

Intro transient course using Python for the exercises

Transient Groundwater Flow Course at IHE, Delft, Feb 2024

Prof. dr.ir. T.N.Olsthorn

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 further deepens the insight in the character of the analytical solutions we'll discover in the course.

Although exploring solutions looks like numerical 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 analysis, 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, numerical 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 conveniently use analytical 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 other hand, analytical solutions are used to verify numerical 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 environment, with a large number of programs and over 1000 packages that may be useful if you are

a data scientist. You don't have to worry 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 all the offered programs, and also, you don't want to install such a huge distribution on your 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, using 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 various versions 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 window, click through a few questions and then press install (see fig. 5)

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

0.2.2 Installing certificates may be a good idea too

There is a message on certificates on the congratulations 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 to their site (see Fig 7).

The certificates can be downloaded and then 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 the *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 type '*python3 -m*' for the command simply means that python and not the shell on your computer runs the command. The command shell

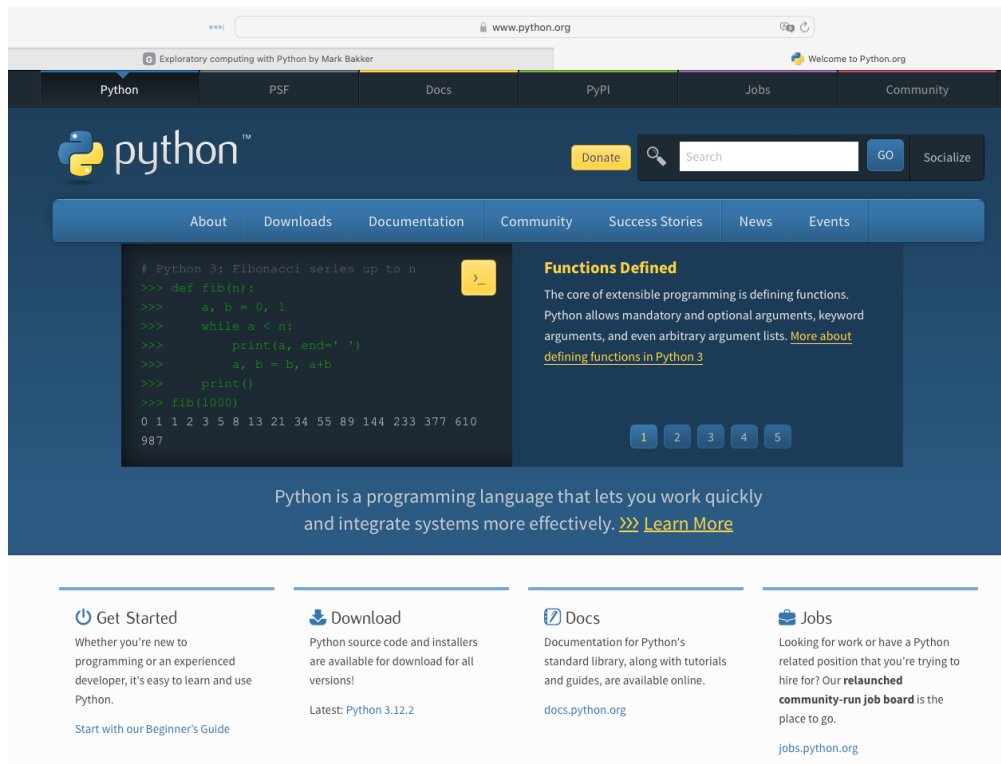



Figure 1: The site to go when starting with Python



[Donate](#)

[GO](#)
[Socialize](#)

[About](#)
[Downloads](#)
[Documentation](#)
[Community](#)
[Success Stories](#)
[News](#)
[Events](#)

Python 3.12.2

Release Date: Feb. 6, 2024

This is the second maintenance release of Python 3.12

Python 3.12 is the newest major release of the Python programming language, and it contains many new features and optimizations. 3.12.2 is the latest maintenance release, containing more than 350 bugfixes, build improvements and documentation changes since 3.12.1.

Major new features of the 3.12 series, compared to 3.11

New features

- More flexible f-string parsing, allowing many things previously disallowed ([PEP 701](#)).
- Support for the buffer protocol in Python code ([PEP 688](#)).
- A new debugging/profiling API ([PEP 669](#)).
- Support for isolated subinterpreters with separate Global Interpreter Locks ([PEP 684](#)).
- Even more improved error messages. More exceptions potentially caused by typos now make suggestions to the user.
- Support for the Linux [perf](#) profiler to report Python function names in traces.
- Many large and small performance improvements (like [PEP 709](#) and support for the BOLT binary optimizer), delivering an estimated 5% overall performance improvement.

Type annotations

- New type annotation syntax for generic classes ([PEP 695](#)).
- New override decorator for methods ([PEP 698](#)).

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

Files

Version	Operating System	Description	MDS Sum	File Size	GPG	Sigstore	SBOM
Gzipped source tarball	Source release		4e64a004f8ad9af1a75607cfd0d5a8c8	27116462	SIG	.sigstore	SPDX
XZ compressed source tarball	Source release		e7c178b97bf87f7cc677b94d614f7b3c	20591308	SIG	.sigstore	SPDX
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later	f88981146d943b5517140fa96e96f153	45586819	SIG	.sigstore	
Windows embeddable package (32-bit)	Windows		787d286b66a3594e697134ca3b97d7fe	9858866	SIG	.sigstore	
Windows embeddable package (64-bit)	Windows		ded837d78a1efa7ea47b31c14c756faa	11068186	SIG	.sigstore	
Windows embeddable package (ARM64)	Windows		1ffc0d4ea3f02a1b4dc2a6e74f75226d	10296740	SIG	.sigstore	
Windows installer (32-bit)	Windows		bc4d721cf44a52fa9e19c1209d45e8c3	25320328	SIG	.sigstore	
Windows installer (64-bit)	Windows	Recommended	44abfae489d87cc005d50a9267b5d58d	26667456	SIG	.sigstore	
Windows installer (ARM64)	Windows	Experimental	f769b05cd9d336d2d6e3f6399cb573be	25882872	SIG	.sigstore	

Figure 3: Installers further down on the same page

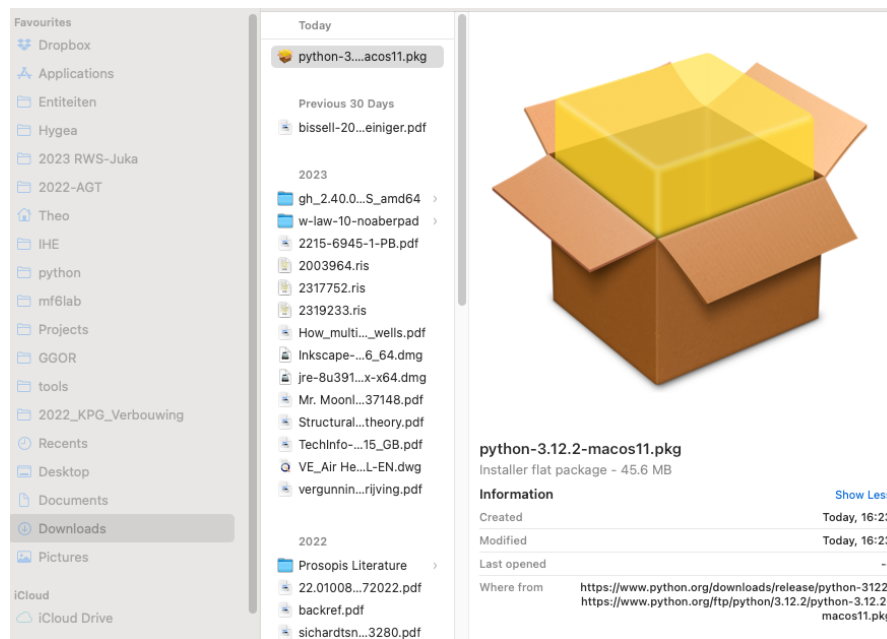


Figure 4: Find the downloaded installer in your download folder. For mac it is in \sim /Downloads and has a size of 45.6 MB.

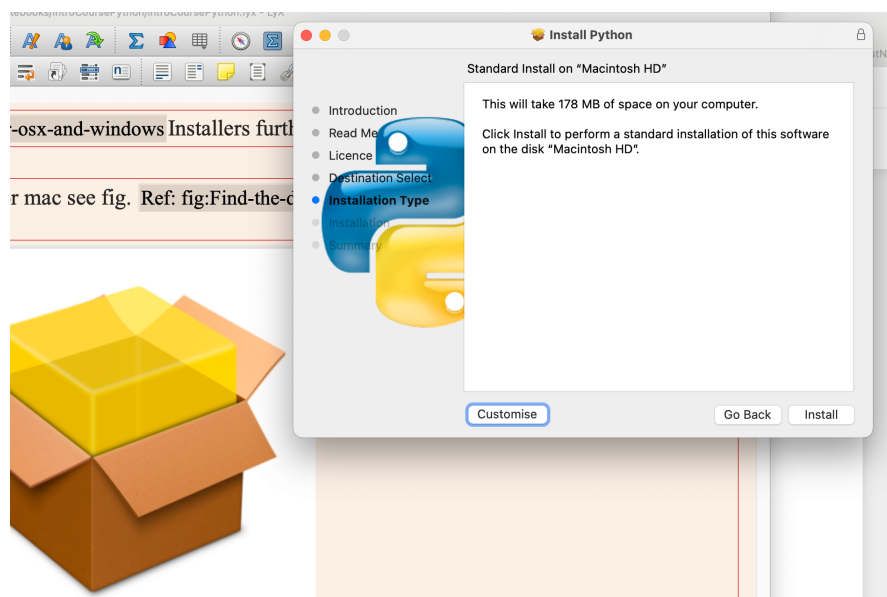


Figure 5: In the pop-up installation window, go through the questions and press install.

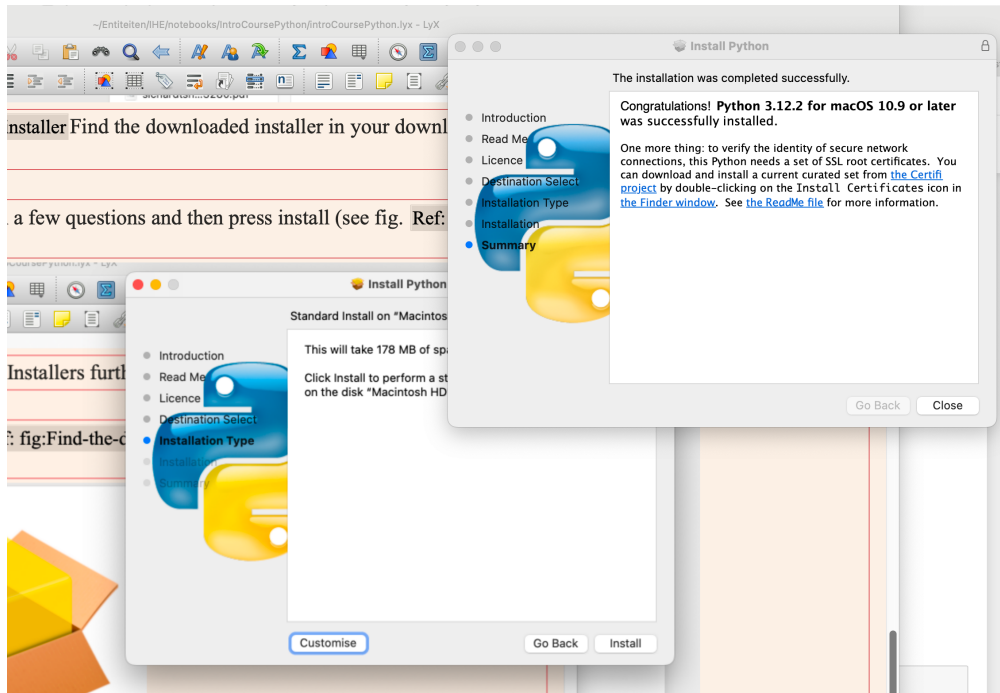


Figure 6: Installation finished after about one or two minutes.

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 install *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)

certifi 2024.2.2

pip install certifi

Released: Feb 2, 2024

Python package for providing Mozilla's CA Bundle.

Navigation

Project description

Release history

Download files

Project links

Homepage

Source

Statistics

GitHub statistics:

Stars: 739

Forks: 282

Open issues: 5

Open PRs: 0

View statistics for this project via [Libraries.io](#), or by using [our public dataset on Google BigQuery](#)

Meta

Project description

Certifi provides Mozilla's carefully curated collection of Root Certificates for validating the trustworthiness of SSL certificates while verifying the identity of TLS hosts. It has been extracted from the [Requests](#) project.

Installation

certifi is available on PyPI. Simply install it with `pip`:

```
$ pip install certifi
```

Usage

To reference the installed certificate authority (CA) bundle, you can use the built-in function:

```
>>> import certifi
>>> certifi.where()
'/usr/local/lib/python3.7/site-packages/certifi/cacert.pem'
```

Or from the command line:

```
$ python -m certifi
/usr/local/lib/python3.7/site-packages/certifi/cacert.pem
```

Figure 7: Certificate site

```
hit ~ $python
PP zsh: command not found: python
:/ ~ $python3 -m pip install --upgrade pip
su Requirement already satisfied: pip in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (24.0)
PP ~ $python
PP zsh: command not found: python
PP ~ $python3 -m pip install certifi
PP Collecting certifi
PP   Downloading certifi-2024.2.2-py3-none-any.whl.metadata (2.2 kB)
PP   Downloading certifi-2024.2.2-py3-none-any.whl (163 kB)
PP     163.8/163.8 kB 2.4 MB/s eta 0:00:00
PP Installing collected packages: certifi
PP Successfully installed certifi-2024.2.2
PP ~ $
```

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

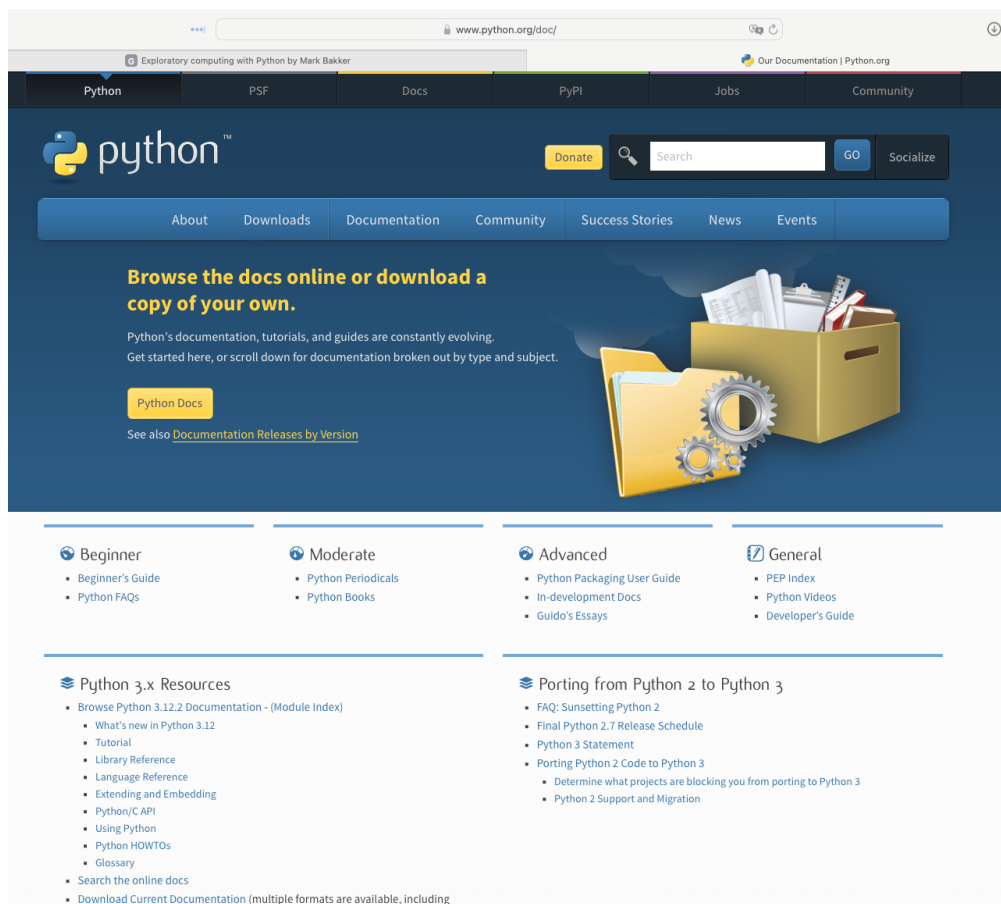


Figure 9: www.python.org/doc webpage

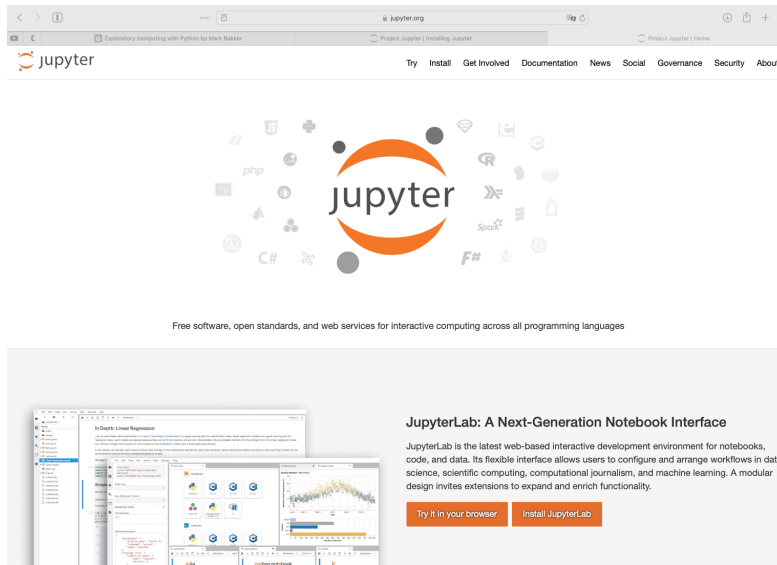


Figure 10: Jupyter.org main page

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 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, navigate 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 terminal of console window/application and type the command

```
python3 -m venv transient_venv
```

This command will create the virtual 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.

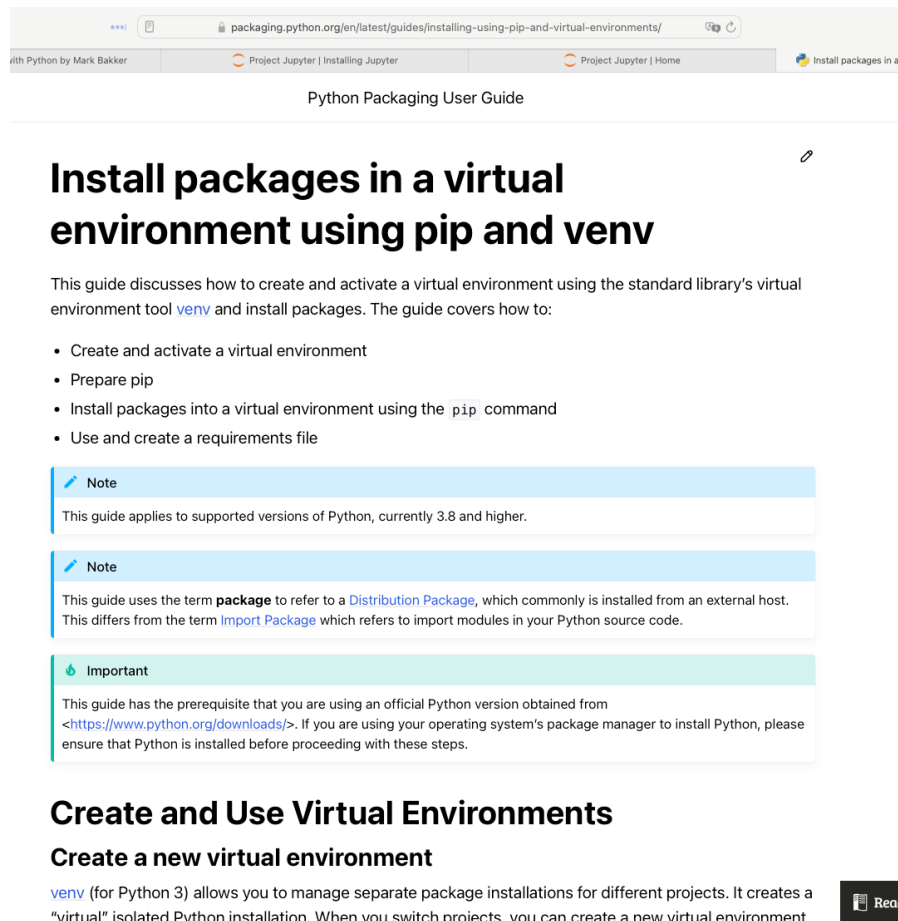


Figure 11: Python Packaging User Guide

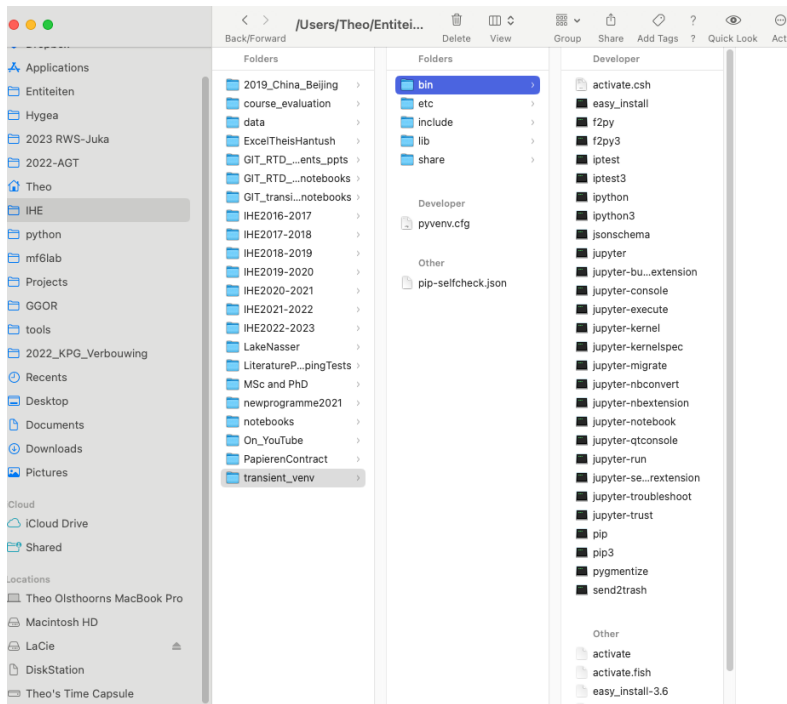


Figure 12: Virtual environment *transient_venv* on the hard disk in the project directory *IHE*

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

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 everything 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 purposes 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 transient_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 Civil 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 thorough them to learn Python to fast way for technical students.

The rest we'll do in class.

