The Pandas Library

pandas

- The pandas library contains useful tools for data analysis in python.
  - We will often use pd as shorthand for pandas

```
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

- With pandas, you can:
  - Store data in a similar format to a table in Excel
  - Quickly analyze and process data
  - Quickly sort data
  - Cut away data that does not meet some criterion
  - And much more!

DataFrames

- A **DataFrame** is an object that stores data in rows and columns, similar to a database.
  - You can think of a DataFrame as kind of like an Excel table.

A DataFrame, named myData		
First Name	**Last Name**	**Goals Scored**
Bob	Smith	0
Joe	Kepler	1
Ann	Faraday	3
Mia	Smith	0
Tim	Ohm	10

```
myData = pd.DataFrame() #create an empty DataFrame object

myData['First Name'] = ['Bob', 'Joe', 'Ann', 'Mia', 'Tim']
myData['Last Name'] = ['Smith', 'Kepler', 'Faraday', 'Smith', 'Ohm']
myData['Goals Scored'] = [0, 1, 3, 0, 10]

myData.head()
```

	First Name	Last Name	Goals Scored
0	Bob	Smith	0
1	Joe	Kepler	1
2	Ann	Faraday	3
3	Mia	Smith	0
4	Tim	Ohm	10

Reading in data from a .csv file

- The pandas library makes reading in data from a .csv file into a DataFrame object very simple.

- If the .csv data has a header row (i.e., column names), the read_csv() function automatically names the columns in the DataFrame.

```
#This will read in the csv file and create an object called a Data Frame
data = pd.read_csv("/kaggle/input/galacticcoordswithgaia/gaiaDataNearSun.csv")
```

Methods

- **Methods** are like functions, but are specifically tied to an object, like a DataFrame.
  - len() is a function, as it can be applied to a list or a DataFrame or other data types.
  - .head() is a method, as it can only be called from a DataFrame object: myData.head()

Exploring the data

- Here are a few useful methods for exploring your data once it is stored in a DataFrame:
  - `.head()` – This displays the first 5 rows of data in a DataFrame. Alternatively, you can choose the number of rows displayed by including a different number as an argument.
  - `.tail()` – this displays the last 5 rows of data if no argument is included. Otherwise, the number included as an argument will be the number of rows shown.

```
df.head(6)
```

	a	b
0	0	1
1	1	11
2	2	21
3	3	31
4	4	41
5	5	51

```
df.tail(6)
```

	a	b
4	4	41
5	5	51
6	6	61
7	7	71
8	8	81
9	9	91

Creating, Referencing Columns in a DataFrame

- To create a new column in a DataFrame:
  - Use square brackets and single quotes to name the new column, and use the assignment operator to assign new data to the column.

```
myData = pd.DataFrame() #create an empty DataFrame object
myData['X'] = [2, 3, 5, 7, 11, 13, 17, 19]

myData.head()
```

	X
0	2
1	3
2	5
3	7
4	11

Creating, Referencing Columns in a DataFrame

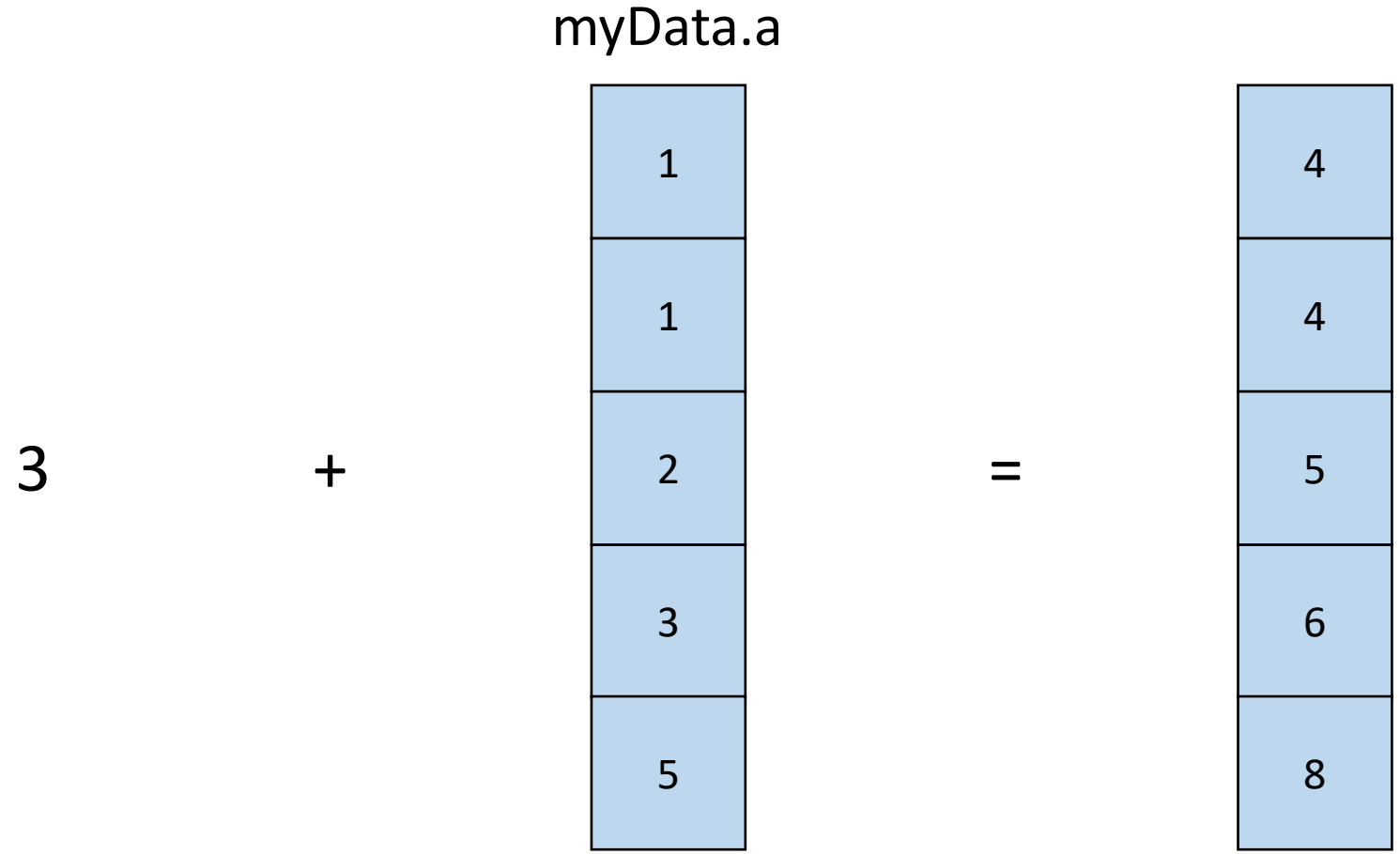
- To reference an existing column:
  - Type the DataFrame's name followed by a period and then the column name you'd like to reference.

