

Jupyter Notebooks

The screenshot displays the JupyterLab web interface. The browser window title is 'Notebook_Int... - JupyterLab'. The address bar shows the URL 'noto.epfl.ch/user/ch-epfl-249277/lab/tree/cs-290-responsible-software-2'. The JupyterLab interface includes a menu bar (File, Edit, View, Run, Kernel, Git, Nbgrader, Tabs, Settings, Help) and a file browser on the left. The file browser shows a directory structure: '/ cs-290-responsible-software-2025 / Notebook-0-Introduction /'. The main area displays a notebook titled 'Notebook_Introduction.ipynl'. The notebook content includes a title 'Introduction: Python and ethical dilemmas', a license notice for CC BY 4.0 International, and a 'Learning Goals' section with bullet points about Python syntax, Pandas, and Matplotlib.

■ **Interactive documents** with explanations and executable code

■ Execute in a **browser** with <http://noto.epfl.ch>

(as an alternative, VS Code can be used but we don't provide support for it)

Getting Started with Jupyter Notebooks and noto

Equipment for Using Notebooks in noto

■ Computer

- Personal laptop
- Machines in the computer rooms:
 - ◆ Default OS
 - ◆ Or Virtual Machine

■ Recent web browser

- **Firefox, Chrome / Chromium**
- Avoid Safari or Edge!

Personal Workspace on noto

The screenshot displays the JupyterLab interface on the noto platform. The browser address bar shows the URL: `noto.epfl.ch/user/ch-epfl-249277/lab/tree/cs-290-responsible-software-202`. The left sidebar contains a file browser with a red circle around the folder icon. The file browser shows the following structure:

- / cs-290-responsible-software-2025 /
- Notebook-0-Introduction /
- res (3 days ago)
- tests (3 days ago)
- Notebook_Introduction.ipynb (3 days ago)

The central notebook editor displays the following content:

Introduction: Python and ethical dilemmas

Notebook by Athina Papageorgiou Koufidou, Cécile Hardebolle and the Responsible software team (2025).

Except where otherwise noted, the content of this notebook is licensed under a [Creative Commons Attribution International License \(CC BY 4.0 International\)](#).

Introduction

Welcome to the first exercise session of Responsible Software!

The bottom status bar shows: `Mode: Command`, `Ln 1, Col 1`, `Notebook_Introduction.ipynb`, and `0`.

Workspace =
online storage
for files

Navigating Notebooks

The image shows a JupyterLab interface with a browser window at the top displaying the URL `noto.epfl.ch/user/ch-epfl-249277/lab/tree/git_Noto/responsible-soft`. The main interface is divided into three panes:

- Left Pane (Table of Contents):** A hierarchical list of notebook sections. The section `1.1. Variables and data types` is selected. A red circle highlights the hamburger menu icon (three horizontal lines) in the top-left corner of this pane.
- Center Pane (Code Editor):** Displays a code cell with the following content:

```
[1]: scenario = "A self-driving car with brake failure is heading towards five ped  
  
You can use the print function to display the content of a variable and the type  
function to display its type.  
  
ns  
cell below to see the output.  
  
ario)  
print("The type of the scenario variable is", type(scenario))  
  
A self-driving car with brake failure is heading towards five pedestrians cro  
ssing the street. The car can swerve to the other lane, hitting one pedestria  
n instead. What should the autopilot do?  
The type of the scenario variable is <class 'str'>
```
- Right Pane (Launcher):** Shows a toolbar with icons for file operations and a `git` button.

A red callout bubble with the text **Interactive outline** points to the hamburger menu icon in the left pane.

Executing Code

The image shows a JupyterLab notebook window with several annotations:

