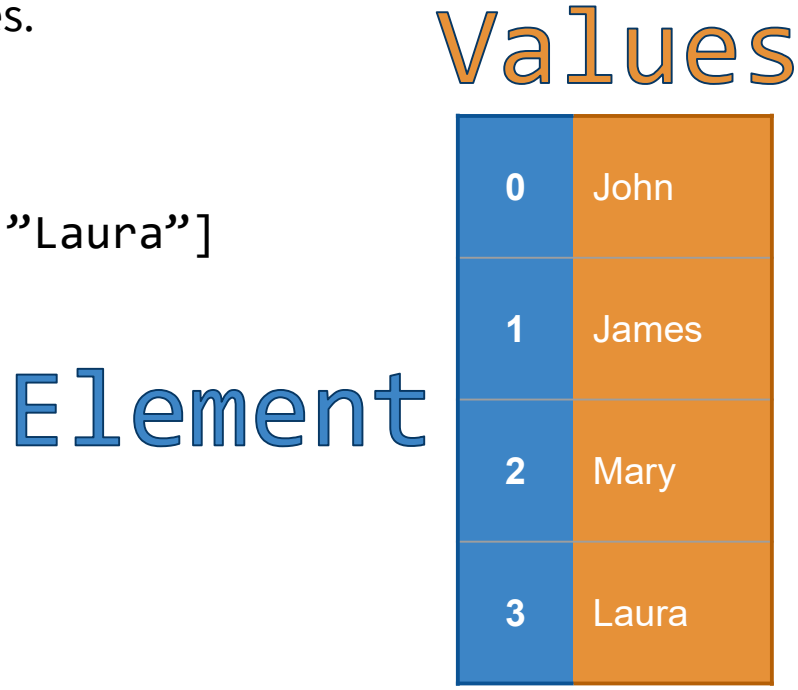
Parallel Arrays (Revision)

There is nothing new about ‘parallel’ arrays

Remember that a **1-D array** can only hold one instance of a variable at any given element so for example if storing 4 names.

e.g.

```
Names = [“John”,”James”,”Mary”,”Laura”]
```



But what if we are storing names and ages

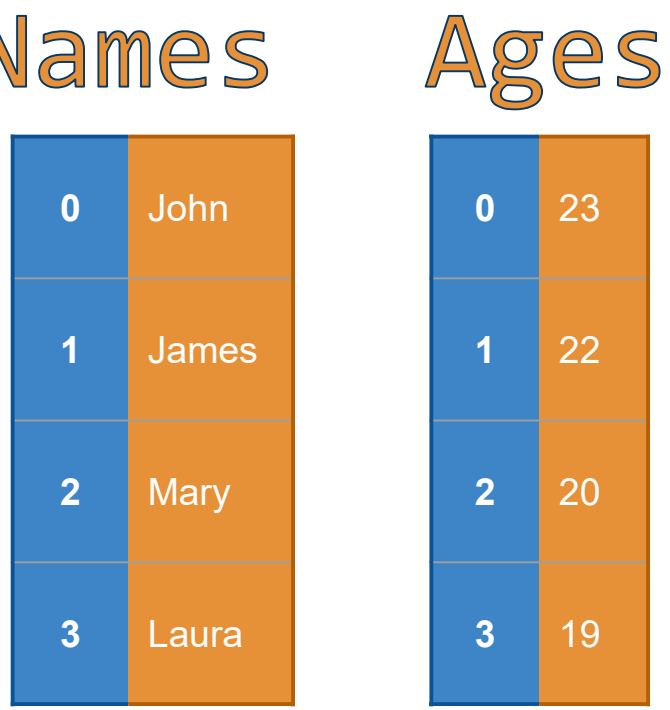
We can store the names in an array called **`Names`** and the ages in an array called **`Ages`**

Such as shown to the right

These are known as **parallel arrays**

- as they are arranged in parallel with each other

You occasionally did this in National 5



But what if we are storing names and ages

We know that the data in the names array is arranged in parallel with the ages array

So….

names[0] is **ages[0]** years old

`John is 23 years old`

names[1] is **ages[1]** years old

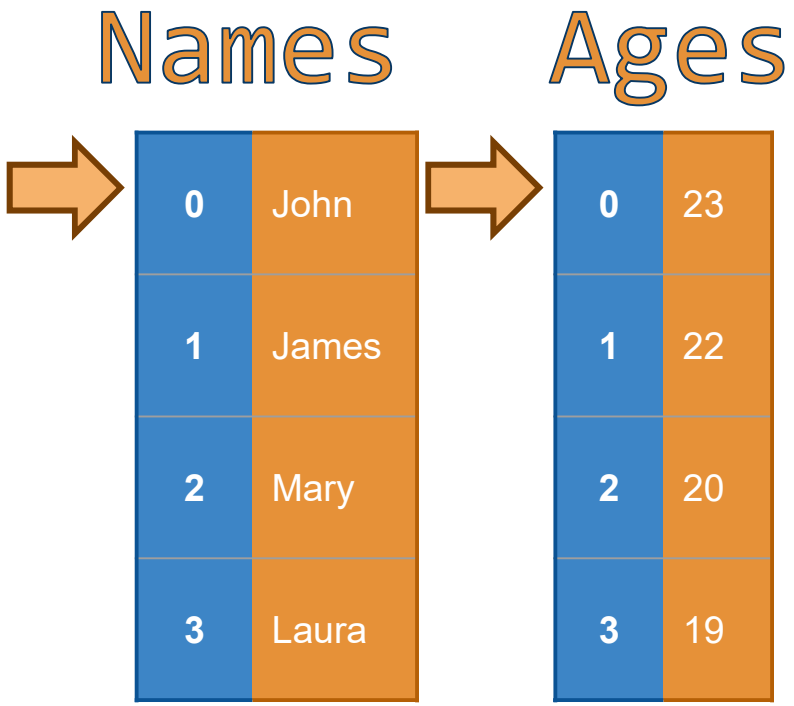
`James is 22 years old`

names[2] is **ages[2]** years old

`Mary is 20 years old`

names[3] is **ages[3]** years old

`Laura is 19 years old`



Using parallel arrays

A teacher is storing the following pieces of data about a class of 20 pupils.

Forename, surname, form class, prelim mark, coursework mark.

They will then calculate a total mark and average mark for the pupil

Question:

How can parallel arrays be used to store all of the data required?

Parallel Arrays and Data Flow

As a reminder the data flow from the BMI program is shown below:

Subroutine	IN	OUT
Get Data From File		`weight(),height()`
Calculate BMI	`weight(),height()`	`BMI()`
BMI Summary	`BMI()`	`under,ideal,over`
DisplayDetails	`weight(),height(),BMI(),under,ideal,over`	

You will notice that we need to be passing **multiple parallel arrays** (weight,height and BMI) in order to allow subprograms to operate