Virtual Observatory ASTRO-CC Training Event : software installation instructions

Here are installation instructions for the ASTRO-CC Virtual Observatory Training event. If you have any questions or trouble installing, you can reach :

— mark.allen@astro.unistra.fr
— matthieu.baumann@astro.unistra.fr
— manon.marchand@astro.unistra.fr
— hendrik.heinl@inaf.it

and we'll do our best to assist you.

1 Aladin and TOPCAT

1.1 Installation

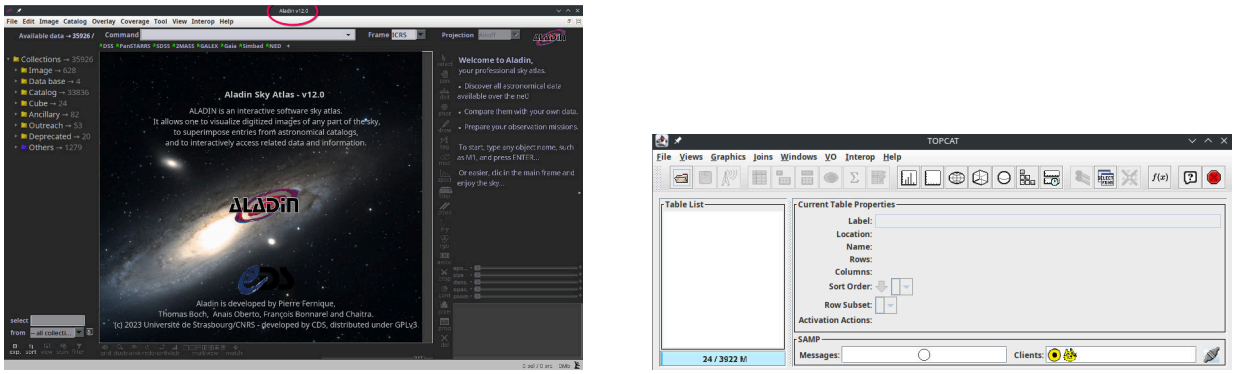
Aladin and TOPCAT are both developed in Java. This means that you need a Java virtual Machine to execute these software. Follow the instructions on this link.

Then, you can download the files corresponding to your operating system respectively on Aladin's website and TOPCAT's website.

TIP : If you're using Ubuntu, Debian, or Arch linux, your package managers have Aladin desktop registered under `aladin` and TOPCAT under `topcat`.

1.2 Checking the installation

Figure 1 shows the starting windows of Aladin and TOPCat. The Aladin version should be at least v12 to be able to follow the tutorials, see the top bar of the window as highlighted in the figure.



(a) Aladin (b) TOPCAT

FIGURE 1 – Launch screens for Aladin and TOPCAT

2 VOSA : the Virtual Observatory SED Analyzer

For the VOSA tutorial, you don't need to install anything, but it would be better to create an account on the VOSA website if you don't already have one.

3 Python

In this section, we describe how to install a working Python environment on your machine. This is optional as we will also provide an online install-free version of the tutorials.

3.1 Getting python

Python is already installed on a lot of operating systems. To check if you already have it, open a terminal [1] and type :

```
python3 --version
```

The response should look something like `Python 3.12.2`. The version should be comprised between 3.10 and 3.14 ([2].) If you are on Windows, it is possible that you'll have to type `py` instead of `python3`.

If this worked, you are good to go and you can go to the virtual environment setup.

If neither of these work, or if your version is too old, follow the installation instructions for Python.

3.2 Setting a virtual environment

It is good practice to have one virtual environment per project.

What problem does a virtual environment solve ? The more Python projects you have, the more likely it is that you need to work with different versions of Python libraries, or even different versions of Python itself. Newer versions of libraries for one project can break compatibility in another project. Virtual environments are independent groups of Python libraries, one for each project. Packages installed for one project will not affect other projects or the operating system's packages. Python comes bundled with the `venv` module to create virtual environments, we will use it here [a].

a. If you already use an other virtual environment package, such as `miniconda` or `uv`, use your own rather than `venv`.

To do this, open your terminal in the directory you want to work in, and do (note that the point at the end is important) :

```
python3 -m venv .
```

This will create the files needed for this virtual environment. We can activate the environment with :

On windows :

```
# In cmd.exe
venv\Scripts\activate.bat
# In PowerShell
venv\Scripts\Activate.ps1
```

On Linux and MAC :

```
source bin/activate
```

The virtual environment is now activated. You can check it by looking at the list of python packages currently installed in this virtual environment with `pip3`, the Python package manager :

1. On windows, the terminal is called `Command prompt`.
2. These are the currently valid python versions you can always check which version is currently valid on Python's life-cycle page.

```
pip3 list
```

You should only have pip3. We can install the libraries we need in this fresh environment [3].

```
pip3 install jupyterlab>=4.0 mocpy>=0.20.0 ipyaladin>=0.8.0 astroquery>=0.4.11
```

And launch jupyterlab with

```
python3 -m jupyterlab
```

And voilà, you are ready to code in python ! When you want to switch off Jupyterlab and the virtual environment, you can do `ctrl + C` in the terminal from which you launched the notebook, say `y` to confirm that you want to leave. When the prompt is available, do `deactivate` to leave the virtual environment.

3.3 Checking the installation

If you've never used Jupyterlab, we advise you to watch this short video introducing the interface. When you launch JupyterLab (with `python3 -m jupyterlab`), you should see the square Python button in figure 2a. Uppon clicking on it, a notebook appears. Copy paste

```
import mocpy
import astroquery
import pyvo
import ipyaladin
```

in the first cell, press `shift+enter` to execute the first cell. If it looks like figure 2b, then you're good to go !

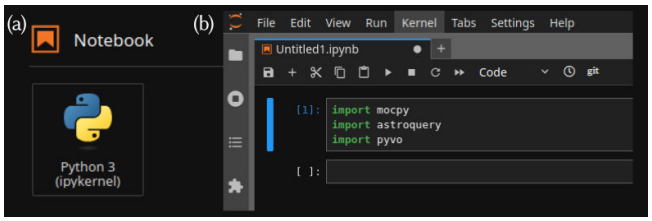


FIGURE 2 – Checking the Python setup. (a) To create a new notebook, click on the python square button. (b) How the inside of a notebook looks like.

4 Conclusion

Don't hesitate to join us with any question (related to the installation, to the content of the workshop, to anything VO...).

Also keep in mind that there is a time dedicated to you personal questions/projects. You can prepare data files, scientific goals, outreach questions, ... and we'll do our best to see how these could be solved (or not, maybe you need something that is yet to be built ?) with VO tools.

See you all at the training event !

3. doing `pip3 list` after this will return a lot of packages. These are all the dependencies of these 4 libraries.