

# Interactive Computing with JupyterLab, IJulia and IPyParaview

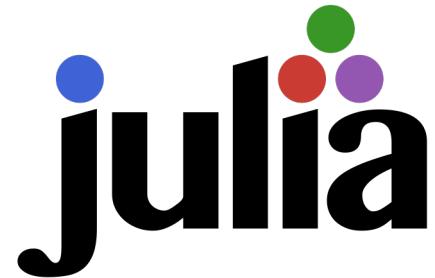
Tim Robinson

Sam Omlin

Jean Favre

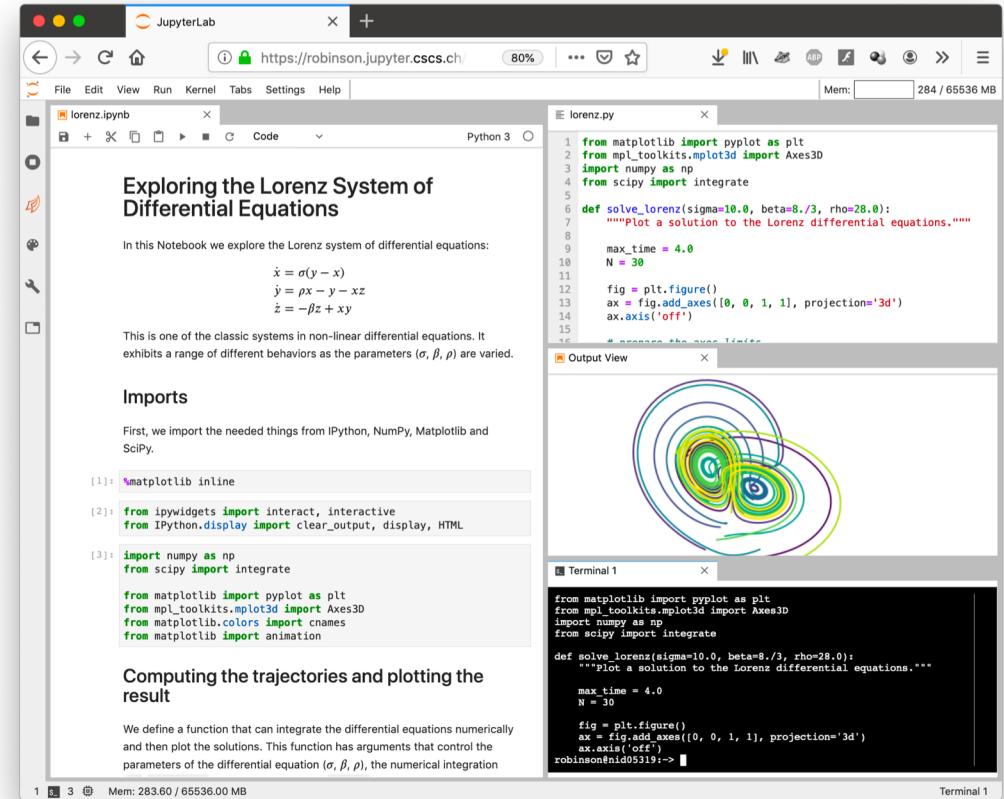
# Agenda

- New functionality in JupyterLab
  - GPU dashboards
  - Code formatting with Black
  - Creating a custom kernel
- IJulia kernel for using Julia in JupyterLab
- ParaView for visualization in JupyterLab



# JupyterLab on Piz Daint

- JupyterHub provides JupyterLab servers on demand at <https://jupyter.csccs.ch>
- Anyone with access to Piz Daint can log in with their CSCS credentials (Kerberos/LDAP)
- Servers are launched on compute nodes of Piz Daint (GPU or multicore)
- You have dedicated access to the full compute capability of the node
- Special queues for interactive computing grow and shrink automatically according to demand



The screenshot shows a JupyterLab interface running in a web browser window. The title bar says "JupyterLab" and the address bar shows the URL "https://robinson.jupyter.csccs.ch". The main area displays a Jupyter Notebook titled "Exploring the Lorenz System of Differential Equations". The notebook contains Python code for solving the Lorenz equations and plotting the results. The code includes imports for matplotlib, ipywidgets, numpy, and scipy, along with a function definition for "solve\_lorenz". The resulting 3D plot of the Lorenz attractor is shown in the "Output View". Below the notebook, a terminal window labeled "Terminal 1" shows the same command-line session, indicating that the computation was run directly from the terminal.

```
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
from scipy import integrate

def solve_lorenz(sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""
    max_time = 4.0
    N = 30

    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # generate the sunfractal
    lorenz = np.load('lorenz.npy')

    # plot the trajectory
    ax.plot(lorenz[:, 0], lorenz[:, 1], lorenz[:, 2], c='cyan', linewidth=2)
```

```
#matplotlib inline
from ipywidgets import interact, interactive
from IPython.display import clear_output, display, HTML

import numpy as np
from scipy import integrate

from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.colors import cnames
from matplotlib import animation

Exploring the Lorenz System of Differential Equations

In this Notebook we explore the Lorenz system of differential equations:

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of different behaviors as the parameters ( $\sigma$ ,  $\beta$ ,  $\rho$ ) are varied.

Imports

First, we import the needed things from IPython, NumPy, Matplotlib and SciPy.

Computing the trajectories and plotting the result

We define a function that can integrate the differential equations numerically and then plot the solutions. This function has arguments that control the parameters of the differential equation ( $\sigma$ ,  $\beta$ ,  $\rho$ ), the numerical integration
```

```
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
from scipy import integrate

def solve_lorenz(sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""

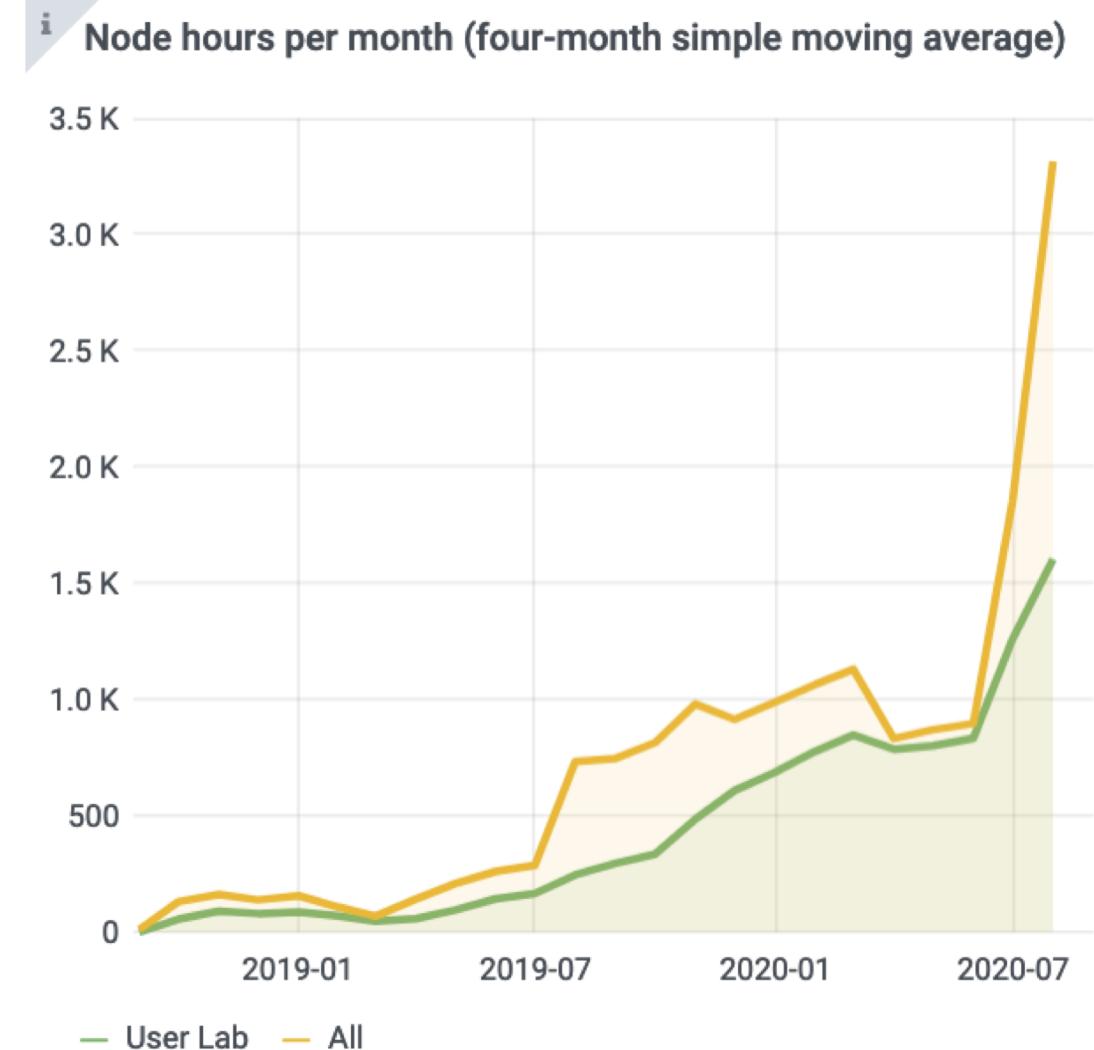
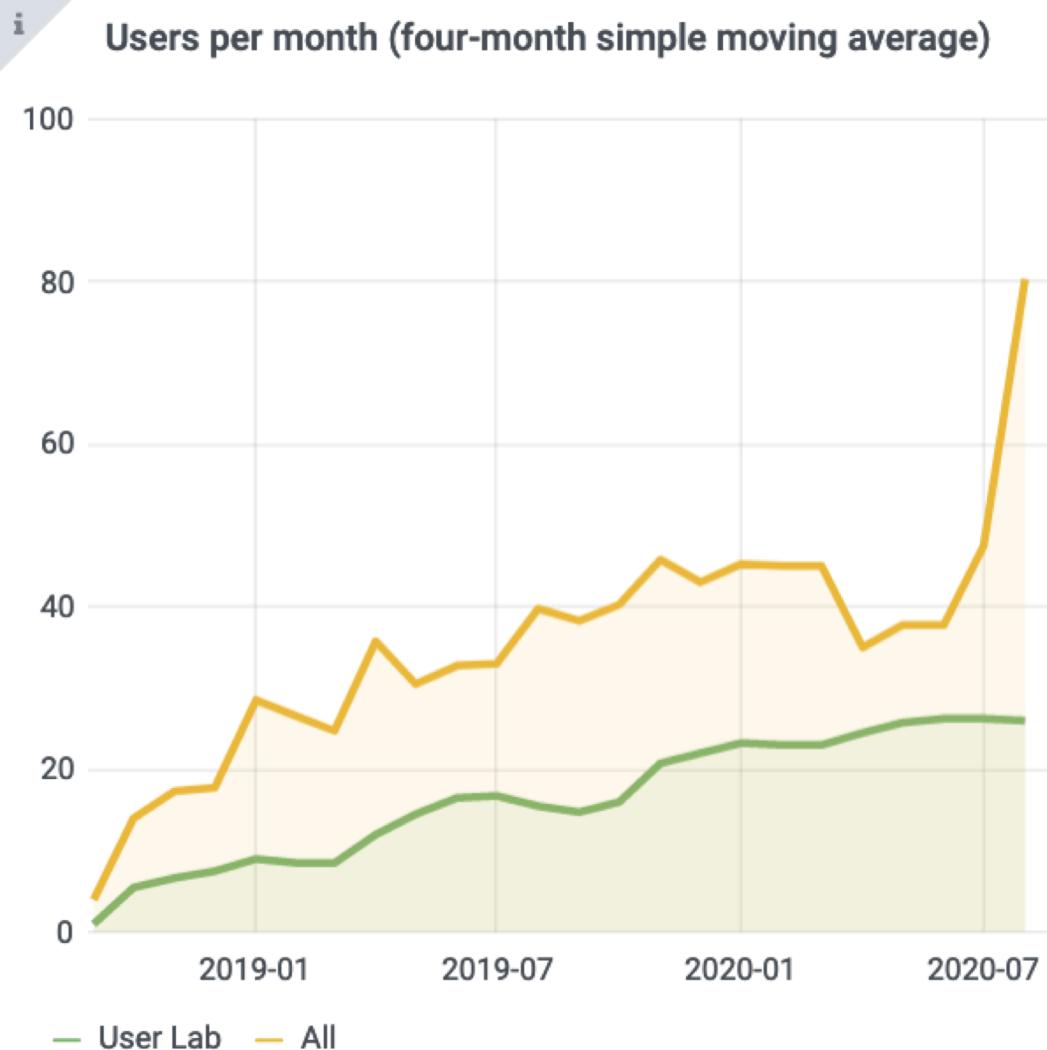
    max_time = 4.0
    N = 30

    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

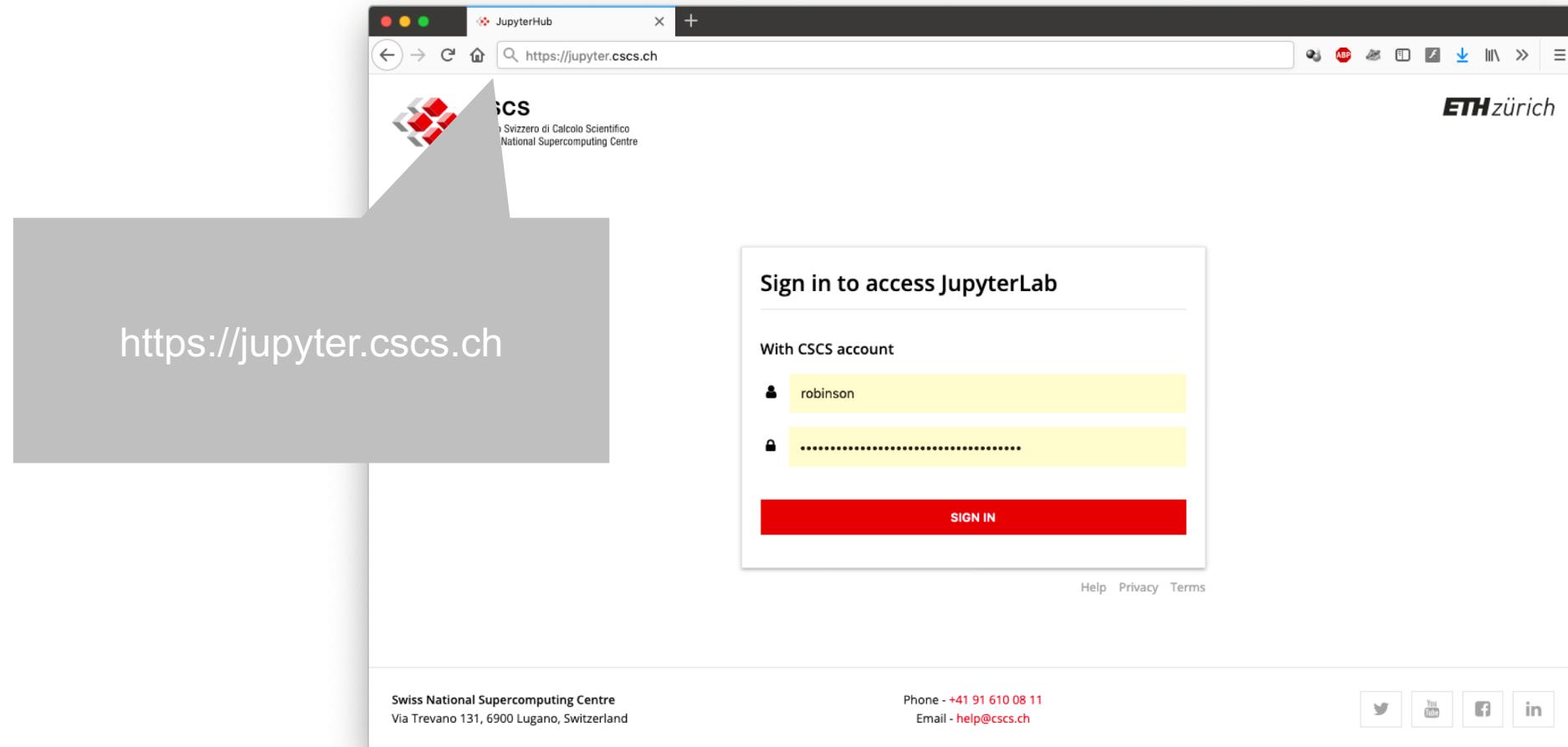
    # generate the sunfractal
    lorenz = np.load('lorenz.npy')

    # plot the trajectory
    ax.plot(lorenz[:, 0], lorenz[:, 1], lorenz[:, 2], c='cyan', linewidth=2)
```