```
myData['Y'] = myData.X + 2
myData.head()
```

	X	Y
0	2	4
1	3	5
2	5	7
3	7	9
4	11	13

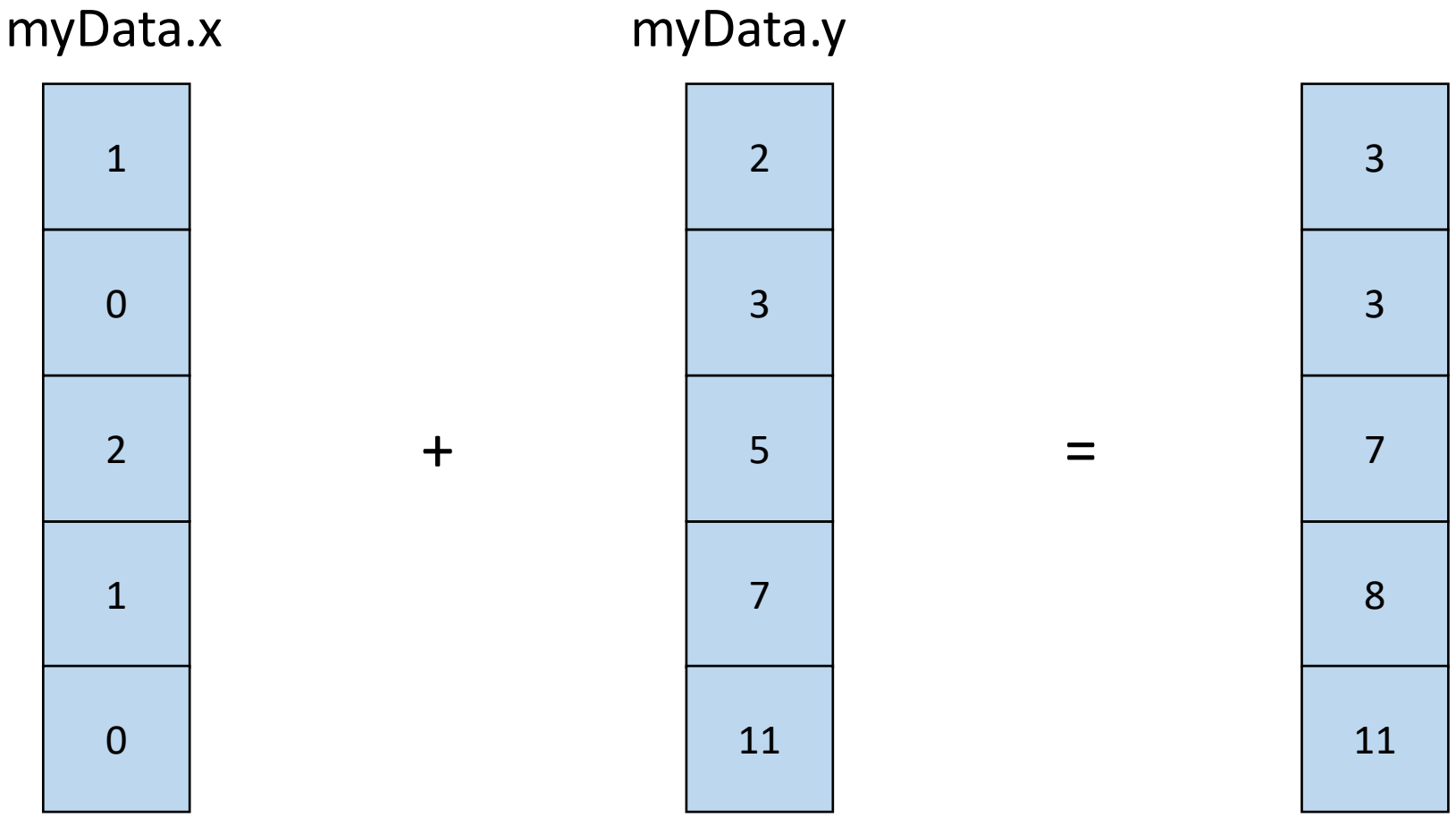
Visualizing Math with DataFrames:

