Intro transient course using Python for the exercises

Transient Groundwater Flow Couse at IHE, Delft, Feb 2024

Prof. dr.ir. T.N.Olsthoorn

2024-02-08

0.1 Intro, why using Python

We will use *Python* for the exercises. We'll do that in so-called *Jupyter notebooks*. It's by far the most convenient environment to explore the analytical solution we're dealing with in our course and the relations between parameters as well as their impact on the behavior of the solutions. Instead of filling in some numbers in a given expression to get for instance the drawdown by a well at a certain time and distance, i.e. one point at a time, with Python we can deal with all points and all times combined and see the relation in graphical form. This is the modern way of studying theory, namely by exploring it in depth using compute software that allows us to deal effortlessly with large number of values, i.e. not with individual points but with entire graphs at once. It provides more insight and also invites exploring what-if scenario's, which futher deepens the insight in the character of the analytical solutions we'll discover in the course.

Although exploring solutions looks like numerial work, it isn't, it is bringing analytical expressions to live in graphs and show their generalities and general behavior. Numerical analysis normally means modeling groundwater flow by invoking computer code that takes a large volume of data, spread out in numbers over a large number of computation cells or nodes, yielding, after convergence was successful, the heads in all computation cells or nodes as well as their water budget over time. Numerical models normally produce just numbers, no insight. Insight comes from analyis, i.e. from using analytical solutions that encompass and show behavior of the underlying groundwater system, allowing prediction of what will happen if some parameters change as well as generalizing statements for other systems, which cannot be obtained from numerical simulation.

Of course, numerial groundwater models can deal with more complex systems and more varied input, but cannot provide the insight that comes from analytical solutions, although the latter generally only handles simpler systems. In practice both numerical and analytical modeling and solutions go hand in hand, and we can conviniently use anaytical solutions to investigate a groundwater system in more general form, without being too much off the answer given by the more complex numerical model. On the onther hand, analytical solutions are used to verify numerial models or parts thereof in an independent way.

Using Python allows us convenient in-depth visual analysis of relations in and between analytical solutions and often of their generalizations. Python is nowadays the world's most used data science language and we can use it on our computers for free. We only have to practice it, which means, to use it for as many purposes as we can, to become good at it, even better than we were in Excel, which is also a good tool, but much more limited.

0.2 Installing python and Jupyter on your computer

Nowadays, installing python has become easy. The most famous site for downloading and installing Python for data scientists is Anaconda. It brings a complete enviroment, with a large number of programs and over 1000 packages that may be useful if your are

a data scientist. You don't have to wory about internal complexities of installing each package and can start immediately with Jupyter notebooks, full-fledged python project development using Spider and much more. However, chances are that you don't need 99% of these over 1000 packages, and neither need al the offered programs, and also, you don't want in install such a huge distribution on you harddisk to begin with. Alternatively you may install miniconda, which is a mini-version of anaconda. It's good and sufficient, if you install some packages that you may need, usign a simple command to accomplish that. The third way is to just install *python* and the 4 or so packages you need from the bottom up. It's easy if you just start with the site '*www.python.org*', see fig 1.

0.2.1 Installing Python3

Clicking 'Download', you get some information on the latest Python version and the release (see fig. 2).

Again further down, you find installers of varous version for different computer systems. For mac install the one for macOS and for Windows install the recommended one (see fig. 3

Find the downloaded installer in your download folder (for mac see fig. 4)

In the pop-up windown, click through a few questions and then press install (see fig. 5)

After one or two minutes, the installation is finished with some congratuations (see fig. 6)

0.2.2 Installing certificates may be a good idea too

There is a message on certificates on the congfratuations pop-up window. We can just as well install that to be up-to-date with it. Press the blue 'the Certificate project' which brings you the their site (see Fig 7).

The certificates can be downloaded and the pressed upon to install. But we can just as well the general Python package installer 'pip' (see the page. It tells to type 'pip install certifi' on the command line. That is, in a terminal window on Mac or in the console window on Windows. Fig. 8 shows a mac terminal window in which I installed the certificates, after a few lines of trial:

Note that '~ $' is the prompt. On the first line I typed python, which does not work on my system. The *zsh* that runs the command in the terminal tells me that hte *command was not found*. I have to type *python3* instead. Then just type the command as follows: *python3 -m pip install certifi* and await its installation. The last line tells you that the installation was successful. Having to top '*python3 -m*' for the command simply means that python and not the shell on your computer runs the command. The command shell

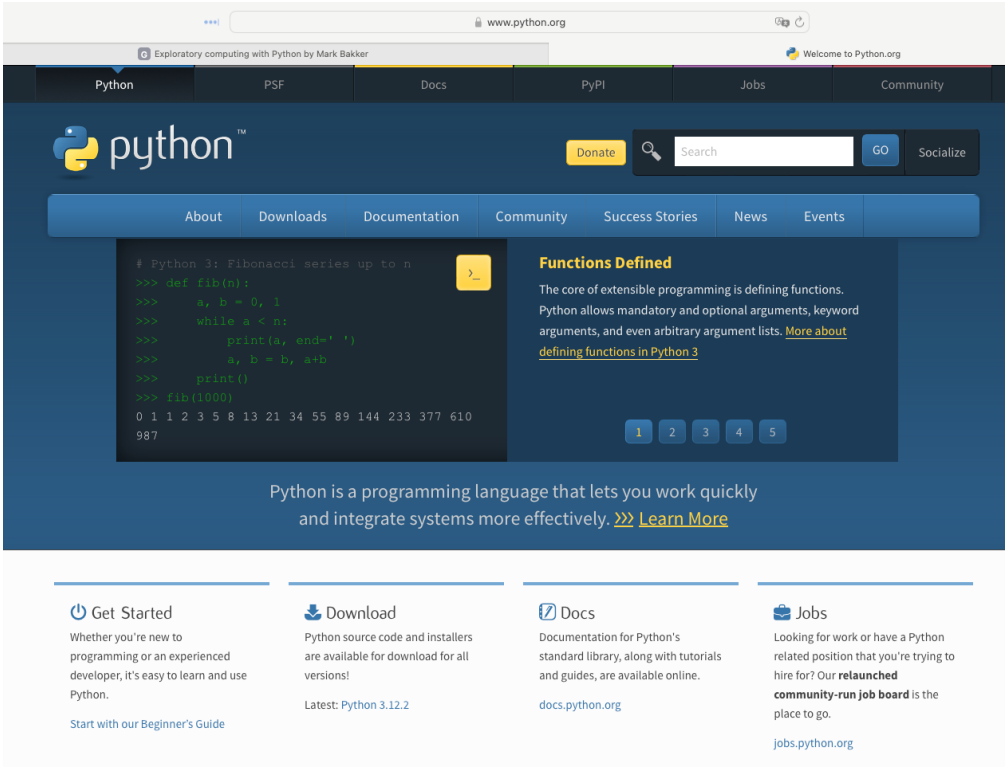

Figure 1: The site to go when starting with Python

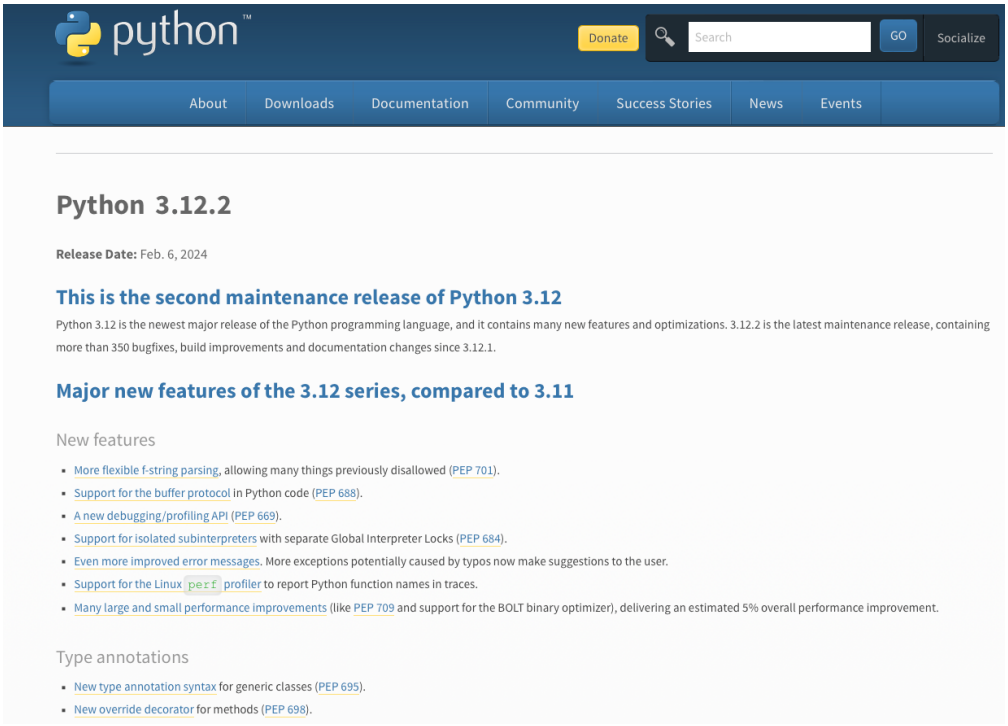

Figure 2: Version and release information of the newest Python version

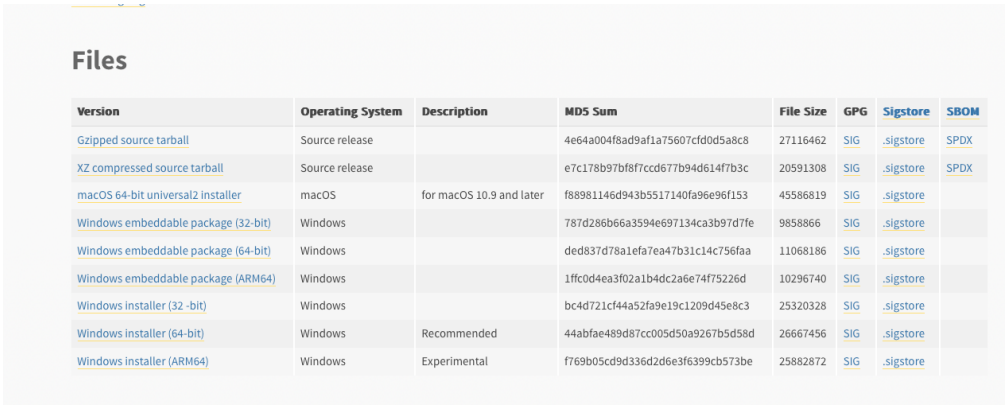

Figure 3: Installers further down on the same page

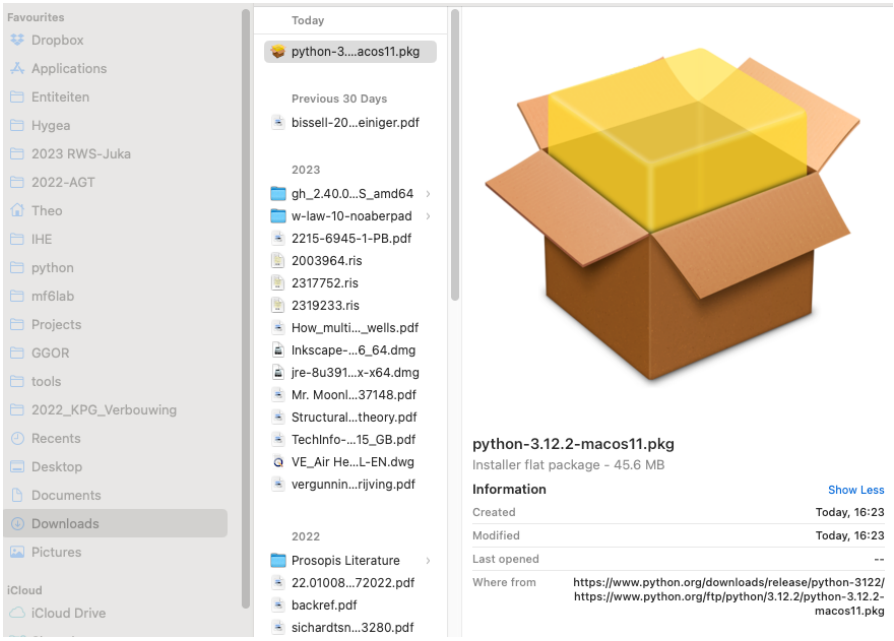

Figure 4: Find the downloaded installer in your download folder. For mac it is in ~/Downloads and has a size of 45.6 MB.

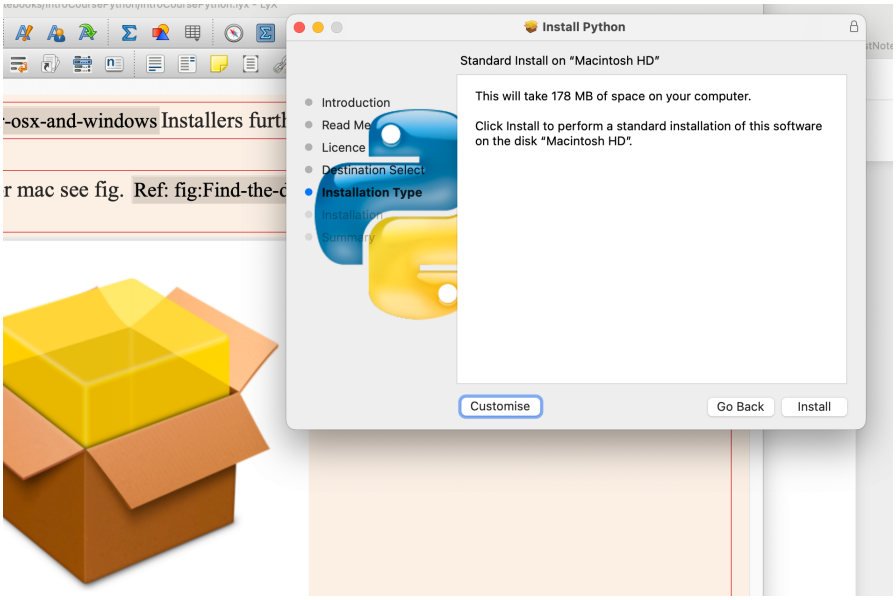

Figure 5: In the pop-up installation window, go throuogh the questions and press install.

