3. The libraries NumPy and SciPy

Markus Reinert[1]

Leibniz Institute of Baltic Sea Research Warnemünde (IOW)

30 April 2021

[1] ✉ markus.reinert@io-warnemuende.de

Overview of today's course

Python course 3

- Installation of NumPy and SciPy
- Overview of NumPy and SciPy
- Comparison of lists and arrays
- Creation of arrays
- Usage of arrays (see the course notebook)

Installation of NumPy and SciPy

- If you are on conda: already installed.
- Otherwise, run in a terminal:
  `$ python -m pip install numpy scipy`
  (You might need to write python3 instead of python.)
- More information: `https://scipy.org/install.html`

Write at the beginning of every script where you use NumPy:

```
import numpy as np
```

and if you use anything from SciPy:

```
from scipy import stats, optimize, special, ...
```

(import only what you need)

Overview of NumPy and SciPy

NumPy

`https://numpy.org/doc/stable/reference/`

- common mathematical functions: `np.sqrt(x)`, `np.sin(x)`, `np.cos(x)`, `np.exp(x)`, ...
- basic statistics: `np.mean(a)`, `np.std(a)`
- random numbers with simple distributions: `np.random.randint(n)` (random integer from {0, 1, ..., n-1})
- basic linear algebra
- solving linear or polynomial equations
- arrays (→ next slide)

Scipy

`https://docs.scipy.org/doc/scipy/reference/`

- less common functions: `special.gamma(x)`, Bessel functions, ...
- advanced statistics: `stats.skew(x)`, `stats.norm.pdf(x)`
- random numbers from complicated distributions: `stats.poisson.rvs(mu)`
- advanced linear algebra
- solving "any" equation (for example with `optimize.newton`)

Comparison of lists and arrays

Lists	Arrays
one-dimensional (but can contain other lists)	zero-, one-, two-, . . . , *n*-dimensional
can be extended	have a fixed length / shape
can contain any data	can contain only one data type
data access with indices	data access with indices or conditions
list-operations (count, index, . . .)	mathematical operations (efficiently)

Typical ways of creating arrays

1. from a list (or similar): `np.array(l)`

2a. filled with 0s or 1s: `np.zeros(shape)`, `np.ones(shape)`

2b. filled with 0s or 1s, but with shape and data type of another array:
`np.zeros_like(a)`, `np.ones_like(a)`

3a. numerical 1D-ranges:
`np.arange(start, stop, step)`, `np.linspace(start, stop, num)`

3b. to go from 1D-ranges to 2D: `X, Y = np.meshgrid(x, y)`

4. from a text file: `np.loadtxt(file_name)`

Usage of arrays (see the course notebook)

- determine size, shape, and number of dimensions
- selection of single and multiple entries, rows, columns
- mathematics
- broadcasting
- selection by conditions