- Adding a number and a column:



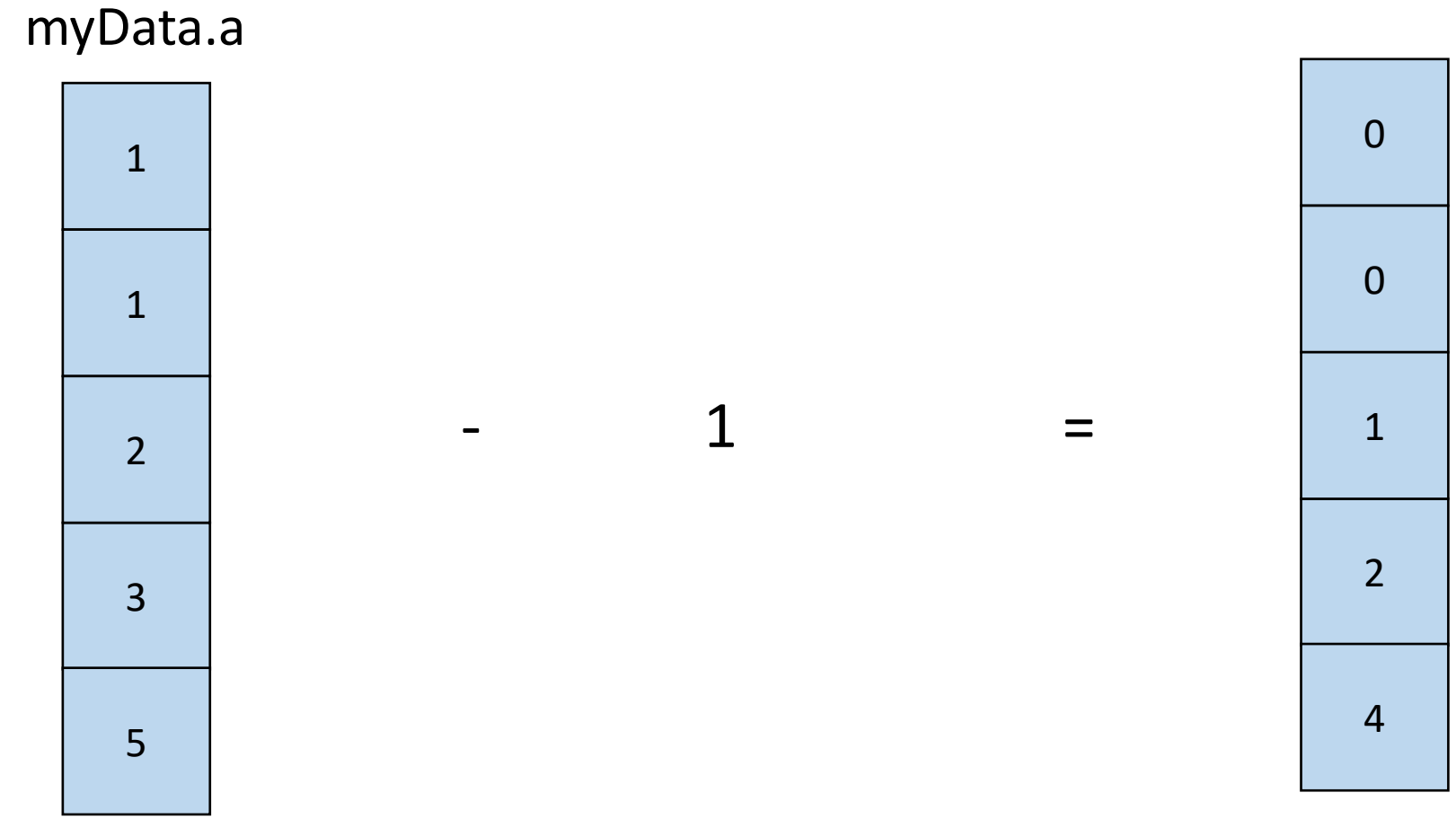
Visualizing Math with DataFrames:

- Adding two columns:



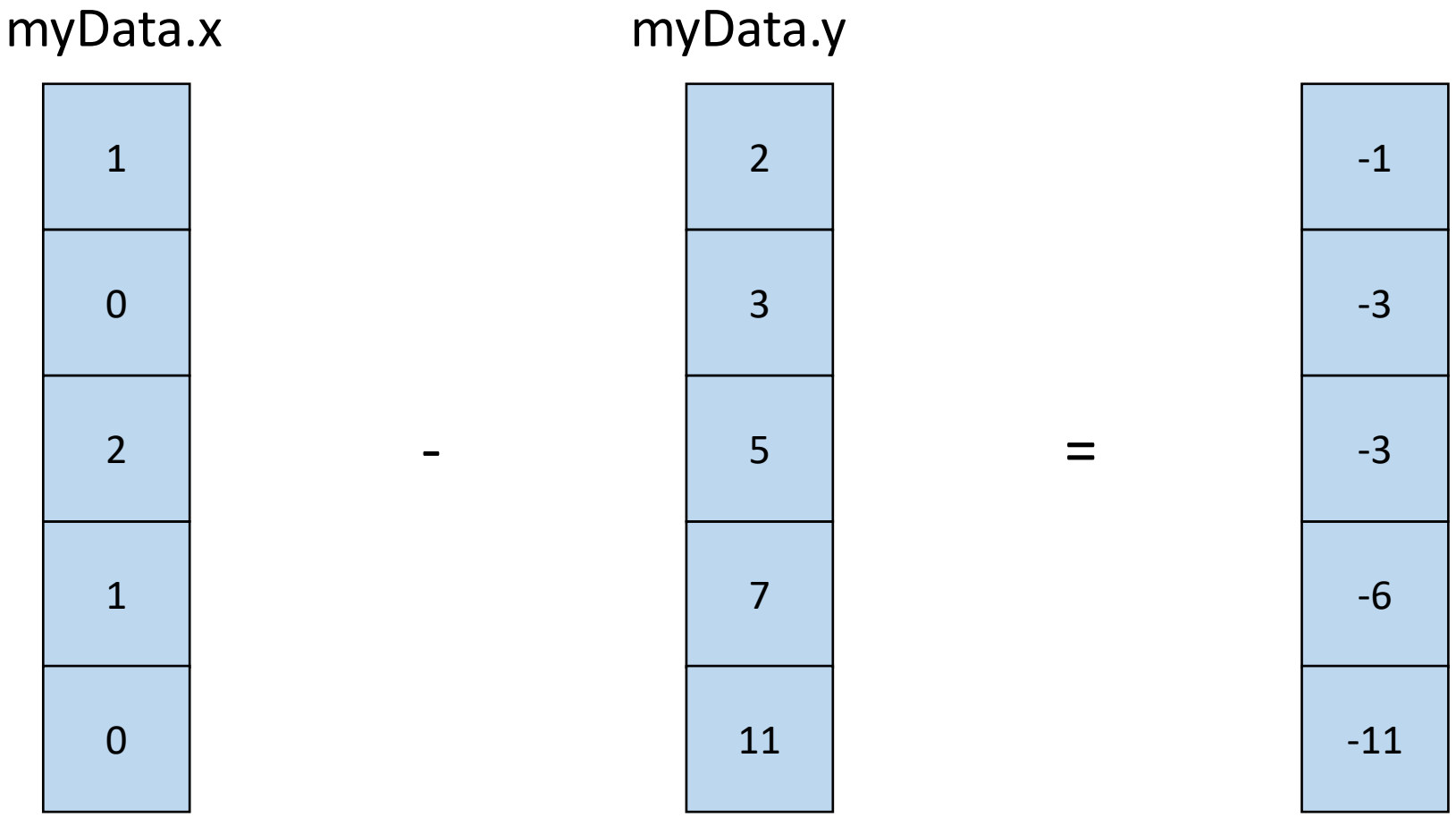
Visualizing Math with DataFrames:

- Subtracting a number and a column:



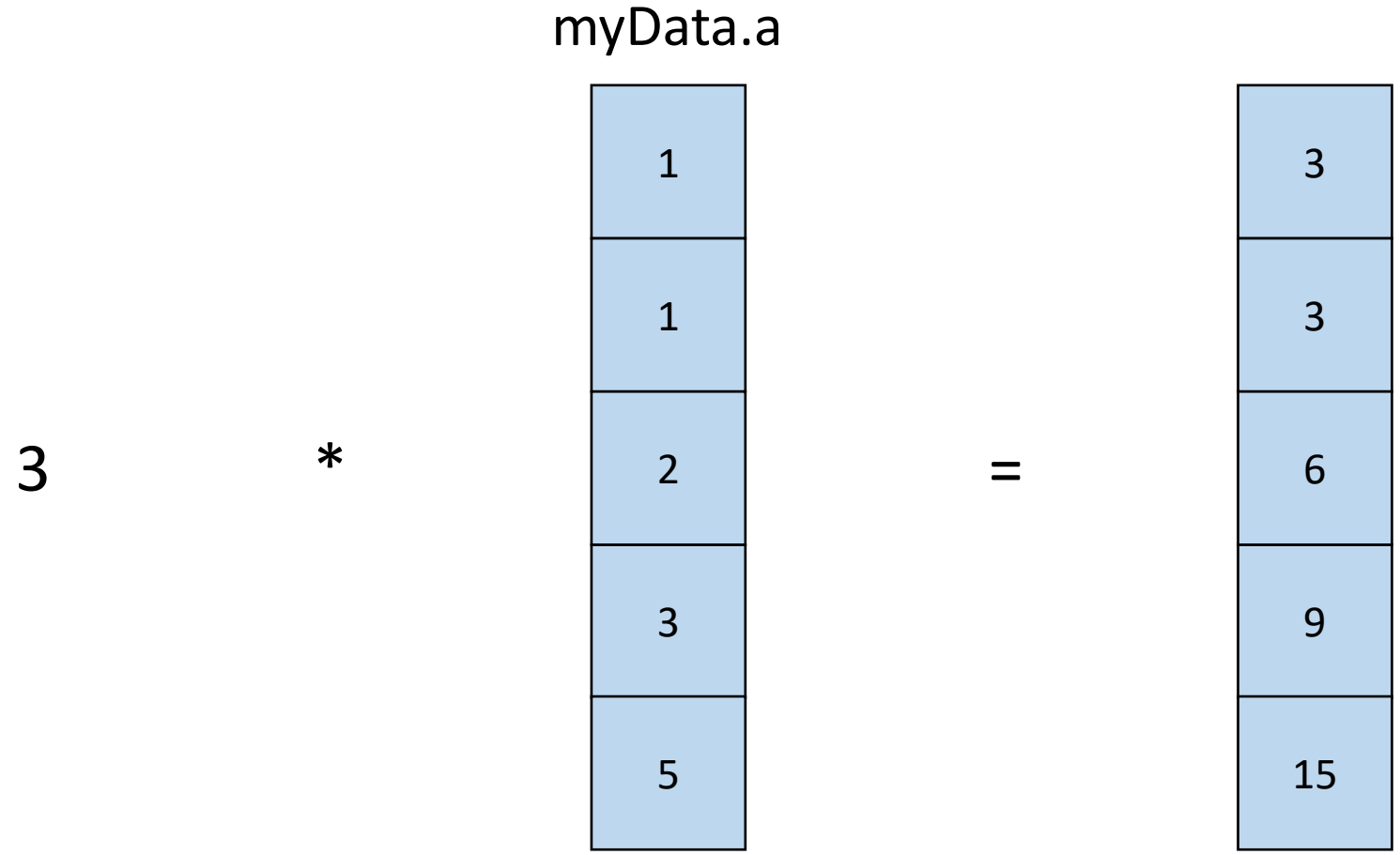
