure to add these exact numbers in the right order.



notes	times
1	5
1	5.5
3	6
1	7
6	8
5	9
length: 6	length: 6

- Here’s how songs will be stored:

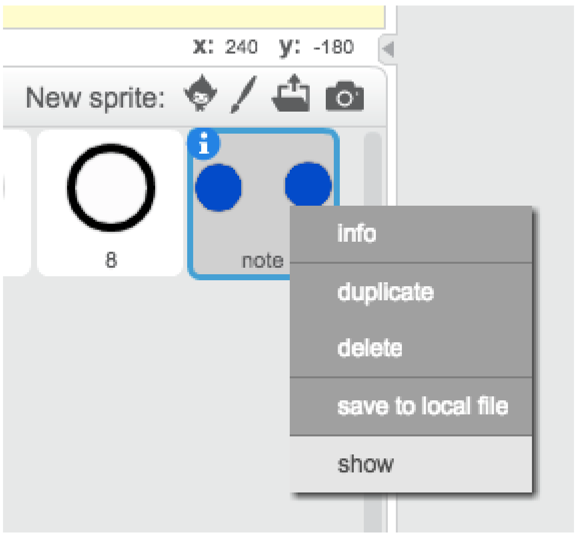
  - The `notes` list stores the notes of the song, in order (from 1 to 15).
  - The `times` list is used to store the times that each note is played.



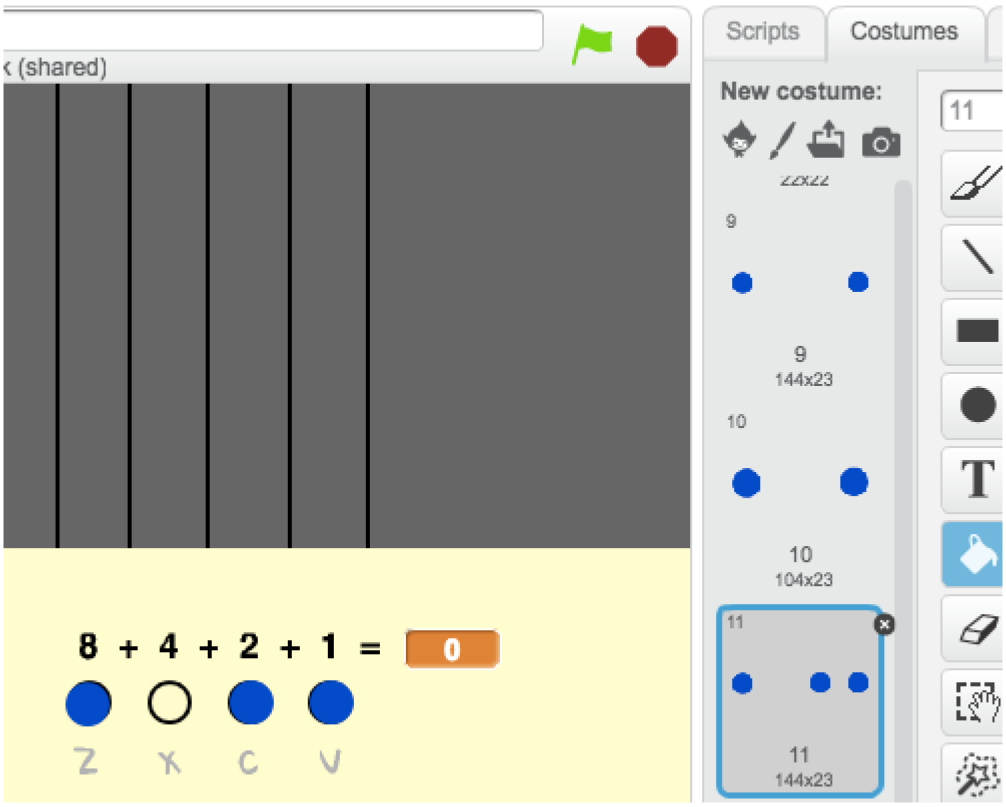
notes	times
1	5
1	5.5
3	6
1	7
6	8
5	9
length: 6	length: 6

  So for the data above:

  - 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…
- Right-click on the ‘note’ sprite and click **show**.



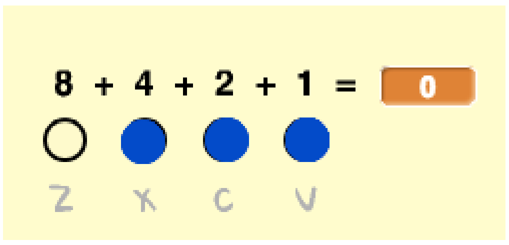
- If you click **Costumes**, you should see that the sprite has 15 different costume, one for each note.



- Add code to create a clone of the ‘note’ sprite for every note to be played. Each clone should be created two seconds before the time the note should be played, and should then move down the stage.

This will give the clone two seconds to move down the screen. You'll create the code to move your clones in a little bit!

Nothing will seem to happen when you test your code, 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.



I need a hint

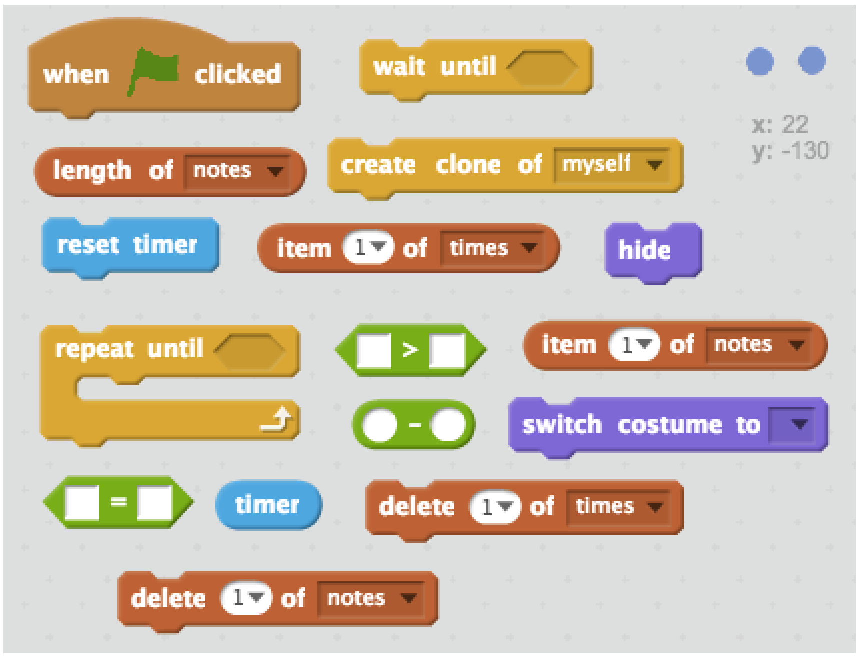
When the `flag is clicked` the 'note' sprite should `hide`, and the `timer` should be `reset`.

You should then `wait until` the timer is `greater than` the next note to be played, which will be the `time` at the `start of the list` (`minus 2 seconds`).

The costume for the 'note' sprite should then be set to the next `note` to be played (the `note` at the start of the list), before a `clone` of the note sprite is created.

The items at the start of the `notes` and `times` lists can then be `deleted`, and the entire process should be `repeated until` there are no notes left.

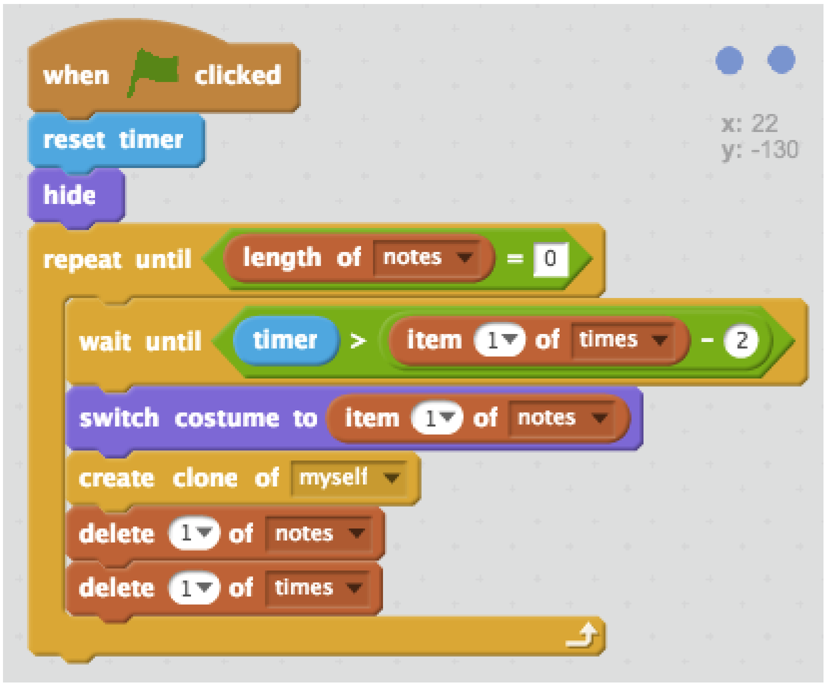
Here are the code blocks you'll need:



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

x: 22
y: -130

This is what your code should look like:



```
when 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 [notes v]
    delete (1 v) of [times v]
end
```

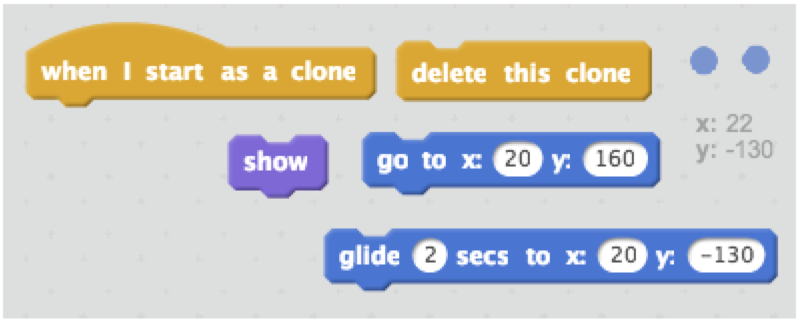
x: 22
y: -130

- Now add code to make each note clone glide from the top to the bottom of the screen before being deleted.

I need a hint

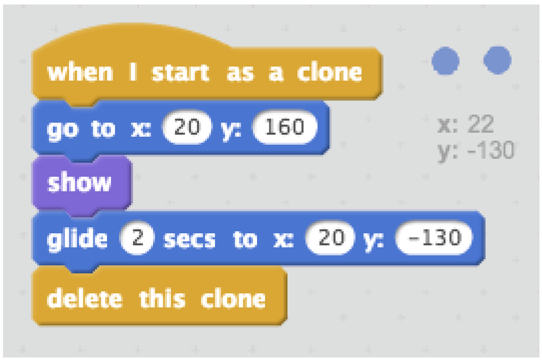
When each `clone starts`, it should be `shown` and should `go to` the top of the screen. The clone should then `glide` for `2 seconds` until it reaches the four 'key' sprites, at which point the clone can be `deleted`.

Here are the code blocks you'll need:



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

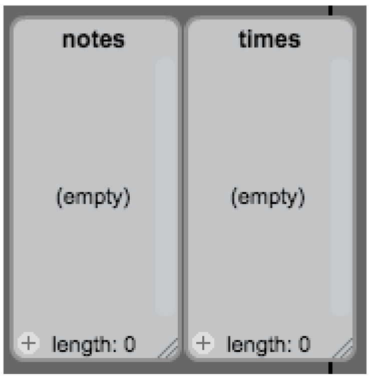
This is what your code should look like:



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

Storing songs

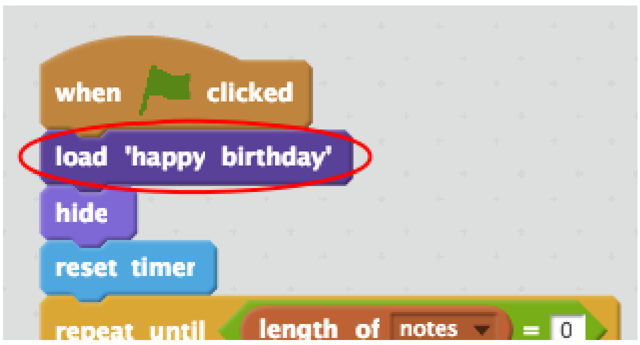
You've made it so that notes are removed from the lists once they've been played, so you'll be left with empty lists:



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.

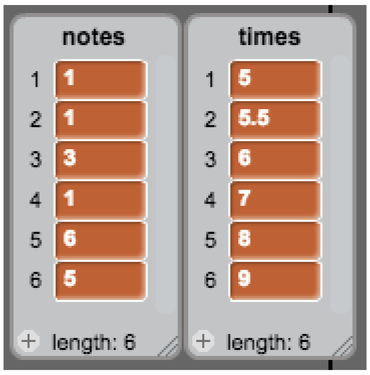
- Make a block called `load 'happy birthday'` that clears both the `notes` and `times` lists, and then adds the numbers back into both lists.

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



```
when flag clicked
load 'happy birthday'
hide
reset timer
repeat until <(length of notes) = 0>
```

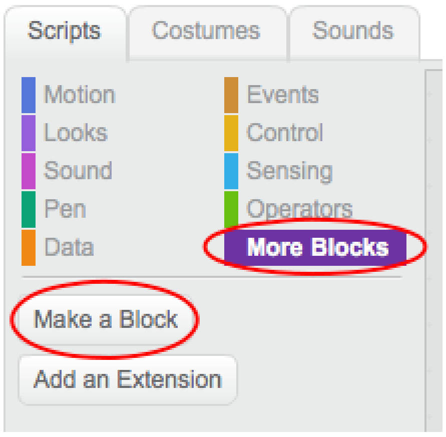
Each of your lists should now contain five numbers.



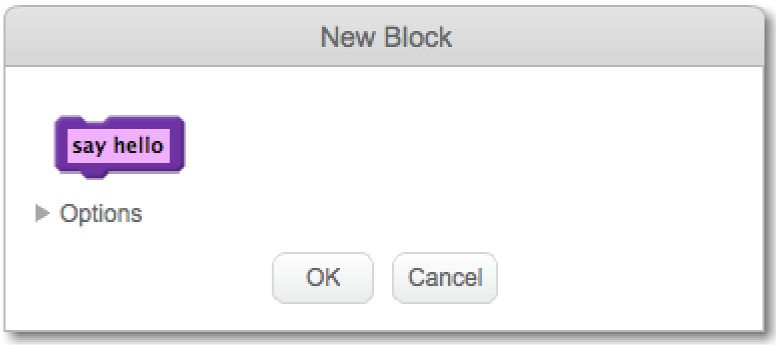
	notes		times
1	1	1	5
2	1	2	5.5
3	3	3	6
4	1	4	7
5	6	5	8
6	5	6	9
	length: 6		length: 6

Making a block

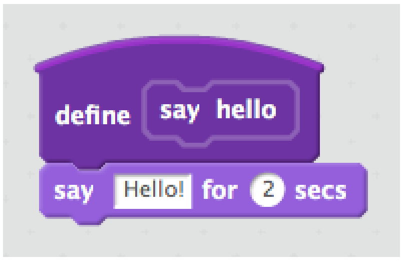
- Click the **Scripts** tab, then on **More Blocks**, and then click **Make a Block**.



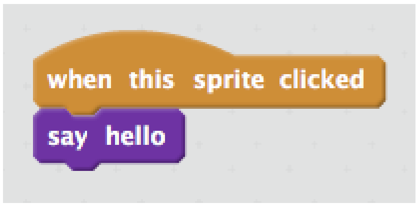
- Give your new block a name and then click **OK**.



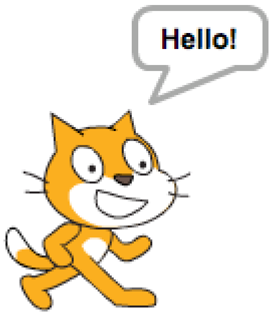
- You will see a new `define` block. Attach code to this block.



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



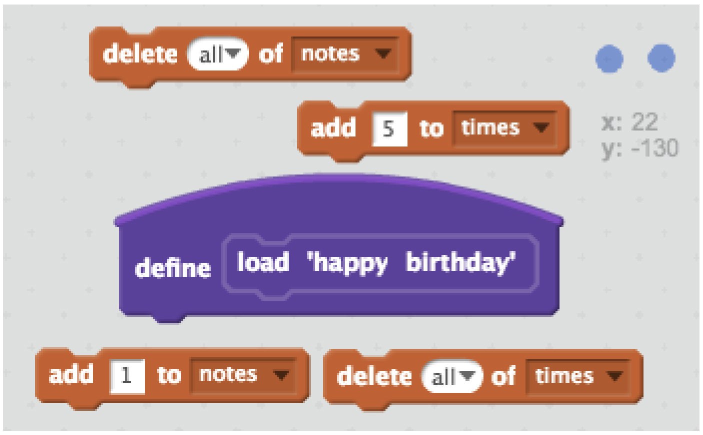
- The code attached to your new `define` block is run whenever the block is used.



I need a hint

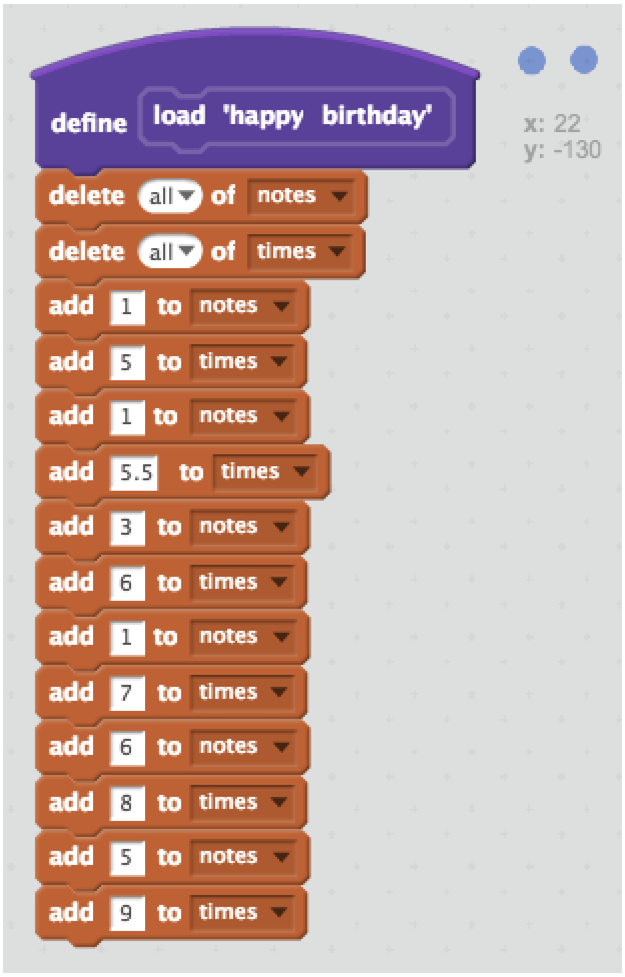
When your new block is run, `all items` should be deleted from both the `notes` and `times` lists. Each of the five numbers should then be `added` to both lists.

Here are the code blocks you'll need:



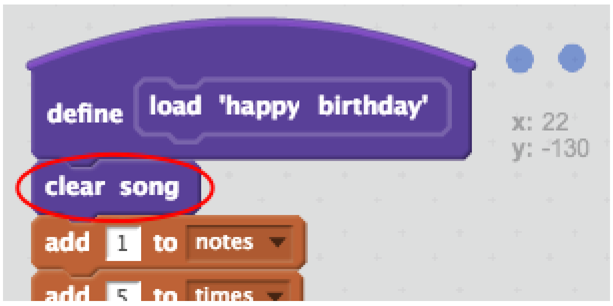
```
delete all of notes
add 5 to times
define load 'happy birthday'
add 1 to notes
delete all of times
```

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
```

- The code above is difficult to read. Make another block called `clear song`, which deletes all items from both lists. Use this block before adding to the lists.



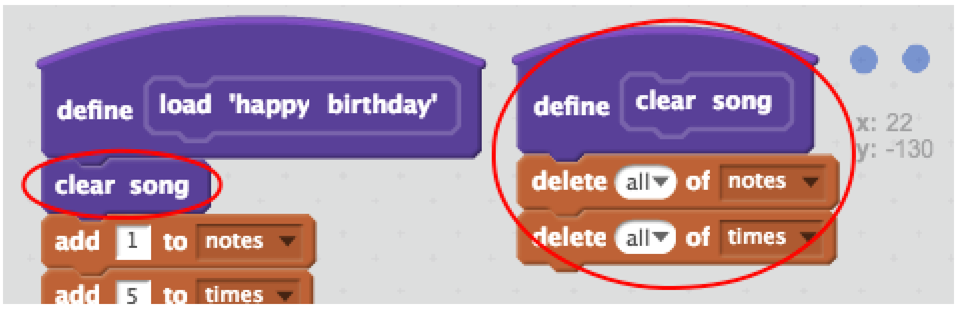
```
define load 'happy birthday'
clear song
add 1 to notes
add 5 to times
```

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

I need a hint

Make a block called 'clear song' that `deletes` all items from both lists.

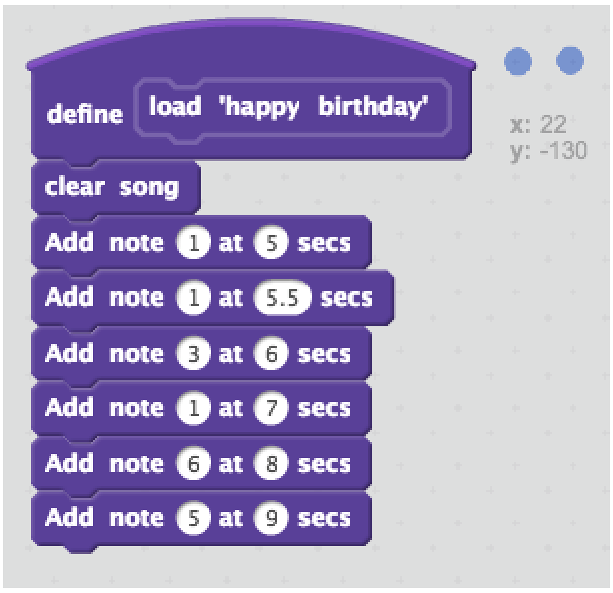
This is what your code should look like:



```
define load 'happy birthday'
clear song
add 1 to notes
add 5 to times
```

```
define clear song
delete all of notes
delete all of times
```

- You could make your code even easier to read by making another block which allows you to specify a note to be played along with a time.

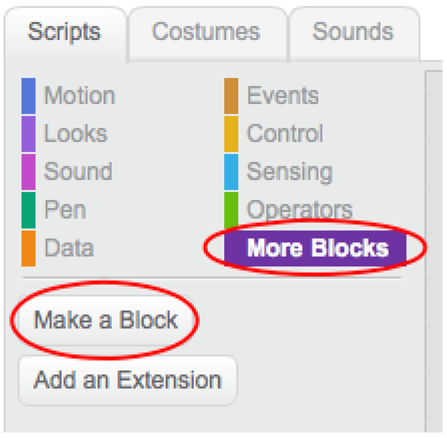


```
define load 'happy birthday'
clear song
Add note 1 at 5 secs
Add note 1 at 5.5 secs
Add note 3 at 6 secs
Add note 1 at 7 secs
Add note 6 at 8 secs
Add note 5 at 9 secs
```

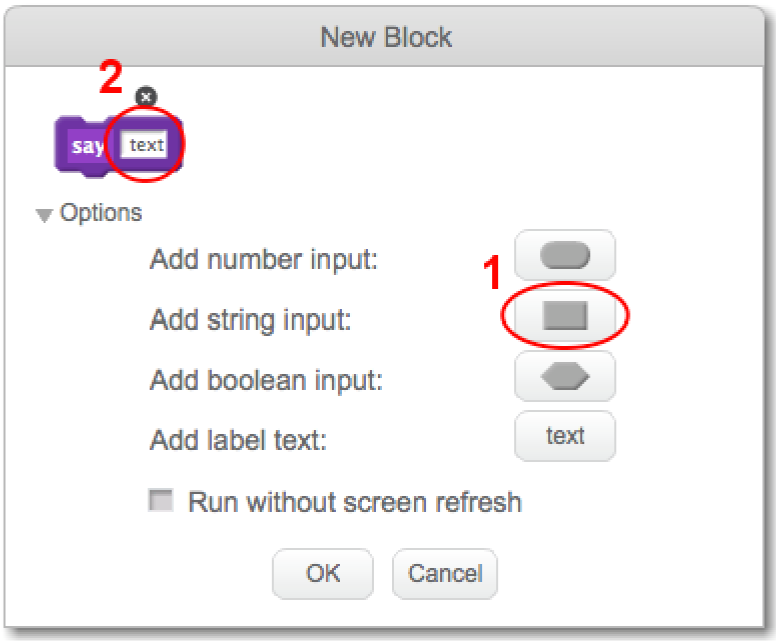
When you test your code, it should work just as it did before.

Making a block with parameters

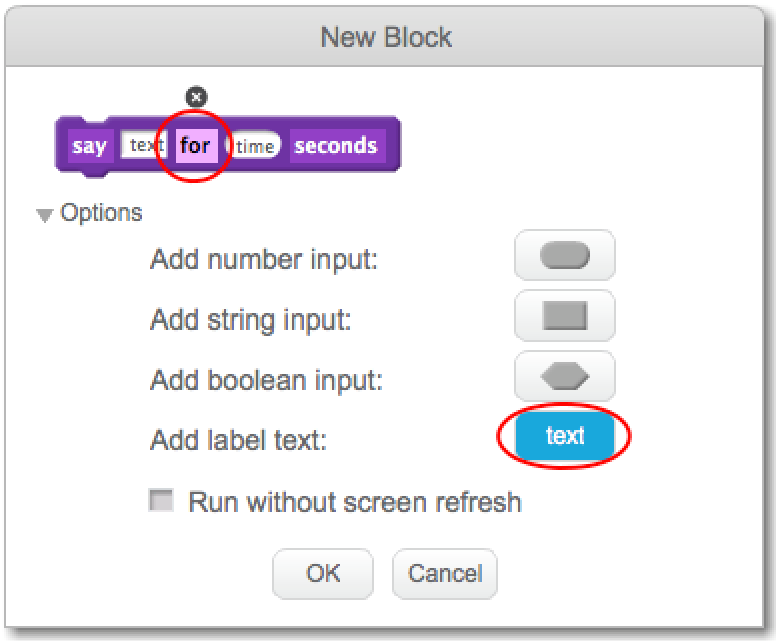
- Click on **More Blocks** in the Scripts tab, then click on **Make a Block**.



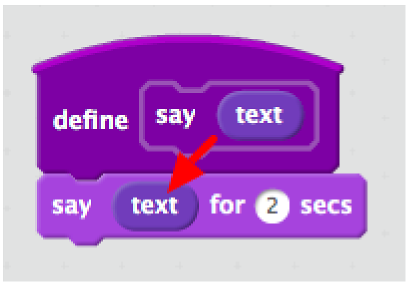
- You can create blocks that have 'gaps' to add data. These 'gaps' are called **parameters**. To add parameters, click **Options**, then click the type of data you want to add and then give your data a name.



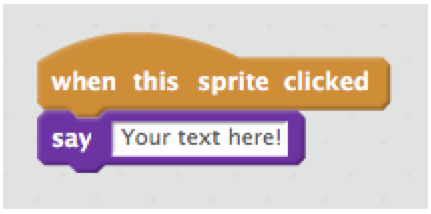
- If you want to add some text between parameters, you can add label text:



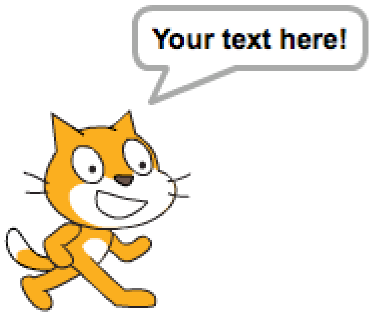
- You can then define your new block, and use the data by dragging the circular blocks to use them in your code.



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



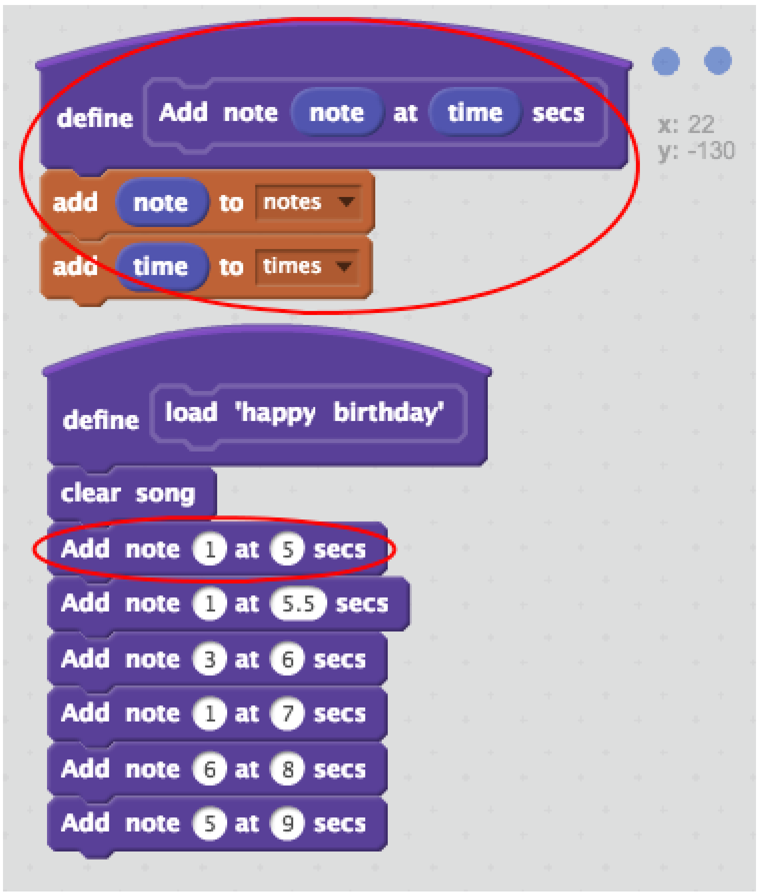
- Use the new `define` block with the gaps you have filled in by attaching code to it and adding it to your script.



I need a hint

Make a block that takes a `note` and a `time` and `adds` both numbers to the lists.

This is what your code should look like:



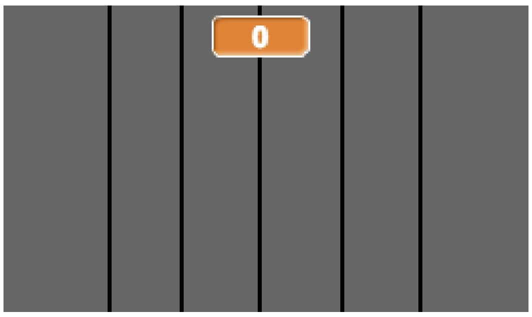
```
define Add note (note) at (time) secs
add (note) to notes
add (time) to times