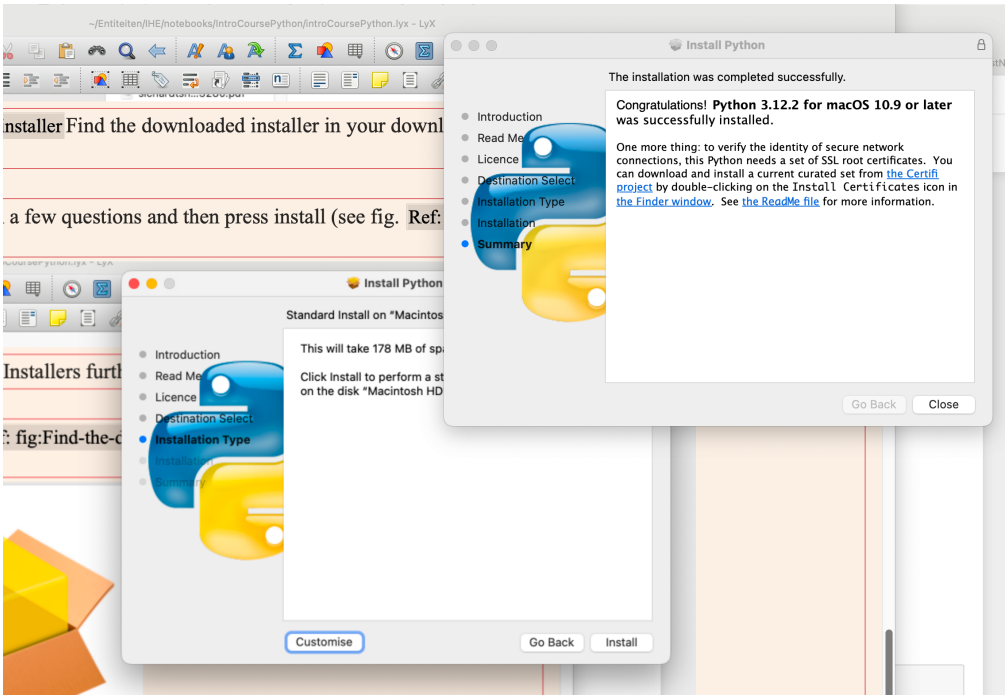

Figure 6: Installation finished after about one or two minutes.

of the terminal or console window runs *python* and *python* runs the rest according to the arguments on the rest of the line, but that's a detail.

0.2.3 Documentation for reference

Back to *www.python.org*, click *Documentation* to find a lot of documentation for both beginners and expert on using Python (see Fig. 9)

0.2.4 Installing Jupyter lab (and jupyter notebook)

But we're not yet at the tutorials. First thing to do next is to installd *jupyter*, the *notebook* interpreter, which allows us to interactively work with Python in a *jupyter* or *Ipython notebook*, they are the same, in which we can type python command, define functions, objects, chunks of python code, while executing it and get the result back which may also be graphs.

Let's move on to installing Jupyter Lab. Move to the website *jupyter.org*. (see Fig. 10)

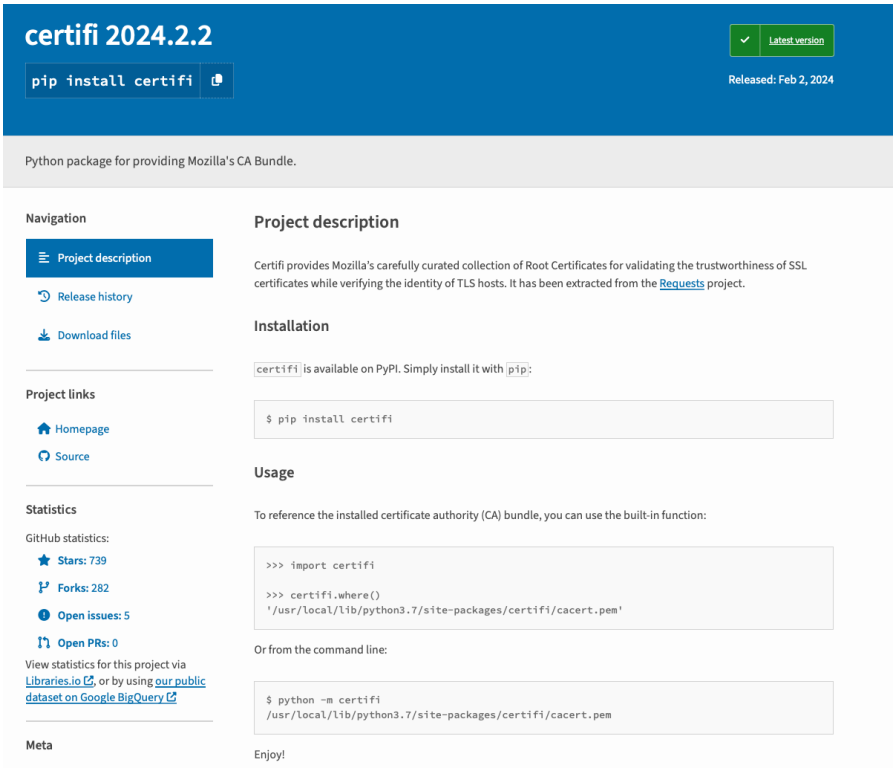

Figure 7: Certifate site

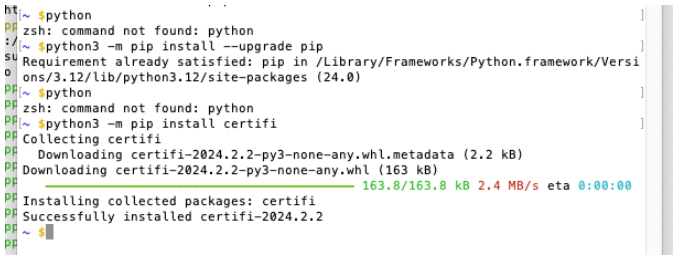

Figure 8: pip installation from Mac terminal window (on Windows the Windows console)

