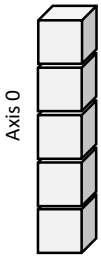
NumPy Cheat Sheets: Tips and Tricks

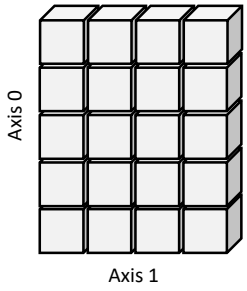
What's the Meaning of Axes and Shape properties?

1D NumPy Array



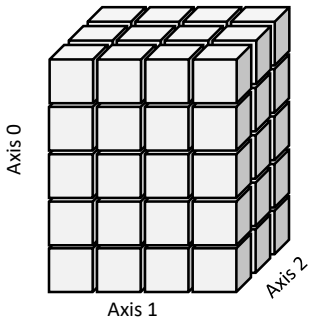
→ a.ndim = 1 *„axis 0"*
→ a.shape = (5,) *„five rows"*
→ a.size = 5 *„5 elements"*

2D NumPy Array



→ a.ndim = 2 *„axis 0 and axis 1"*
→ a.shape = (5, 4) *„five rows, four cols"*
→ a.size = 20 *„5*4=20 elements"*

3D NumPy Array



→ a.ndim = 3 *„axis 0 and axis 1"*
→ a.shape = (5, 4, 3) *„5 rows, 4 cols, 3 levels"*
→ a.size = 60 *„5*4*3=60 elements"*

What's Broadcasting?

Goal: bring arrays with different shapes into the same shape during arithmetic operations. NumPy does that for you!

```
import numpy as np

salary = np.array([2000, 4000, 8000])
salary_bump = 1.1

print(salary * salary_bump)
# [2200. 4400. 8800.]
```

- For any dimension where first array has size of one, NumPy conceptually copies its data until the size of the second array is reached.
- If dimension is completely missing for array B, it is simply copied along the missing dimension.

What's Boolean Indexing?

Goal: create a new array that contains a fine-grained element selection of the old array

```
import numpy as np

a = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

indices = np.array([[False, False, True],
                    [False, False, False],
                    [True, True, False]])

print(a[indices])
# [3 7 8]
```

We create two arrays "a" and "indices". The first array contains two-dimensional numerical data (=data array). The second array has the same shape and contains Boolean values (=indexing array). You can use the indexing array for fine-grained data array access. This creates a new NumPy array from the data array containing only those elements for which the indexing array contains "True" Boolean values at the respective array positions. Thus, the resulting array contains the three values 3, 7, and 8.

How to Search Arrays? The np.nonzero() Trick

Goal: find elements that meet a certain condition in a NumPy array

- Step 1: Understanding np.nonzero()

```
import numpy as np

X = np.array([[1, 0, 0],
              [0, 2, 2],
              [3, 0, 0]])

print(np.nonzero(X))
# (array([0, 1, 1, 2], dtype=int64), array([0, 1, 2, 0],
dtype=int64))
```

The result is a tuple of two NumPy arrays. The first array gives the row indices of non-zero elements. The second array gives the column indices of non-zero elements.

- Step 2: Use np.nonzero() and broadcasting to find elements

```
import numpy as np

## Data: air quality index AQI data (row = city)
X = np.array(
  [[ 42, 40, 41, 43, 44, 43 ], # Hong Kong
   [ 30, 31, 29, 29, 29, 30 ], # New York
   [ 8, 13, 31, 11, 11, 9 ], # Berlin
   [ 11, 11, 12, 13, 11, 12 ]] # Montreal

cities = np.array(["Hong Kong", "New York", "Berlin",
"Montreal"])

# Find cities with above average pollution
polluted = set(cities[np.nonzero(X > np.average(X))[0]])
print(polluted)
```

The Boolean expression "X > np.average(X)" uses broadcasting to bring both operands to the same shape. Then it performs an element-wise comparison to determine a Boolean array that contains "True" if the respective measurement observed an above average AQI value. The function np.average() computes the average AQI value over all NumPy array elements. Boolean indexing accesses all city rows with above average pollution values.