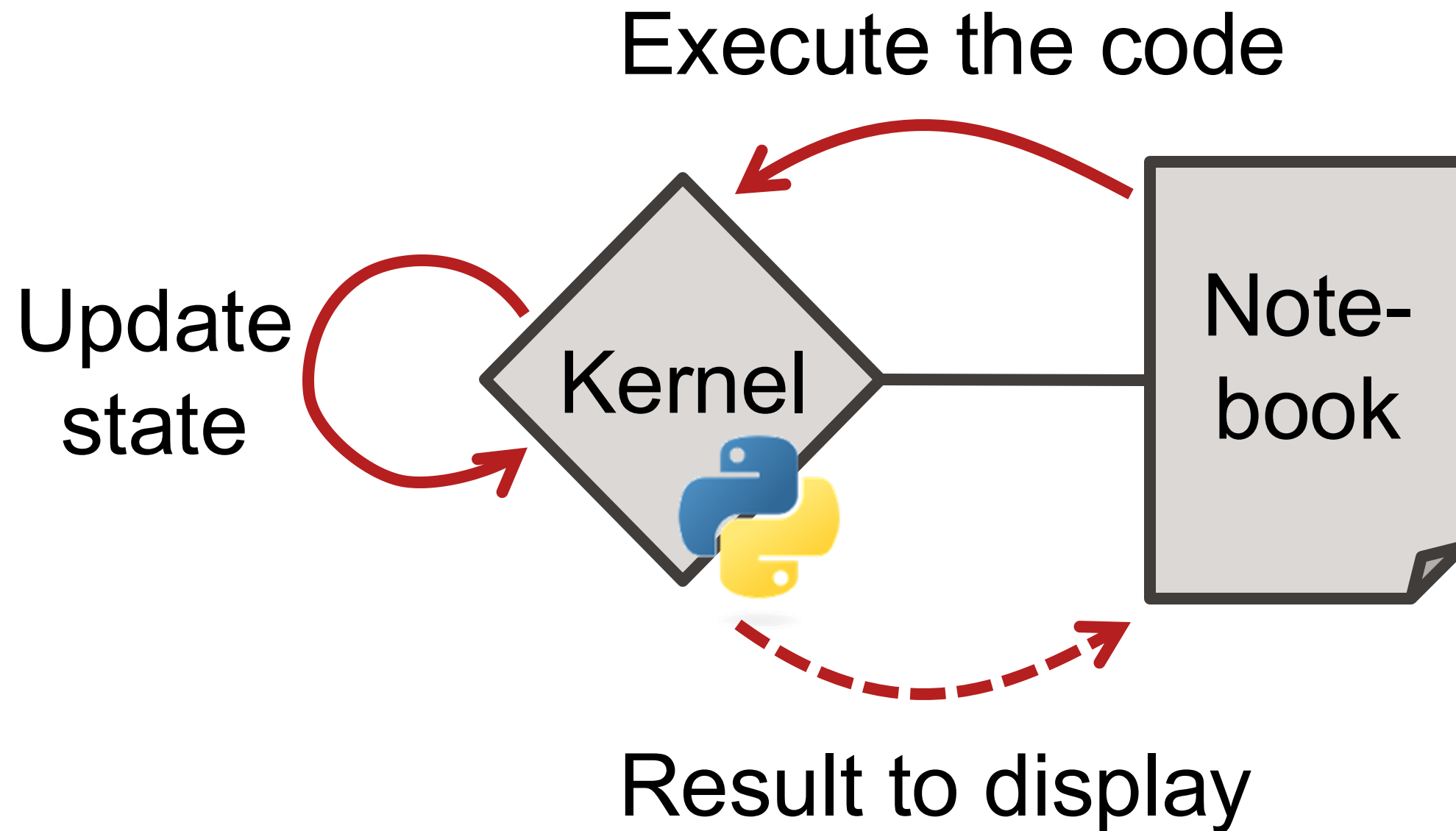
- A red speech bubble at the top center contains the text "Shift + enter". A red circle highlights the play button icon in the notebook's toolbar, with a red arrow pointing from the speech bubble to it.
- A red speech bubble on the right side contains the text "Code cell". A red arrow points from this bubble to a code cell containing the following code:

```
[1]: scenario = "A self-driving car with brake failure is heading towards five ped"
```
- A red speech bubble at the bottom left contains the text "Execution number". A blue arrow points from this bubble to the "[1]:" prefix of the code cell above.
- Below the first code cell is an "Instruction" box with the text "Execute the cell below to see the output."
- Below the instruction box is a second code cell with the following code:

```
[2]: print(scenario)
print("The type of the scenario variable is", type(scenario))
```
- The output of the second code cell is displayed below it:

```
A self-driving car with brake failure is heading towards five pedestrians cro
ssing the street. The car can swerve to the other lane, hitting one pedestria
n instead. What should the autopilot do?
The type of the scenario variable is <class 'str'>
```

How do Notebooks Work?



The kernel:

- Maintains the **execution state** behind the notebook = all the variables with their values
- Is **separate from the notebook**

Where is the Kernel?

The image shows a JupyterLab interface. On the left sidebar, the 'Kernels' panel is highlighted with a red circle and a red arrow pointing to it. A red speech bubble with the text 'List of active kernels' is positioned over the Kernels panel. The main area displays a code cell with the following content:

```
[1]: scenario = "A self-driving car with brake failure is heading towards five ped
```

The code cell also contains instructions and a code block:

Instructions

Execute the cell below to see the output.

```
print(scenario)
print("The type of the scenario variable is", type(scenario))
```

The output of the code cell is:

```
A self-driving car with brake failure is heading towards five pedestrians cro
ssing the street. The car can swerve to the other lane, hitting one pedestria
n instead. What should the autopilot do?
The type of the scenario variable is <class 'str'>
```

The bottom status bar shows 'Simple', '1', 'main', 'Python3 | Idle', 'Mode: Command', 'Ln 1, Col 1', 'Notebook_Introduction.ipynb', and '0'.

Can I see the Execution State?

Displaying variables in the notebook:

■ `print(...)`

```
[5]: scenario = "A self-driving car with brake failure is heading towards five pedest  
print(scenario)
```

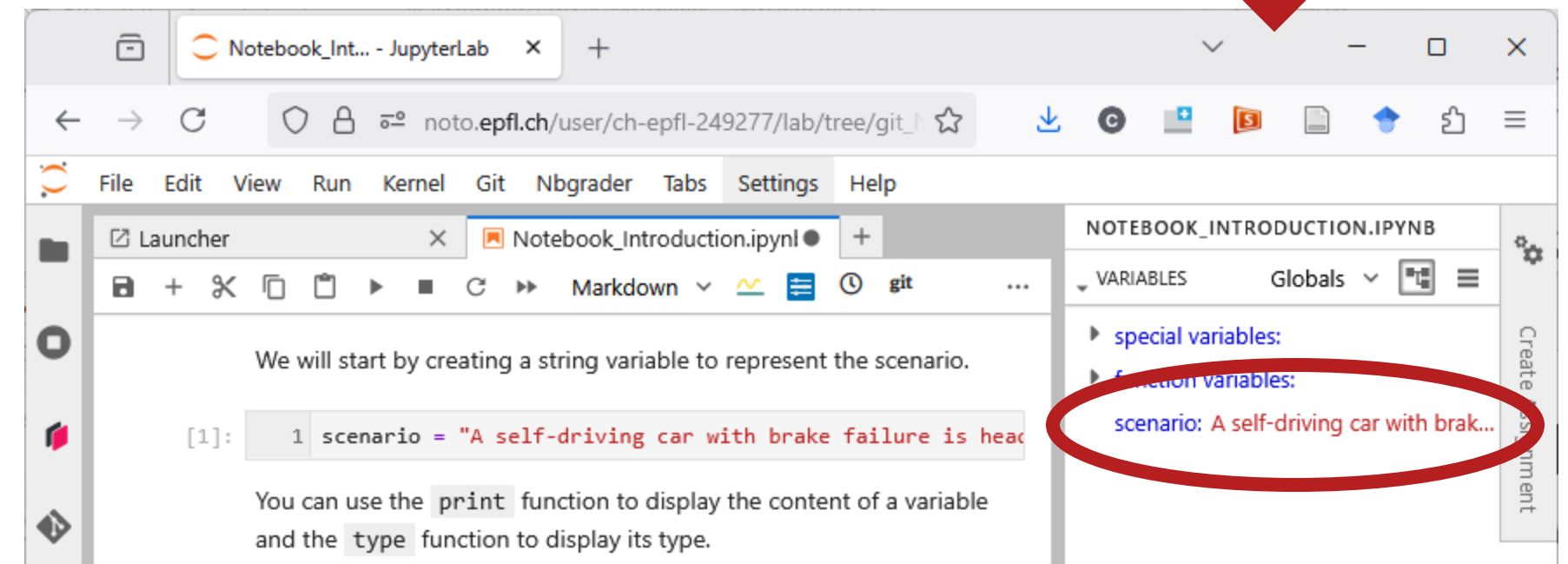
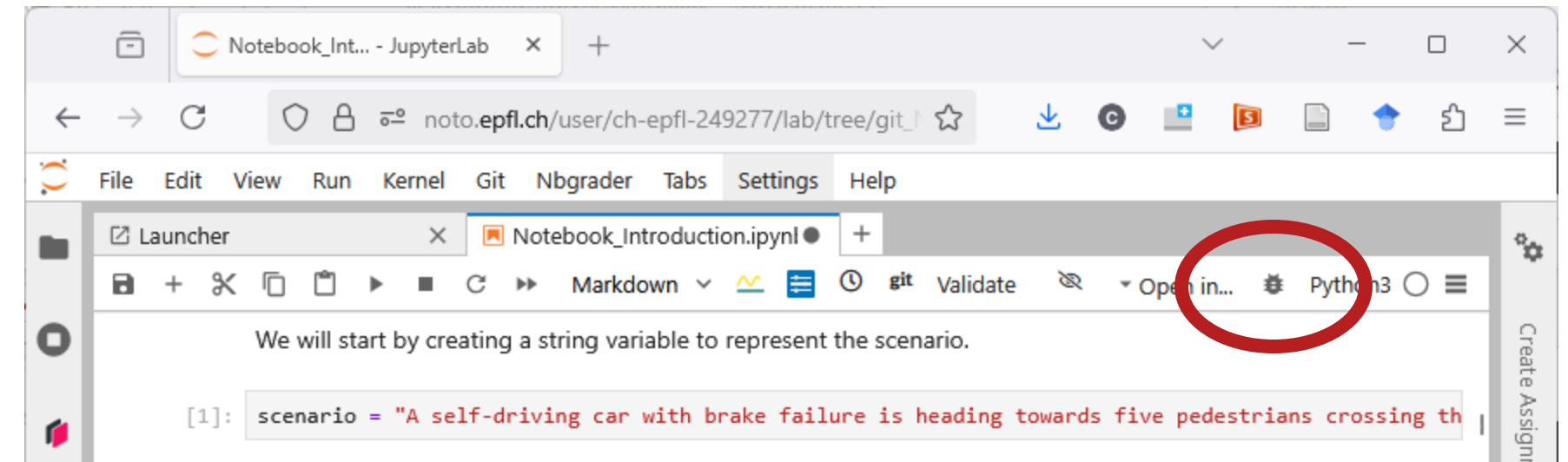
A self-driving car with brake failure is heading towards five pedestrians crossing the street. The car can swerve to the other lane, hitting one pedestrian instead. What should the autopilot do?

