# Living Data Workshop 2

Link to this slideshow: http://bit.ly/LDWS2-19

# In Workshop 1

- Introduction to coding in Processing
- Drawing with code
- Loops and variables

# In the pre-work

**You have:**

- Loaded some data into Processing
- Used loops to read each row of data, one-by-one

# In this workshop...

- Floats
- Mapping variables
- Drawing with data
- Capturing your own data on your mobile

# Exploring our data

CSV stands for **C**omma **S**eparated **V**alues

One row per line. Each column is separated by a comma

```
2019,2,26,3720,2.45,113,0:35

2019,2,27,0,0.00,0,0:00

2019,2,28,737,0.52,22,0:06

2019,3,1,6586,4.61,200,1:03

2019,3,2,15545,10.88,474,2:13
```

# Exploring our data

CSV stands for **C**omma **S**eparated **V**alues

**One row per line**. Each column is separated by a comma

| | | | | | | |
|---|---|---|---|---|---|---|
| 2019, | 2, | 26, | 3720, | 2.45, | 113, | 0:35 |
| 2019, | 2, | 27, | 0, | 0.00, | 0, | 0:00 |
| 2019, | 2, | 28, | 737, | 0.52, | 22, | 0:06 |
| 2019, | 3, | 1, | 6586, | 4.61, | 200, | 1:03 |
| 2019, | 3, | 2, | 15545, | 10.88, | 474, | 2:13 |

# Exploring our data

The first row *might* be a header. Headers **describe** the data.

| year, | month, | day, | steps, | dist, | kcal, | duration |
|---|---|---|---|---|---|---|
| 2019, | 2, | 26, | 3720, | 2.45, | 113, | 0:35 |
| 2019, | 2, | 27, | 0, | 0.00, | 0, | 0:00 |
| 2019, | 2, | 28, | 737, | 0.52, | 22, | 0:06 |
| 2019, | 3, | 1, | 6586, | 4.61, | 200, | 1:03 |
| 2019, | 3, | 2, | 15545, | 10.88, | 474, | 2:13 |

# Loading Data (recap)

Create a variable called **myData** with a type of **Table**

```
Table myData;

myData = loadTable("hours_data.csv", "header");
```

Store the loaded data in **myData**

Opens a data file on your computer

The *name* of the file to open.

Tells Processing to treat the first-row as *a header* instead of as data.

Processing Reference: https://processing.org/reference/loadTable_.html

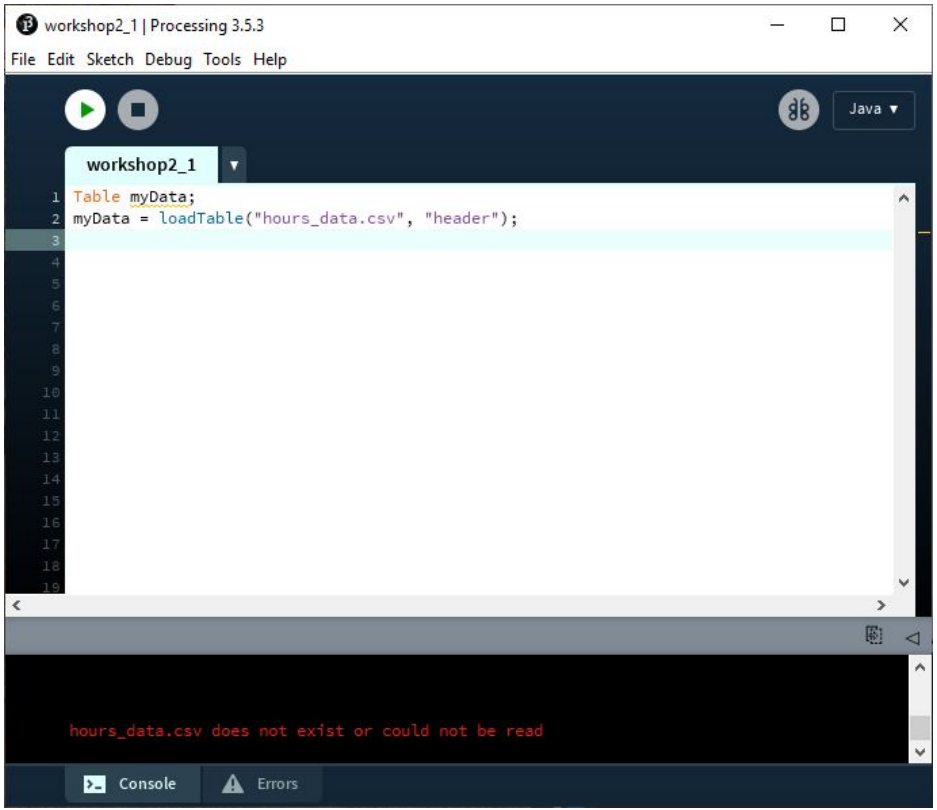
# What could go wrong?

You may get this message in the console:

```
hours_data.csv does not exist or
could not be read
```

Check two things:

- Filename is *exactly* correct
  - Does it end with .csv?
  - No typos
  - Capitalisation and spaces matter
- File has been dragged into Processing *(see Workshop 2 Sheet)*



# Reading Data (recap)

Create a variable called `numRows` with type `Int`

On the Table called `myData`, find the number of rows.

```
int numRows = myData.getRowCount();
println("There are", numRows, "rows of data");
```

Print some info to the console.

Create a message that includes the number of rows.

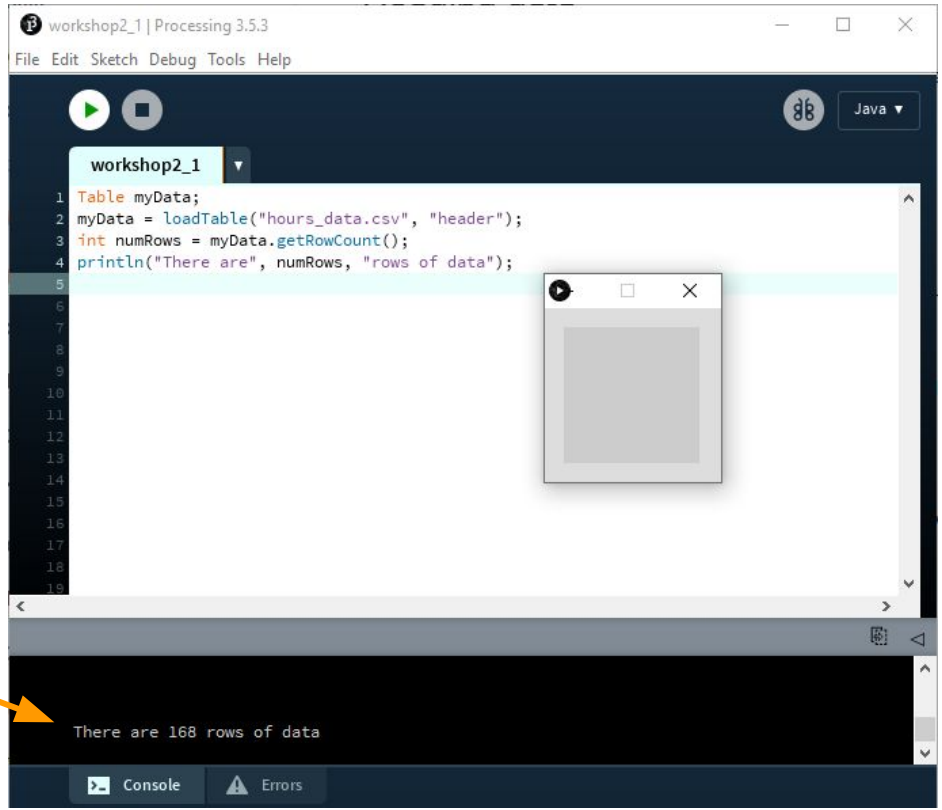
Processing Reference: https://processing.org/reference/Table_getRowCount_.html

# Checking that it works

Nothing shows up in the sketch window…

Because we haven't told it to draw anything.

But we can see our message in the console.



# Reading Data (recap)

Set up a loop to run `numRows` number of times.

```
for (int i = 0; i < numRows; i++) {
  float value = myData.getFloat(i, "Steps (average per min)");
  println(value);
}
```

Get a value from a column in `myData`

What the column is called (i.e. it's header label)

Which row number to get the data from

Print this value to the console

Processing Reference: https://processing.org/reference/Table_getFloat_.html

# Floats

In this example, we added a new concept - a datatype called `float`

Short for floating point - lets you have decimal points:

i.e.

```
int   pi = 3;

float pi = 3.14159;
```

Processing Reference: https://processing.org/reference/float.html

# Floats

Remember that you **can't assign a `float` to an `int`**

Doing this:

```
int   pi = 3.14159;
```

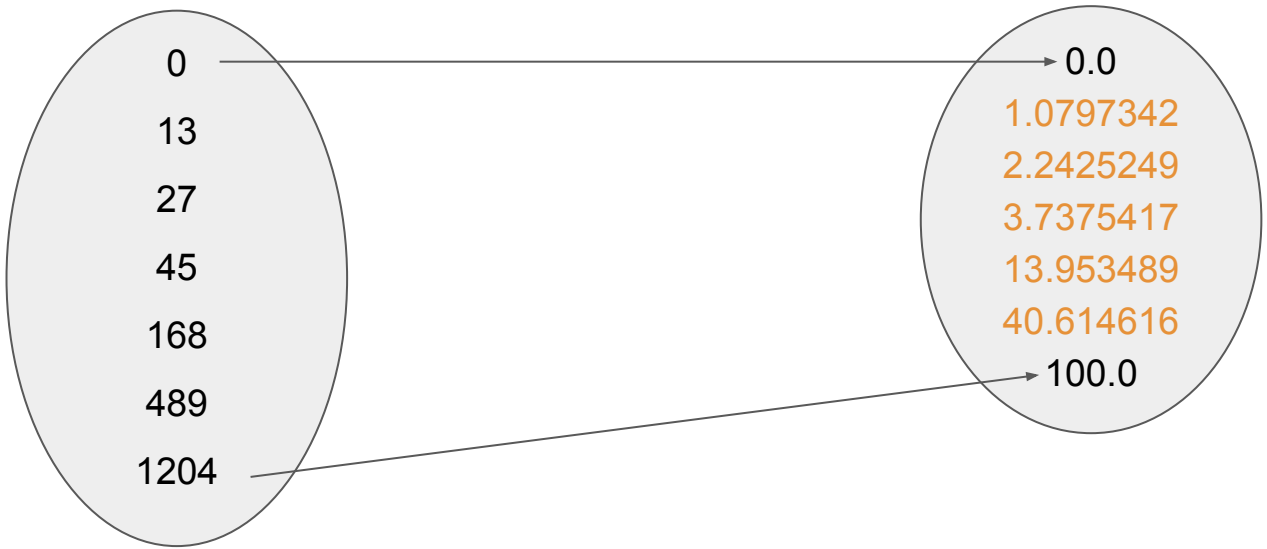
Results in an error:

```
cannot convert from float to int
```

Processing Reference: https://processing.org/reference/float.html

# Mapping

In maths, a mapping refers to applying a *function* to convert one **range** of data, to **another**.



# Why do mapping?

Your data is lots of:

- Very small numbers:
  0.3555, 0.200550, 0.455555
- Very big numbers
  12323, 343453, 123123, 3212312

But your output needs:

- To fit within the width of the screen
- Stay within the bounds of what a colour code can be (i.e. max 255)

# map() function

Syntax

```
map(value, start1, stop1, start2, stop2);
```

Example:

```
float value = 489;
float mapValue = map(value, 0, 1204, 0, 100);
println(mapValue);
```

What is output to the console?

Parameters

| | |
|---|---|
| `value` | value to be converted |
| `start1` | lower bound of current range |
| `stop1` | upper bound of current range |
| `start2` | lower bound of the target range |
| `stop2` | upper bound of the target range |

Processing Reference: https://processing.org/reference/map_.html

# Mapping our data

Before we can use `map()` properly, we need to know two more things about our data:

1. What's the maximum value?
2. What's the minimum value?

We could do this by hand… but. Let's use code.

# Exercise

# Finding the range of values

```
... // Leave the existing code here

float minValue = 0;
float maxValue = 0;

for(int i = 0; i < numRows; i++) {
  float value = myData.getFloat(i, "Steps (average per min)");

  minValue = min(value, minValue);
  maxValue = max(value, maxValue);

  println(value);
}

println("Minimum value: ", minValue);
println("Maximum value: ", maxValue);
```

# Drawing with data

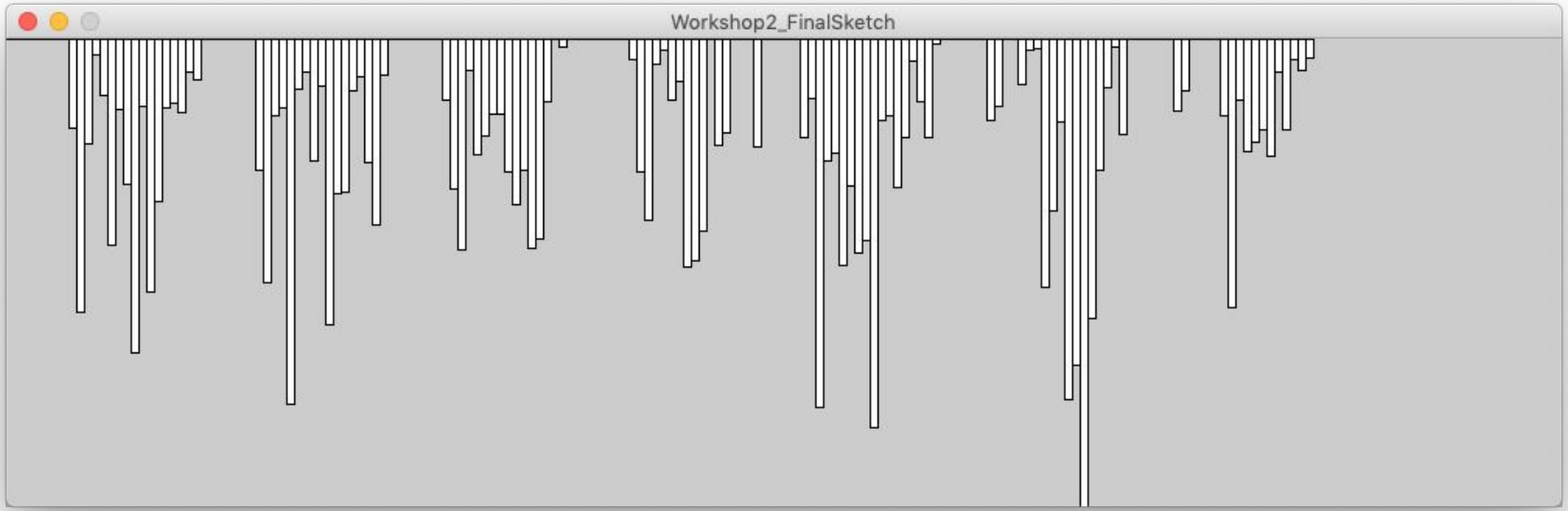
```
... // Leave the existing code here

// Create a second loop to make a drawing
for(int i = 0; i < numRows; i++) {
  float value = myData.getFloat(i, "Steps (average per min)");

  float barHeight = map(value, minValue, maxValue, 0, height);

  rect(i*5, 0, 5, barHeight);
}
```

# The Result



# Capturing your own data

# Capturing your own data

We are going to use your own activity as a source of data. We will keep track of your *hourly* or *daily* **step count**.

We need you to install an app to help capture this data.

GET IT ON Google Play

Download on the App Store

Android

GET IT ON Google Play

Apple

Download on the App Store

# Important note

You **do not need** to create an account for either app

**On iOS:**

HealthKit is enabled by default. QSAccess will collect your logged data.

**On Android:**

Accupedo will start logging once you run it for the first time.

# Sample Data

We have provided some data for you, captured on both Android and iOS.

Download these files from Dropbox.

http://bit.ly/LDWS2-DATA-19

# Downloading your own data

We’ll walk you through the process of downloading your data in the pre-work for Workshop 3.