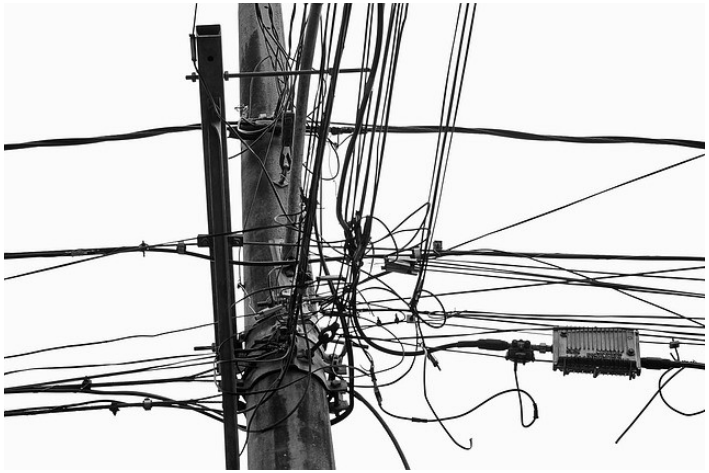
Table of contents (../toc.ipynb)

Python installation

Before we are able to start with coding, we need some knowledge about Python installation.



First, there is Python 2 and Python 3. The last Python 2 version is Python 2.7.17 and was released in October 2019. Official support for Python 2 has stopped on January 2020, see this note (https://www.python.org/dev/peps/pep-0373/).

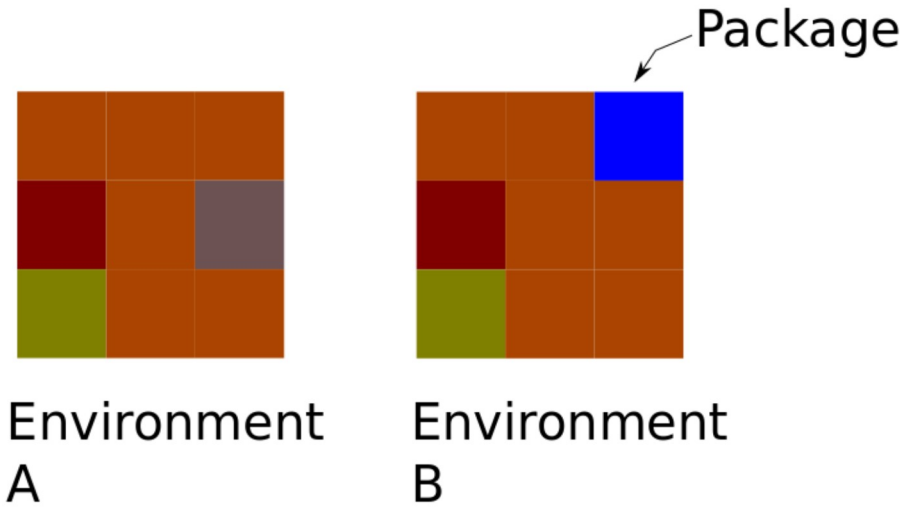
Python 3 was released in 2008, but it took quite some time to convert all libraries from Python 2 to Python 3.

Some packages might still lack behind. If you need to learn how to convert Python 2 to Python 3, have a look here (http://python3porting.com/bookindex.html).

Anyway, we will focus on **Python 3**.

Pip vs Conda

Luckily, Python is supported by package management and environment systems. These programs allow to define, share and resolve Python configurations. Hence, you can share your specific setup with someone very easily.



There are two major Python package installers:

- **Pip** (https://pypi.org/project/pip/)

- **Conda** (https://docs.conda.io/en/latest/)

- Pip is a lightweight Python only package manger.
    - Pip covers packages from Python Package Index **pypi** (https://pypi.org/).
    - Pip does not support virtual environments out of the box (requires virtualenv or venv package).
    - Pip does not check dependencies. Hence, you need to take more care when defining environments.

- Conda is a general package manager (more than just Python) and a bit more heavy weight.
    - Conda covers Python packages from **Anaconda cloud** (https://repo.anaconda.com/pkgs/).
    - Conda checks dependencies.
    - Conda allows to install nearly all tools which are required for data science.
- Let us start with Conda (Pip commands are very similar):

Exercise: Install and try Miniconda (5 minutes)

- There is a lightweight version of Conda, called Miniconda. Please install it from https://repo.anaconda.com/ (https://repo.anaconda.com/)
- Use `Miniconda3-latest-Windows-x86_64.exe` or `Miniconda3-latest-Linux-x86_64.sh`
- On Windows, open a `Anaconda prompt`, just press `Win` key and type `anaconda prompt` to find it
- Type `conda --version` to see if conda is installed

Exercise: Learn Anaconda commands (20 minutes)

- Please follow the tutorial of Anaconda to learn how to work with environments: bit.ly/tryconda (https://bit.ly/tryconda)
- Take a look at Coda cheat sheet (https://docs.conda.io/projects/conda/en/latest/user-guide/cheatsheet.html) to learn more conda commands

Working with requirements files

- It is very common to share a python environment through a `requirements.txt` file.
- Souch a file could look like this

Content of `requirements.txt`

```
python=3.6.2
numpy
```

Exercise: Work with requirements.txt (10 minutes)

- Create a folder
- Open ananconda prompt (or use pip) and navigate to this folder
- Create a `requirements.txt` file with text from above
- Create a virtual enviroment with `conda create -p .\_venv --file requirements.txt`
- Activate this environment
- Check installed packages with `conda list` and `python --version`
- Add `matplotlib` to list in `requirements.txt`
- Install new package with `conda install --file requirements.txt`

Conda channels and the .condarc file

- Conda packages are organized in channels
- For instance, the widely used sklearn package is hosted in anaconda channel, not in default channel

Exercise: Work with channels (5 minutes)

- Try to install sklearn in your latest environment with `conda install scikit-learn`
- If it fails with `PackagesNotFoundError`
    - Find sklearn in https://anaconda.org/anaconda/scikit-learn (https://anaconda.org/anaconda/scikit-learn)
    - Use conda install with channnel `conda install -c anaconda scikit-learn`
- Add this channel to `.condarc file` with `conda config --add channels anaconda`
- View `.condarc` file in your user home directory

Exercise: Take a look at Pip docu (5 minutes)

- Please open the documentation of Pip (https://packaging.python.org/tutorials/installing-packages/) and compare the commands between Conda and Pip
- Try `python --version` and `pip --version` commands in a terminal

Play with Python

Now that we have a running Python installation, we can start to run some Python code.

- The Python interpreter can be entered with `python`
- Each line starts now with `>>>`
- Exit to normal shell with `exit()`

Exercise: Python (2 minutes)

- Activate your environment (if not done yet)
- Type `python`
- Type `print("hello world")`
- Type `a = 1`
- Type `b = 3`
- Type `c = a + b`
- Type `print(c)`
- Go to shell with `exit()`

```
(C:\Users\Stephan\PycharmProjects\tmp\_venv) C:\Users\Stephan\PycharmProjects\tm
p>python
Python 3.6.2 |Anaconda, Inc.| (default, Sep 30 2017, 11:52:29) [MSC v.1900 64 bi
t (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
>>> a = 1; b = 3; c = a + b
>>> print(c)
4
>>> exit()

(C:\Users\Stephan\PycharmProjects\tmp\_venv) C:\Users\Stephan\PycharmProjects\tm
p>
```

IPython

Although we could directly use Python now after this installation in a terminal, it is more convenient to use IPython (https://ipython.readthedocs.io/en/stable/overview.html), which is a powerfull interactive Python shell. IPython is an enhanced interactive Python shell which supports

- Tab completion
- Object introspection
- System shell access
- Searching through modules and namespaces with `*` wildcards
- ...

Exercise: IPython (2 minutes)

- Install ipython in your environment with `conda install ipython`
- Open ipython with `ipython`
- Type `print("hello world")`
- Exit ipython with `quit`

```
(C:\Users\Stephan\PycharmProjects\tmp\_venv) C:\Users\Stephan\PycharmProjects\tm
p>ipython
Python 3.6.2 |Anaconda, Inc.| (default, Sep 30 2017, 11:52:29) [MSC v.1900 64 bi
t (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.10.2 -- An enhanced Interactive Python. Type '?' for help.

In [1]: print("hello world")
hello world

In [2]: quit
```

Run Python files

The interactive Python shells are convenient for small experiments but we need to learn how to run longer series of Python commands from Python files.

The default Python file extension is `.py` and here is the content of a `test.py` file.

```
# file test.py

for i in range(9):
    print("Hello world times", i)
```

What does this script?

Exercise: Run a Python script (2 minutes)

- Run `test.py` with `python test.py` (note that we are in shell not in python interpreter)
- Enter IPython with `ipython`
- Run the file again with `%run test.py`

```
(C:\Users\Stephan\PycharmProjects\tmp\_venv) C:\Users\Stephan\PycharmProjects\py
-algorithms-4-automotive-engineering\00_intro>python test.py
Hello world times 0
Hello world times 1
Hello world times 2
Hello world times 3
Hello world times 4
Hello world times 5
Hello world times 6
Hello world times 7
Hello world times 8
```