Getting started with Python

- Google colab: https://colab.research.google.com/notebooks/welcome.ipynb You find a ready-to-go computing environment here. You can start notebooks from scratch, upload notebooks, and save them to your google drive. This tutorial is based on google colab!

- Run Python on your own PC (advanced)

    - If you're using a Linux system like Ubuntu, Python3 is probably already pre-installed or you can easily install it with any linux package manager. In addition, `conda` (https://docs.conda.io) or `pip` (https://pypi.org/project/pip/) should be installed, which are both Python package managers. These you'll need for downloading and installing external Python software like e.g. `numpy`.
    - If you're using Windows, we recommend to use "WSL" (Windows Subsystem Linux, https://learn.microsoft.com/de-de/windows/wsl/install) and install Python and `conda` there. If you're using a Windows PC, I recommend using WSL and VS Code! (see below)
- Recommended standard packages: `numpy`, `scipy`, `matplotlib`, `jupyterlab`. See also how to use virtual environments and install Python packages below.

More Tutorials

- https://www.codecademy.com/
- https://www.w3schools.com/python/
- https://exeter-data-analytics.github.io/python-intro/first_script.html

Python as a script language

Notebooks are of course not the only way to write and execute Python code. Notebooks are fine for smaller projects and calculations and especially to get started! But once you start to build larger projects or you need to work with other people's code and collaborate on software project, you also need to write and use Python scripts (text files ending with .py) and know how to import the content of these scripts into your code. In principle, this works like a notebooks with only one cell (and usually not interactive). Usually, all functions and classes of a software package are defined in .py files, and notebooks are meant for interactive use and prototyping.

This all goes beyond the scope of this tutorial, but you can find e.g. more information here:

- https://exeter-data-analytics.github.io/python-intro/first_script.html
- https://python-packaging-tutorial.readthedocs.io/en/latest/setup_py.html

Advanced editors

- `Jupyterlab` : https://jupyter.org/ is a browser-based programming environment. It looks similar to Google colab, but it's hosted on your own computer (so no Internet connection needed!). You can use it for both Python scripts and notebooks.
- `Visual Studio Code` : https://code.visualstudio.com/ is a free, lightweight programming environment by Microsoft. It runs on both Windows and Linux systems. You can use it under Windows to access your Python installation in WSL (see above). If you're using a Windows PC, I recommend using WSL and VS Code!

Virtual Environments

Virtual environments allow encapsulation of python packages. This is useful if your analysis scripts depend on different versions of e.g. numpy, or you would like to try out experimental updates of certain packages without corrupting your analysis environment. This may be important if you are using packages that are still being developed. You may want to use the same environment in one year from now, or you may want to check easily if your analysis changes with new packages.

VS code recognizes virtual environments and enables easy use of the packages installed therein.

conda

(my recommendation, it's currently the easiest setup I ever did)

Step by step introduction: https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

Summary:

- `conda create --name myenv` where myenv is your chosen name for the virtual environment (could be anything, e.g. your project name). Follow the steps during creation of the env.
- `conda install -n myenv scipy` installs the `scipy` package into your env.
- Start the environment with `conda activate myenv`. The commandline should now begin with the name `(myenv)`. The env is now ready to use.
- Deactivate it with `conda deactivate`.

pipenv

Another easy way to work with environments is with `pipenv`: https://pipenv.pypa.io/en/latest/install/#installing-pipenv

Follow the instructions there to built a new environment for your project. Summary:

- First step: `pip install --user pipenv`. The `--user` option ensures that `pipenv` is separated better from your system environment.
- Go to your project folder, this will then be connected to this particular environment.
- First package + building the env: `pipenv install numpy`
- You can install more packages with `pipenv install scipy` (and more, of course)
- Start the environment with `pipenv shell`. You should see the commandline beginning with `(your_env_name)`. The env is now ready to use.
- Deactivate it with `deactivate`.

Code versioning

Once you are coding more frequently, you might want to start using versioning and development platforms like https://github.com/ . Again, this is beyond the scope of this tutorial, but think about building your software in your own GitHub repository to

1. Save your progress frequently and be less scared about losing any scripts or messing up your programs.
2. Work collaboratively.
3. Share and publish your code.
4. ... use many other helpful tools provided by GitHub.

Summary of resources and tutorials

(probably incomplete)

General tutorials:

- https://www.codecademy.com/
- https://www.w3schools.com/python/
- https://exeter-data-analytics.github.io/python-intro/first_script.html
- https://beapython.dev/2019/12/23/writing-your-first-python-script/

Python setup:

- https://www.scaler.com/topics/python/install-python-on-linux/
- https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html
- https://pipenv.pypa.io/en/latest/install/#installing-pipenv
- https://python-packaging-tutorial.readthedocs.io/en/latest/setup_py.html

Editors:

- https://code.visualstudio.com/
- https://jupyter.org/
- https://www.sublimetext.com/
- https://www.vim.org/ (very old-school)

Specific tutorials:

- numpy: https://numpy.org/doc/stable/user/quickstart.html
- data visualization: https://matplotlib.org/stable/tutorials/index.html (matplotlib), https://seaborn.pydata.org/tutorial.html (seaborn)
- scipy: https://www.tutorialspoint.com/scipy/scipy_quick_guide.htm
- pandas: https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html

In [ ]: