FAIM Python Course – Session 2
Data Handling & Analysis

benjamin.titze@fmi.ch

Outline

Prerequisites: Python 3.6+, Jupyter Notebook, NumPy, SciPy, pandas, Matplotlib

- Could you run the code from *pre-check.py* inside your Jupyter Notebook?
- You may need to install the required packages: `pip install numpy`, …

I. How Python handles data

- Data types / Immutable and mutable objects
- Passing by reference vs passing by value

II. Brief intros to the core Python data science libraries

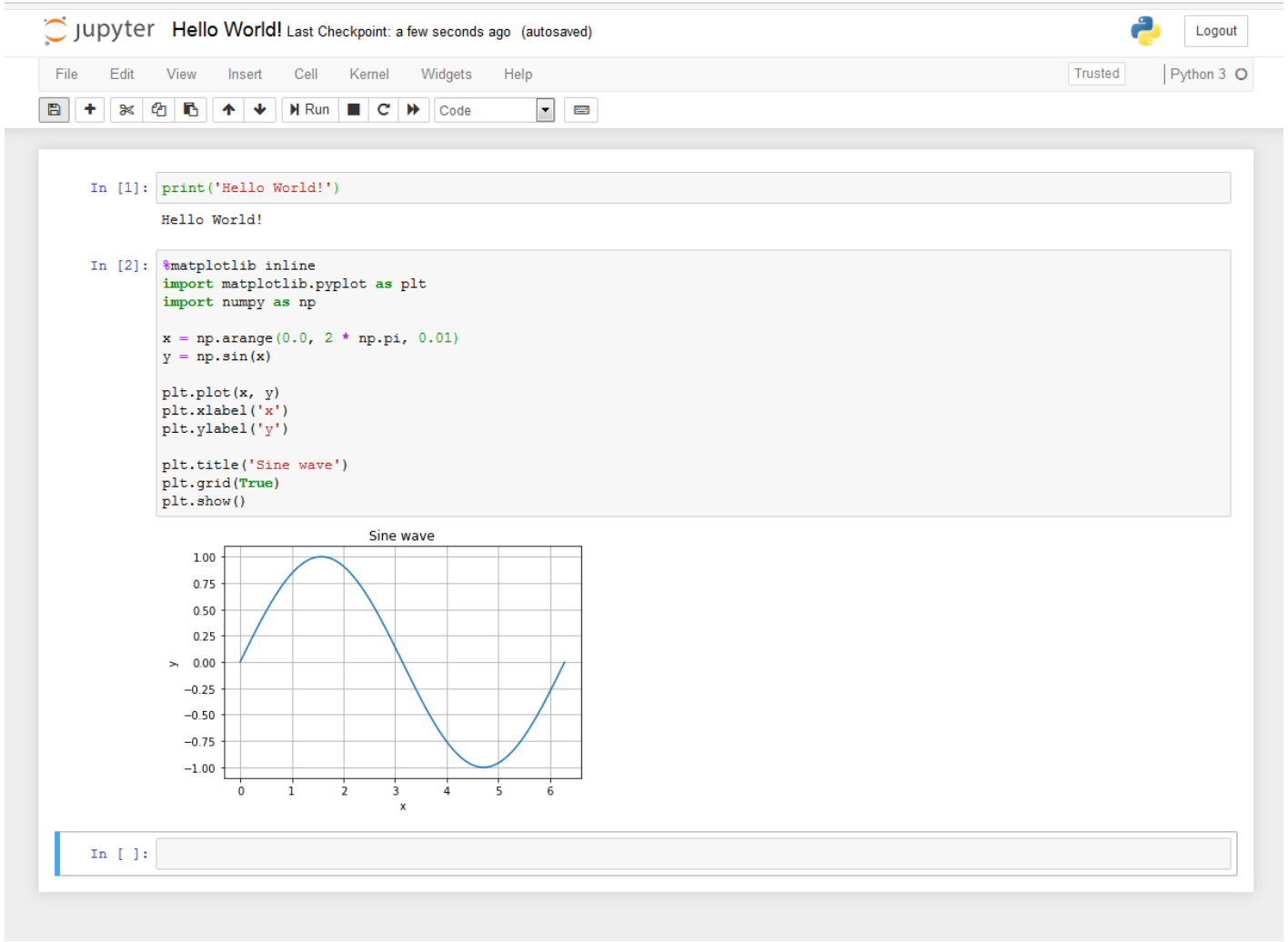
- Processing data with NumPy, SciPy and pandas
- Visualization with Matplotlib

III. Working with an example dataset (a movie database)

Duration: ~90 min, max. 2h

Jupyter Notebook

Browser-based interactive programming environment (previously IPython Notebook)



Named after the core languages supported: **Ju**lia, **Py**thon, and **R**.

A common tool for data science / scientific computing in both academia and industry.

Run from command line: `jupyter notebook` or launch via Anaconda Navigator.

Start up Jupyter Notebook and open a new Python 3 notebook!

Overview: Python data types

- `int` – Integer (whole numbers, like *`253`*, *`-12`*)
- `float` – floating-point numbers such as `-243.1998`
- `bool` – Boolean, named after mathematician George Boole: *`True`* or *`False`*
- `str` – String of characters, for example: `'FMI'`, `'世界'`, `'2 tables'`

The above are Python's basic *immutable* built-in types.

- `tuple` – an immutable sequence of objects: `('a string', 89, False)`

The following are *mutable*:

- `list` – a sequence of objects, e.g., `[5, 6, 7, 1]` or `[True, 'true']`
- `dict` – a 'dictionary', an unordered mapping of keys to values:
  `{'name': 'Anne', 'age': 29}`
- `set` – an unordered set of distinct objects: `{'cow', 'dog', 'horse'}`

Fundamental concepts

- In Python, everything is an *object*!
- Use `type()` to return the type of any object. You will notice that even the basic data types are *classes*, for example: `<class 'int'>` (in Jupyter Notebook, only `int` is shown)
- Use `id()` to return the 'identity' of an object, which is its location in memory (for CPython). It's unique and constant during an object's lifetime.
- **Mutable objects**, for example *lists*, can be changed (= modified in place) and will remain pointed to a fixed location in memory.
- **Immutable objects**, for example *ints*, cannot be changed, only reassigned to new values. If reassigned, they point to a new location in memory.

Let's look at concrete examples in our Jupyter Notebook!

→ datatypes.ipynb

Passing by reference vs passing by value

- When you call a function in Python, pay attention to whether you're passing arguments that are *mutable* or *immutable*.
- Passing a mutable object, for example a dictionary, means that you are passing a ***reference*** to the object. If you change it inside the function, the object will also be changed outside the function!
- If you pass an immutable object, for example a string: In this case, the function can only use the ***value*** of this string. If the object is changed inside the function, it has no effect on the outside scope.

Examples
→ datatypes.ipynb

Python's data science ecosystem: the core libraries

(pronounced *Num Pie*) – https://numpy.org
A library for fast manipulation of multidimensional numerical data. The basic type *ndarray* is a multidimensional array with zero-based indexing.

→ numpy-intro.ipynb

(pronounced *Sigh Pie*) – https://www.scipy.org
A scientific computing library (built with NumPy arrays) with various modules for numerical integration, linear algebra, image processing, Fourier transforms, signal processing, statistical functions…

→ scipy-intro.ipynb

pandas (**pan**el **da**ta **s**tructures) – https://pandas.pydata.org
Various tools to work with structured data sets (similar to SQL/Excel functionality). Useful for handling tables, for example measurement results. The basic types are *DataFrame* and *Series*.

→ pandas-intro.ipynb

https://matplotlib.org
MATLAB-like visualization support through the pyplot module: 2D and 3D plots, histograms, multipanel figures; works in Jupyter Notebooks.

→ matplotlib-intro.ipynb

Conventional import abbreviations

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Please use these abbreviations consistently! If you don’t, it will confuse others who are reading your code…

For SciPy, just import the module you need, for example:

```
from scipy import fftpack
```

or

```
import scipy.fftpack
```

Please make sure these imports work. Install the libraries if necessary.

Brief introductions to NumPy, SciPy, pandas and Matplotlib,
+ hands-on exercise with example dataset

→ Jupyter Notebook

For the hands-on exercise, load this dataset with pandas and explore it:

movies_dataset.csv

```
import pandas as pd
df = pd.read_csv(…)
df.head(…)
…
```