University of Zurich[UZH]

**Department of Physics**

# Where the fun really starts – Useful python libraries for scientific computing

Scientific Programming with Python

Christian Elsasser

# Setup

The idea in this lecture is that you can directly follow the commands in the jupyter notebok to better understand what we can achieve with the discussed libraries

- Download material
  `$ wget http://www.physik.uzh.ch/~python/python/lecture7/material_usefullib_lec.tar.gz`
- Unpack it
  `$ tar xvf material_usefullib_lec.tar.gz`
- Enter the directory
  `$ cd material_usefullib_lec`
- . . . and open the notebook
  `$ ipython notebook Lecture.ipynb`
- Also you need to install `feedparser`
  `$ sudo aptitude install python3-feedparser`

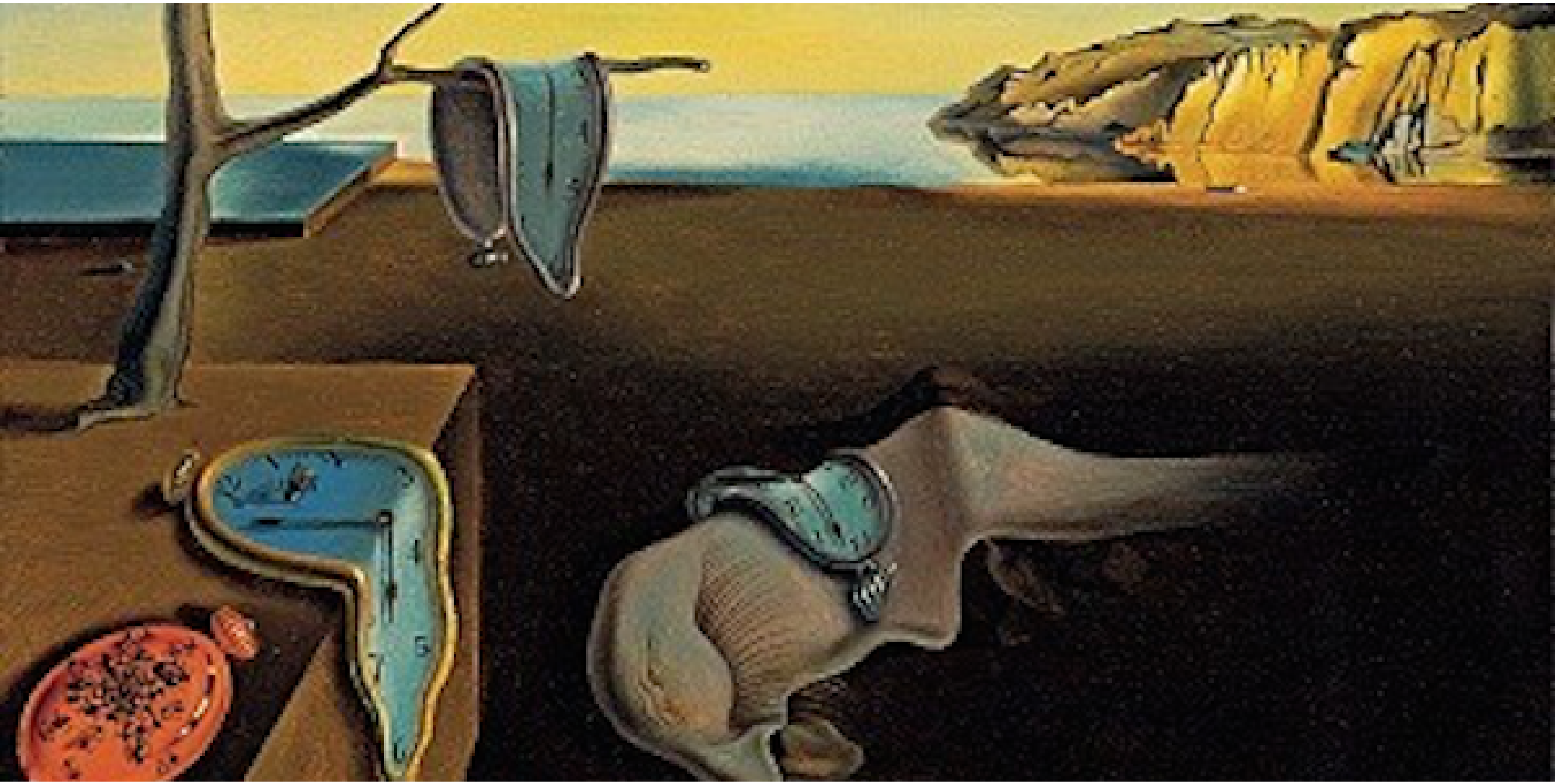
# Introduction

- So far rather high-level topics; now it is time to learn more about the right tools!

# Covered Areas

- Time & date
  - `datetime`
- Data handling
  - `csv`
  - `json`
  - `sqlite3`
  - `pandas`
- Web tools
  - `requests/urllib`
  - `feeder`
  - `BeautifulSoup`
- Advanced visualisation
  - `seaborn`
  - `bokeh`

# Time & Date



## `datetime`

https://docs.python.org/3.4/library/datetime.html

- Collection of classes to manipulate date and time
- Most important class `datetime` to represent date (year, month, day) and time (hour, minute, second, millisecond)
- `strptime` and `strftime` to load and dump dates from and to a string, respectively $\rightarrow$ format defined via standard time fields (*i.e.* %Y for four-digit year, %b for three-letter month abbreviation, etc. using locale information)
- Timezone info encodable via abstract base class of `tzinfo`, *e.g.* `pytz`
- `timedelta` as difference between `datetime` objects allowing to make calculations

# Data Handling

## `csv`

https://docs.python.org/3.4/library/csv.html

- Module to read and write data from a “comma”-separated file object (`csv.reader` and `csv.writer`)
- Specifications format (delimiters, string identifier, etc.) via keyword arguments
- Processing of in- and output needs to be sanatised manually
- Dialect/format can be also “sniffed” by (`csv.Sniffer`)
- `csv.DictReader` and `csv.DictWriter` allow to read into dictionnaires with header information and to dump dictionnaries

# `json`

https://docs.python.org/3.4/library/json.html
http://docs.python-guide.org/en/latest/scenarios/json/

- Interface between python's data structure and the JavaScript Object Notation (JSON)
- Translation using `json.dumps` and `json.loads` from and to strings
- ... or directly into or from a file using `json.dump` and `json.load`
- Possibility to define parsing rules via derived classes from `JSONEncoder` and `JSONDecoder`

# json

https://docs.python.org/3.4/library/json.html
http://docs.python-guide.org/en/latest/scenarios/json/

- Interface between python's data structure and the JavaScript Object Notation (JSON)
- Translation using `json.dumps` and `json.loads` from and to strings
- . . . or directly into or from a file using `json.dump` and `json.load`
- Possibility to define parsing rules via derived classes from `JSONEncoder` and `JSONDecoder`

## Python ↔ JSON

| | | |
|---|---|---|
| `dict` | ↔ | `object` |
| `list (tuple)` | ↔ | `array` |
| `None` | ↔ | `null` |

- All encoded data are python-internally treated as string!

# sqlite3

https://www.sqlite.org

*What is SQLite?*

- SQLite is a light-weight (= server-less) SQL-type (= spreadsheet-based) database system.
- The database processes are direclty embedded in the front-end program.
- Due to the outsourced write-interlock handling write intensive programs will suffer.
- Some SQL commands are not valid (removing column, rename column).

# sqlite3

https://www.sqlite.org

## A few typical (SQL) commands

| Purpose | Command |
|---|---|
| Retrieve all the data from a table | `SELECT * FROM <table>` |
| Retrieve some columns (`c1`,`c2`) from a table `t` with a condition | `SELECT c1,c2 FROM t WHERE <cond>` |
| Group entries according to values | `SELECT SUM(c1),AVG(c2) FROM t GROUP BY c3,c4` |
| Add new entry | `INSERT INTO t (c1,c2) values (v1,v2)` |
| Delete an entry | `DELETE FROM t WHERE c1=v1 AND c2=v2` |

## `sqlite3`

https://docs.python.org/3.4/library/sqlite3.html

- Database operations on sqlite3 databases
- `sqlite3.conntect` to get a handler on the database
- Default output of (part of) a row is a list → possibility to change the behaviour via the `row_factory` variable of the database
- Use `?` as placeholder insted of concatenating the SQL command by Python string operations
- Use `executemany()` to run same SQL command with several parameter sets
- All executed commands need to be commited before closing the connection (`<dbvariable>.commit()`)