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
```

Keeping score

Improve your game by giving the player points each time the correct note is played.

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

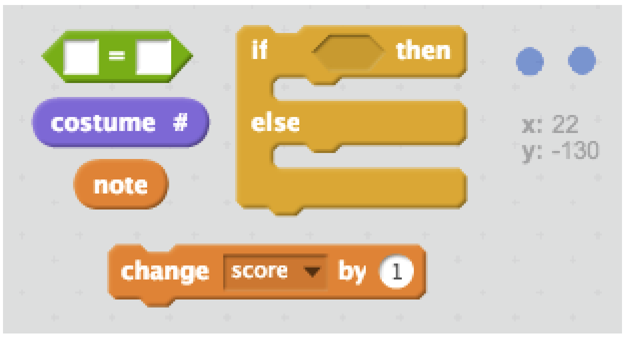


- Add to the player's score whenever they play the correct note at the correct time. Remember to set their score to `0` at the start of the game.

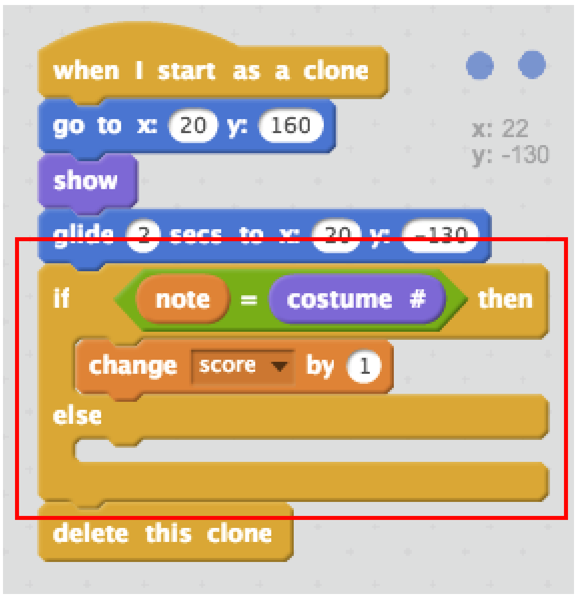
I need a hint

`Before each clone is deleted`, it should check to see `if` the `note` is `equal to` the `costume number`. If they are the same, the score can be `changed`.

Here are the code blocks you'll need:

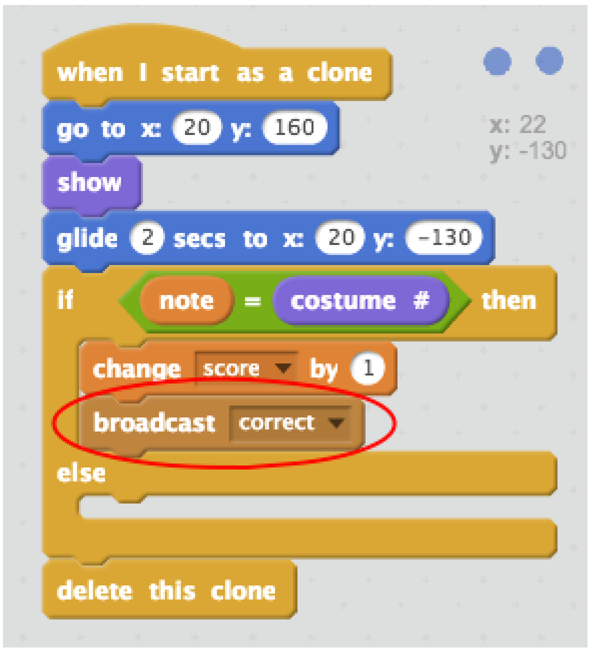


This is what your code should look like:



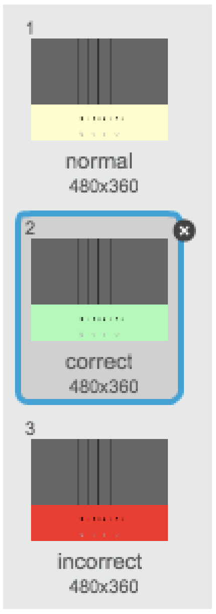
```
when I start as a clone
go to x: (20) y: (160)
show
glide (2) secs to x: (20) y: (-130)
if <(note) = (costume #)> then
    change [score v] by (1)
else
end
delete this clone
```

- 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 #)> then
    change [score v] by (1)
    broadcast [correct v]
else
end
delete this clone
```

- Add code to your Stage to briefly change how it looks when the player plays the correct note. A costume has been provided for you.

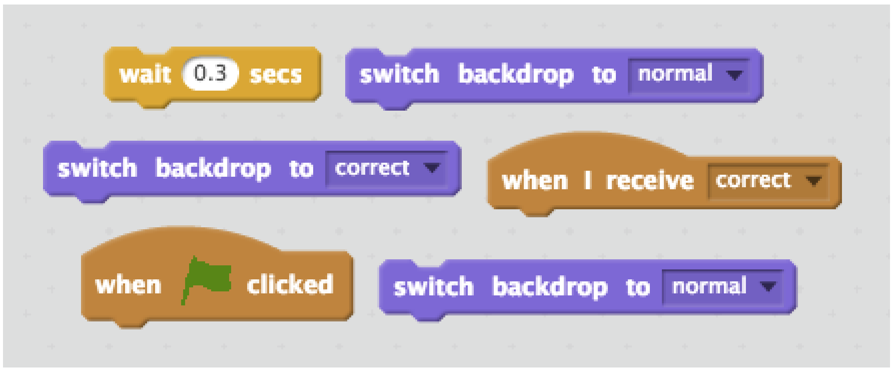


I need a hint

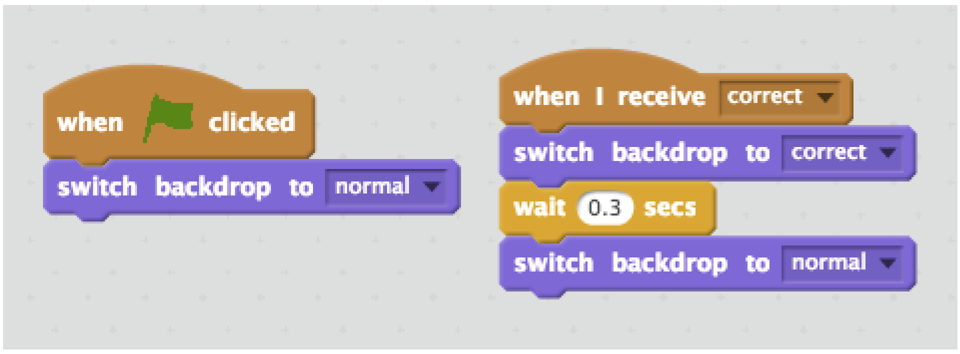
When your stage receives the ‘correct’ message, it should `switch costume`, `wait` for a short time before `switching back`.

You might also need to add code to `set the costume` to normal when the `flag is clicked`.