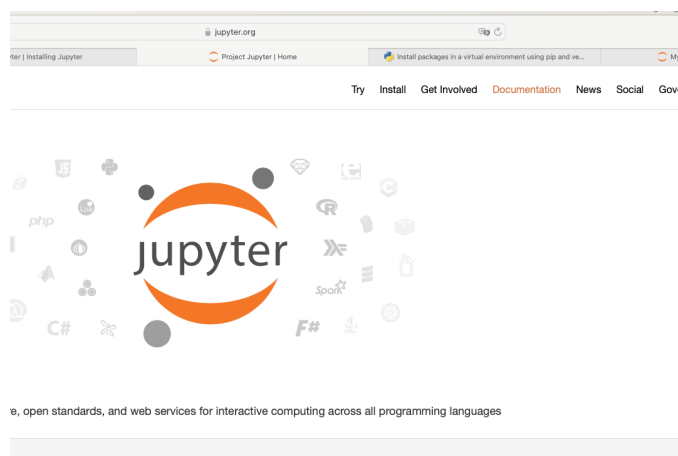


Figure 13: Where to find jupyter documtation?

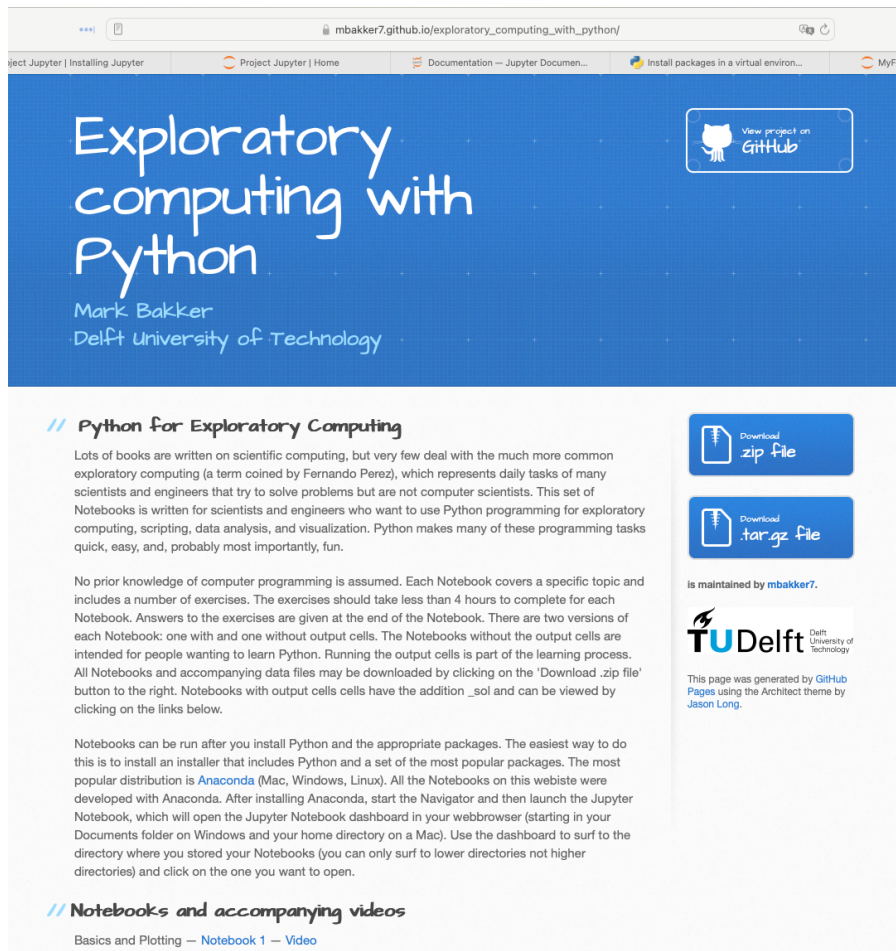


Figure 14: Mark Bakker's site met tutorials for explanatory computing using *Jupyter* notebooks.