Visualizing Math with DataFrames:

- Subtracting two columns:



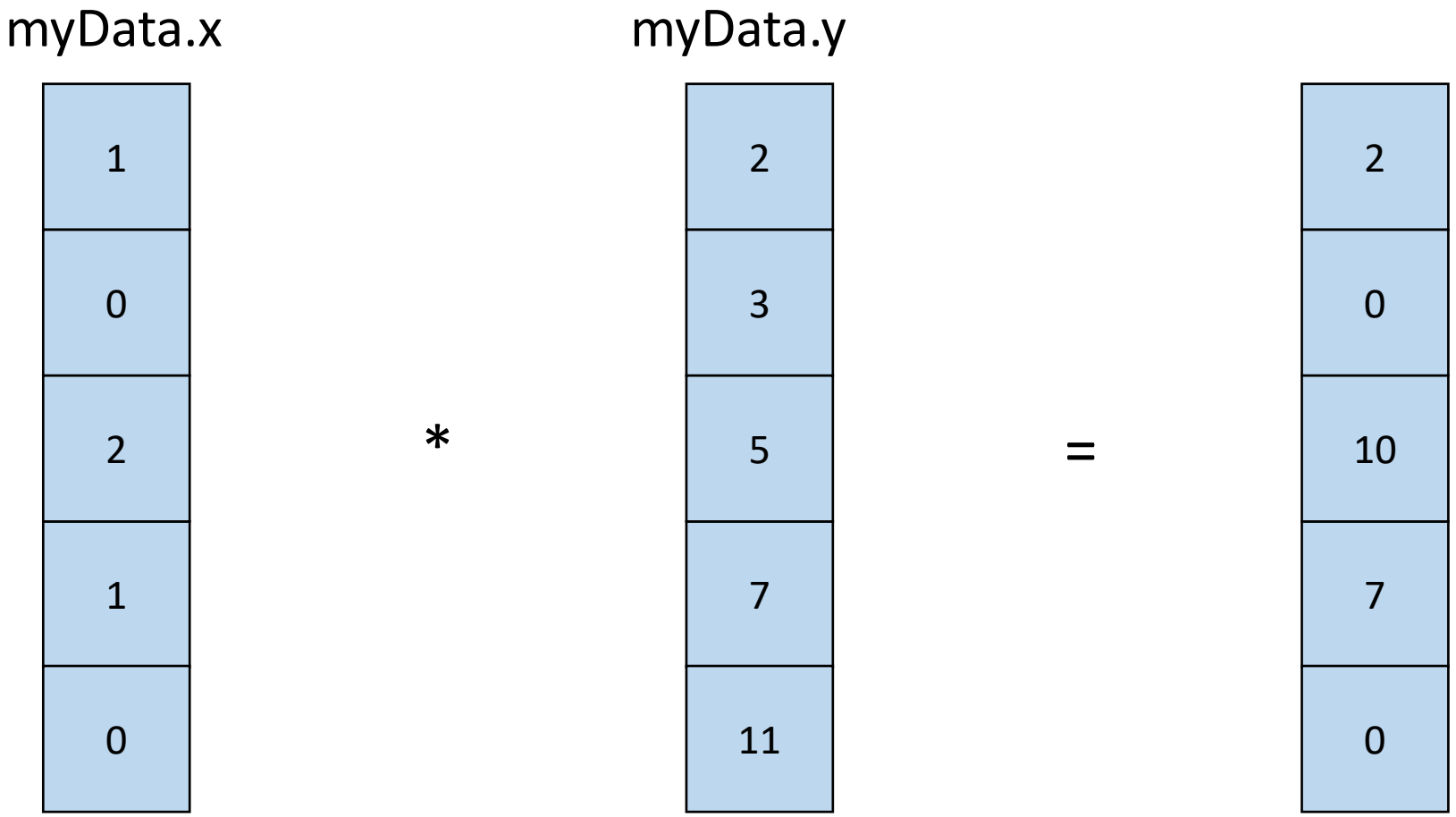
Visualizing Math with DataFrames:

- Multiplying a number and a column:



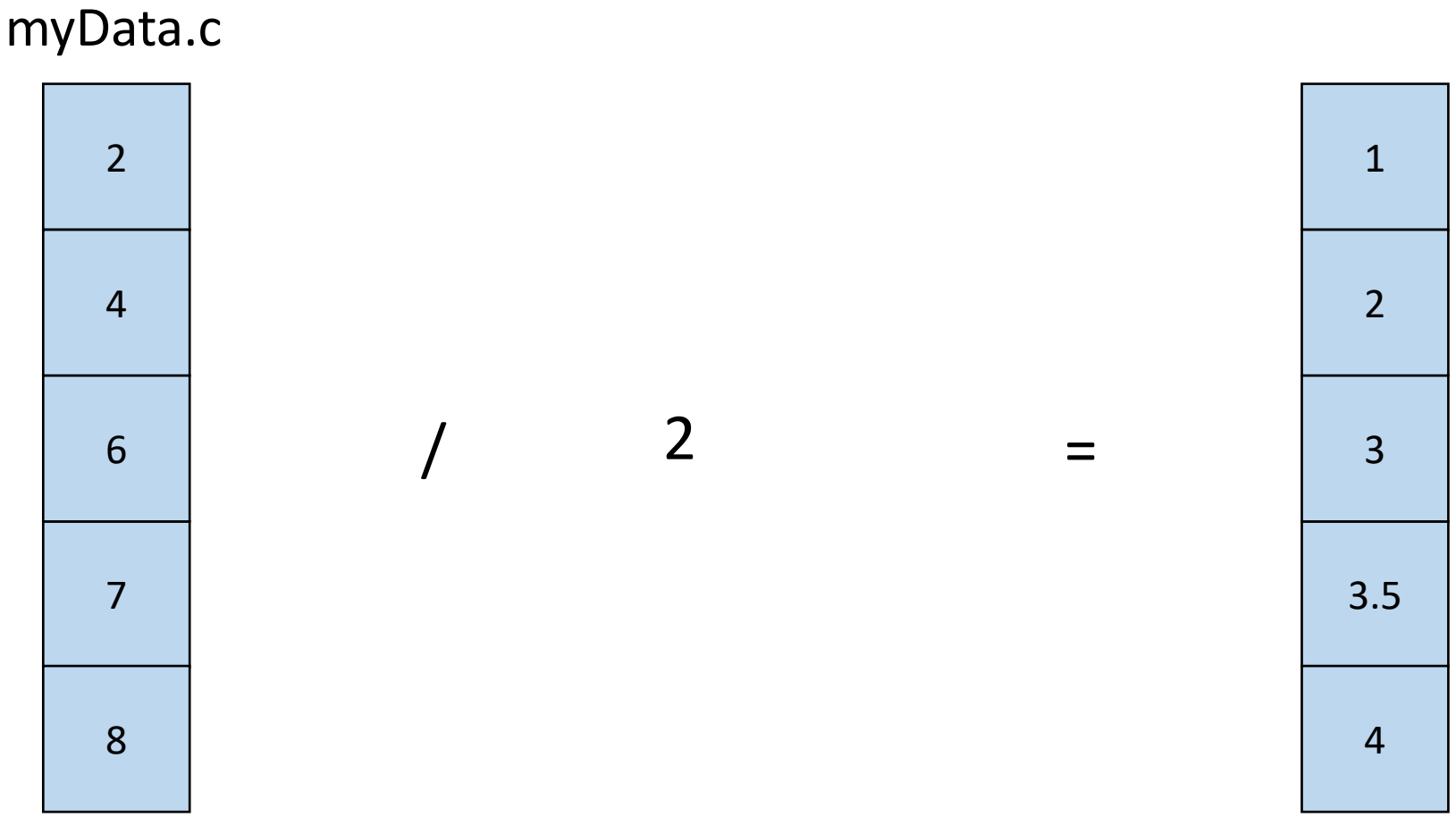
Visualizing Math with DataFrames:

- Multiplying two columns:



Visualizing Math with DataFrames:

- Dividing a column by a number:



Visualizing Math with DataFrames:

- Dividing a column by another column: