Summarizing data

INTRODUCTION TO NUMPY

Izzy Weber
Core Curriculum Manager, DataCamp

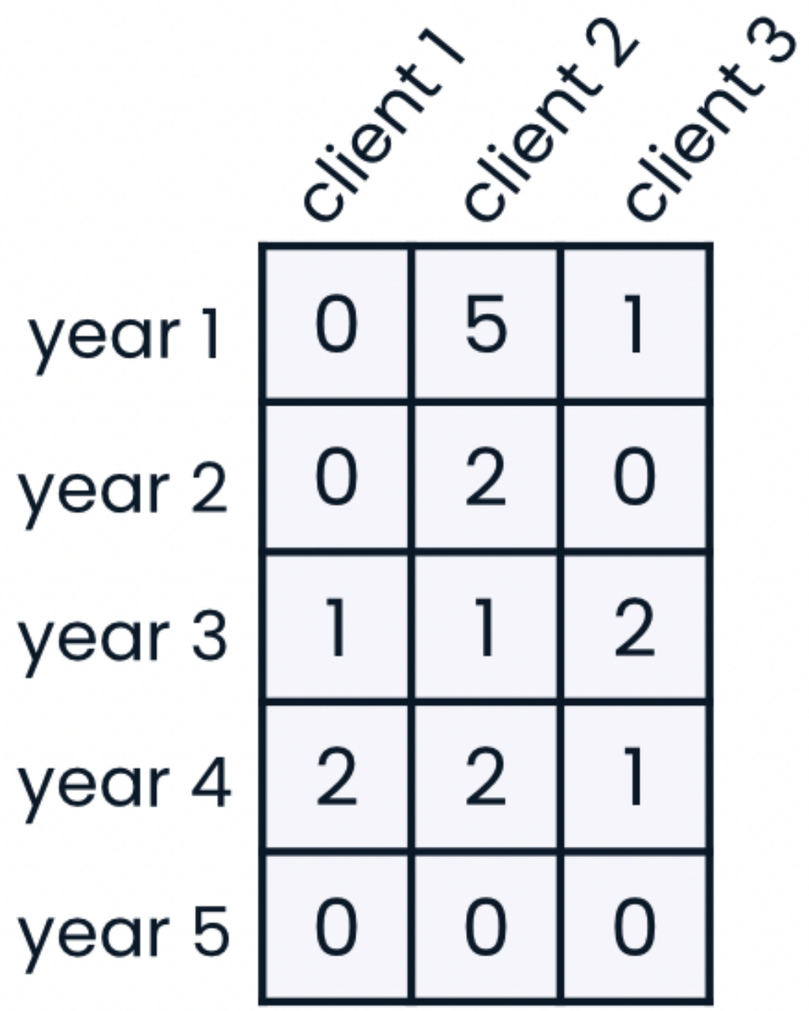
Aggregating methods

- `.sum()`
- `.min()`
- `.max()`
- `.mean()`
- `.cumsum()`

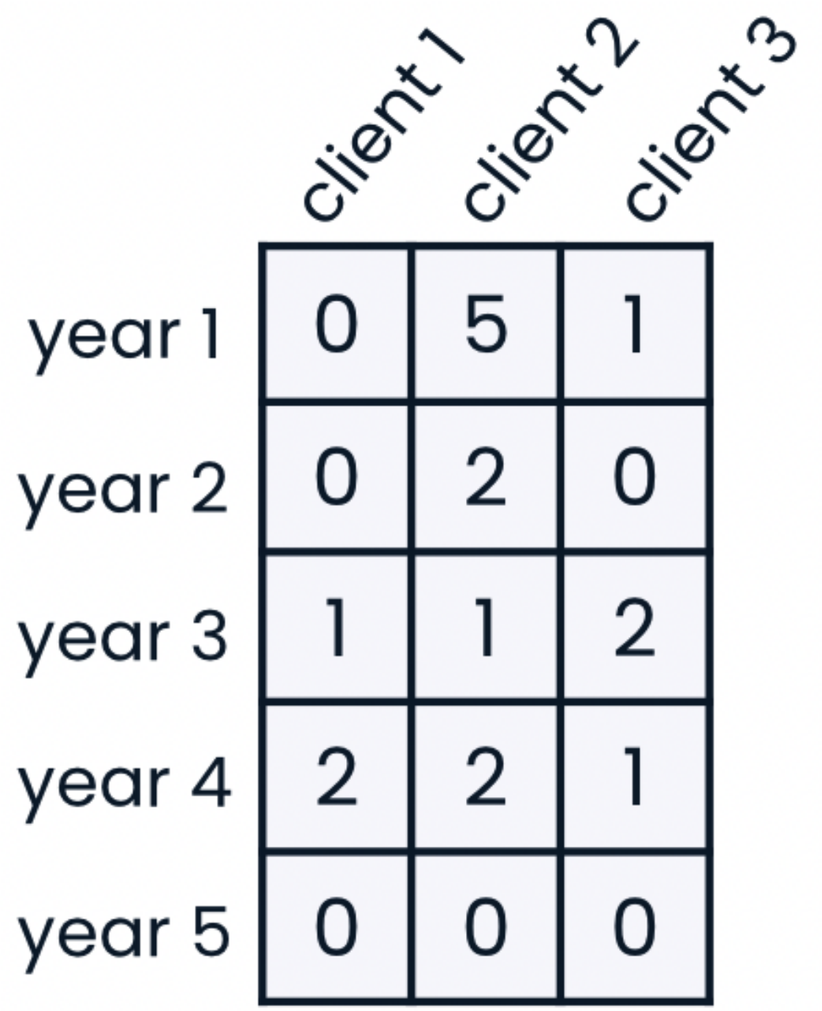
Our data

```
security_breaches
```

```
array([[0, 5, 1],
       [0, 2, 0],
       [1, 1, 2],
       [2, 2, 1],
       [0, 0, 0]])
```



Summing data

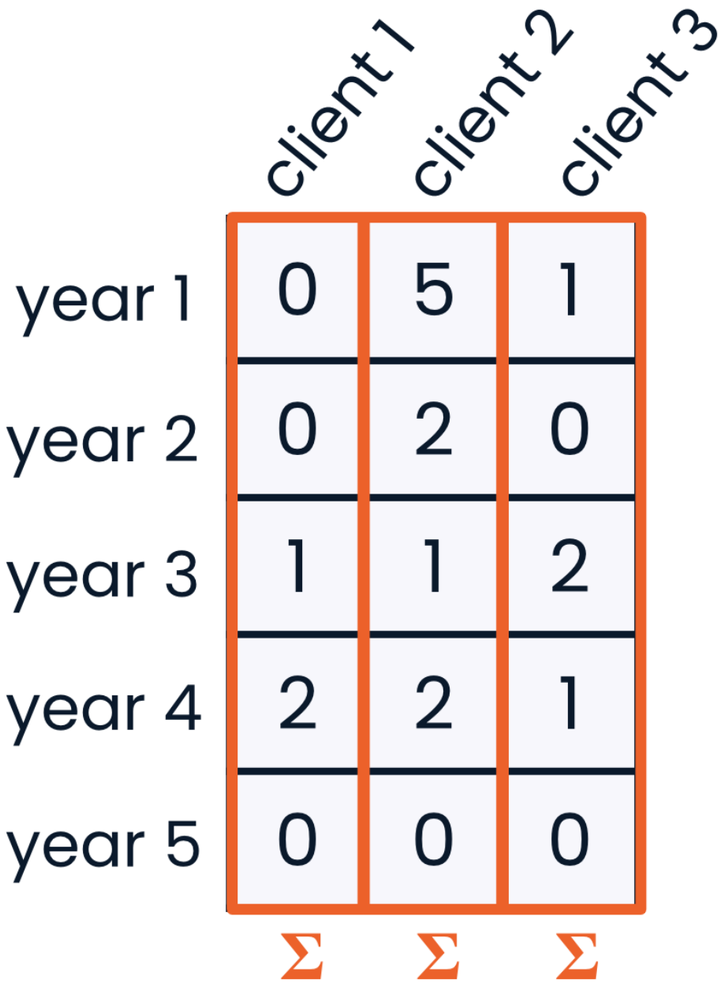


	client 1	client 2	client 3
year 1	0	5	1
year 2	0	2	0
year 3	1	1	2
year 4	2	2	1
year 5	0	0	0

```
security_breaches.sum()
```

```
17
```

Aggregating rows




```
security_breaches.sum(axis=0)
```

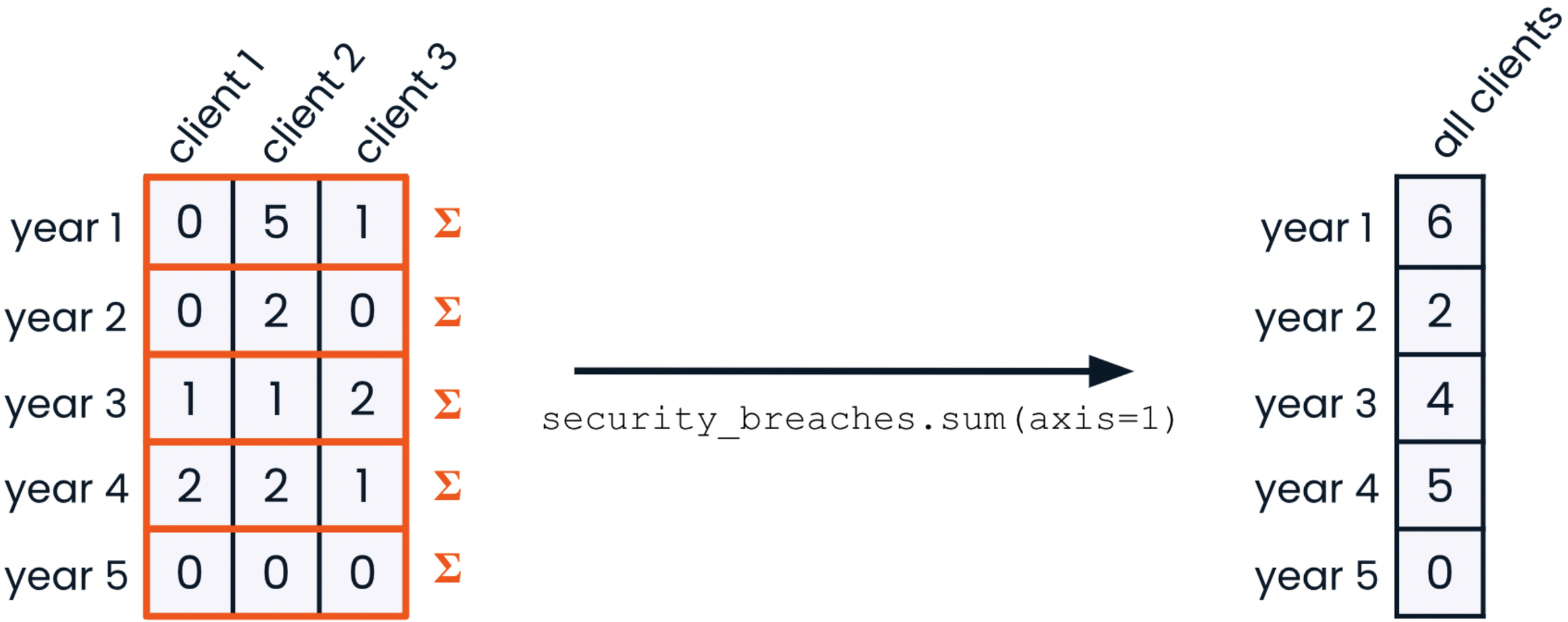
```
array([ 3, 10,  4])
```

Aggregating columns

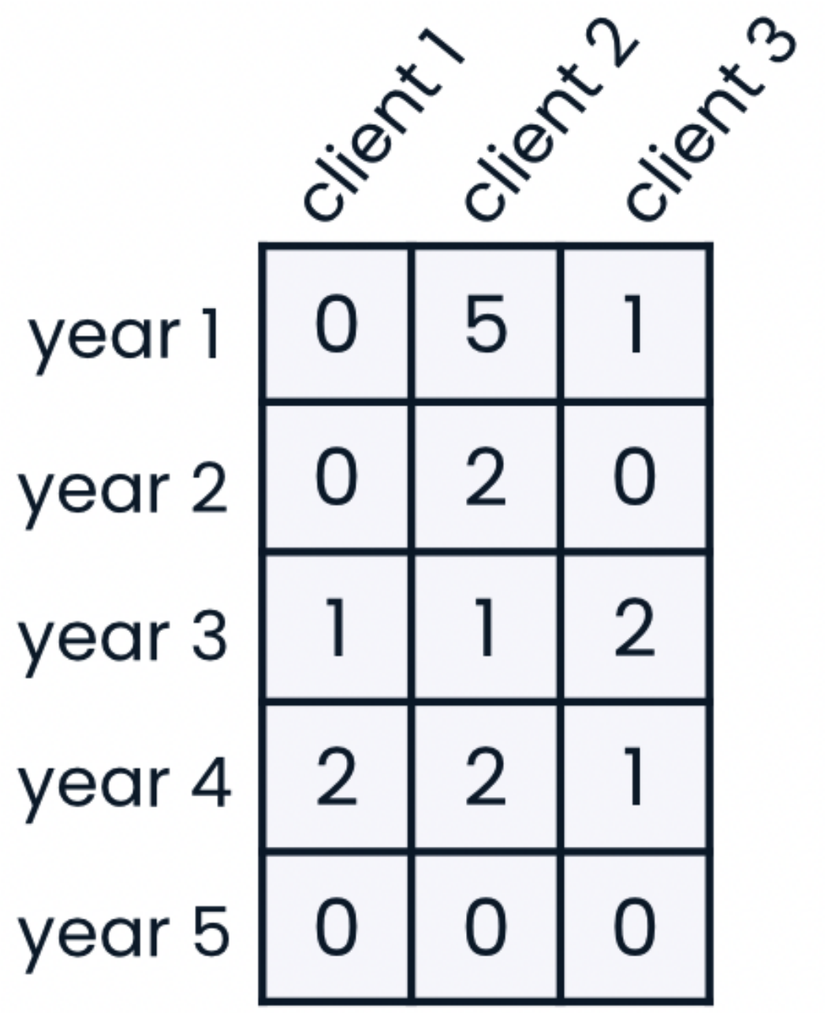

```
security_breaches.sum(axis=1)
```

```
array([6, 2, 4, 5, 0])
```

Making sense of the axis argument



Minimum and maximum values



	client 1	client 2	client 3
year 1	0	5	1
year 2	0	2	0
year 3	1	1	2
year 4	2	2	1
year 5	0	0	0

```
security_breaches.min()
```

```
0
```

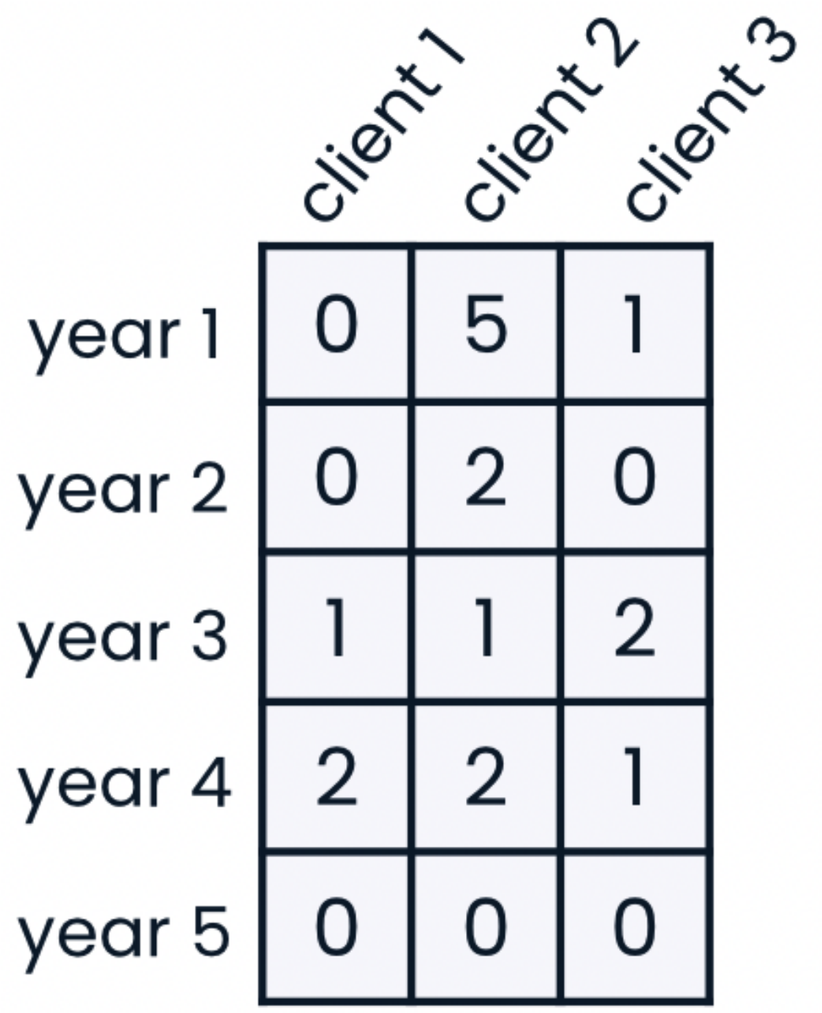
```
security_breaches.max()
```

```
5
```

```
security_breaches.min(axis=1)
```

```
array([0, 0, 1, 1, 0])
```

Finding the mean



	client 1	client 2	client 3
year 1	0	5	1
year 2	0	2	0
year 3	1	1	2
year 4	2	2	1
year 5	0	0	0

```
security_breaches.mean()
```

```
1.1333333333333333
```

```
security_breaches.mean(axis=1)
```

```
array([2., 0.6667, 1.3333, 1.6667, 0.])
```

The keepdims argument

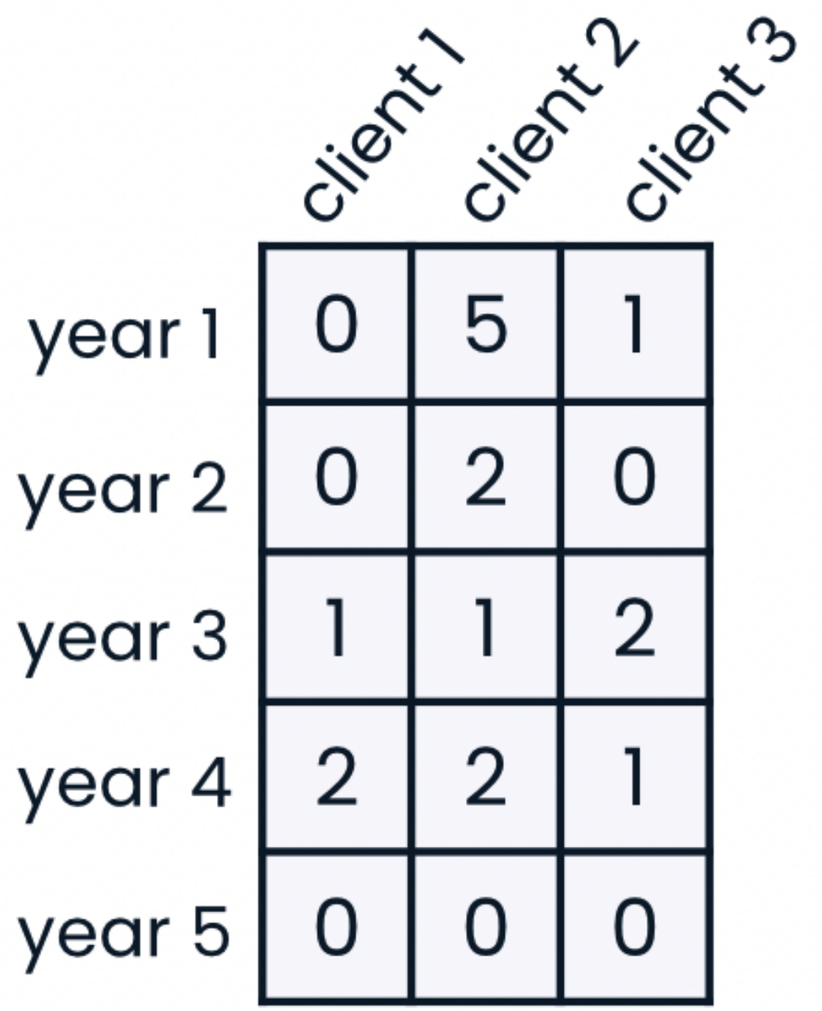
```
security_breaches.sum(axis=1)
```

```
array([6, 2, 4, 5, 0])
```

```
security_breaches.sum(axis=1, keepdims=True)
```

```
array([[6],
       [2],
       [4],
       [5],
       [0]])
```

Cumulative sums



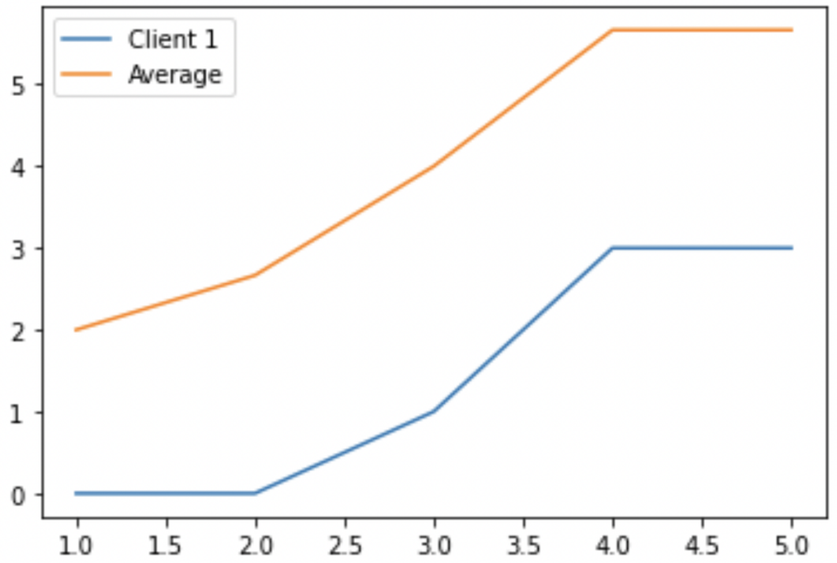
	client 1	client 2	client 3
year 1	0	5	1
year 2	0	2	0
year 3	1	1	2
year 4	2	2	1
year 5	0	0	0

```
security_breaches.cumsum(axis=0)
```

```
array([[ 0,  5,  1],
       [ 0,  7,  1],
       [ 1,  8,  3],
       [ 3, 10,  4],
       [ 3, 10,  4]])
```

Graphing summary values

```
cum_sums_by_client = security_breaches.cumsum(axis=0)
plt.plot(np.arange(1, 6), cum_sums_by_client[:, 0], label="Client 1")
plt.plot(np.arange(1, 6), cum_sums_by_client.mean(axis=1), label="Average")
plt.legend()
plt.show()
```



Let's practice!

INTRODUCTION TO NUMPY

Vectorized operations

INTRODUCTION TO NUMPY

Izzy Weber
Core Curriculum Manager, DataCamp

A little help from C

Vectorized operations

```
np.arange(1000000).sum()
```

```
499999500000
```

Speed compared to Python


```
array = np.array([[1, 2, 3], [4, 5, 6]])
for row in range(array.shape[0]):
    for column in range(array.shape[1]):
        array[row][column] += 3
```

```
array([[4, 5, 6],
       [7, 8, 9]])
```

NumPy syntax

```
array = np.array([[1, 2, 3], [4, 5, 6]])
array + 3
```

```
array([[4, 5, 6],
       [7, 8, 9]])
```

Multiplying by a scalar

```
array = np.array([[1, 2, 3], [4, 5, 6]])
array * 3
```

```
array([[ 3,  6,  9],
       [12, 15, 18]])
```

Adding two arrays together

```
array_a = np.array([[1, 2, 3], [4, 5, 6]])
array_b = np.array([[0, 1, 0], [1, 0, 1]])
array_a + array_b
```

```
array([[1, 3, 3],
       [5, 5, 7]])
```

Multiplying two arrays together

```
array_a = np.array([[1, 2, 3], [4, 5, 6]])
array_b = np.array([[0, 1, 0], [1, 0, 1]])
array_a * array_b
```

```
array([[0, 2, 0],
       [4, 0, 6]])
```

Not just for math

```
array = np.array([[1, 2, 3], [4, 5, 6]])
array > 2
```

```
array([[False, False, True],
       [True,  True,  True]])
```

Vectorize Python code!

```
array = np.array(["NumPy", "is", "awesome"])
len(array) > 2
```

```
True
```

```
vectorized_len = np.vectorize(len)
```

```
vectorized_len(array) > 2
```

```
array([ True, False,  True])
```

Let's practice!

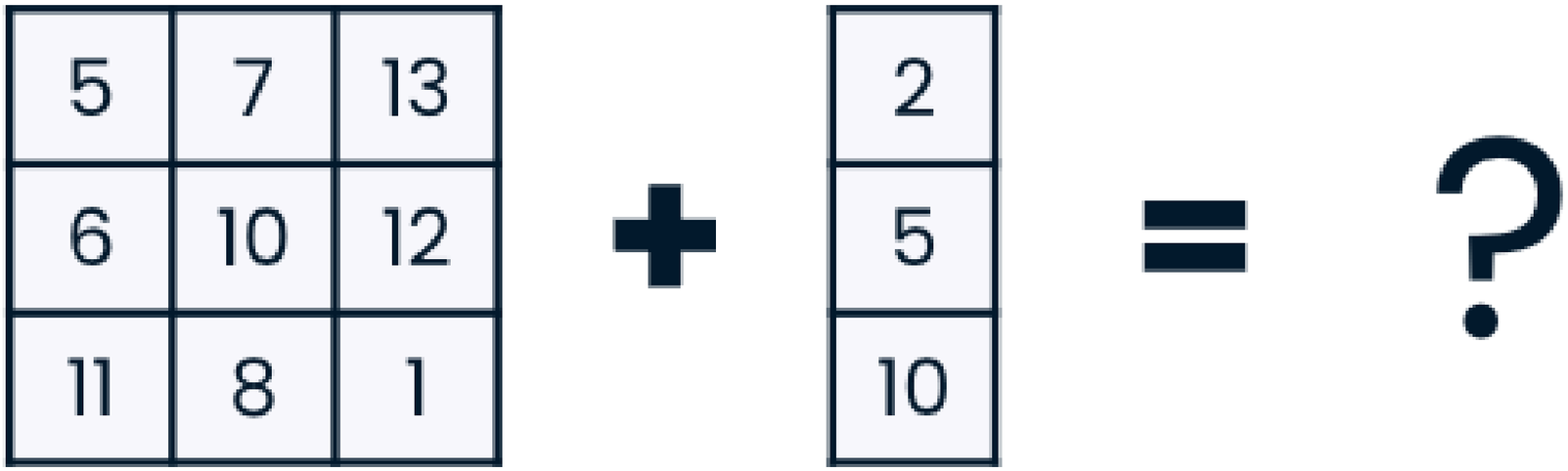
INTRODUCTION TO NUMPY

Broadcasting

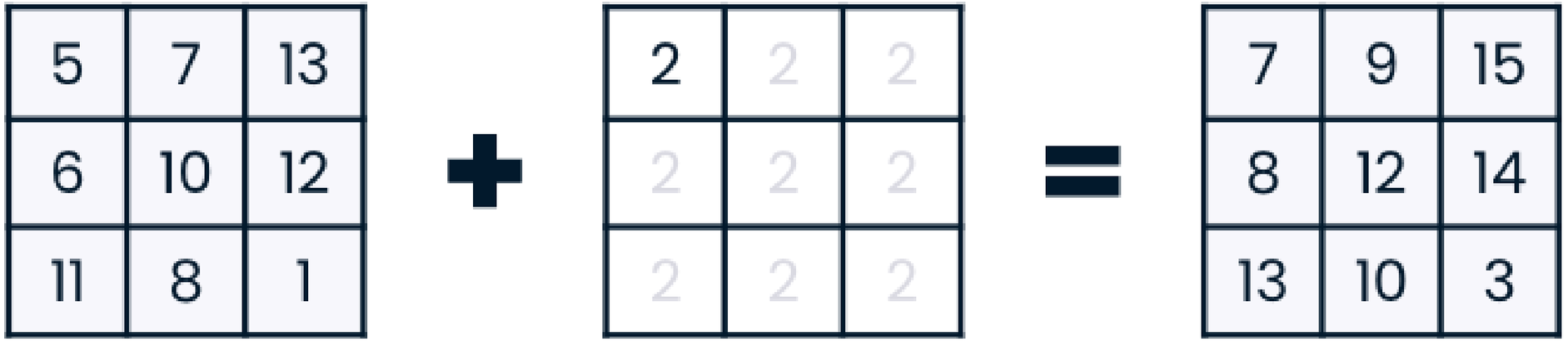
INTRODUCTION TO NUMPY

Izzy Weber
Core Curriculum Manager, DataCamp

Broadcasting introduced

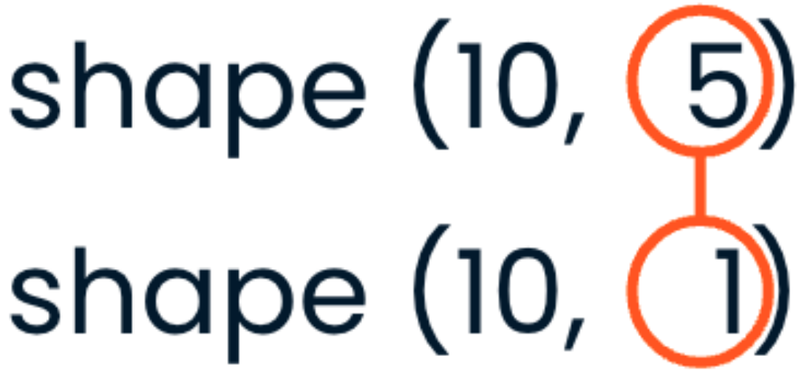


Broadcasting a scalar



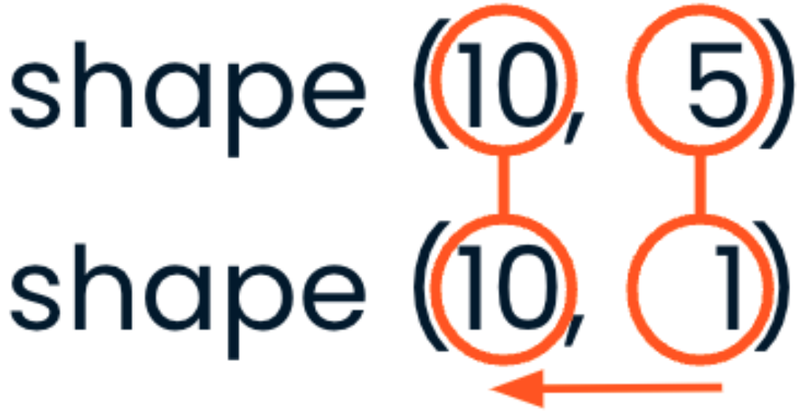
Compatibility rules

- NumPy compares sets of array dimensions from right to left



Compatibility rules

- NumPy compares sets of array dimensions from right to left
- Two dimensions are compatible when...
  - One of them has a length of one or
  - They are of equal lengths
- All dimension sets must be compatible



Broadcastable or not?

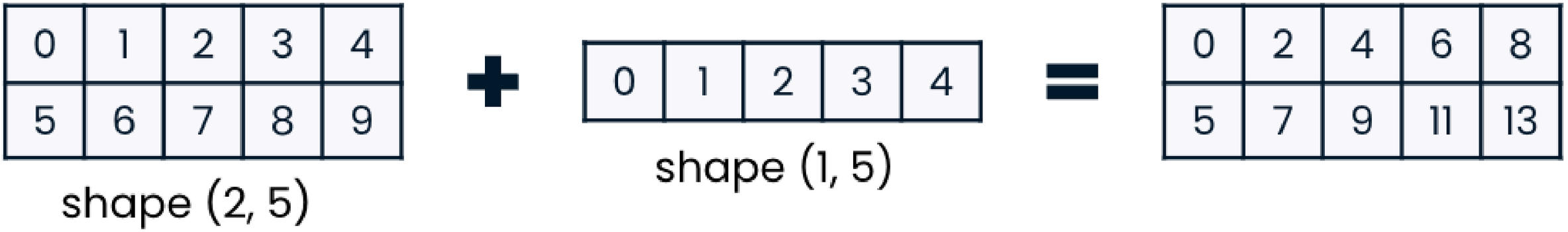
Broadcastable shapes:

- `(10, 5)` and `(10, 1)`
- `(10, 5)` and `(5, )`

Shapes which are not broadcastable:

- `(10, 5)` and `(5, 10)`
- `(10, 5)` and `(10, )`

Broadcasting rows

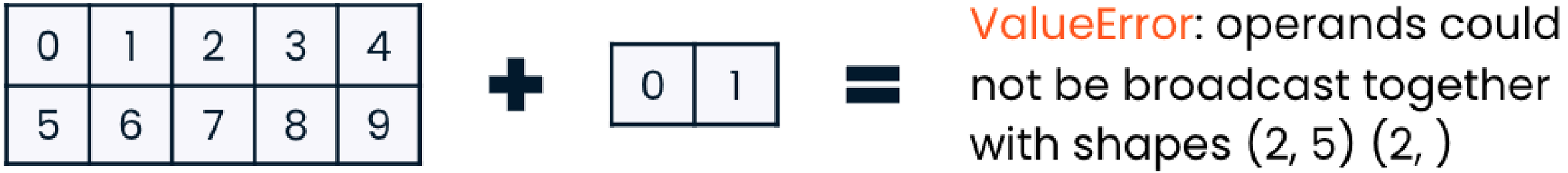



```
array = np.arange(10).reshape((2, 5))
array + np.array([0, 1, 2, 3, 4])
```

```
array([[ 0,  2,  4,  6,  8],
       [ 5,  7,  9, 11, 13]])
```

Broadcasting rows

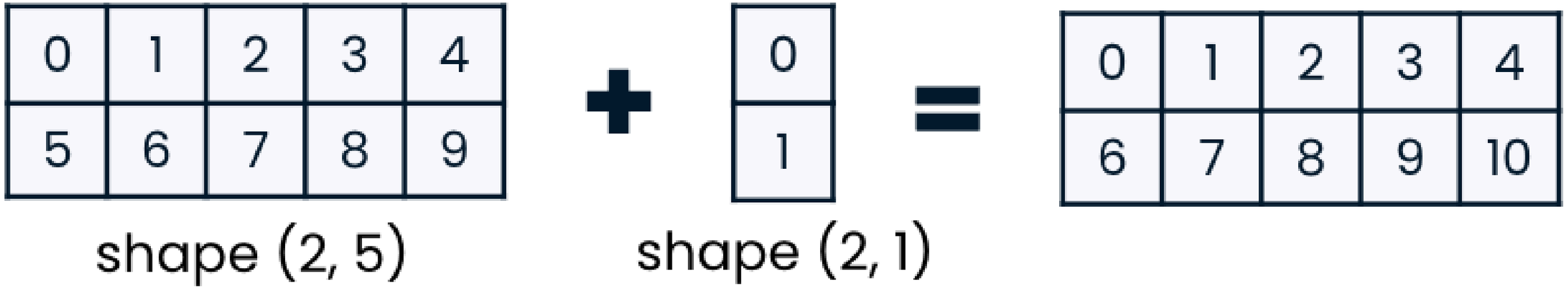
Incompatible broadcasting




```
array = np.arange(10).reshape((2, 5))
array + np.array([0, 1])
```

```
ValueError: operands could not be broadcast together with shapes (2,5) (2,)
```

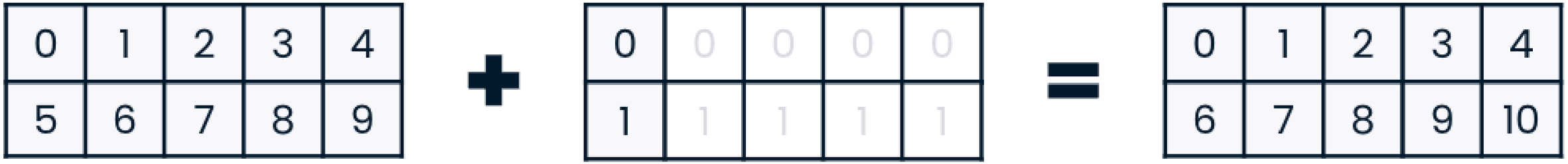
Broadcasting columns




```
array = np.arange(10).reshape((2, 5))
array + np.array([0, 1]).reshape((2, 1))
```

```
array([[ 0,  1,  2,  3,  4],
       [ 6,  7,  8,  9, 10]])
```

Broadcasting columns



Other operators

Let's practice!

INTRODUCTION TO NUMPY