An Introduction to Numpy

Data Science 2 / Data & AI 3

Karel de Grote Hogeschool

Agenda

1. Introduction
2. Indexing, slicing and reshaping
3. Computation on NumPy Arrays
   - Vectorized operations
   - Agregations
   - Broadcasting
4. Boolean Arrays and Masks
5. Fancy indexing

Cursus/
Boek

canvas Oefeningen

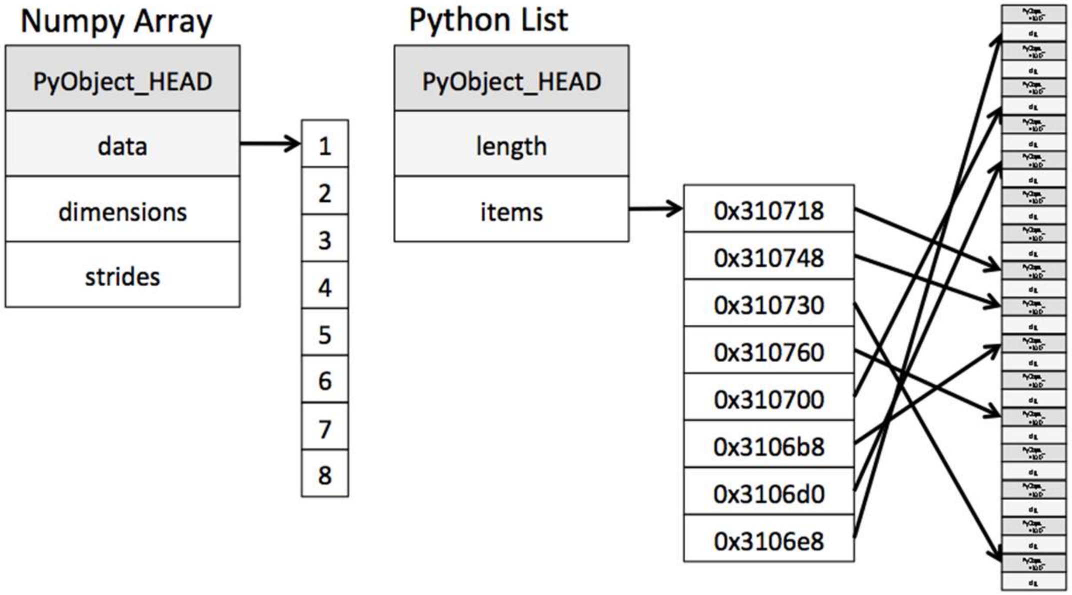
NumPy

1

Understanding data types

What is Numpy

- NumPy: package for scientific computing in Python
- Python library that provides multidimensional array
- Routines for fast operations on arrays,
  - Mathematical
  - Logical
  - Shape manipulation
  - Selecting
  - Basic linear algebra
  - Basic statistical operations



Array creation

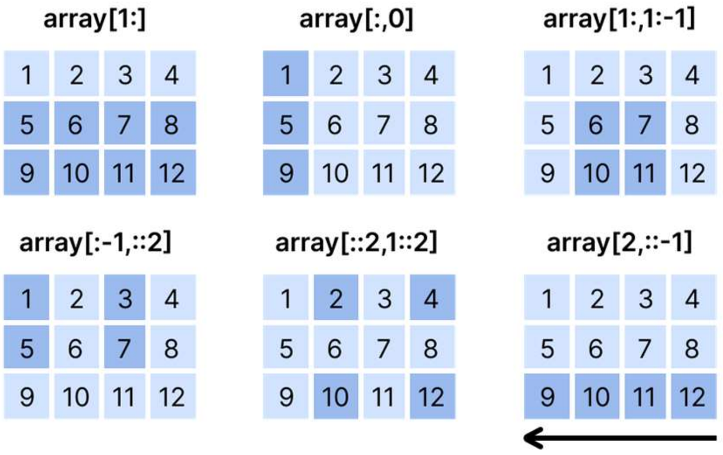
```
# import numpy package
import numpy as np


# Creating Arrays from Python Lists
a = np.array([1, 4, 2, 5, 3])
# Mind recent version of Python have a built-in package
“array” that also define a data type “array”.  That is way
we commonly import Numpy with alias “np” to make clear we
use Numpy arrays.


# Creating Arrays from Scratch
np.random.randint(0, 10, (3, 3))
np.zeros(3, dtype=int)        # array([0, 0, 0])
np.ones((2, 1), dtype=float)  # array([[ 1.],[ 1.]])
# random integers in [0, 10 )
np.arange(1, 5)               # [1, 2, 3, 4]
```

NumPy

2

The basics of Numpy arrays

Indexing, slicing and reshaping



Indexing, slicing and reshaping

```
# Array Attributes
x = np.array([[1, 2, 3],
              [4, 5, 6]])
print("x ndim: ", x.ndim)     # 2
print("x shape:", x.shape)    # (2, 3), 2 rows and 3 columns
print("x size: ", x.size)     # 6

# Array Indexing
x[1, 2]                       # 6
x[0]                          # [1, 2, 3]
x[-1]                         # [4, 5, 6]

# Array Slicing
x[0, :2]                      # [1, 2]
x[:, 1]                       # [2, 5]
x[:, :2]                      # [[1, 2], [4, 5]]

# Reshaping
np.arange(1, 10).reshape((3, 3))
```

Make sure you are fluent in indexing and slicing arrays!