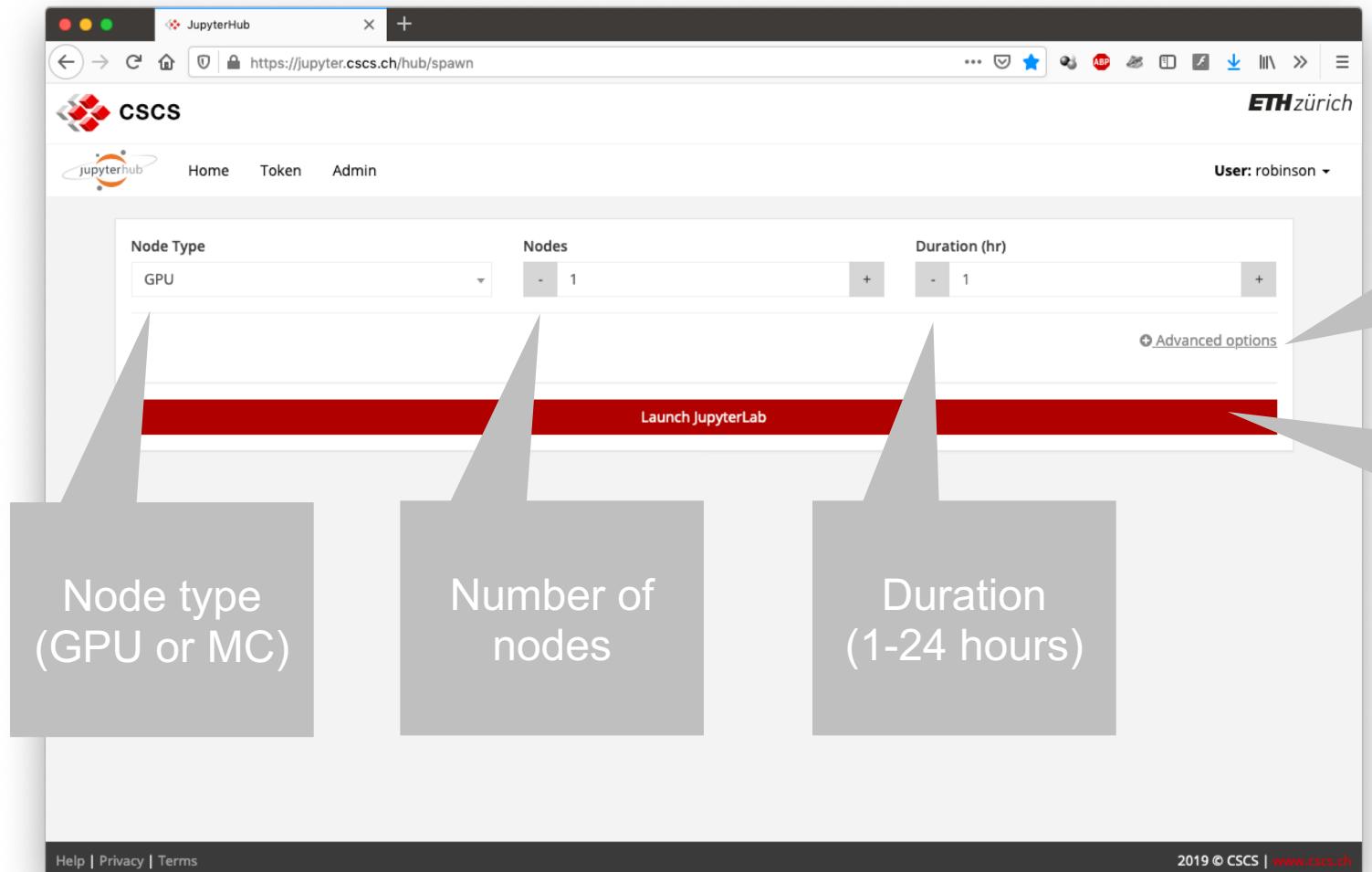
# JupyterLab usage over time



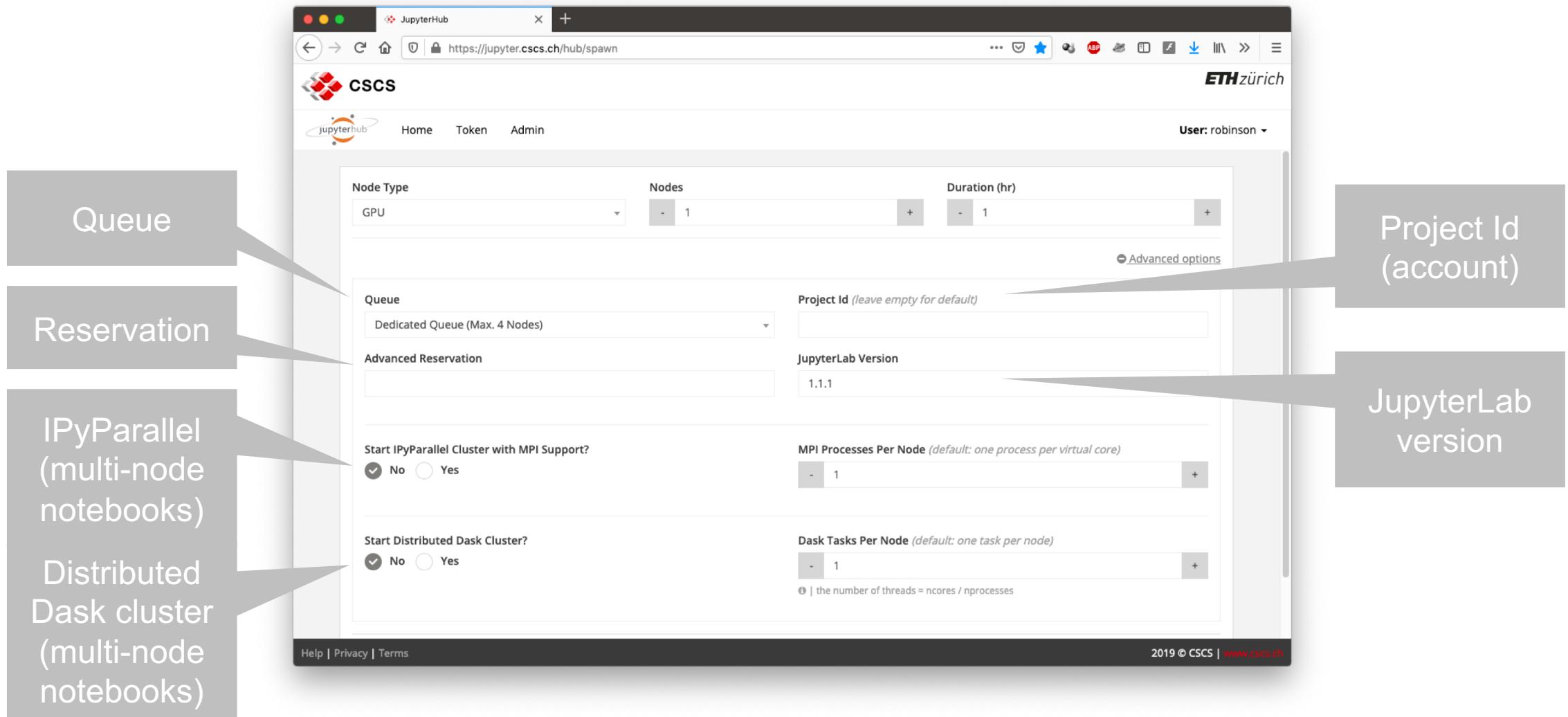
# Logging in



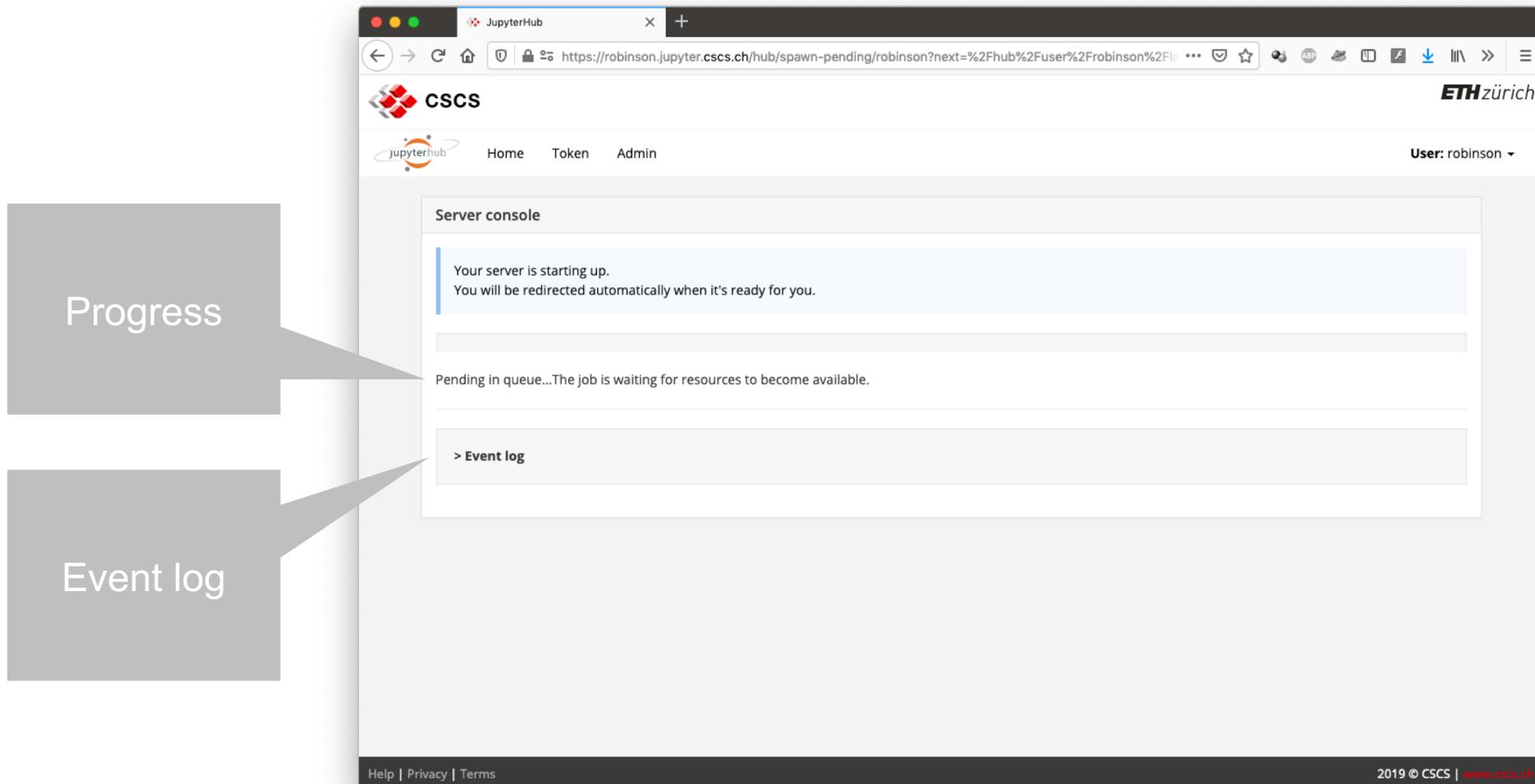
# Basic options



# Advanced options



# Launching



# JupyterLab interface

Standard output and stderr written to:

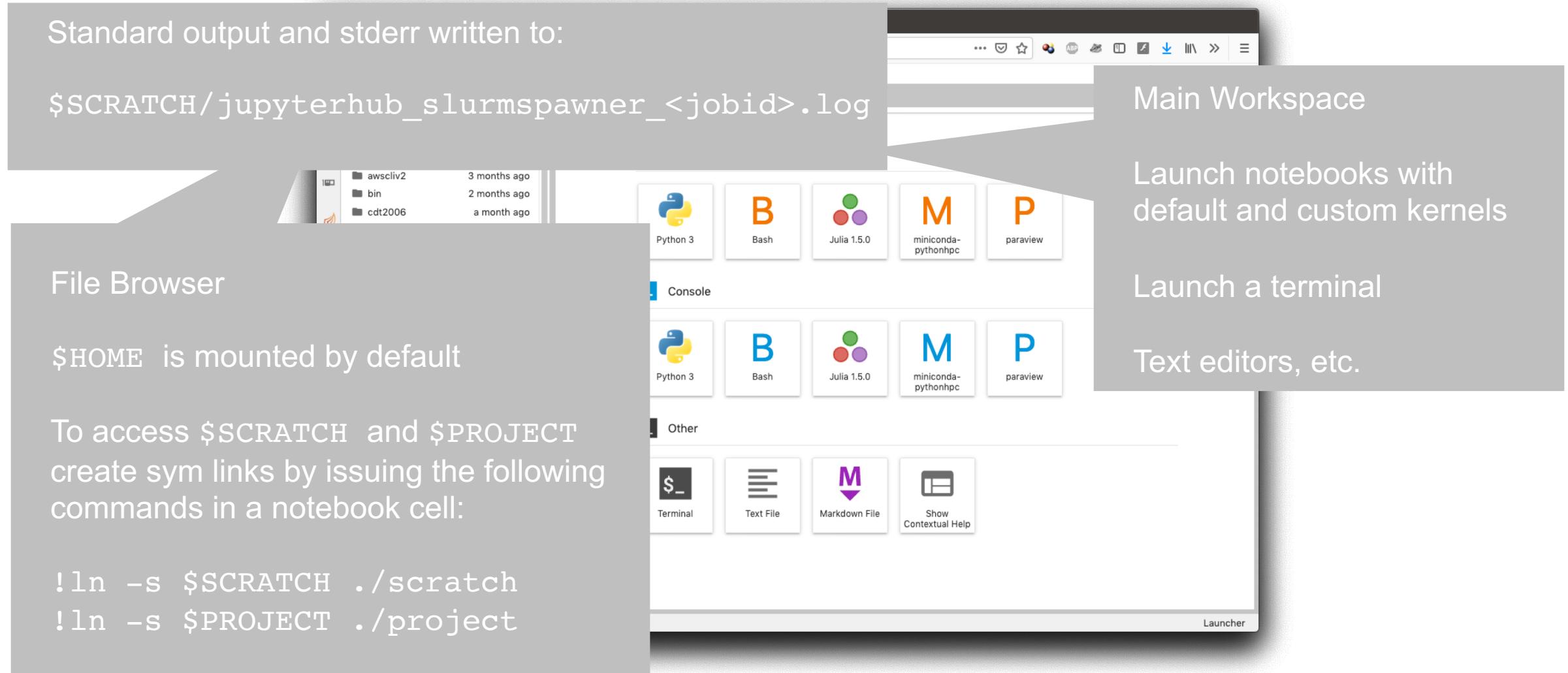
```
$SCRATCH/jupyterhub_slurmspawner_<jobid>.log
```

File Browser

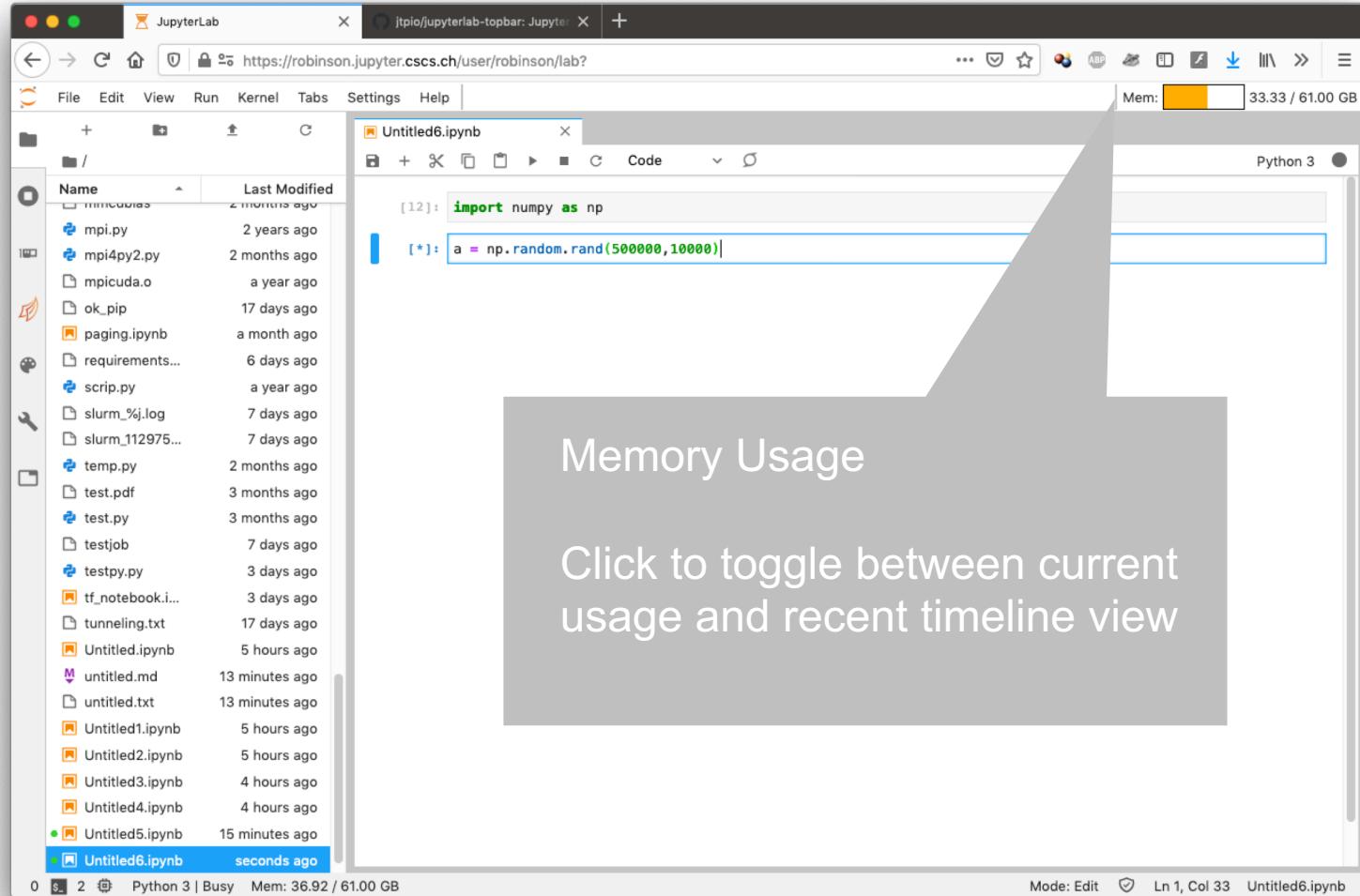
`$HOME` is mounted by default

To access `$SCRATCH` and `$PROJECT` create sym links by issuing the following commands in a notebook cell:

```
!ln -s $SCRATCH ./scratch  
!ln -s $PROJECT ./project
```



# Memory usage monitor





**CSCS**

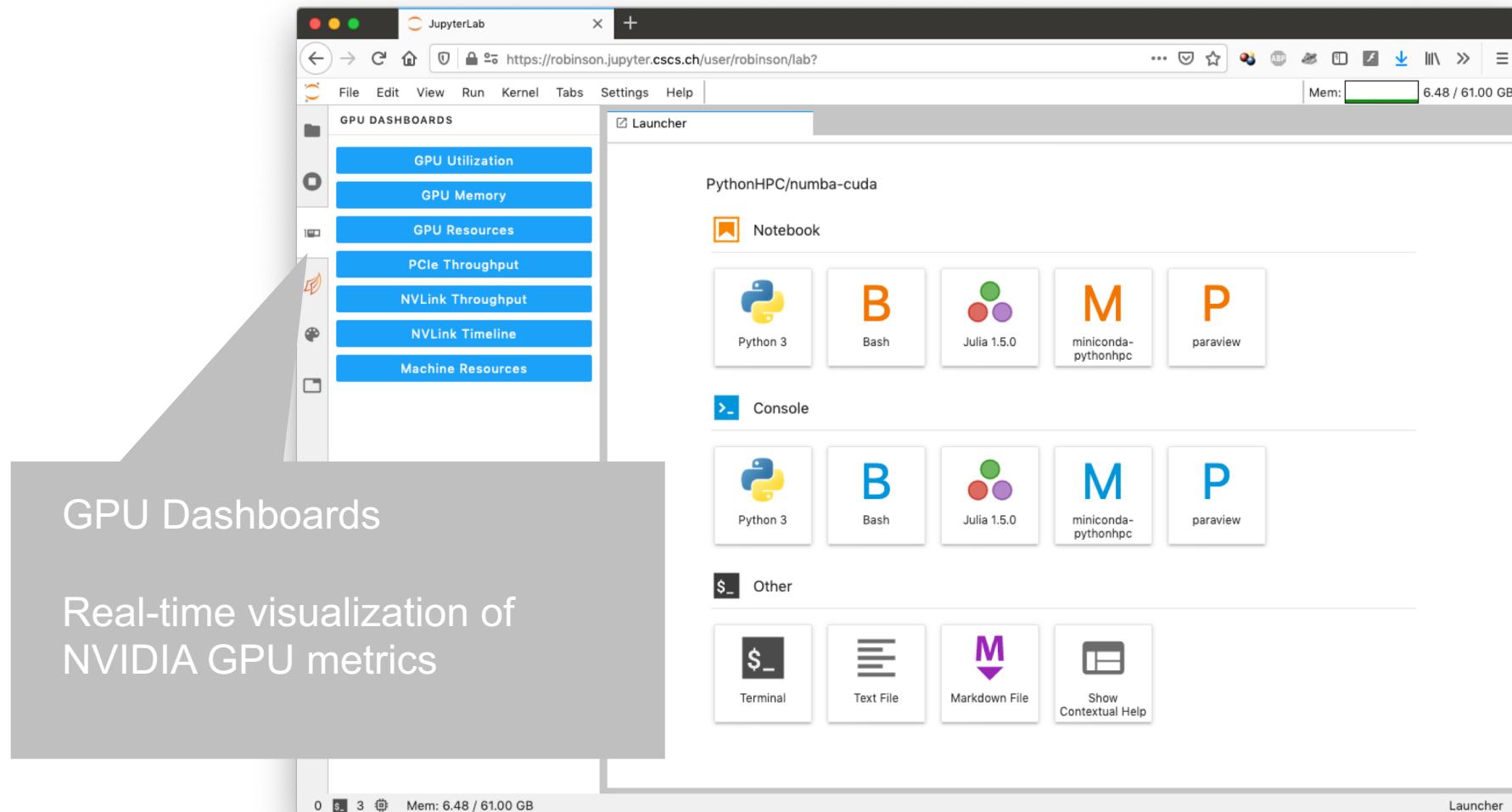
Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

**ETH**zürich

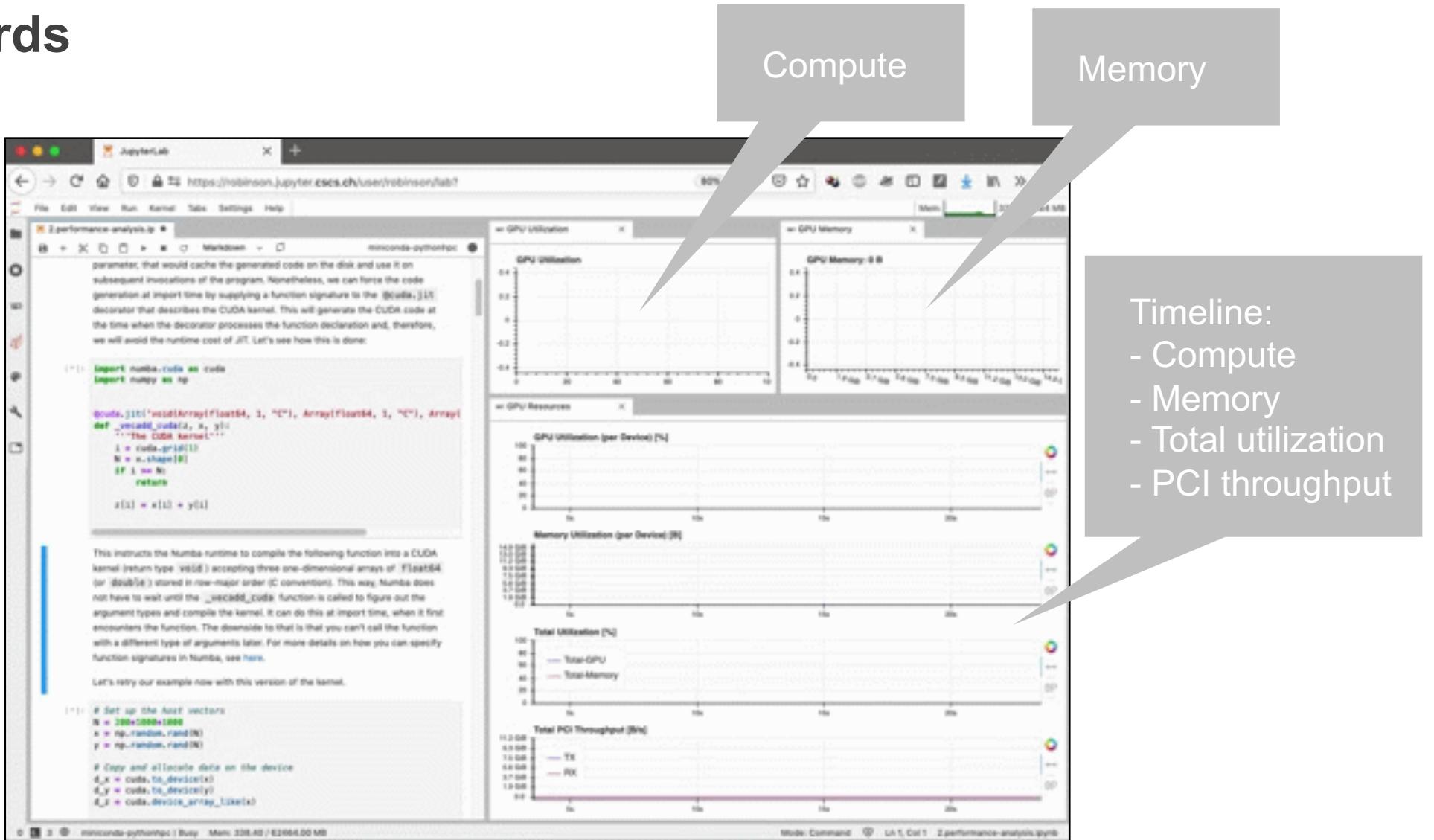
# GPU dashboards

---

# GPU dashboards



# GPU dashboards





**CSCS**

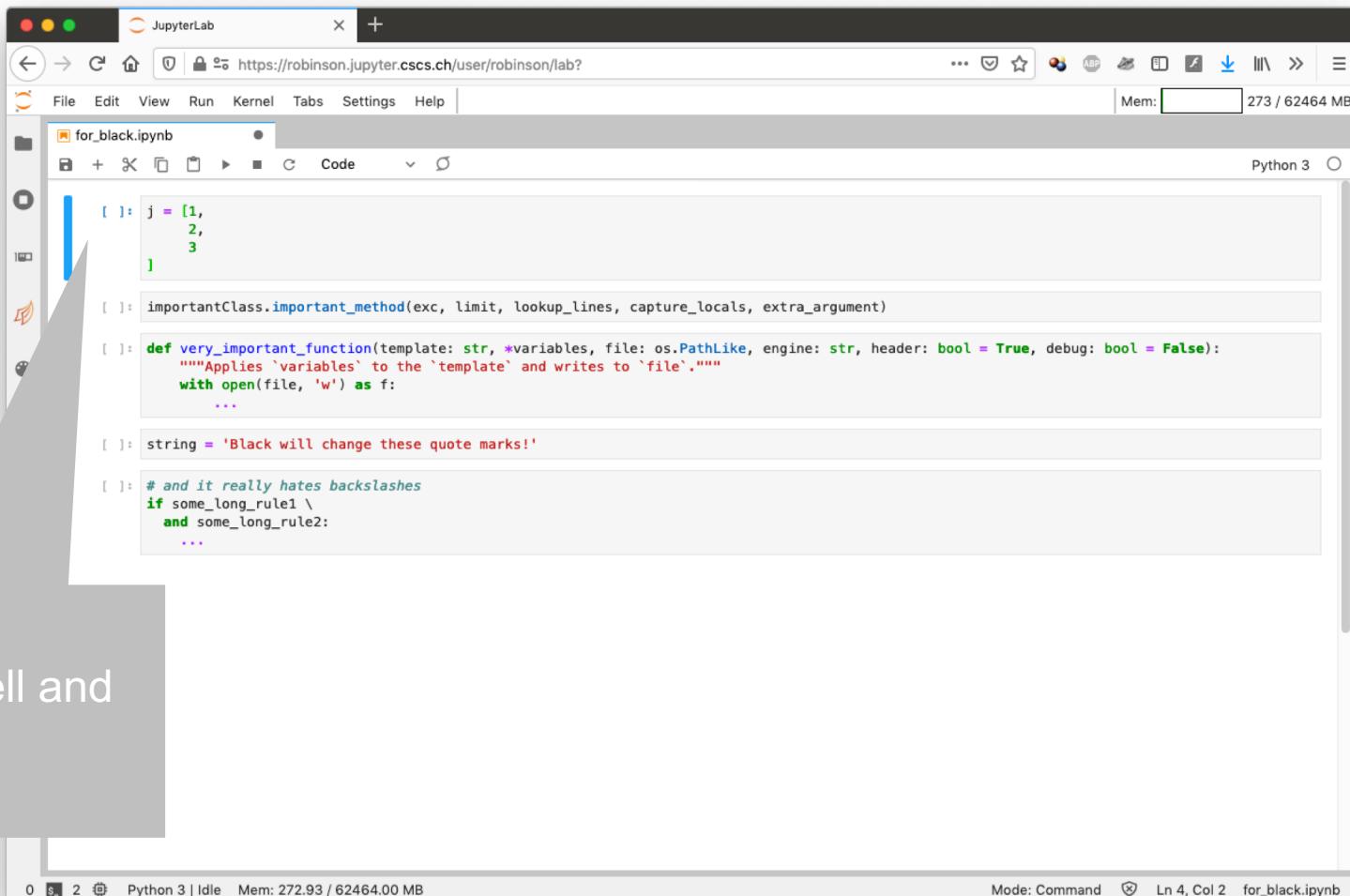
Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

**ETH**zürich

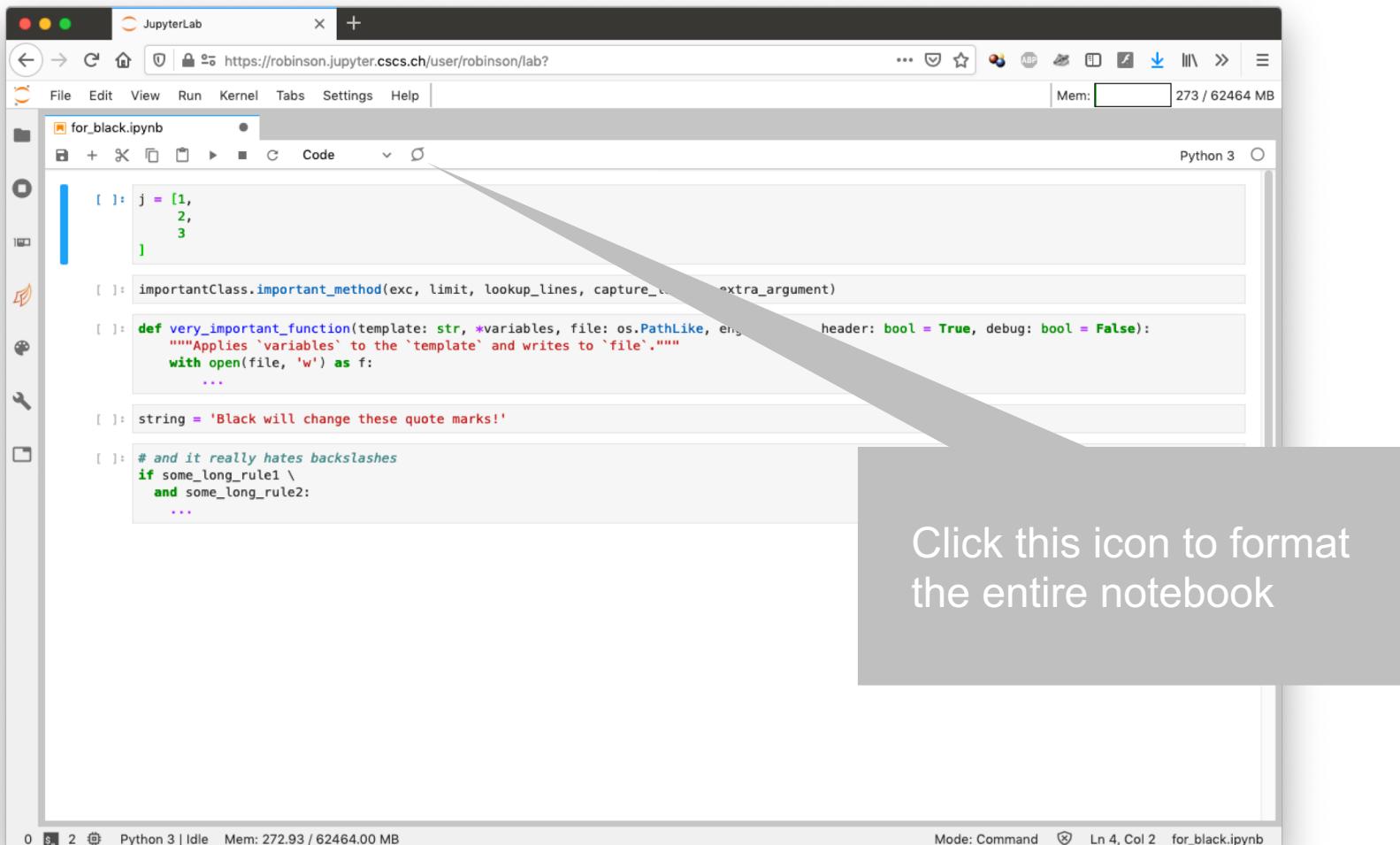
# JupyterLab code formatter

---

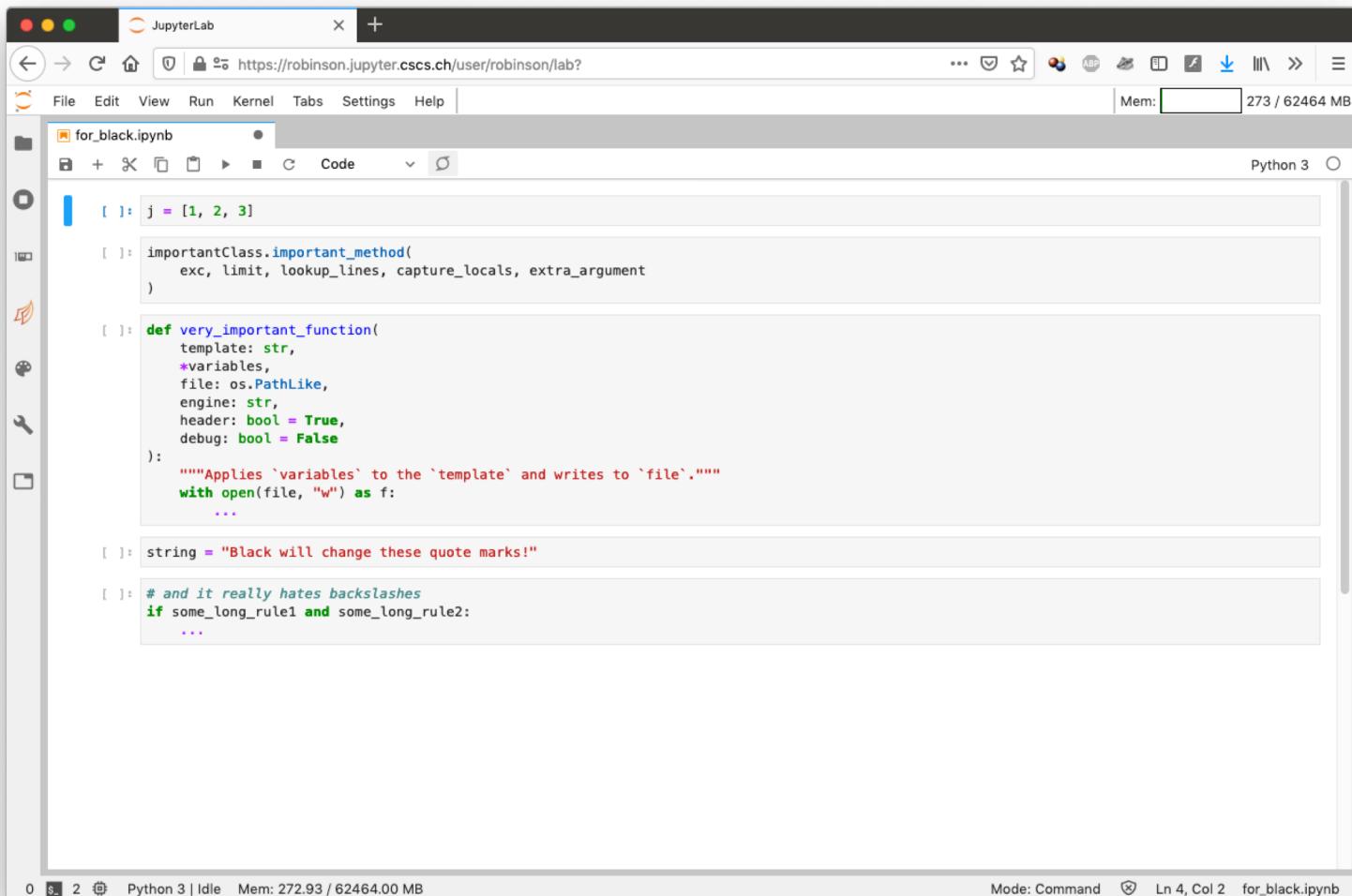
# Code formatter



# Code formatter

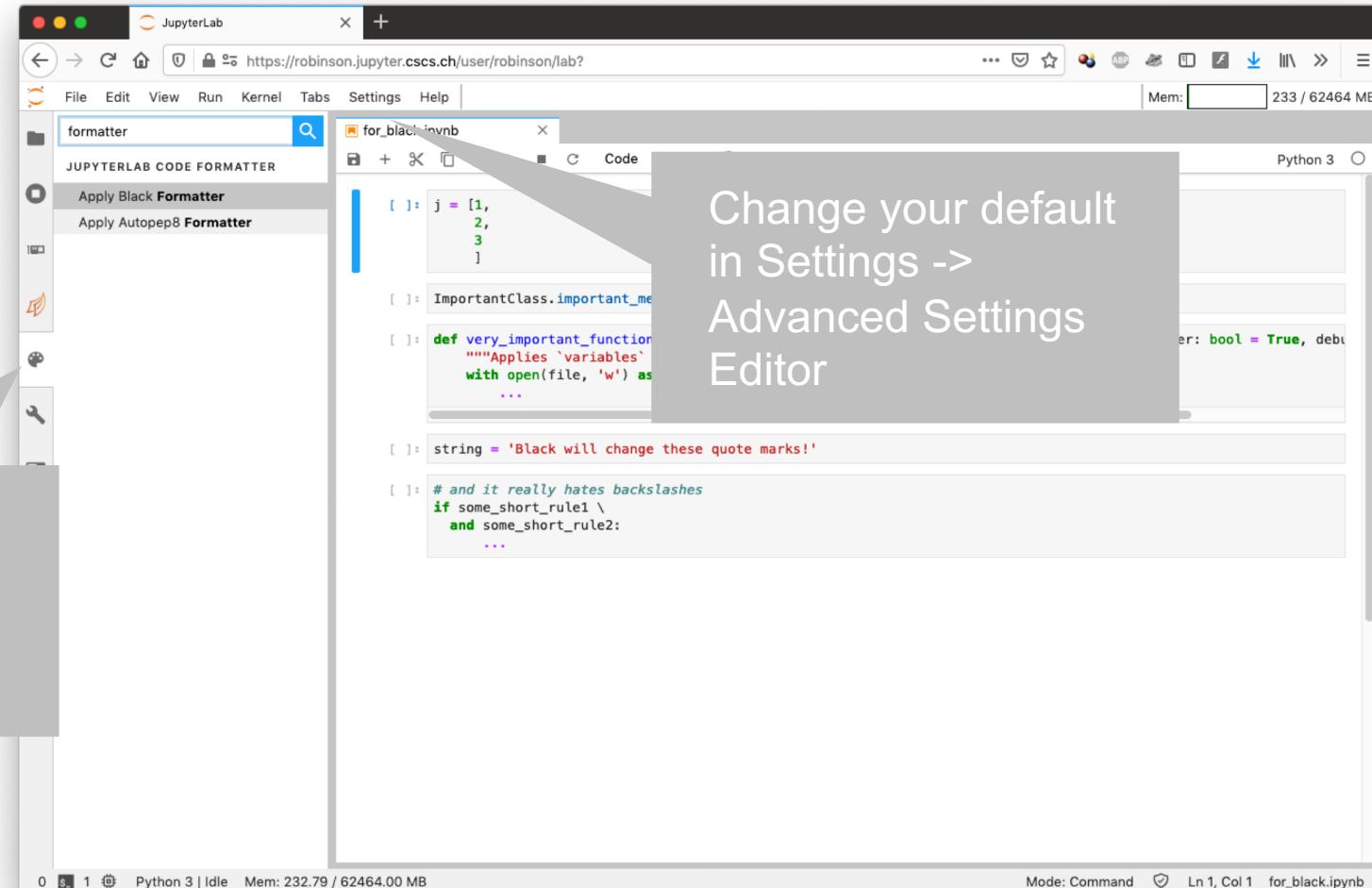


# Code formatter



# Code formatter

Select your own  
formatter in the  
command palette





**CSCS**

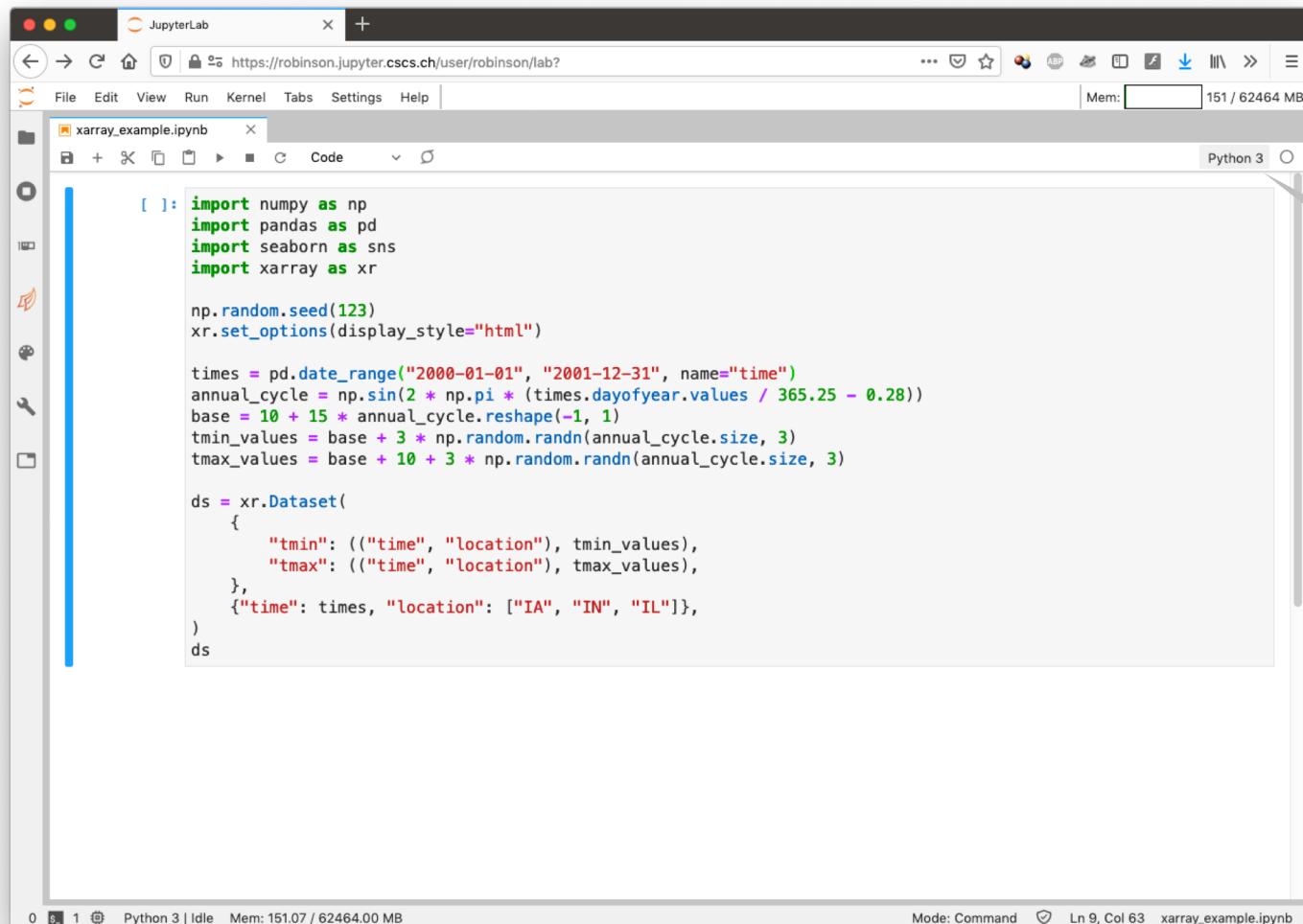
Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

**ETH**zürich

# Customizing your Python environment

---

# Adding Python modules with pip



The screenshot shows a JupyterLab interface with a code cell containing Python 3 code. The code imports numpy, pandas, seaborn, and xarray, and generates a dataset with annual temperature cycles for three locations. A callout box points to the "Python 3" label in the top right corner of the code cell, indicating it is the default kernel.

```
[ ]: import numpy as np
import pandas as pd
import seaborn as sns
import xarray as xr

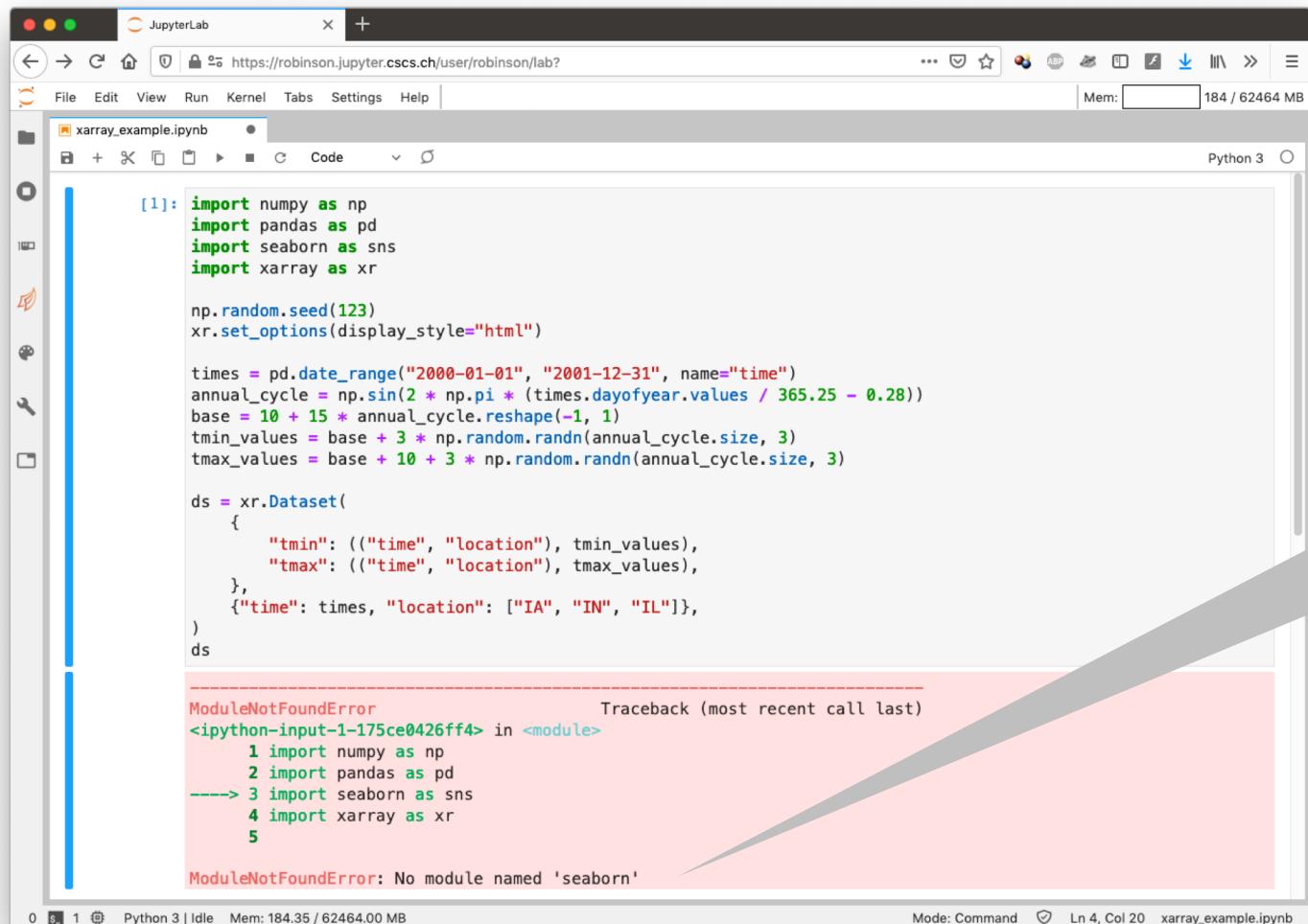
np.random.seed(123)
xr.set_options(display_style="html")

times = pd.date_range("2000-01-01", "2001-12-31", name="time")
annual_cycle = np.sin(2 * np.pi * (times.dayofyear.values / 365.25 - 0.28))
base = 10 + 15 * annual_cycle.reshape(-1, 1)
tmin_values = base + 3 * np.random.rand(annual_cycle.size, 3)
tmax_values = base + 10 + 3 * np.random.rand(annual_cycle.size, 3)

ds = xr.Dataset(
    {
        "tmin": (("time", "location"), tmin_values),
        "tmax": (("time", "location"), tmax_values),
    },
    {"time": times, "location": ["IA", "IN", "IL"]},
)
ds
```

Default  
kernel:  
“Python 3”

# Adding Python modules with pip



The screenshot shows a JupyterLab interface with a code cell containing Python code. The code imports numpy, pandas, seaborn, and xarray, and then defines a dataset with time and location dimensions. A ModuleNotFoundError is raised when attempting to import seaborn.

```
[1]: import numpy as np
import pandas as pd
import seaborn as sns
import xarray as xr

np.random.seed(123)
xr.set_options(display_style="html")

times = pd.date_range("2000-01-01", "2001-12-31", name="time")
annual_cycle = np.sin(2 * np.pi * (times.dayofyear.values / 365.25 - 0.28))
base = 10 + 15 * annual_cycle.reshape(-1, 1)
tmin_values = base + 3 * np.random.rand(annual_cycle.size, 3)
tmax_values = base + 10 + 3 * np.random.rand(annual_cycle.size, 3)

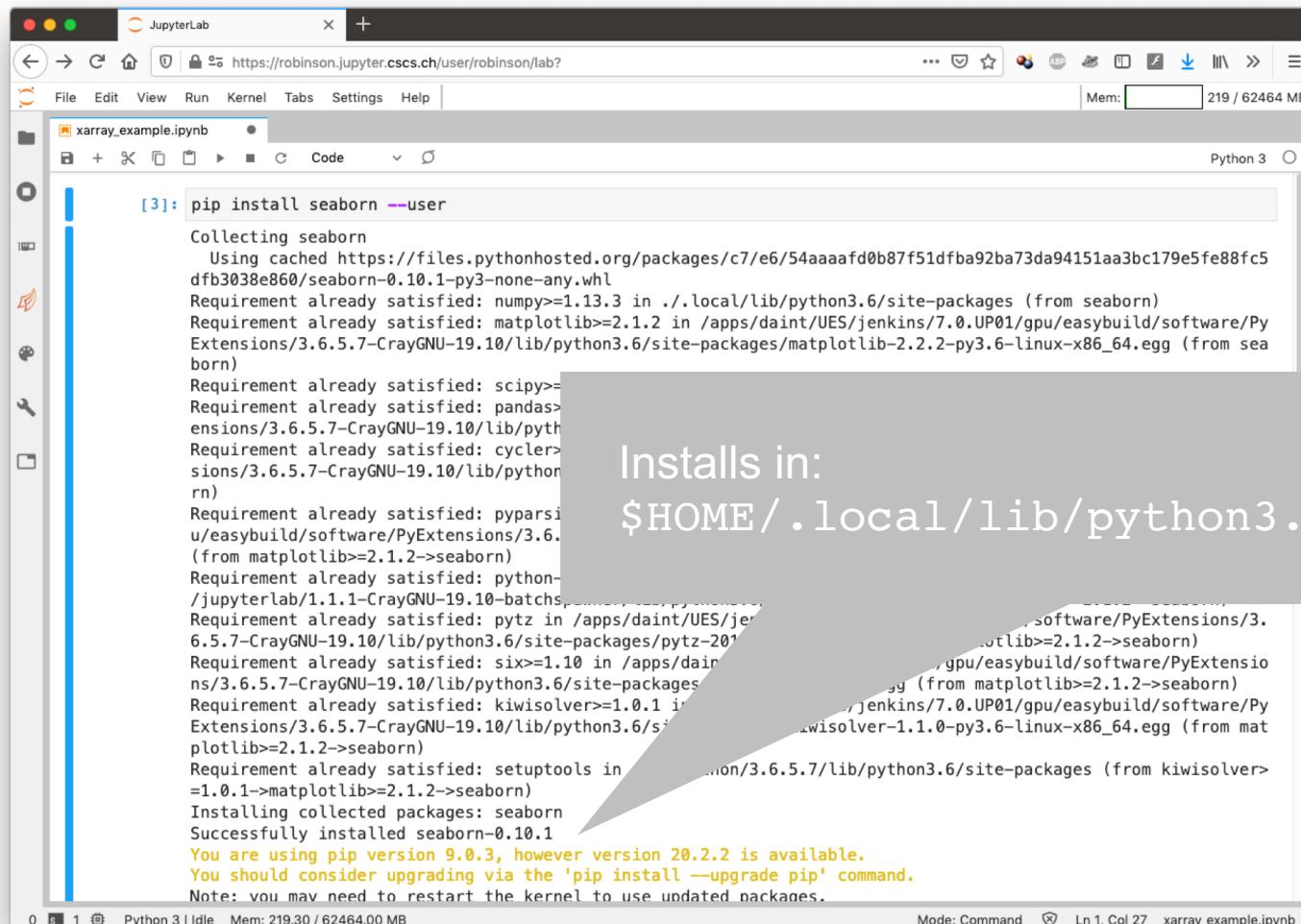
ds = xr.Dataset(
    {
        "tmin": (("time", "location"), tmin_values),
        "tmax": (("time", "location"), tmax_values),
    },
    {"time": times, "location": ["IA", "IN", "IL"]},
)
ds
```

```
ModuleNotFoundError                         Traceback (most recent call last)
<ipython-input-1-175ce0426ff4> in <module>
      1 import numpy as np
      2 import pandas as pd
----> 3 import seaborn as sns
      4 import xarray as xr
      5

ModuleNotFoundError: No module named 'seaborn'
```

Missing:  
seaborn,  
xarray, ...

# Adding Python modules with pip



A screenshot of a JupyterLab interface. The top bar shows the title "JupyterLab" and the URL "https://robinson.jupyter.csccs.ch/user/robinson/lab?". The main area is a terminal window titled "[3]:" containing the command:

```
pip install seaborn --user
```

The output of the command is displayed below:

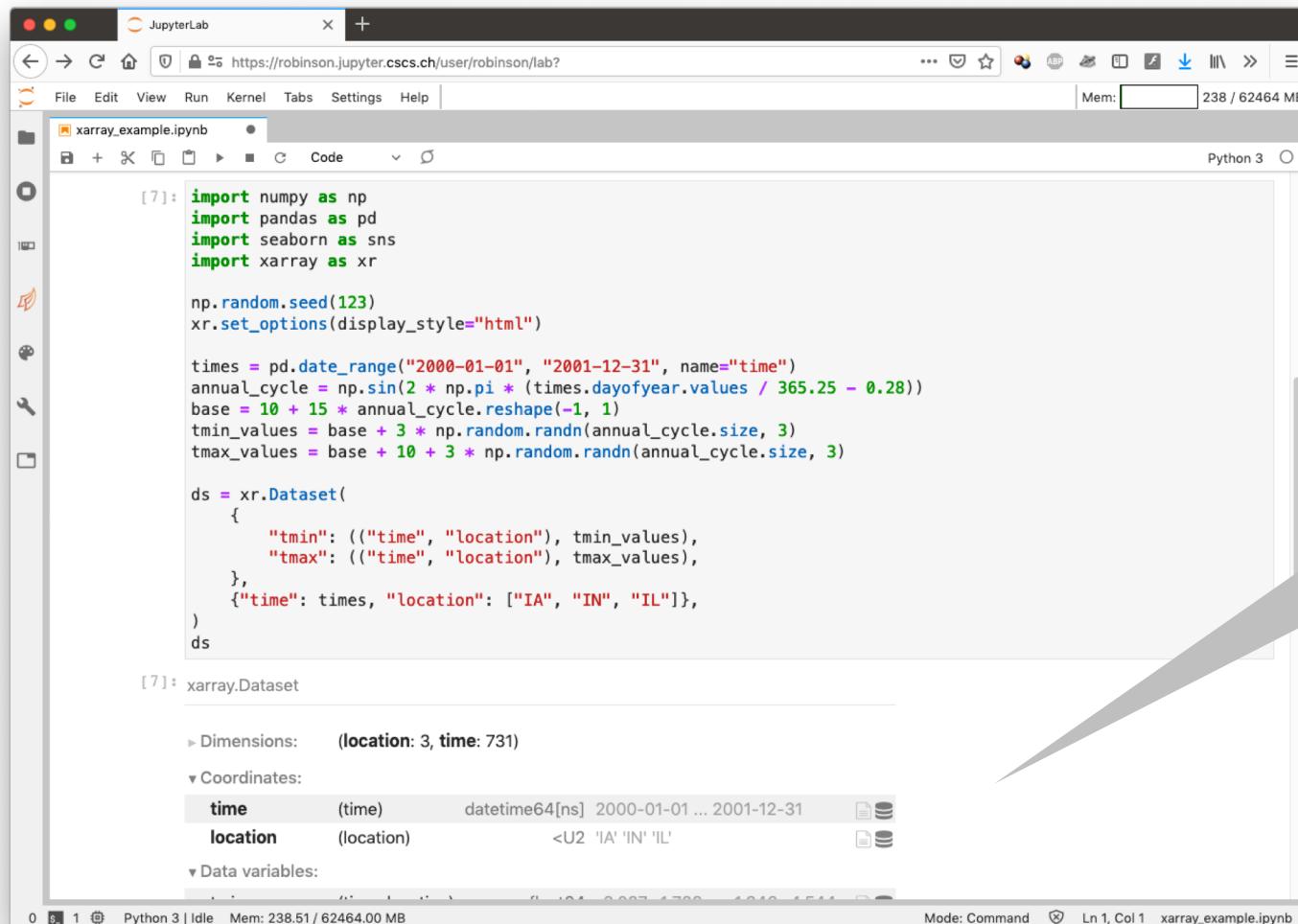
```
Collecting seaborn
  Using cached https://files.pythonhosted.org/packages/c7/e6/54aaaafdf0b87f51dfba92ba73da94151aa3bc179e5fe88fc5
  Requirement already satisfied: numpy>=1.13.3 in ./local/lib/python3.6/site-packages (from seaborn)
  Requirement already satisfied: matplotlib>=2.1.2 in /apps/daint/UES/jenkins/7.0.0.P01/gpu/easybuild/software/Py
  Extentions/3.6.5.7-CrayGNU-19.10/lib/python3.6/site-packages/matplotlib-2.2.2-py3.6-linux-x86_64.egg (from sea
  born)
  Requirement already satisfied: scipy>=
  Requirement already satisfied: pandas>
  ensions/3.6.5.7-CrayGNU-19.10/lib/pyt
  Requirement already satisfied: cycler>
  sions/3.6.5.7-CrayGNU-19.10/lib/pytho
  rn)
  Requirement already satisfied: pypari
  u/easybuild/software/PyExtensions/3.6.
  (from matplotlib>=2.1.2->seaborn)
  Requirement already satisfied: python-
  /jupyterlab/1.1.1-CrayGNU-19.10-batchs
  Requirement already satisfied: pytz in /apps/daint/UES/jen
  6.5.7-CrayGNU-19.10/lib/python3.6/site-packages/pytz-201
  Requirement already satisfied: six>=1.10 in /apps/dain
  ns/3.6.5.7-CrayGNU-19.10/lib/python3.6/site-packages
  Requirement already satisfied: kiwisolver>=1.0.1 in /app
  jenkins/7.0.0.P01/gpu/easybuild/software/Py
  Extentions/3.6.5.7-CrayGNU-19.10/lib/python3.6/si
  Requirement already satisfied: setuptools in /app/jen
  6.5.7/lib/python3.6/site-packages (from kiwisolver>
  =1.0.1->matplotlib>=2.1.2->seaborn)
  Installing collected packages: seaborn
  Successfully installed seaborn-0.10.1
  You are using pip version 9.0.3, however version 20.2.2 is available.
  You should consider upgrading via the 'pip install --upgrade pip' command.
  Note: you may need to restart the kernel to use undated packages.
```

The bottom status bar shows "Mode: Command" and "Ln 1, Col 27".

Installs in:

\$HOME/.local/lib/python3.x/site-packages/

# Adding Python modules with pip



A screenshot of a JupyterLab interface. The title bar says "JupyterLab" and the address bar shows the URL "https://robinson.jupyter.csccs.ch/user/robinson/lab?". The main area displays a code cell numbered [7] containing Python code to import various modules and create a dataset. The code imports numpy, pandas, seaborn, and xarray, sets a random seed, and creates a dataset with time and location coordinates. Below the code, the output shows the dataset's dimensions and coordinates. A large gray arrow points from the bottom right towards the "Success!" text.

```
[7]: import numpy as np
import pandas as pd
import seaborn as sns
import xarray as xr

np.random.seed(123)
xr.set_options(display_style="html")

times = pd.date_range("2000-01-01", "2001-12-31", name="time")
annual_cycle = np.sin(2 * np.pi * (times.dayofyear.values / 365.25 - 0.28))
base = 10 + 15 * annual_cycle.reshape(-1, 1)
tmin_values = base + 3 * np.random.randn(annual_cycle.size, 3)
tmax_values = base + 10 + 3 * np.random.randn(annual_cycle.size, 3)

ds = xr.Dataset(
    {
        "tmin": (("time", "location"), tmin_values),
        "tmax": (("time", "location"), tmax_values),
    },
    {"time": times, "location": ["IA", "IN", "IL"]},
)
ds
```

[7]: xarray.Dataset

► Dimensions: (location: 3, time: 731)

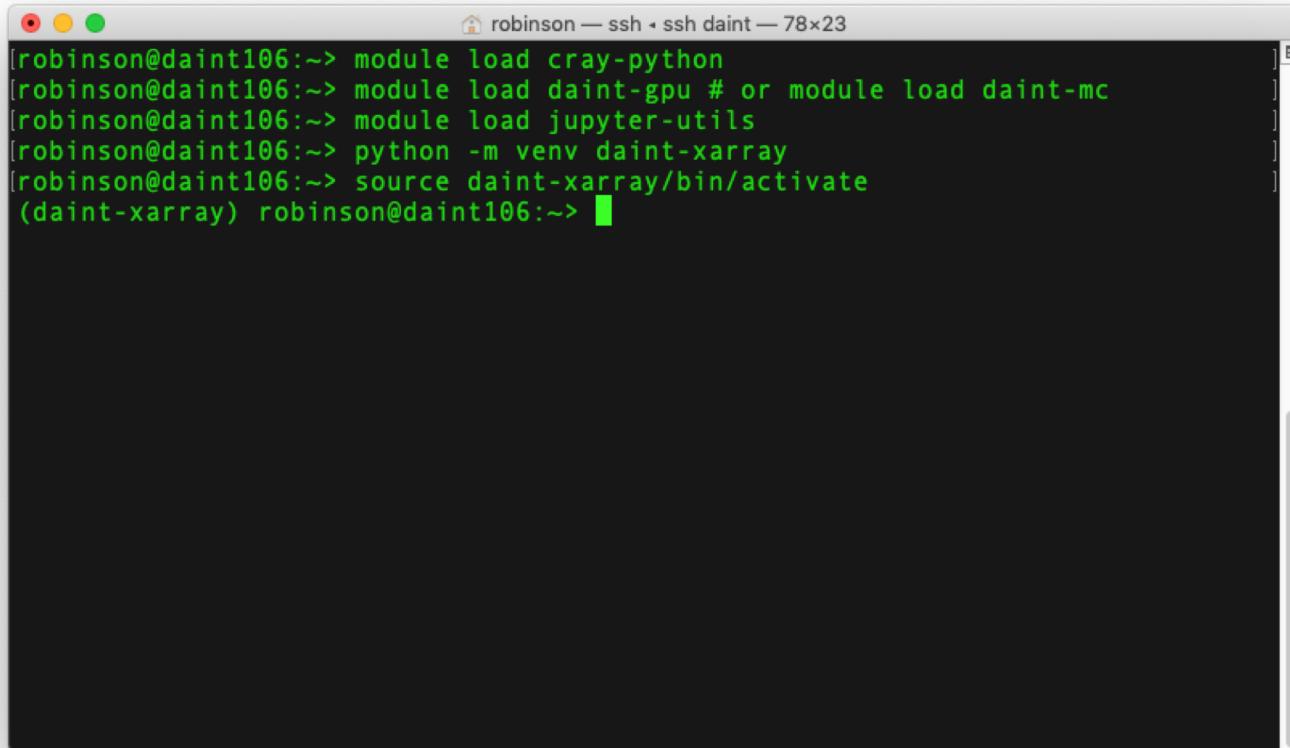
▼ Coordinates:

time	(time)	datetime64[ns]
		2000-01-01 ... 2001-12-31

▼ Data variables:

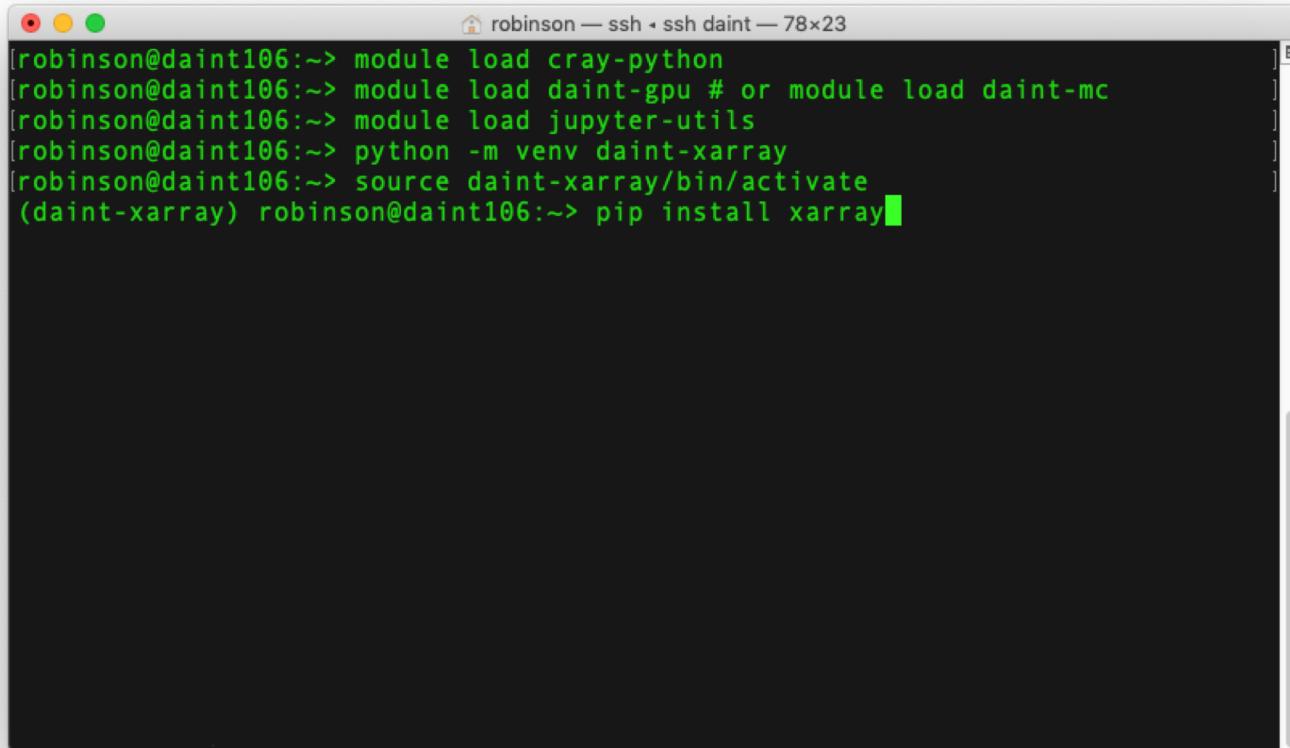
Success!

# Creating and installing a custom kernel (recommended)



```
robinson — ssh — ssh daint — 78x23
[robinson@daint106:~] module load cray-python
[robinson@daint106:~] module load daint-gpu # or module load daint-mc
[robinson@daint106:~] module load jupyter-utils
[robinson@daint106:~] python -m venv daint-xarray
[robinson@daint106:~] source daint-xarray/bin/activate
(daint-xarray) robinson@daint106:~] █
```

# Creating and installing a custom kernel (recommended)



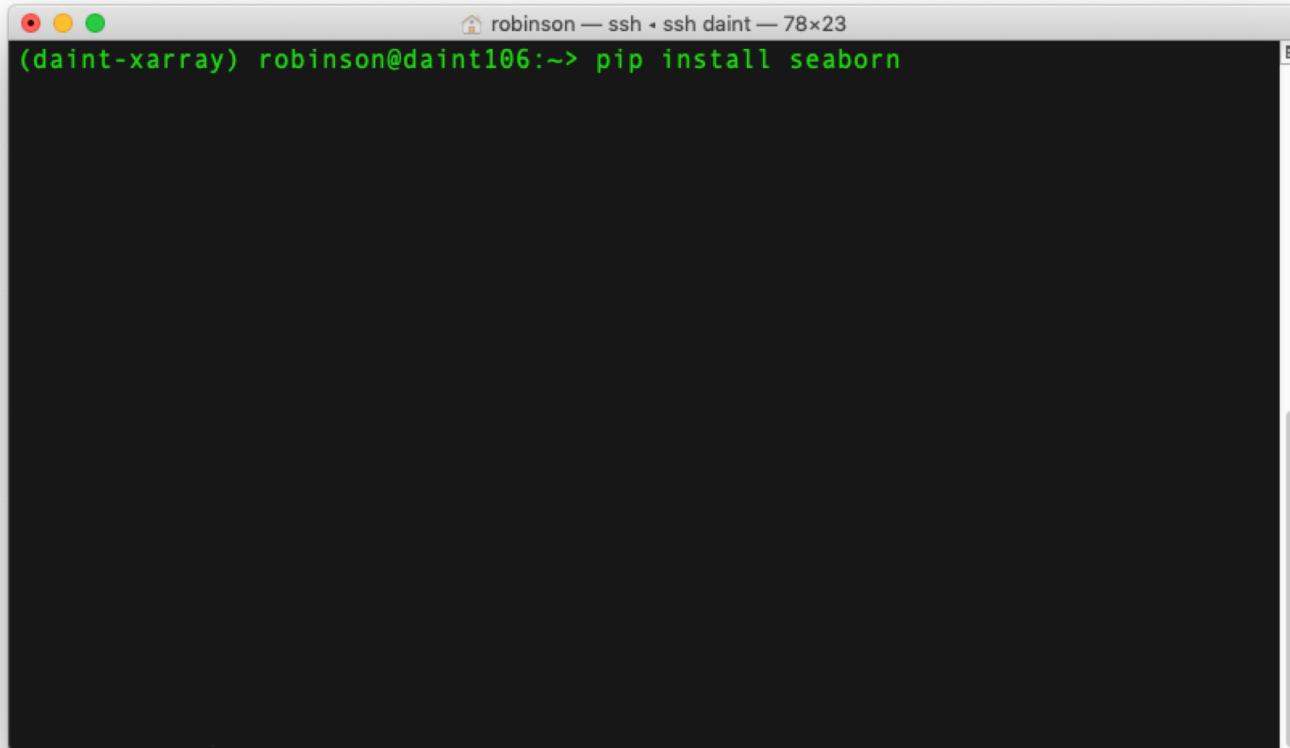
The screenshot shows a terminal window titled "robinson — ssh - ssh daint — 78x23". The window contains the following command sequence:

```
[robinson@daint106:~] module load cray-python  
[robinson@daint106:~] module load daint-gpu # or module load daint-mc  
[robinson@daint106:~] module load jupyter-utils  
[robinson@daint106:~] python -m venv daint-xarray  
[robinson@daint106:~] source daint-xarray/bin/activate  
(daint-xarray) robinson@daint106:~] pip install xarray
```

# Creating and installing a custom kernel (recommended)

```
robinson — ssh -> ssh daint — 78x23
Using cached https://files.pythonhosted.org/packages/a1/c6/9ac4ae44c24c787a1
738e5fb34dd987ada6533de5905a041aa6d5bea4553/pandas-1.1.1-cp36-cp36m-manylinux1
_x86_64.whl
Collecting python-dateutil>=2.7.3 (from pandas>=0.25->xarray)
  Using cached https://files.pythonhosted.org/packages/d4/70/d60450c3dd48ef875
86924207ae8907090de0b306af2bc5d134d78615cb/python_dateutil-2.8.1-py3-none
-any.whl
Collecting pytz>=2017.2 (from pandas>=0.25->xarray)
  Using cached https://files.pythonhosted.org/packages/4f/a4/879454d49688e2fad
93e59d7d4efda580b783c745fd2ec2a3adf87b0808d/pytz-2020.1-py2.py3-none-any.whl
Collecting six>=1.5 (from python-dateutil>=2.7.3->pandas>=0.25->xarray)
  Using cached https://files.pythonhosted.org/packages/ee/ff/48bde5c0f013094d7
29fe4b0316ba2a24774b3ff1c52d924a8a4cb04078a/six-1.15.0-py2.py3-none-any.whl
Installing collected packages: setuptools, numpy, six, python-dateutil, pytz,
pandas, xarray
  Found existing installation: setuptools 39.0.1
    Uninstalling setuptools-39.0.1:
      Successfully uninstalled setuptools-39.0.1
Successfully installed numpy-1.19.1 pandas-1.1.1 python-dateutil-2.8.1 pytz-20
20.1 setuptools-49.6.0 six-1.15.0 xarray-0.16.0
You are using pip version 9.0.3, however version 20.2.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
(daint-xarray) robinson@daint106:~>
```

# Creating and installing a custom kernel (recommended)

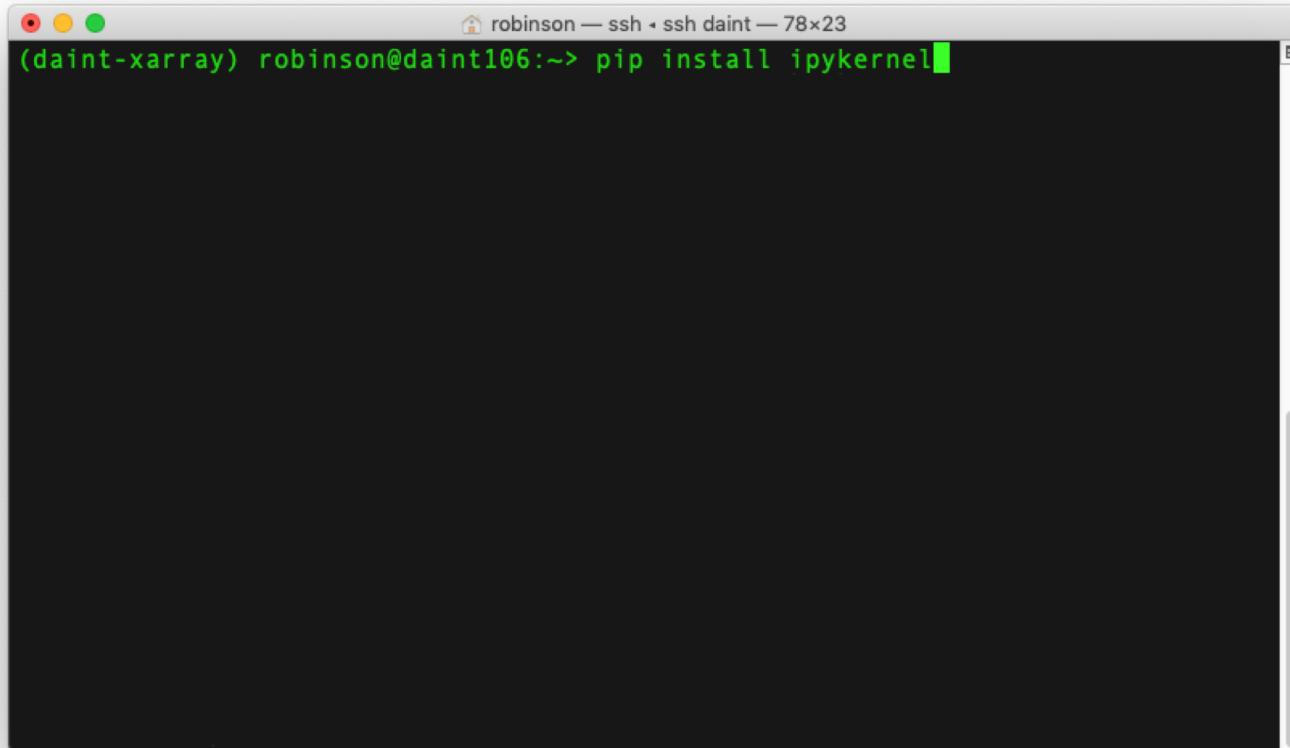


A screenshot of a terminal window titled "robinson — ssh - ssh daint — 78x23". The window shows a single line of green text: "(daint-xarray) robinson@daint106:~> pip install seaborn". The terminal has a dark background and light-colored text.

# Creating and installing a custom kernel (recommended)

```
robinson — ssh -> ssh daint — 78x23
nux1_x86_64.whl
Requirement already satisfied: python-dateutil>=2.1 in ./daint-xarray/lib/python3.6/site-packages (from matplotlib>=2.1.2->seaborn)
Collecting pillow>=6.2.0 (from matplotlib>=2.1.2->seaborn)
  Using cached https://files.pythonhosted.org/packages/30/bf/92385b4262178ca22b34f82e0e09c2922eb351fe39f3cc7b8ba9ea555b41/Pillow-7.2.0-cp36-cp36m-manylinux1_x86_64.whl
Collecting pyparsing!=2.0.4,!>=2.1.2,!>=2.1.6,>=2.0.3 (from matplotlib>=2.1.2->seaborn)
  Using cached https://files.pythonhosted.org/packages/8a/bb/488841f56197b13700afd5658fc279a2025a39e22449b7cf29864669b15d/pyparsing-2.4.7-py2.py3-none-any.whl
Requirement already satisfied: pytz>=2017.2 in ./daint-xarray/lib/python3.6/site-packages (from pandas>=0.22.0->seaborn)
Requirement already satisfied: six in ./daint-xarray/lib/python3.6/site-packages (from cycler>=0.10->matplotlib>=2.1.2->seaborn)
Installing collected packages: certifi, cycler, kiwisolver, pillow, pyparsing, matplotlib, scipy, seaborn
Successfully installed certifi-2020.6.20 cycler-0.10.0 kiwisolver-1.2.0 matplotlib-3.3.1 pillow-7.2.0 pyparsing-2.4.7 scipy-1.5.2 seaborn-0.10.1
You are using pip version 9.0.3, however version 20.2.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
(daint-xarray) robinson@daint106:~>
```

# Creating and installing a custom kernel (recommended)

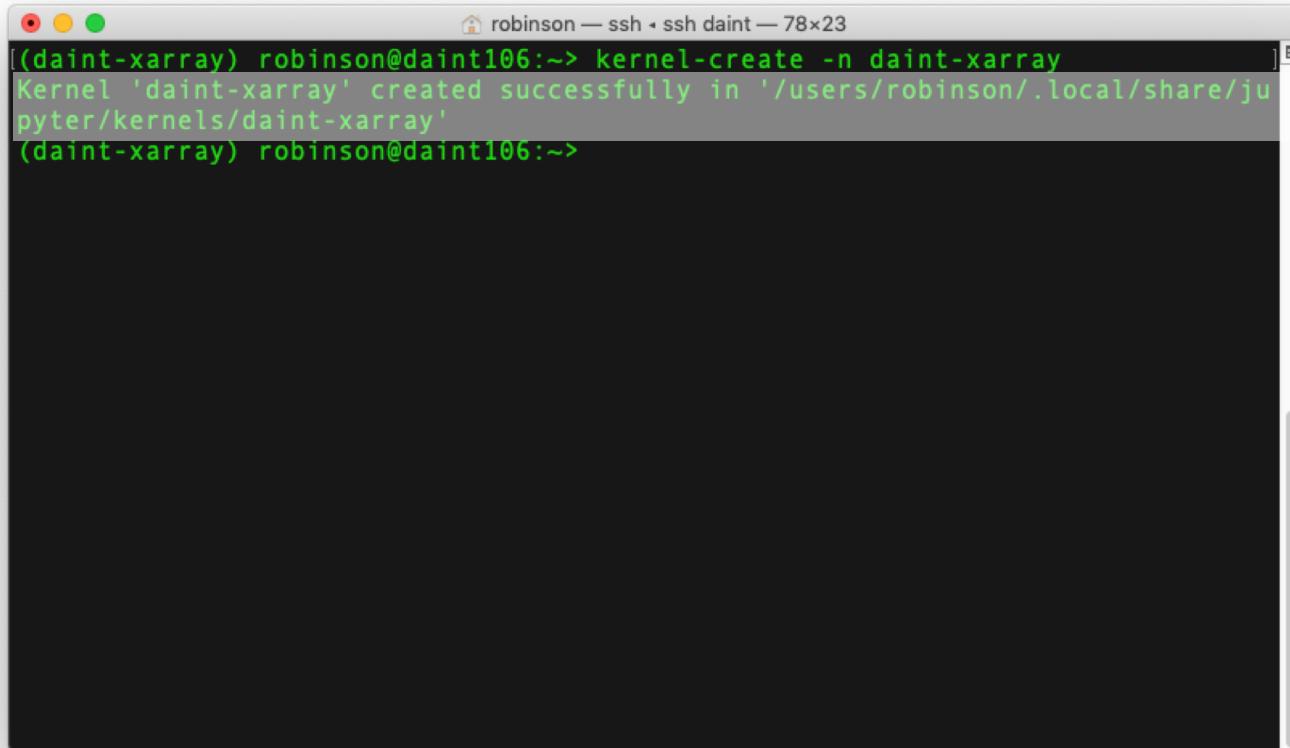
A screenshot of a terminal window titled "robinson — ssh - ssh daint — 78x23". The window has three colored window control buttons (red, yellow, green) at the top left. The terminal prompt shows "(daint-xarray) robinson@daint106:~>". A green cursor is visible at the end of the command "pip install ipykernel".

```
(daint-xarray) robinson@daint106:~> pip install ipykernel
```

# Creating and installing a custom kernel (recommended)

```
robinson — ssh -> ssh daint — 78x23
Using cached https://files.pythonhosted.org/packages/93/d1/e635bdde32890db5a
eb2ffbd17e74f68986305a4466b0aa373b861e3f00/parso-0.7.1-py2.py3-none-any.whl
Collecting wcwidth (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython
>=5.0.0->ipykernel)
  Using cached https://files.pythonhosted.org/packages/59/7c/e39aca596badaf1b7
8e8f547c807b04dae603a433d3e7a7e04d67f2ef3e5/wcwidth-0.2.5-py2.py3-none-any.whl
Collecting ptyprocess>=0.5 (from pexpect; sys_platform != "win32"->ipython>=5.
0.0->ipykernel)
  Using cached https://files.pythonhosted.org/packages/d1/29/605c2cc68a9992d18
dada28206eeada56ea4bd07a239669da41674648b6f/ptyprocess-0.6.0-py2.py3-none-any.
whl
Installing collected packages: tornado, decorator, ipython-genutils, traitlets
, jupyter-core, pyzmq, jupyter-client, backcall, pickleshare, parso, jedi, wcw
idth, prompt-toolkit, ptyprocess, pexpect, pygments, ipython, ipykernel
  Running setup.py install for tornado ... done
Successfully installed backcall-0.2.0 decorator-4.4.2 ipykernel-5.3.4 ipython-
7.16.1 ipython-genutils-0.2.0 jedi-0.17.2 jupyter-client-6.1.7 jupyter-core-4.
6.3 parso-0.7.1 pexpect-4.8.0 pickleshare-0.7.5 prompt-toolkit-3.0.6 ptyproces
s-0.6.0 pygments-2.6.1 pyzmq-19.0.2 tornado-6.0.4 traitlets-4.3.3 wcwidth-0.2.
5
You are using pip version 9.0.3, however version 20.2.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
(daint-xarray) robinson@daint106:~>
```

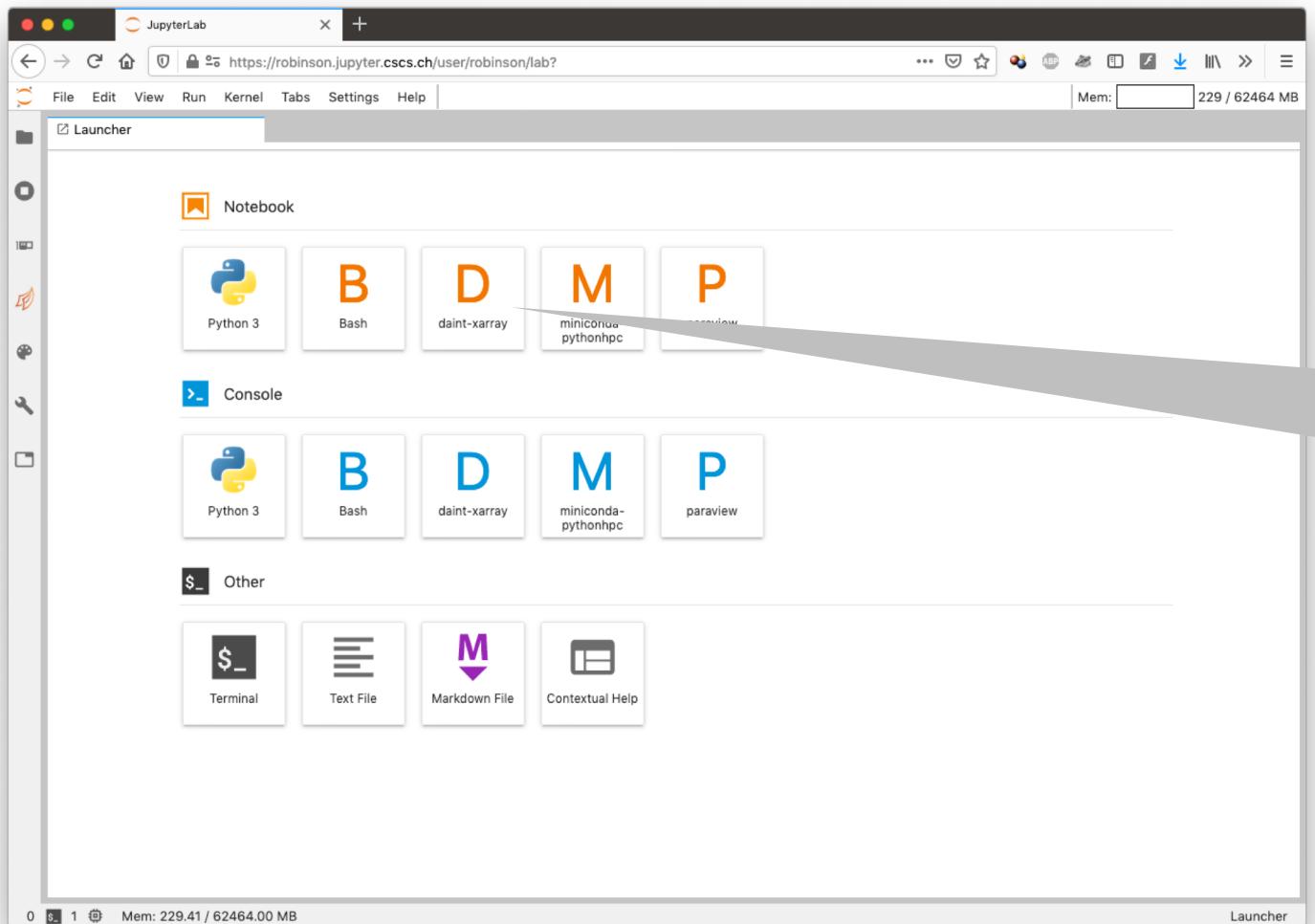
# Creating and installing a custom kernel (recommended)



A screenshot of a terminal window titled "robinson — ssh - ssh daint — 78x23". The window shows the command "kernel-create -n daint-xarray" being run, followed by the message "Kernel 'daint-xarray' created successfully in '/users/robinson/.local/share/jupyter/kernels/daint-xarray'". The terminal has a dark background with light green text.

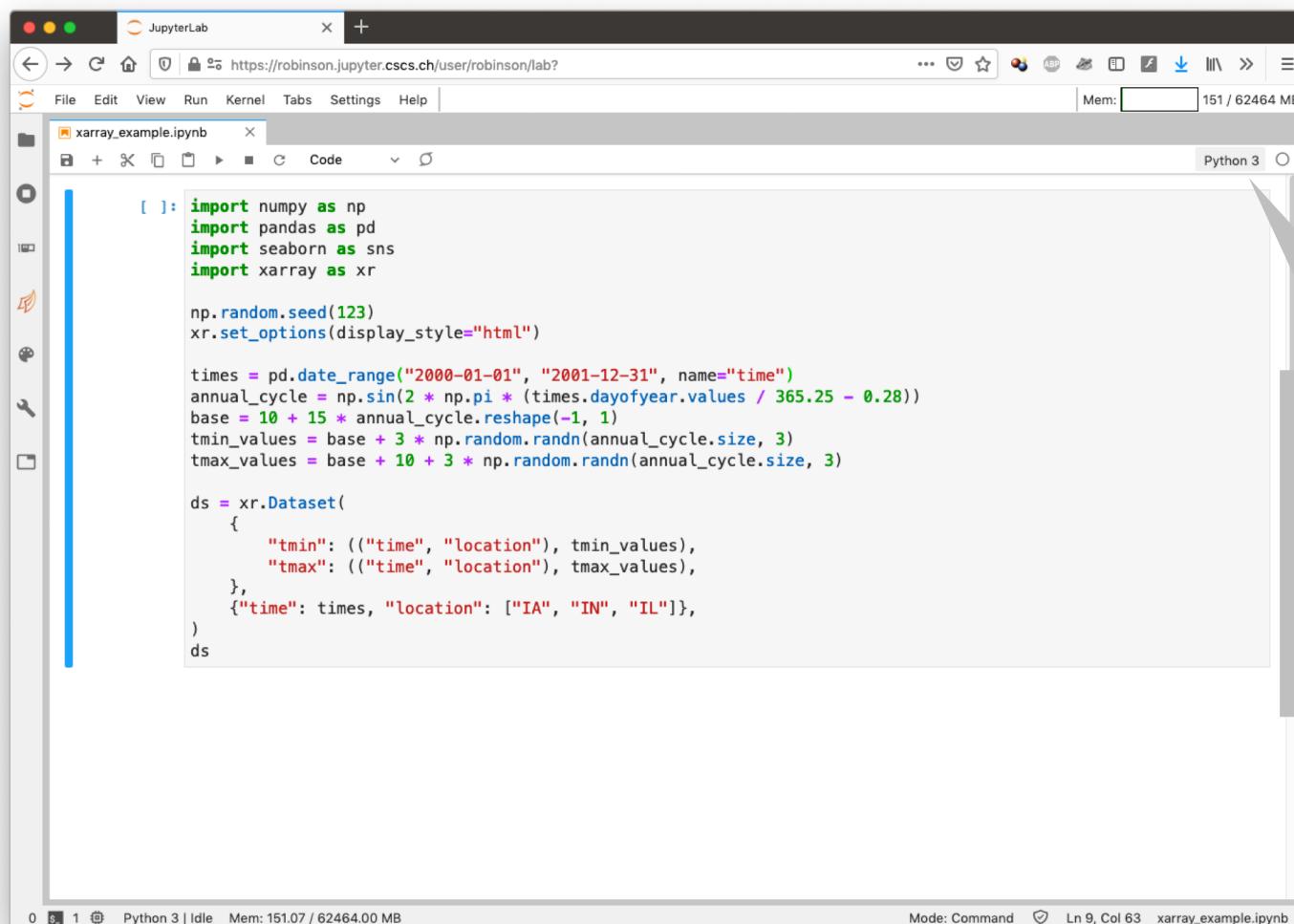
```
(daint-xarray) robinson@daint106:~> kernel-create -n daint-xarray
Kernel 'daint-xarray' created successfully in '/users/robinson/.local/share/jupyter/kernels/daint-xarray'
(daint-xarray) robinson@daint106:~>
```

# Using the new kernel



daint-xarray  
kernel will  
appear  
immediately

# Using the new kernel



The screenshot shows a JupyterLab interface with a notebook titled "xarray\_example.ipynb". A code cell in the first panel contains Python code for generating a dataset. A tooltip with the text "Click to change kernel" points to the kernel selection dropdown in the top right corner of the code cell.

```
[ ]: import numpy as np
import pandas as pd
import seaborn as sns
import xarray as xr

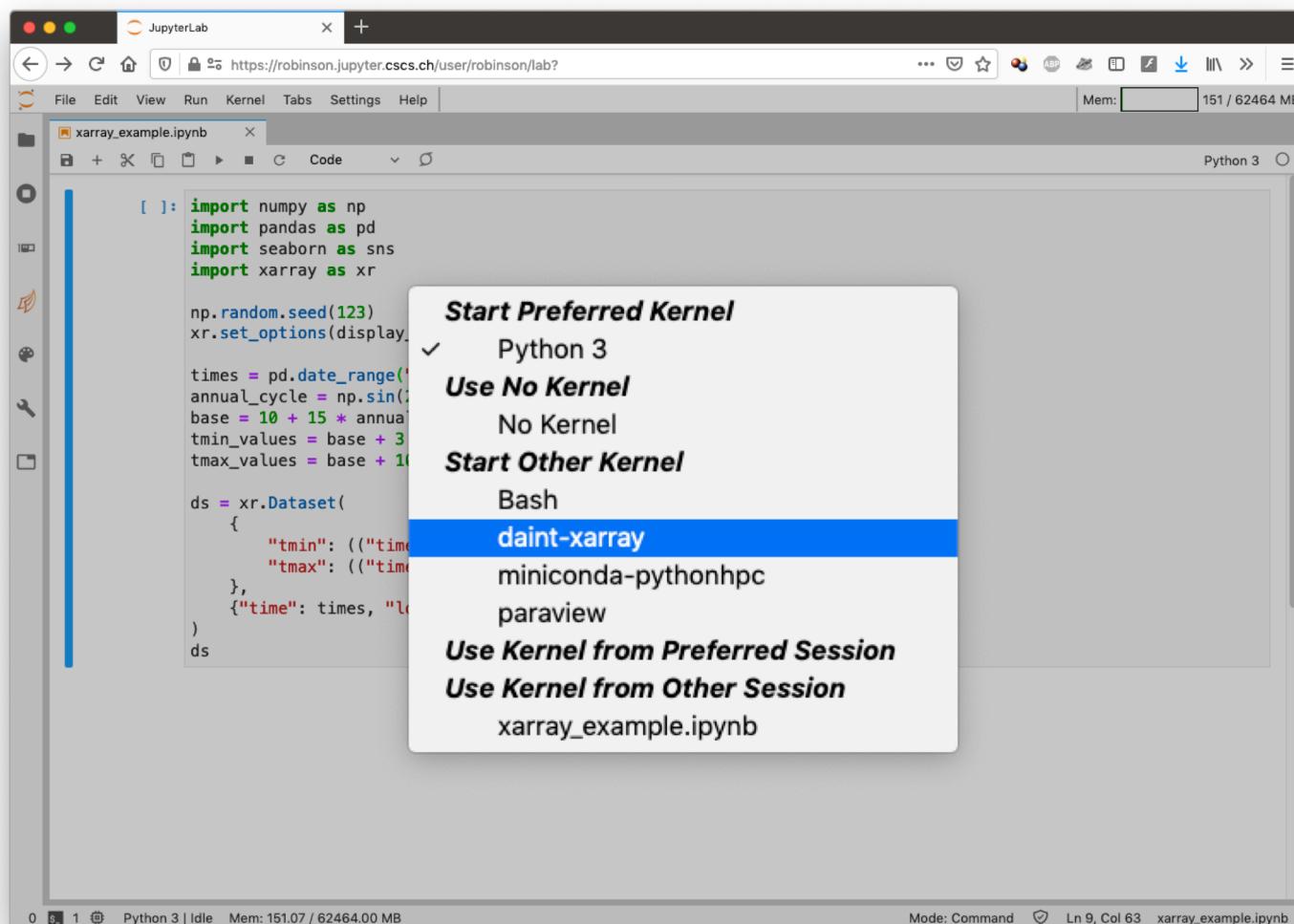
np.random.seed(123)
xr.set_options(display_style="html")

times = pd.date_range("2000-01-01", "2001-12-31", name="time")
annual_cycle = np.sin(2 * np.pi * (times.dayofyear.values / 365.25 - 0.28))
base = 10 + 15 * annual_cycle.reshape(-1, 1)
tmin_values = base + 3 * np.random.randn(annual_cycle.size, 3)
tmax_values = base + 10 + 3 * np.random.randn(annual_cycle.size, 3)

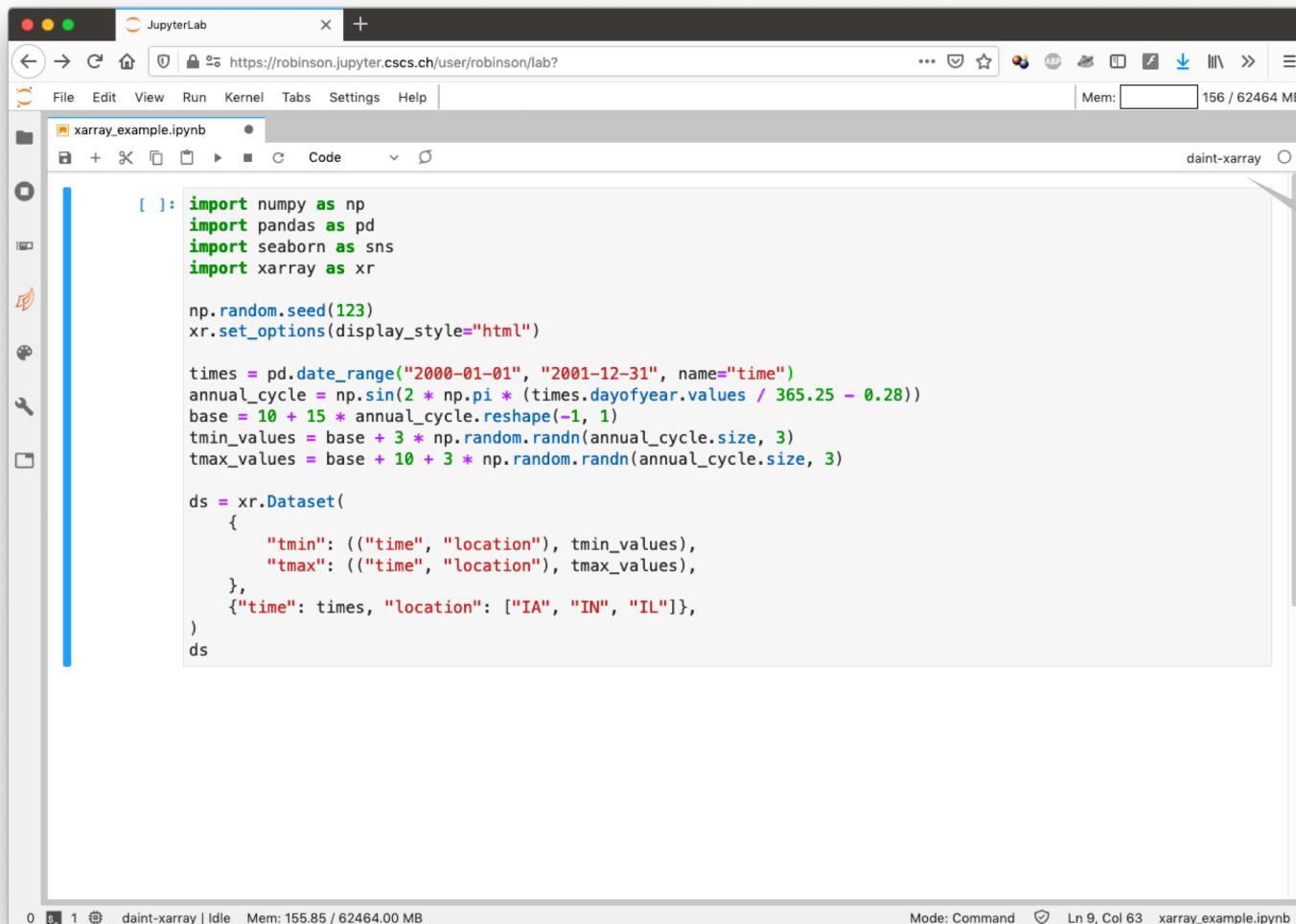
ds = xr.Dataset(
    {
        "tmin": ((["time", "location"], tmin_values),
                  {"time": times, "location": ["IA", "IN", "IL"]}),
        "tmax": ((["time", "location"], tmax_values),
                  {"time": times, "location": ["IA", "IN", "IL"]})
    }
) ds
```

Click to  
change  
kernel

# Using the new kernel



# Using the new kernel



The screenshot shows a JupyterLab interface with a single code cell containing Python code for generating a dataset using xarray. The code imports numpy, pandas, seaborn, and xarray, sets a random seed, creates a date range, generates annual cycle data, and creates a Dataset. A tooltip labeled "daint-xarray kernel" points to the kernel name in the top right corner of the interface.

```
[ ]: import numpy as np
import pandas as pd
import seaborn as sns
import xarray as xr

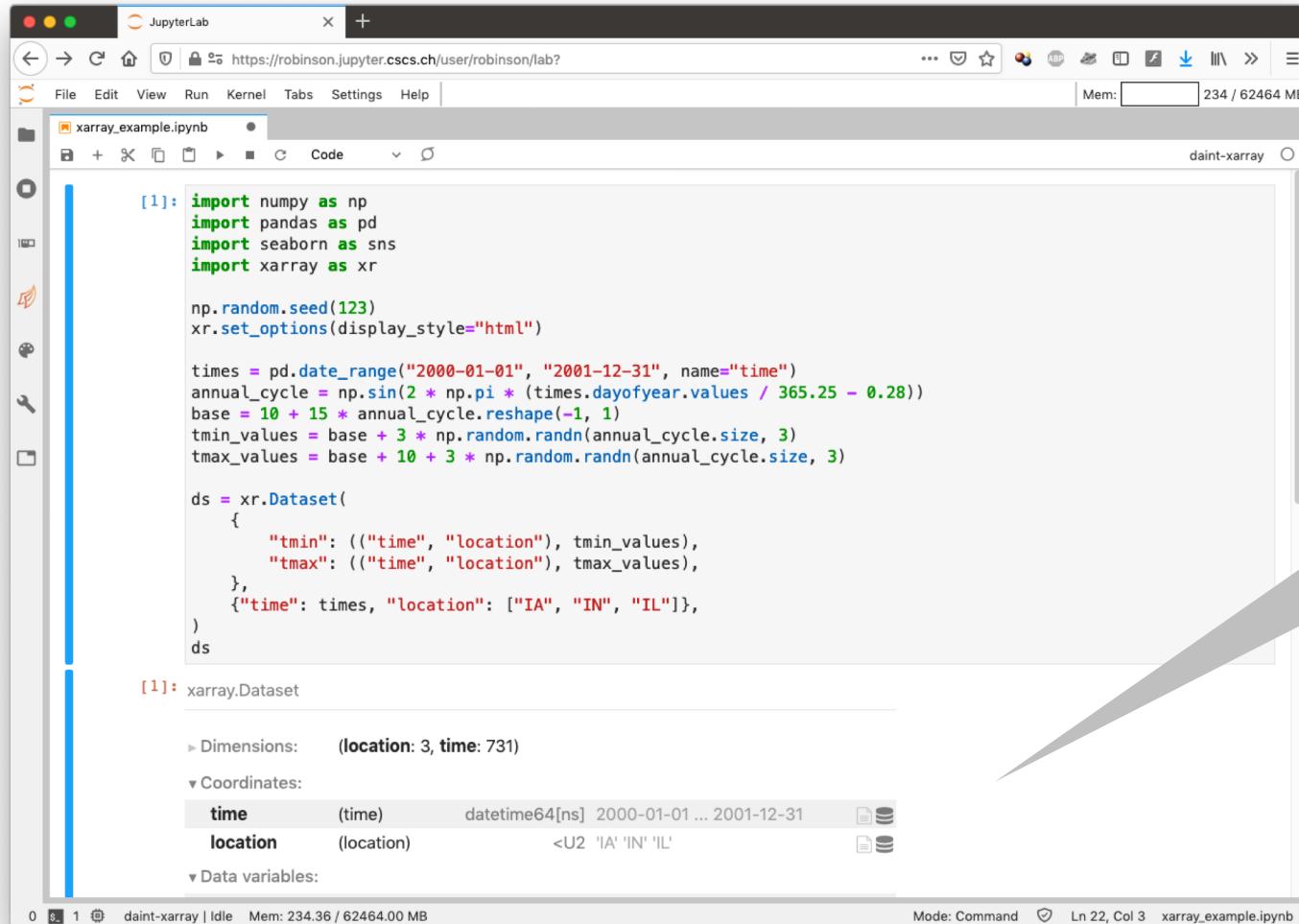
np.random.seed(123)
xr.set_options(display_style="html")

times = pd.date_range("2000-01-01", "2001-12-31", name="time")
annual_cycle = np.sin(2 * np.pi * (times.dayofyear.values / 365.25 - 0.28))
base = 10 + 15 * annual_cycle.reshape(-1, 1)
tmin_values = base + 3 * np.random.rand(annual_cycle.size, 3)
tmax_values = base + 10 + 3 * np.random.rand(annual_cycle.size, 3)

ds = xr.Dataset(
    {
        "tmin": (("time", "location"), tmin_values),
        "tmax": (("time", "location"), tmax_values),
    },
    {"time": times, "location": ["IA", "IN", "IL"]},
)
ds
```

daint-xarray  
kernel

# Using the new kernel



A screenshot of a JupyterLab interface. The title bar says "JupyterLab" and the address bar shows "https://robinson.jupyter.csccs.ch/user/robinson/lab?". The main area displays a code cell [1] containing Python code to generate a dataset. The output below the cell shows the resulting `xarray.Dataset` object with dimensions `(location: 3, time: 731)`. A large gray arrow points from the bottom right towards the output area, and the word "Success!" is written in white on a gray background to the right of the arrow.

```
[1]: import numpy as np
import pandas as pd
import seaborn as sns
import xarray as xr

np.random.seed(123)
xr.set_options(display_style="html")

times = pd.date_range("2000-01-01", "2001-12-31", name="time")
annual_cycle = np.sin(2 * np.pi * (times.dayofyear.values / 365.25 - 0.28))
base = 10 + 15 * annual_cycle.reshape(-1, 1)
tmin_values = base + 3 * np.random.randn(annual_cycle.size, 3)
tmax_values = base + 10 + 3 * np.random.randn(annual_cycle.size, 3)

ds = xr.Dataset(
    {
        "tmin": ((["time", "location"], tmin_values),
                  {"tmax": ((["time", "location"], tmax_values),
                            {"time": times, "location": ["IA", "IN", "IL"]})})
    }
)
ds
```

[1]: `xarray.Dataset`

► Dimensions: `(location: 3, time: 731)`

▼ Coordinates:

time	(time)	datetime64[ns]	2000-01-01 ... 2001-12-31
location	(location)	<U2	'IA' 'IN' 'IL'

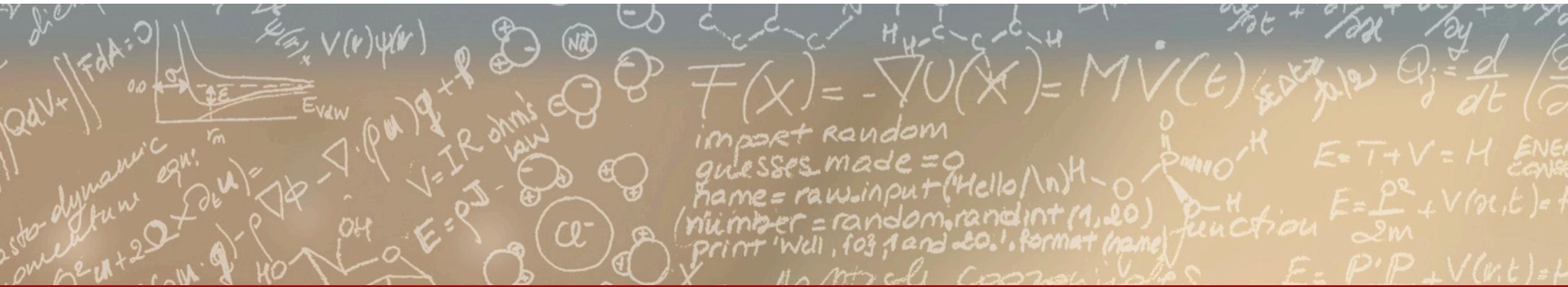
▼ Data variables:



CSCS

Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

ETH zürich



**Have fun with Jupyter notebooks, and please let us know if you find the service useful.  
We're interested in hearing about your use cases!**

robinson@cscs.ch  
help@cscs.ch