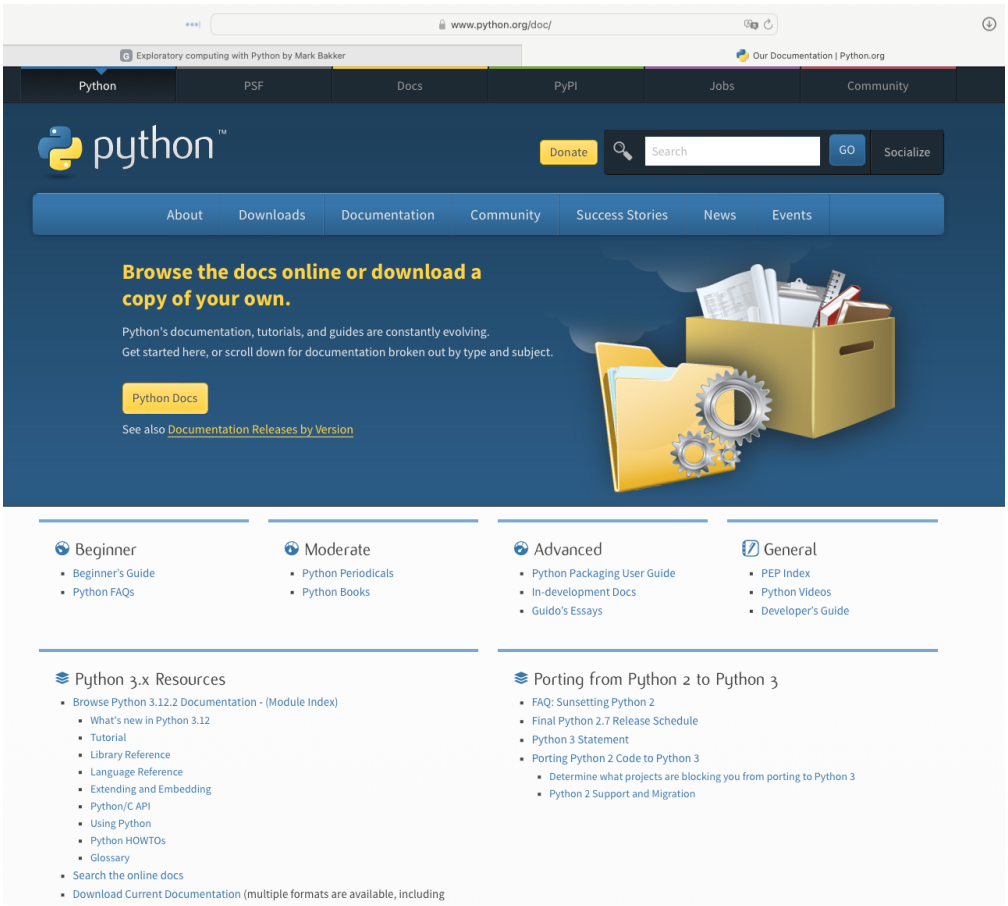

Figure 9: www.python.org/doc webpage

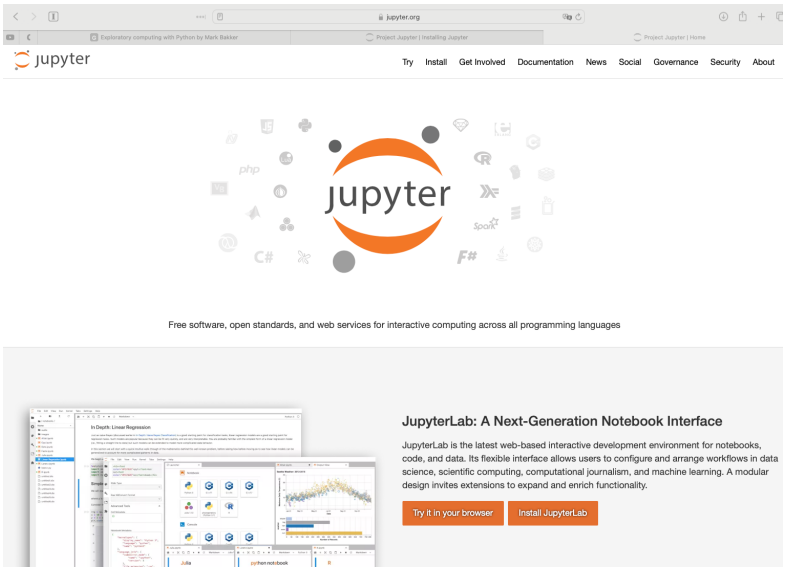

Figure 10: Jupyter.org main page

0.2.5 First create a virtual environment, it's recommended and safe

There is a lot to see on this page and to learn under Documentation. But we want to install it on our computer. The best way is to install in in a virtual environment and use a separate virtual environment for every project. This way, there can be no compatibility issues caused by for instance upgrading certain packages for another project that would break your code. A virtual environment is a clean environment, i.e. a folder in which a clean version of python and its installed packages are stored. You switch to that environment when you start working. In this philosophy you may want to create a virtual environment for this transient course and name it for instance '*transient_venv*' and put it in the directory of other material of your course.

Just Google for '*create venv using pip*' to find the right website, the '*Python Packaging User Guide*' (Fig. 11)

Scroll down to see how to create a virtual environment, a *venv*. Before doing so, naviate to your project directory where you will create the new virtual environment.Then think of a name for the environment, for instance '*transient_venv*' open the termanal of console window/application and type the command

```
python3 –m venv transient_venv
```

This command will create the vitual environment in your project directory. It now sits there as the folder '*transient_venv*' (Fig. 12)

As can be seen the *transient_venv* folder has a set of subfolders of which the bin holds the packages that have been installed in this environment, and which are thus separate from any other environment that I may have on my hard disk.

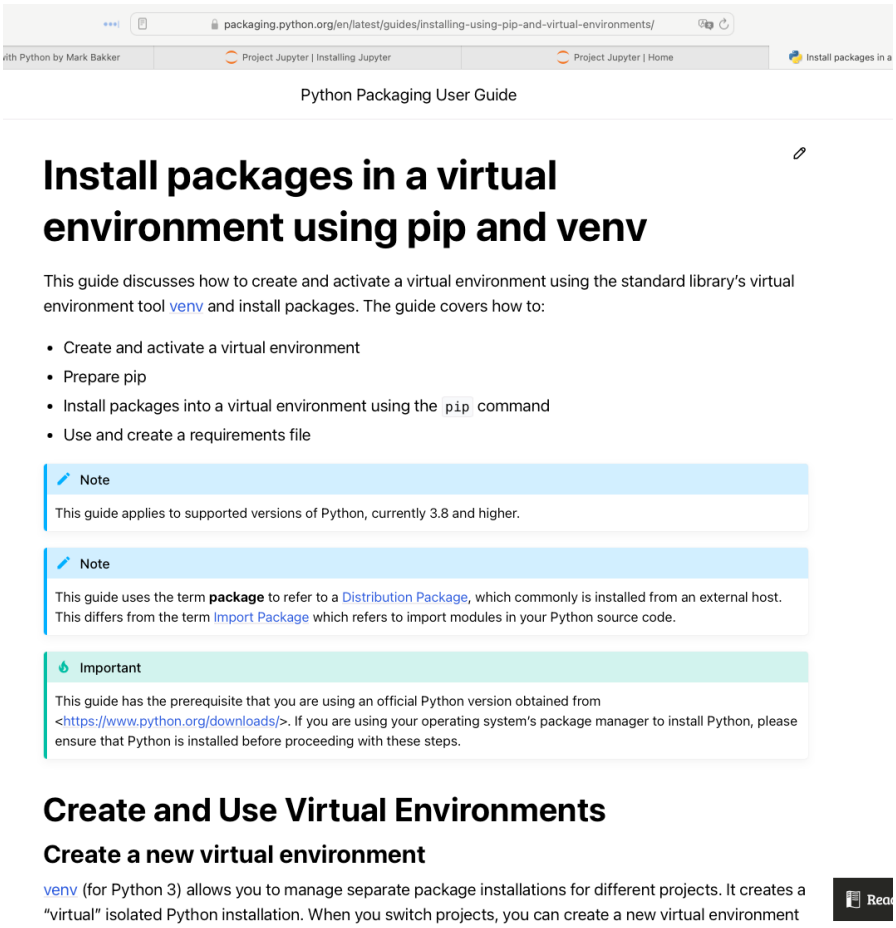

Figure 11: Python Packaging User Guide

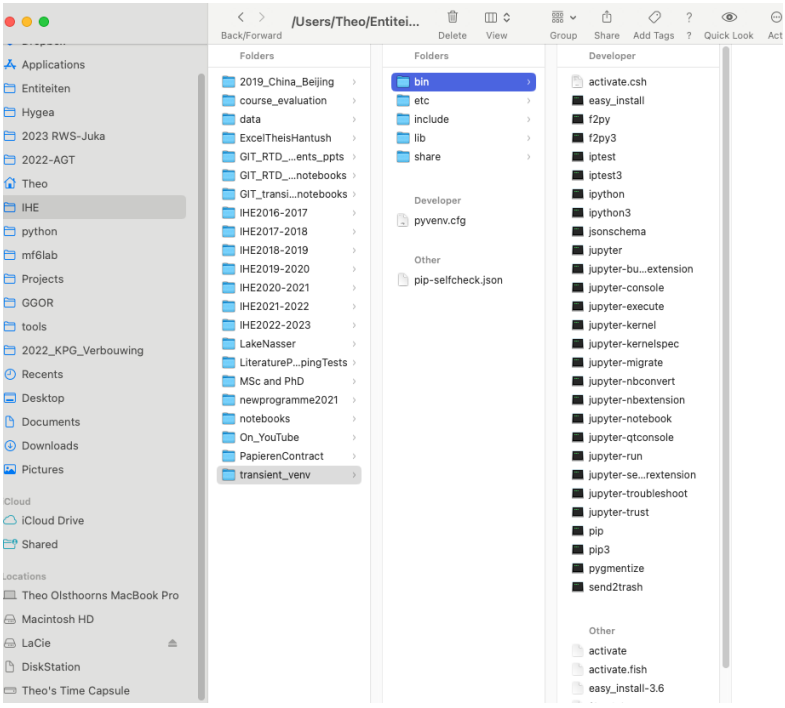

Figure 12: Virtual environment *transient_venv* on the hard disk in the project directory *IHE*

Before we can use the virtual environment we have to activate it. On a mac in the terminal window, type

```
source transient_venv/bin/activate
```

on Windows, in the console window type

```
transient_venv\Scripts\activate
```

On mac, the activated virtual environment is shown in front of the prompt ($) in the terminal window like so

```
(transient_venv) $
```

This may be similar on Windows.

However to confirm that the virtual environment is activated, check the location of your python interpreter by typing the command

```
where python
```

While the virtual environment is activated, pip will install packages in this environment!

If you want to switch projects or leave your virtual environment, *deactivate* it by typing

```
deactivate
```

Always activate the virtual environment to use it.

0.2.6 Installing new packages

Always activate the virtual environment to use it. While the virtual environment is activated, pip will install packages in this environment!

If everythin is ok, pip will work to let you install packages.

Just to make sure it works on Mac in the terminal type the command

```
python3 –m ensurepip —upgrade
```

On Windows in the console window type the command

```
py – m ensurepip —upgrade
```

If you open the file browser to show the *transient_ venv/bin* in which the packages will be installed you can follow the installation while it proceeds by seeing packages added to that folder.

For our pupposes we will install the following packages as follows:

```
pip3 install numpy
pip3 install scipy
pip3 install matplotlib
pip3 install pandas
pip3 install jupyterlab
pip3 install notebook
```

When ready, your tansient_venv/bin should look like Fig. 12.

0.2.7 Working with notebooks

From this point we can start working with notebooks

Navigate to the project directory (where you want your notebooks to be saved). Then type

```
jupyter lab
```

or

```
jupyter notebook
```

The interface should then pop up in which you can start your first notebook and start solving groundwater or any other problems.

0.3 Intro and tutorials

Documentation on how jupyter notebooks work can be found on the *jupyter.org* site under *Documentation*.

After pressing the button *Documentation* look for User Interfaces *Jupyter lab* (and perhaps also *Jupyter notebook*) to see how it works.

0.4 Documentation to get rapidly on speed with using Python for your own projects

It's definitely best to start with examples made for students on how to use Python to solve problems. The best entry in my honest opinion is to get the notebook examples made by Prof. Mark Bakker of TUDelft and used for all 2nd year students of Civel Engineering faculty to get into Python with speed using exactly the tools these students need.

Google for "*Mark Bakker exploratory computing with Python*" to get to the following website:

For excellent tutorials download the zip file and put its contents into a folder which you may name *BakkerExploratory* or so, and go thorugh them to learn Python to fast way for technical students.

The rest we'll do in class.

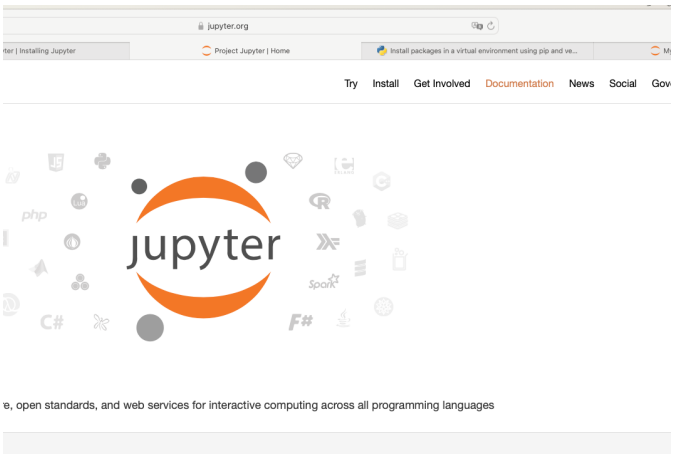

Figure 13: Where to find jupyter documtation?

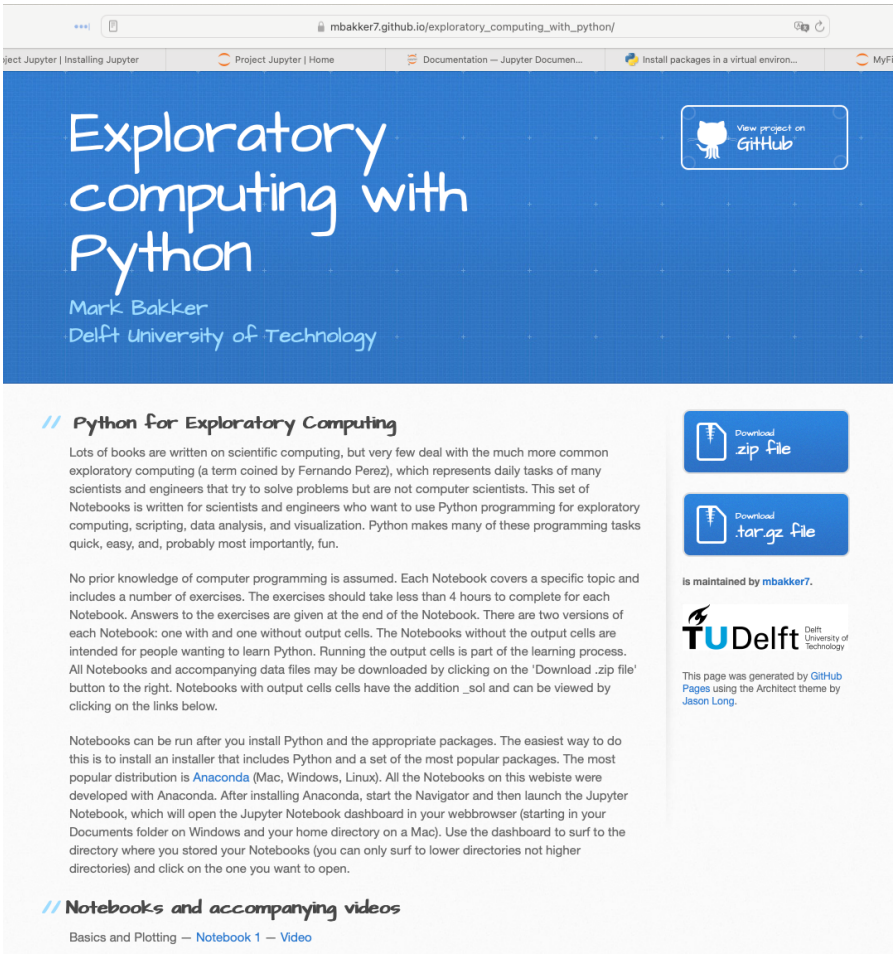

Figure 14: Mark Bakker's site met tutorials for explanatory computing using *Jupyter* notebooks.