Computation on NumPy Arrays

ufuncs

3

Vectorized operations

- Python for loops are slow
- Solution: Numpy’s vectorized operations and functions

When using Numpy & Pandas (later on) **using loops** is suboptimal and **will affect your evaluation scores negatively**!

```
# Array arithmetic
x = np.arange(3)         # [0, 1, 2]
2 ** x                   # [1, 2, 4]
x / np.arange(1, 4)      # [0, 0.5,  0.66666667]
-(0.5*x + 1)             # [-1, -1.5, -2]
np.power(3, x)           # [1, 3, 9]

# Aggregates
np.add.reduce(x)                # 3
np.add.accumulate(x)            # [0, 1, 3]
np.multiply.reduce(x)           #0
np.multiply.accumulate(x)       # [0, 0, 0]
```

Computation on NumPy Arrays

Aggregations

4

Agregations

```
x = np.array([[1, 5, 3],
          [4, 2, 6]])
x.sum                 # 21
x.min                 # 1
x.max(axis=0)         # [1, 2, 3],
                      # axis is dimension that is collapsed
x.max(axis=1)         # [1, 2]

# Statistics
x.mean(axis=0)        # [2.5, 3.5, 4.5]
x.var()
x.std(axis=1)
x.var()
x.median()            # as method
np.median(x)          # as function
np.percentile(x, 25)
```

NumPy

Computation on NumPy Arrays

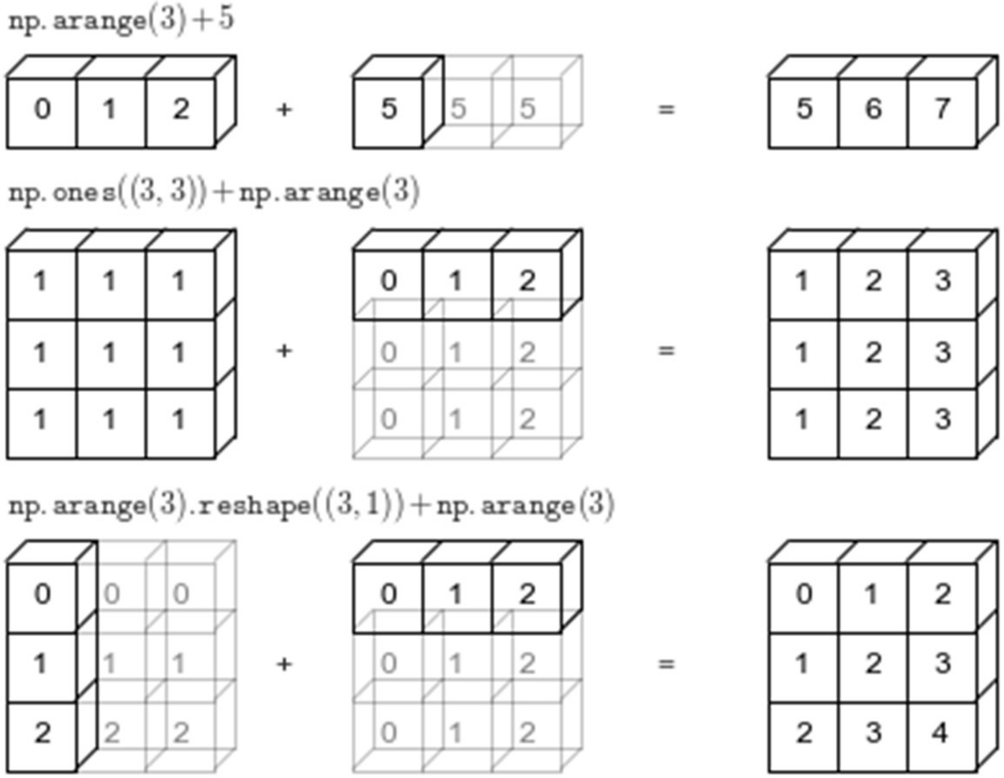
Broadcasting

5

Broadcasting

Rules

1. If the two arrays differ in their number of dimensions, the shape of the one with fewer dimensions is padded with ones on its leading (left) side
2. If the shape of the two arrays does not match in any dimension, the array with shape equal to 1 in that dimension is stretched to match the other shape
3. If in any dimension the sizes disagree and neither is equal to 1 -> error


np.arange(3)+5

np.ones((3, 3))+np.arange(3)

np.arange(3).reshape((3, 1))+np.arange(3)

Take some time to understand the broadcasting principles!

NumPy

6

Boolean Arrays and Masks

Boolean Arrays and Masks

```
x = np.array([[1, 5, 3],
          [4, 2, 6]])
x < 5                         # [[True, False, True], [True,
True, False]]
np.sum(x < 5)                 # 4, False -> 0, True -> 1
np.sum(x < 5, axis=1)         # [2, 2]
np.any(x < 5)                 # True
np.all(x < 5)                 # False
np.sum((x > 2) & (x < 5))            # 2

# Boolean Arrays as Masks
x[x < 5]               # [1, 3, 4, 2], returns 1-dim array
```

NumPy

7

Fancy indexing

Fancy indexing

```
x = np.arange(1,10)
ind = [3, 7, 4, 0]
x[ind]                      # [4, 8, 5, 1]
```