Here are the code blocks you’ll need:


wait 0.3 secs
switch backdrop to normal
switch backdrop to correct
when I receive correct
when clicked
switch backdrop to normal

This is what your code should look like:



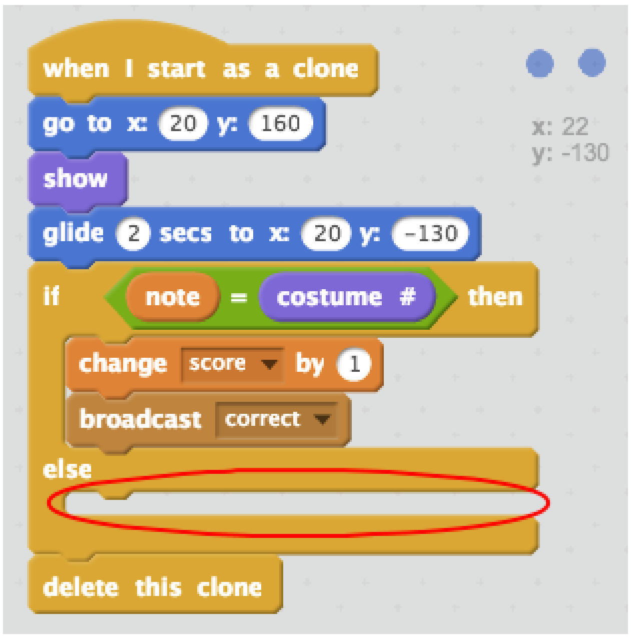
```
when flag clicked
switch backdrop to (normal)

when I receive (correct)
switch backdrop to (correct)
wait (0.3) secs
switch backdrop to (normal)
```

Challenge: incorrect notes

Your Binary Hero game is done now, but there are a few things you could do to make it even better!

For example, can you add code to change how the stage looks if the correct note isn't played?



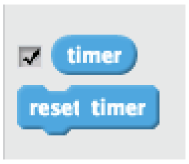
```
when I start as a clone
go to x: (20) y: (160)
show
glide (2) secs to x: (20) y: (-130)
if <(note) = (costume #)> then
    change (score) by (1)
    broadcast (correct)
else

delete this clone
```

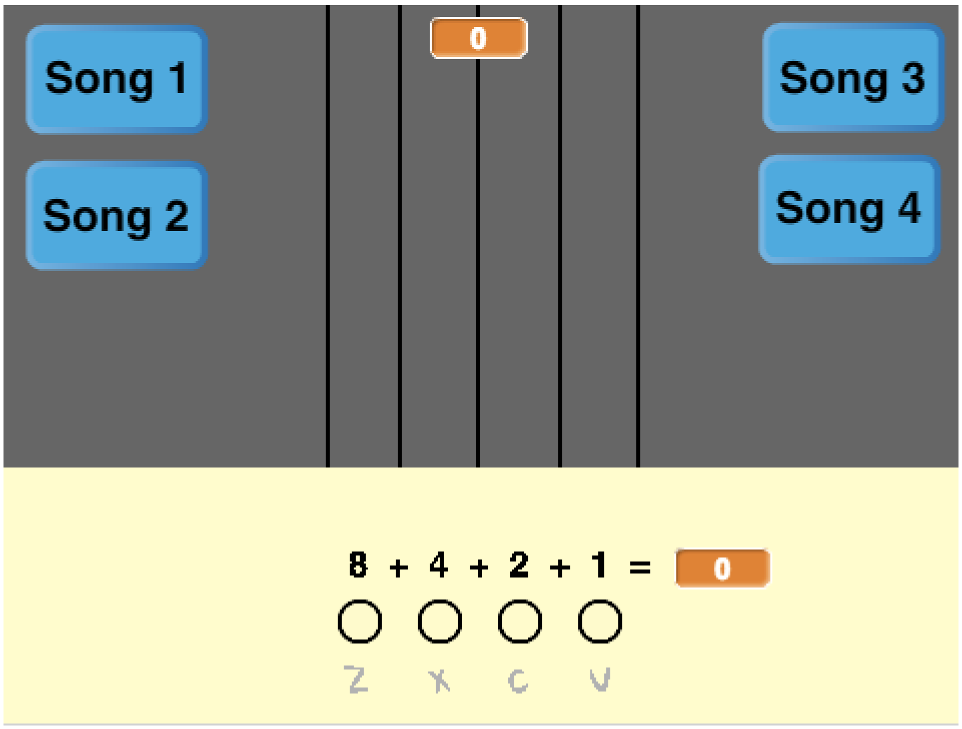
You'll need to add code very similar to the code for when a correct note is played, and a costume has been provided for you.

Challenge: create your own song

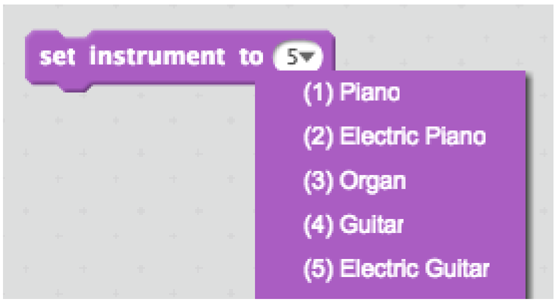
Can you add your own song to the game? Showing the timer will help you get an idea of when the notes of your song should be played.



You could also allow the player to choose a song at the start of the game.



You could even use different instruments for different songs!


set instrument to 5
(1) Piano
(2) Electric Piano
(3) Organ
(4) Guitar
(5) Electric Guitar