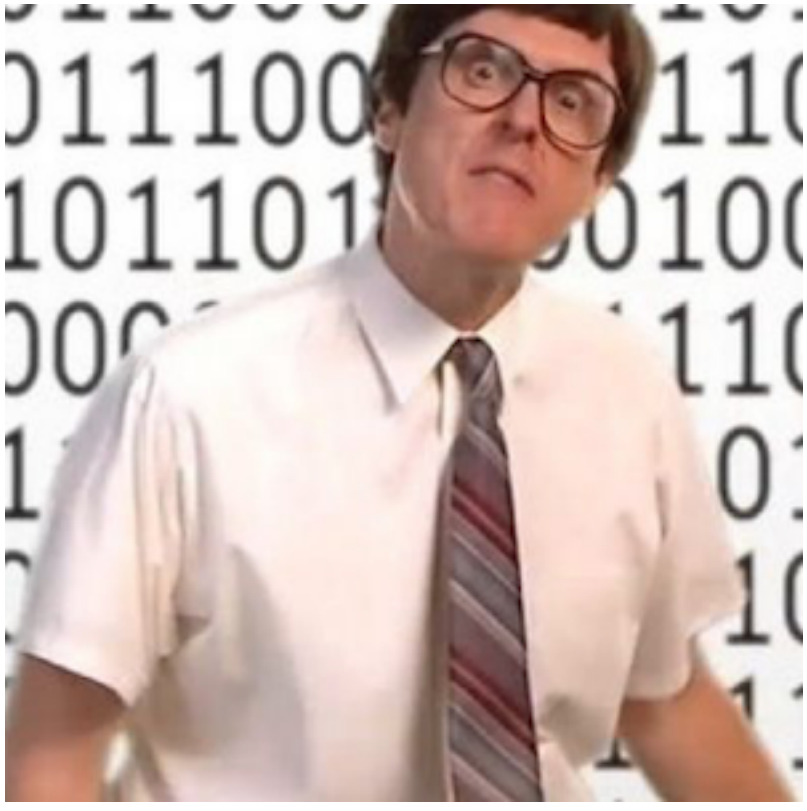
# pandas

https://pandas.pydata.org

- Offering data containers plus corresponding functionality
- *e.g.* Time series `pd.Series` and their notorious functions (*i.e.* rolling-“whatever”-you-want function)
- Many SQL-like data operations (group, merge)
- Interface to a large variety of file formats (Nobody has heard about all of them!)
- Data interface/API to many data repositories (Yahoo Finance, FRED)

# numpy *vs.* pandas

Numpy



fast and good with numbers

Pandas

a bit slow and cool with everything

# Web Tools

# `requests` / `urllib`

http://docs.python-requests.org/en/master/
https://docs.python.org/3.4/library/urllib.html

**requests**

- User-friendly module for HTTP functionality
- POST and GET (and the others) functionality ($\rightarrow$ extraction of web site content, download of files, low-level handling of APIs, etc.)
- Possiblity to specify sessions (`requests.Session`)
- Submission of additional parameters to specifiy proxy, authentification, etc.

**urllib**

- For some functionalities we need to fall back to `urllib`
  - Download files easily
  - Retrieve data from files iteratively

## `feedparser`

https://pythonhosted.org/feedparser/introduction.html

- Handling of RSS and Atom feeds
- Metadata of the news items including short description plus link to the full text

## BeautifulSoup

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

- Parsing of HTML or XML files into a tree structure
- Selection of sections based on tags including their attributes (class, id, name, etc.) possible
- Also extraction of attributes possible (*e.g.* href field for HTML links)
- `parent`, `children`, `siblings` methods allow to navigate in the structure of the document

# Visualisation

## seaborn

https://web.stanford.edu/ mwaskom/software/seaborn/

- Collection of advanced plots (including some analytics capabilities)
  - Heat-map
  - Correlation (incl. clustering)
  - Distributions
  - Kernel density
  - Violin plot
- Many single commands for a combination of plots (with all the good and evil behind it!)
- Based on matplotlib $\Rightarrow$ deep down there are matplotlib options that can/need to be configured
- Fast option to create advanced standard plots (*i.e.* you can spend a lot of time to configure them to your needs)

# bokeh

http://bokeh.pydata.org/

- Interface to build (interactive) JavaScript-based plots and tables
- Interaction via
  - Plot tools (Zoom, selection, mousetip tools, etc.)
  - Buttons, text fields, sliders, drop-down selections, etc.
  - JavaScript
- Alternative to JavaScript's standard approaches (*e.g.* D3.js, Chart.js or home-made tables with jQuery)
- Build-up of standalone HTML files or collection of HTML div's (or embedded in a jupyter/ipython notebook)
- Toolbox for user interactions (zoom, crosshair, selections, etc.) as well as additional interactions (buttons, sliders, etc.)
- Possibility to embed Google maps for the display of geographical locations
- Includes a server to handle client requests

# Advanced Python modules

We omitted any modules with a large and specific purpose → otherwise you would sit here tomorrow

Left to the interested audience to explore them further

- NLTK (www.nltk.org) → Natural language processing
- scikit-learn (scikit-learn.org) → Machine learning
- scikit-learn (scikit-image.org) → Image processing and analysis
- ...

Rapidly growing and improving landscape of python modules, but with still some “whitish” spots (*e.g.* time series) ⇒ Reflection of available alternatives?

# Conclusion

- Large variety of modules (growing every day), not just data analysis, but also for web interface, etc.
- Many packages targeting APIs
  - Twitter → `tweepy`
  - Yandex translator → `yandex.translate`
  - Quandl → `quandl`

  ⇒ Do not reinvent the wheel!
- `pip` is your friend and helper
  - Local installation: `--user` installs it to the `$PYTHONINSTALLPATH`
  - Behind proxy: `--proxy=<proxyname>` *e.g.* `gateway.uzh.ch:7777`
- Learning by doing!
- . . . But knowing what functionalities are available and their potential is half the battle!