Indexing and slicing arrays

INTRODUCTION TO NUMPY

Izzy Weber
Core Curriculum Manager, DataCamp

datacamp

Indexing 1D arrays

```
array = np.array([2, 4, 6, 8, 10])
array[3]
```

```
8
```

Indexing elements in 2D

0	0	4	3	0	0	2	0	9
0	0	5	0	0	9	0	0	1
0	7	0	0	6	0	0	4	3
0	0	6	0	0	2	0	8	7
1	9	0	0	0	7	4	0	0
0	5	0	0	8	3	0	0	0
6	0	0	0	0	0	1	0	5
0	0	3	5	0	8	6	9	0
0	4	2	9	1	0	3	0	0

```
sudoku_game[2, 4]
```

```
6
```

Indexing rows in 2D

0	0	4	3	0	0	2	0	9
0	0	5	0	0	9	0	0	1
0	7	0	0	6	0	0	4	3
0	0	6	0	0	2	0	8	7
1	9	0	0	0	7	4	0	0
0	5	0	0	8	3	0	0	0
6	0	0	0	0	0	1	0	5
0	0	3	5	0	8	6	9	0
0	4	2	9	1	0	3	0	0

```
sudoku_game[0]
```

```
array([0, 0, 4, 3, 0, 0, 2, 0, 9])
```

Indexing columns in 2D

0	0	4	3	0	0	2	0	9
0	0	5	0	0	9	0	0	1
0	7	0	0	6	0	0	4	3
0	0	6	0	0	2	0	8	7
1	9	0	0	0	7	4	0	0
0	5	0	0	8	3	0	0	0
6	0	0	0	0	0	1	0	5
0	0	3	5	0	8	6	9	0
0	4	2	9	1	0	3	0	0

```
sudoku_game[:, 3]
```

```
array([3, 0, 0, 0, 0, 0, 0, 5, 9])
```

Slicing 1D arrays

```
array = np.array([2, 4, 6, 8, 10])
array[2:4]
```

```
array([6,  8])
```

Slicing 2D arrays

0	0	4	3	0	0	2	0	9
0	0	5	0	0	9	0	0	1
0	7	0	0	6	0	0	4	3
0	0	6	0	0	2	0	8	7
1	9	0	0	0	7	4	0	0
0	5	0	0	8	3	0	0	0
6	0	0	0	0	0	1	0	5
0	0	3	5	0	8	6	9	0
0	4	2	9	1	0	3	0	0

```
sudoku_game[3:6, 3:6]
```

```
array([[0, 0, 2],
       [0, 0, 7],
       [0, 8, 3]])
```

Slicing with steps

0	0	4	3	0	0	2	0	9
0	0	5	0	0	9	0	0	1
0	7	0	0	6	0	0	4	3
0	0	6	0	0	2	0	8	7
1	9	0	0	0	7	4	0	0
0	5	0	0	8	3	0	0	0
6	0	0	0	0	0	1	0	5
0	0	3	5	0	8	6	9	0
0	4	2	9	1	0	3	0	0

```
sudoku_game[3:6:2, 3:6:2]
```

```
array([[0, 2],
       [0, 3]])
```

Sorting arrays

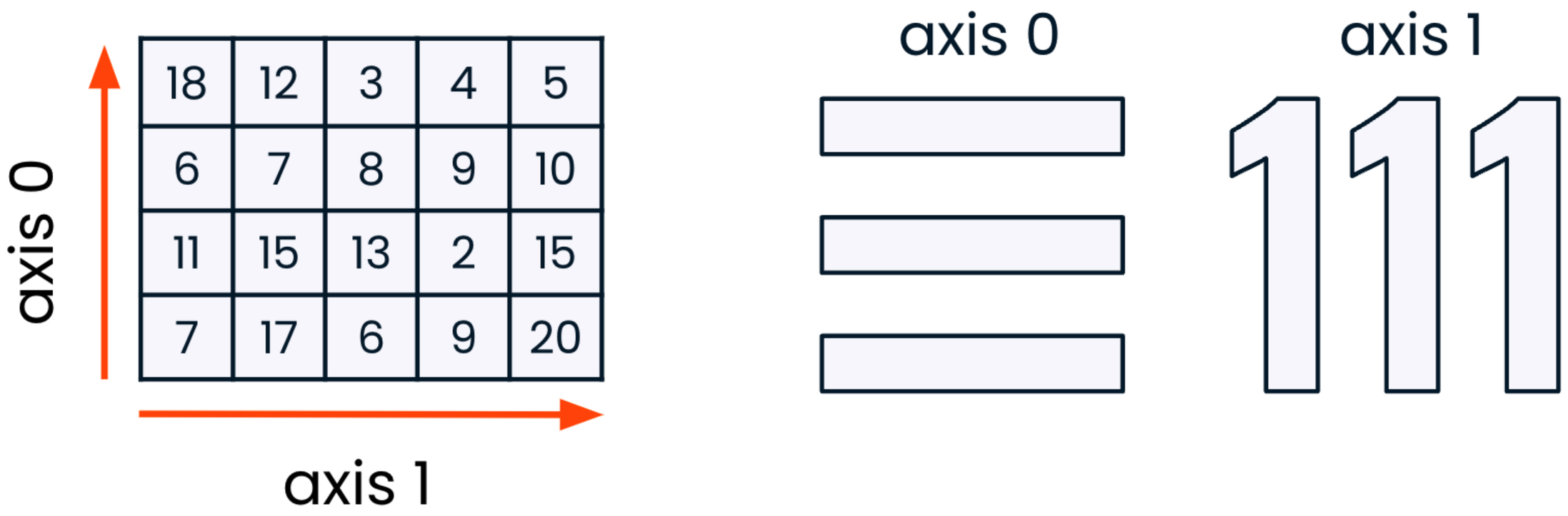
Before sorting:

0	0	4	3	0	0	2	0	9
0	0	5	0	0	9	0	0	1
0	7	0	0	6	0	0	4	3
0	0	6	0	0	2	0	8	7
1	9	0	0	0	7	4	0	0
0	5	0	0	8	3	0	0	0
6	0	0	0	0	0	1	0	5
0	0	3	5	0	8	6	9	0
0	4	2	9	1	0	3	0	0

```
np.sort(sudoku_game)
```

```
array([[0, 0, 0, 0, 0, 2, 3, 4, 9],
       [0, 0, 0, 0, 0, 0, 1, 5, 9],
       [0, 0, 0, 0, 0, 3, 4, 6, 7],
       [0, 0, 0, 0, 0, 2, 6, 7, 8],
       [0, 0, 0, 0, 0, 1, 4, 7, 9],
       [0, 0, 0, 0, 0, 0, 3, 5, 8],
       [0, 0, 0, 0, 0, 0, 1, 5, 6],
       [0, 0, 0, 0, 3, 5, 6, 8, 9],
       [0, 0, 0, 0, 1, 2, 3, 4, 9]])
```

Axis order


axis 0
axis 1
18 12 3 4 5
6 7 8 9 10
11 15 13 2 15
7 17 6 9 20
axis 0
axis 1
1 1 1

Sorting by axis

```
np.sort(sudoku_game)
```

```
array([[0, 0, 0, 0, 0, 2, 3, 4, 9],
       [0, 0, 0, 0, 0, 0, 1, 5, 9],
       [0, 0, 0, 0, 0, 3, 4, 6, 7],
       [0, 0, 0, 0, 0, 2, 6, 7, 8],
       [0, 0, 0, 0, 0, 1, 4, 7, 9],
       [0, 0, 0, 0, 0, 0, 3, 5, 8],
       [0, 0, 0, 0, 0, 0, 1, 5, 6],
       [0, 0, 0, 0, 3, 5, 6, 8, 9],
       [0, 0, 0, 0, 1, 2, 3, 4, 9]])
```

```
np.sort(sudoku_game, axis=0)
```

```
array([[0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 2, 0, 0, 2, 1, 0, 1],
       [0, 4, 3, 0, 0, 3, 2, 0, 3],
       [0, 5, 4, 3, 1, 7, 3, 4, 5],
       [1, 7, 5, 5, 6, 8, 4, 8, 7],
       [6, 9, 6, 9, 8, 9, 6, 9, 9]])
```

Let's practice!

INTRODUCTION TO NUMPY

Filtering arrays

INTRODUCTION TO NUMPY

Izzy Weber
Core Curriculum Manager, DataCamp

datacamp

Two ways to filter

1. Masks and fancy indexing
2. `np.where()`

Boolean masks

```
one_to_five = np.arange(1, 6)
one_to_five
```

```
array([1, 2, 3, 4, 5])
```

```
mask = one_to_five % 2 == 0
mask
```

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

Filtering with fancy indexing

```
one_to_five = np.arange(1, 6)
mask = one_to_five % 2 == 0
one_to_five[mask]
```

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

2D fancy indexing

```
classroom_ids_and_sizes = np.array([[1, 22], [2, 21], [3, 27], [4, 26]])
classroom_ids_and_sizes
```

```
array([[ 1, 22],
       [ 2, 21],
       [ 3, 27],
       [ 4, 26]])
```

```
classroom_ids_and_sizes[:, 1] % 2 == 0
```

2D fancy indexing

```
classroom_ids_and_sizes = np.array([[1, 22], [2, 21], [3, 27], [4, 26]])
classroom_ids_and_sizes
```

```
array([[ 1, 22],
       [ 2, 21],
       [ 3, 27],
       [ 4, 26]])
```

```
classroom_ids_and_sizes[:, 0][classroom_ids_and_sizes[:, 1] % 2 == 0]
```

```
array([1, 4])
```

Fancy indexing vs. np.where()

Fancy indexing

- Returns array of elements

`np.where()`

- Returns array of indices
- Can create an array based on whether elements do or don't meet condition

Filtering with np.where()

```
classroom_ids_and_sizes
```

```
array([[ 1, 22],
       [ 2, 21],
       [ 3, 27],
       [ 4, 26]])
```

```
np.where(classroom_ids_and_sizes[:, 1] % 2 == 0)
```

```
(array([0, 3]),)
```

np.where() element retrieval

```
sudoku_game
```

```
array([[0, 0, 4, 3, 0, 0, 2, 0, 9],
       [0, 0, 5, 0, 0, 9, 0, 0, 1],
       [0, 7, 0, 0, 6, 0, 0, 4, 3],
       [0, 0, 6, 0, 0, 2, 0, 8, 7],
       [1, 9, 0, 0, 0, 7, 4, 0, 0],
       [0, 5, 0, 0, 8, 3, 0, 0, 0],
       [6, 0, 0, 0, 0, 0, 1, 0, 5],
       [0, 0, 3, 5, 0, 8, 6, 9, 0],
       [0, 4, 2, 9, 1, 0, 3, 0, 0]])
```

A tuple of indices

```
row_ind, column_ind = np.where(sudoku_game == 0)
row_ind, column_ind
```

```
(array([0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4,
        4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 8, 8,
        8, 8]),
 array([0, 1, 4, 5, 7, 0, 1, 3, 4, 6, 7, 0, 2, 3, 5, 6, 0, 1, 3, 4, 6, 2,
        3, 4, 7, 8, 0, 2, 3, 6, 7, 8, 1, 2, 3, 4, 5, 7, 0, 1, 4, 8, 0, 5,
        7, 8]))
```

Find and replace

```
np.where(sudoku_game == 0, "", sudoku_game)
```

```
array([['', '', '4', '3', '', '', '2', '', '9'],
       ['', '', '5', '', '', '9', '', '', '1'],
       ['', '7', '', '', '6', '', '', '4', '3'],
       ['', '', '6', '', '', '2', '', '8', '7'],
       ['1', '9', '', '', '', '7', '4', '', ''],
       ['', '5', '', '', '8', '3', '', '', ''],
       ['6', '', '', '', '', '', '1', '', '5'],
       ['', '', '3', '5', '', '8', '6', '9', ''],
       ['', '4', '2', '9', '1', '', '3', '', '']])
```

Let's practice!

INTRODUCTION TO NUMPY

Adding and removing data

INTRODUCTION TO NUMPY

Izzy Weber
Core Curriculum Manager, DataCamp

datacamp

Concatenating in NumPy

18	12	3
6	7	8
11	15	13

+

7	1
23	18
4	11

=

18	12	3	7	1
6	7	8	23	18
11	15	13	4	11

Concatenating rows

```
classroom_ids_and_sizes = np.array([[1, 22], [2, 21], [3, 27], [4, 26]])
new_classrooms = np.array([[5, 30], [5, 17]])
np.concatenate((classroom_ids_and_sizes, new_classrooms))
```

```
array([[ 1, 22],
       [ 2, 21],
       [ 3, 27],
       [ 4, 26],
       [ 5, 30],
       [ 5, 17]])
```

- `np.concatenate()` concatenates along the first axis by default.

Concatenating columns

```
classroom_ids_and_sizes = np.array([[1, 22], [2, 21], [3, 27], [4, 26]])
grade_levels_and_teachers = np.array([[1, "James"], [1, "George"], [3,"Amy"],
                                      [3, "Meehir"]])
np.concatenate((classroom_ids_and_sizes, grade_levels_and_teachers), axis=1)
```

```
array([['1', '22', '1', 'James'],
       ['2', '21', '1', 'George'],
       ['3', '27', '3', 'Amy'],
       ['4', '26', '3', 'Meehir']])
```

Shape compatibility

18	12	3
6	7	8
11	15	13

+

7	1
23	18
4	11
14	7

=

ValueError: all the input array dimensions for the concatenation axis must match exactly

18	12	3
6	7	8
11	15	13

+

7	1
23	18
4	11

=

18	12	3	7	1
6	7	8	23	18
11	15	13	4	11

Dimension compatibility

18	12	3
6	7	8
11	15	13

shape (3, 3)

+

7
23
4

shape (3,)

=

ValueError: all the input arrays must have same number of dimensions

18	12	3
6	7	8
11	15	13

shape (3, 3)

+

7
23
4

shape (3, 1)

=

18	12	3	7
6	7	8	23
11	15	13	4

Creating compatibility

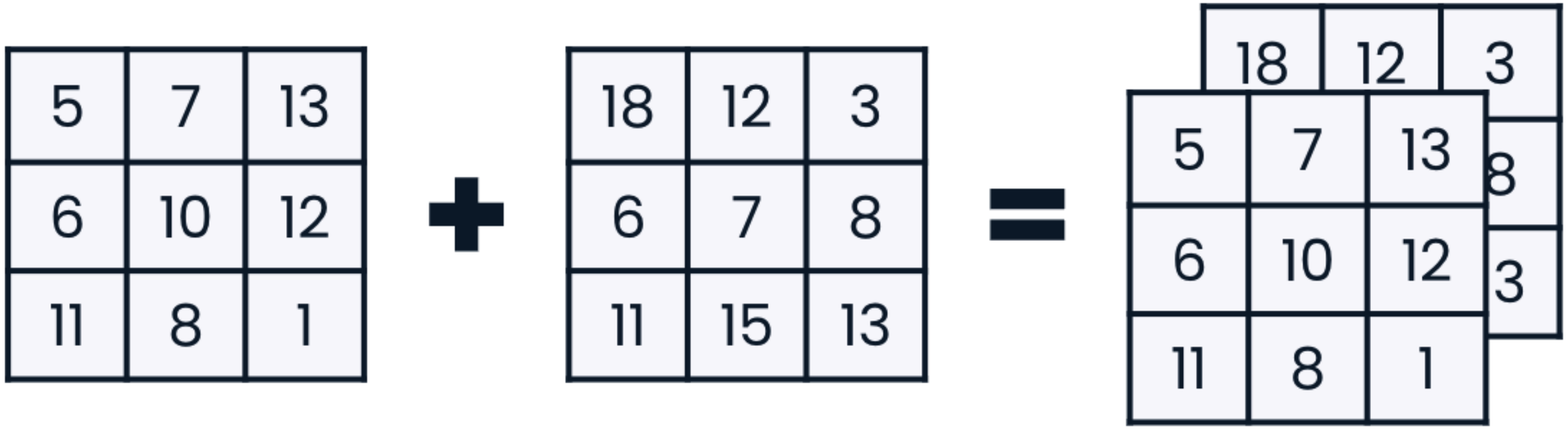
```
array_1D = np.array([1, 2, 3])
column_array_2D = array_1D.reshape((3, 1))
column_array_2D
```

```
array([[1],
       [2],
       [3]])
```

```
row_array_2D = array_1D.reshape((1, 3))
row_array_2D
```

```
array([[1, 2, 3]])
```

Concatenating new dimensions



✔ While this can be done...

🚫 Not with concatenate!

Deleting with np.delete()

```
classroom_data
```

```
array([['1', '22', '1', 'James'],
       ['2', '21', '1', 'George'],
       ['3', '27', '3', 'Amy'],
       ['4', '26', '3', 'Meehir']],)
```

```
np.delete(classroom_data, 1, axis=0)
```

```
array([['1', '22', '1', 'James'],
       ['3', '27', '3', 'Amy'],
       ['4', '26', '3', 'Meehir']])
```

Deleting columns

```
np.delete(classroom_data, 1, axis=1)
```

```
array([['1', '1', 'James'],
       ['2', '1', 'George'],
       ['3', '3', 'Amy'],
       ['4', '3', 'Meehir']]')
```

Deleting without an axis

```
classroom_data
```

```
array([['1', '22', '1', 'James'],
       ['2', '21', '1', 'George'],
       ['3', '27', '3', 'Amy'],
       ['4', '26', '3', 'Meehir']],)
```

```
np.delete(classroom_data, 1)
```

```
array(['1', '1', 'James', '2', '21', '1', 'George', '3', '27', '3', 'Amy',
       '4', '26', '3', 'Meehir'])
```

Let's practice!

INTRODUCTION TO NUMPY