■ Last line of cell

```
[4]: scenario = "A self-driving car with brake failure is heading towards five pedest  
scenario
```

```
[4]: 'A self-driving car with brake failure is heading towards five pedestrians crossing the street. The car can swerve to the other lane, hitting one pedestrian instead. What should the autopilot do?'
```

Using the Python debugger:



The Order in Which You Run Cells Matters!



The screenshot shows a JupyterLab interface with a notebook titled "Notebook_Introduction.ipynl". The notebook contains two code cells. The first cell, labeled "[]:", defines a variable `scenario` with the value "A self-driving car with brake failure is heading towards five pedestrians". The second cell, labeled "[1]:", attempts to print the `scenario` variable and its type. However, a red error message is displayed below the second cell: `NameError: name 'scenario' is not defined`. A red arrow points from the error message back to the first cell, indicating that the variable was not yet defined when the second cell was executed. The notebook's file browser on the left shows the current directory structure, including folders `res` and `tests`, and the notebook file `Notebook_Introduction.ipynb`.

```
[ ]: scenario = "A self-driving car with brake failure is heading towards five pedestrians"

You can use the print function to display the content of a variable and the type function to display its type.

Instructions

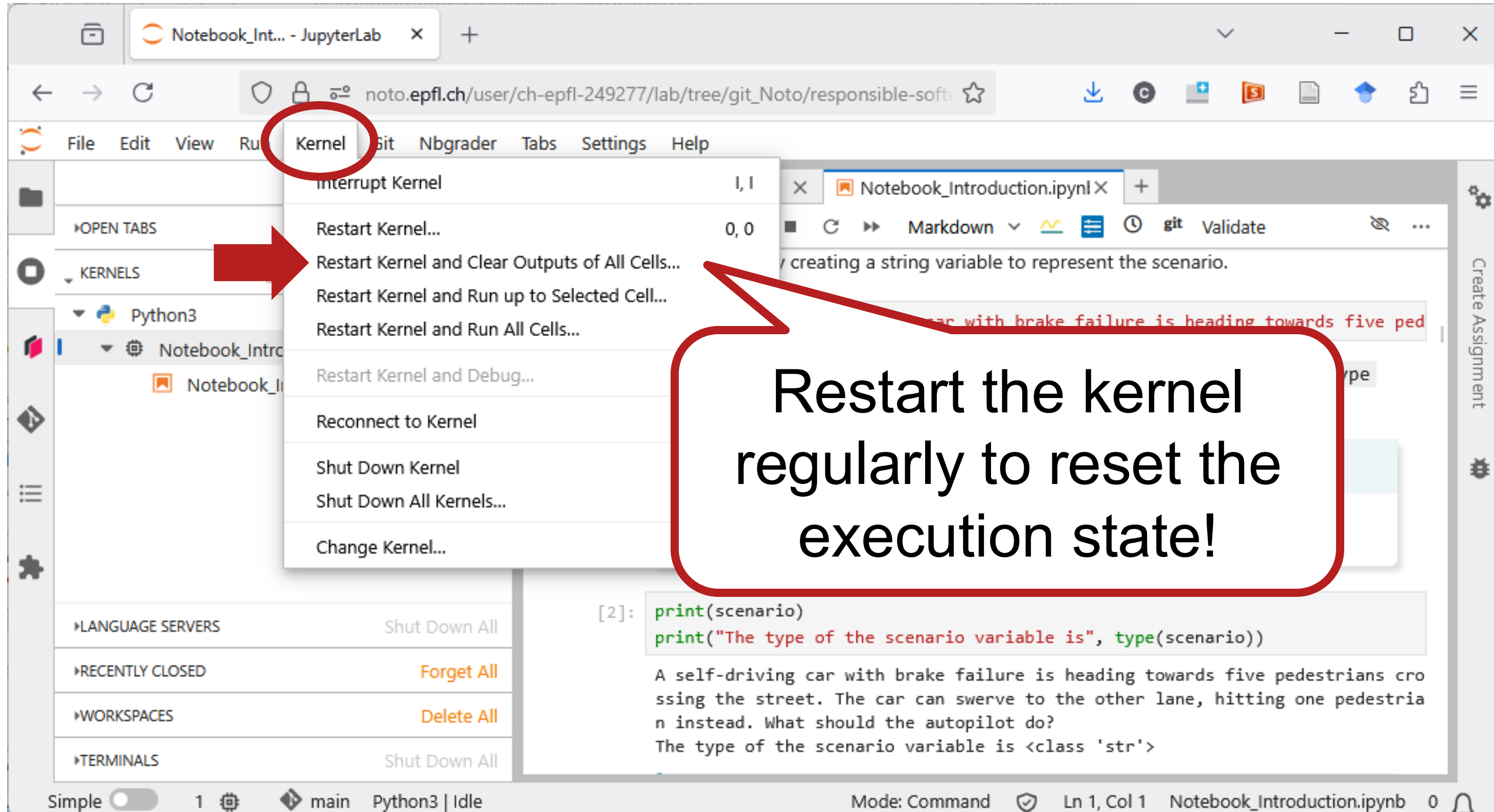
Execute the cell below to see the output.

[1]: print(scenario)
print("The type of the scenario variable is", type(scenario))

-----
NameError                                Traceback (most recent call last)
Cell In[1], line 1
----> 1 print(scenario)
      2 print("The type of the scenario variable is", type(scenario))

NameError: name 'scenario' is not defined
```

Actions on the Kernel



The screenshot displays the JupyterLab interface. The 'Kernel' menu is open, showing various options. A red circle highlights the 'Kernel' menu item, and a red arrow points to the 'Restart Kernel and Clear Outputs of All Cells...' option. A red callout box contains the text: 'Restart the kernel regularly to reset the execution state!'. The background shows a notebook with Python code and its output.

Restart the kernel regularly to reset the execution state!

```
[2]: print(scenario)
print("The type of the scenario variable is", type(scenario))
```

A self-driving car with brake failure is heading towards five pedestrians crossing the street. The car can swerve to the other lane, hitting one pedestrian instead. What should the autopilot do?
The type of the scenario variable is <class 'str'>

Saving Work

The image shows a JupyterLab notebook interface. The browser address bar shows the URL `noto.epfl.ch/us`. The notebook title is `Notebook_Introduction.ipynl`. The left sidebar shows the file explorer with the following structure:

- / cs-290-responsible-software-2025 /
- Notebook-0-Introduction /
- res (3 days ago)
- tests (3 days ago)
- Notebook_Introduction.ipynb (next yr.)

The main notebook area contains the following text and code:

We will start by creating a string variable to represent the scenario.

```
[2]: scenario = "A self-driving car with brake failure is heading towards five ped
```

You can use the `print` function to display the content of a variable and the `type` function to display its type.

Instructions

Execute the cell below to see the output.

```
[3]: print(scenario)
print("The type of the scenario variable is", type(scenario))
```

A self-driving car with brake failure is heading towards five pedestrians crossing the street. The car can swerve to the other lane, hitting one pedestrian instead. What should the autopilot do?
The type of the scenario variable is `<class 'str'>`

At the bottom of the notebook, the status bar shows: `Simple` (toggle), `2` (tabs), `main` (environment), `Python3 | Idle` (kernel), `Mode: Command`, `Ln 1, Col 1`, `Notebook_Introduction.ipynb`, and `0` (notifications).

A red callout box with the text `Ctrl + S` points to the save icon (a floppy disk) in the notebook toolbar. The save icon is also circled in red.

Opening Multiple Notebooks

The screenshot shows the JupyterLab interface. At the top, the browser's tab bar is visible, with two tabs: "Notebook_Int... (2) - JupyterLab" and "Sample_Py3.i... - auto-5". A red oval highlights these tabs. A red speech bubble with a yellow warning triangle icon points to the tabs, containing the text: "Avoid opening noto in multiple browser tabs!".

The JupyterLab interface below shows a file browser on the left with a list of files and folders. The main area displays the content of a notebook titled "Introduction: Python and ethical dilemmas". The notebook content includes:

- A title: "Introduction: Python and ethical dilemmas"
- Text: "Notebook by Athina Papageorgiou Koufidou, Cécile Hardebolle and the Responsible software team (2025)."
- Text: "Except where otherwise noted, the content of this notebook is licensed under a [Creative Commons Attribution International License \(CC BY 4.0 International\)](#)."
- A Creative Commons Attribution (CC BY) license logo.
- A section titled "Introduction" with the text: "Welcome to the first exercise session of Responsible Software!"
- Text: "This tutorial notebook will serve as an introduction to Python basics, as well as some"

The bottom status bar shows "Simple" mode, "Python3 | Idle", and "Mode: Command Ln 1, Col 1 Notebook_Introduction.ipynb 0".

Opening Multiple Notebooks

The screenshot shows the JupyterLab interface with three notebooks open in tabs: Notebook_Introduction.ipyn, 30_Using_git.ipynb, and Sample_Py3.ipynb. A red circle highlights the tabs area. A red callout box points to the tabs with the text "Instead, open the notebooks inside noto".

The main notebook displayed is "Python3 Notebook" with the following content:

First example

This first example will plot a simple

$$e^{-x} = \sum_{i=0}^{\infty} \frac{(-1)^i x^i}{i!}$$

Well... not really. This was to demo $\text{L}^{\text{A}}\text{T}^{\text{E}}\text{X}$ equations, courtesy of `MathJax`.

The equation we'll plot is the following:

$$y = \sin(x) + \cos(x)$$

```
[ ]: import numpy
      from matplotlib import pyplot
```

At the bottom, the status bar shows "Mode: Command", "Ln 1, Col 1", and "Sample_Py3.ipynb 0".

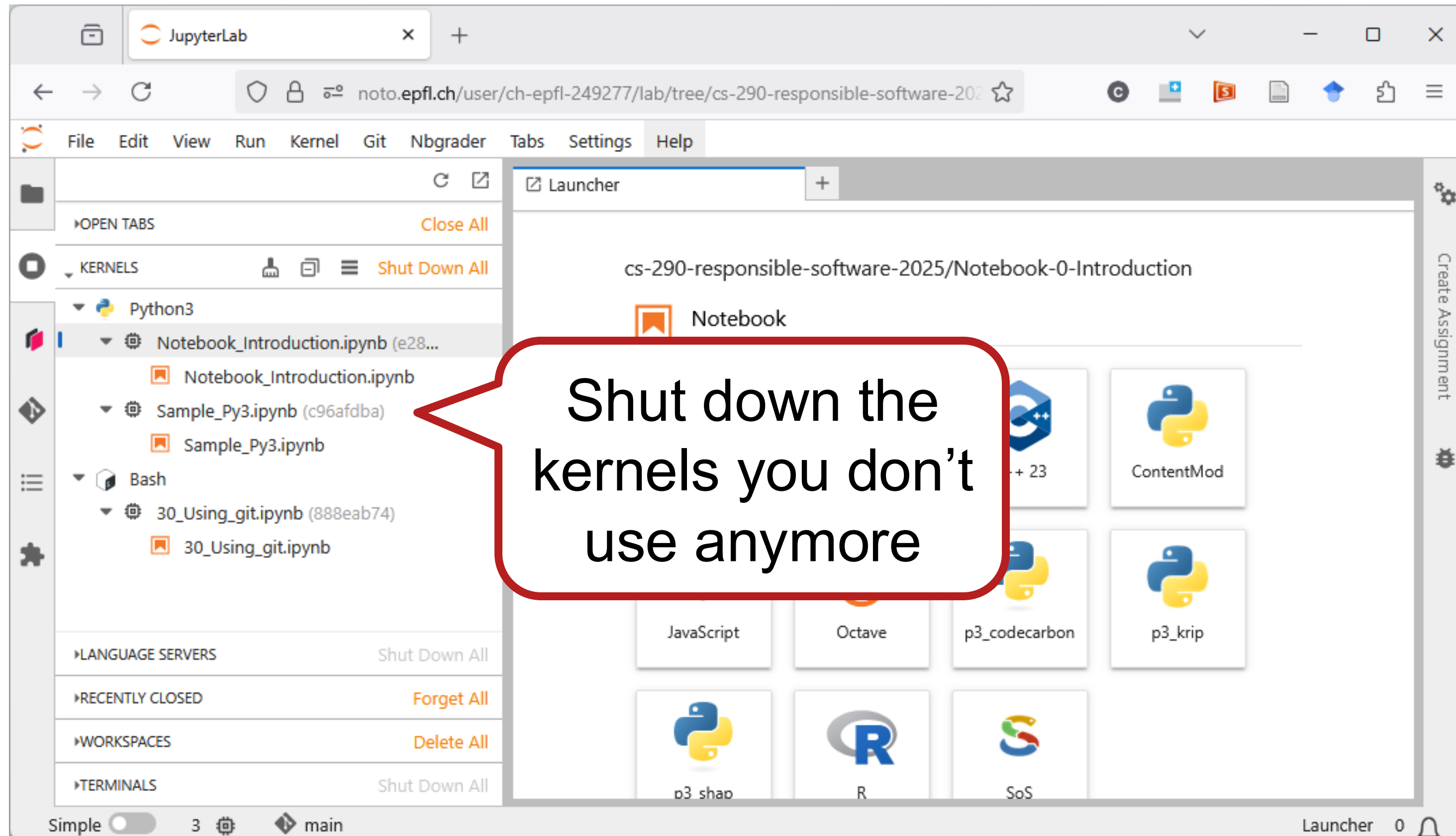
Opening Multiple Notebooks

The screenshot shows the JupyterLab web interface. At the top, there's a browser window with the URL `noto.epfl.ch/user/ch-epfl-249277/lab/tree/Documentation/Examples/Samp`. Below the browser is the JupyterLab menu bar with options: File, Edit, View, Run, Kernel, Git, Nbgrader, Tabs, Settings, Help. The main workspace is split into two panels. The left panel shows a notebook titled 'Notebook_Introduction.ipynl' with the text 'Introduction: Python and al dilemmas'. The right panel shows a notebook titled 'Sample_Py3.ipynb' with the text 'Python3 Notebook' and 'First example'. The status bar at the bottom indicates 'Simple' mode, '3' tabs, 'main' environment, 'Python3 | Idle', 'Mode: Command', 'Ln 1, Col 1', 'Sample_Py3.ipynb', and '0' notifications.

You can collapse the side bar

You can arrange panels side by side! (or any other config you want)

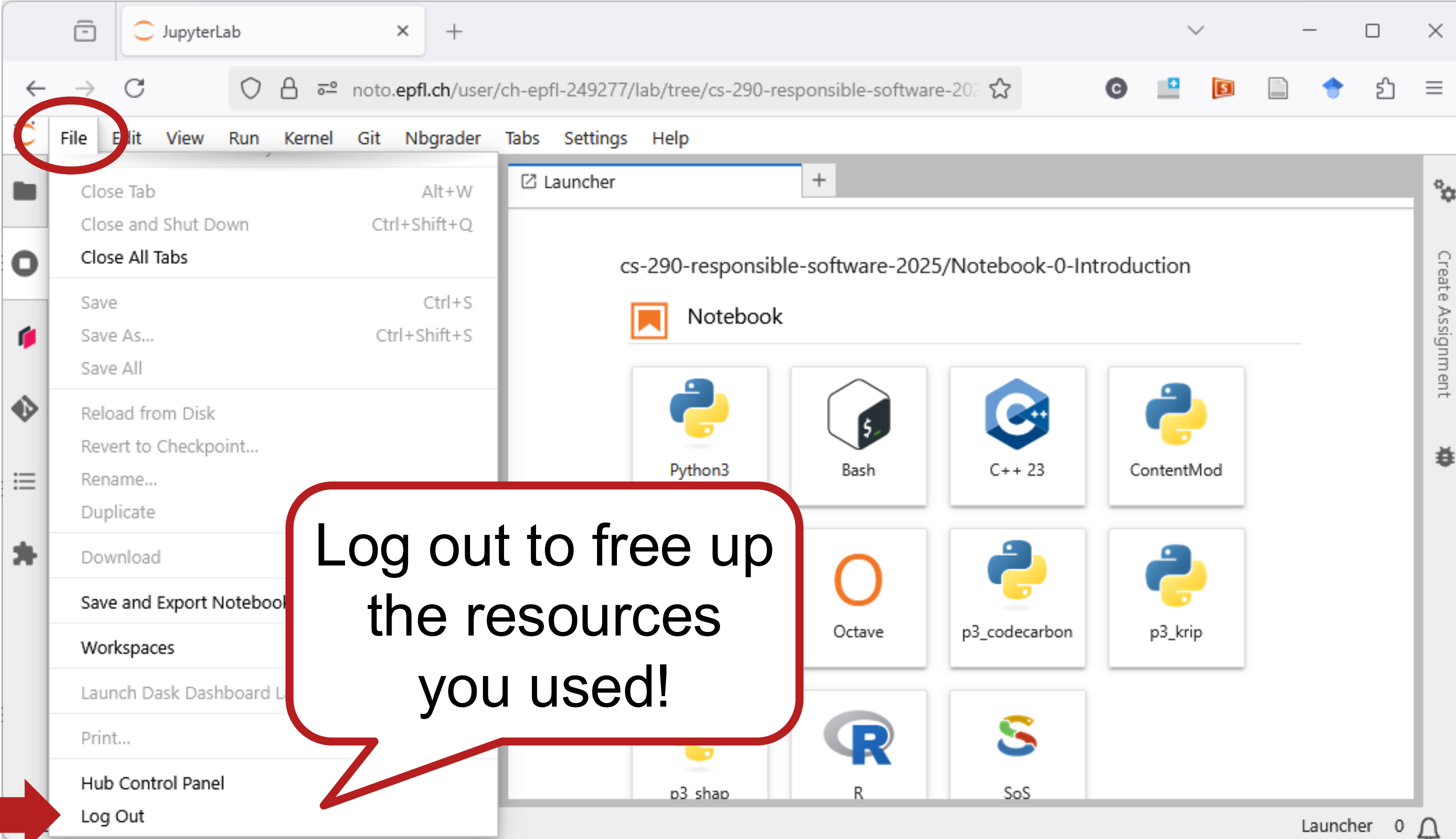
Save Resources!



The image shows a JupyterLab web interface. The browser address bar displays the URL: `noto.epfl.ch/user/ch-epfl-249277/lab/tree/cs-290-responsible-software-2025`. The interface includes a top menu bar with options like File, Edit, View, Run, Kernel, Git, Nbgrader, Tabs, Settings, and Help. On the left, there is a sidebar with sections for OPEN TABS, KERNELS, LANGUAGE SERVERS, RECENTLY CLOSED, WORKSPACES, and TERMINALS. The KERNELS section is expanded, showing several active kernels under Python3 and Bash. A red callout box with a white background and black text is overlaid on the interface, pointing to the kernel list. The callout text reads: "Shut down the kernels you don't use anymore". The main workspace area shows a notebook titled "cs-290-responsible-software-2025/Notebook-0-Introduction". Below the notebook title, there is a grid of kernel selection options, including JavaScript, Octave, p3_codecarbon, ContentMod, p3_krip, p3_shap, R, and SoS. The bottom status bar shows "Simple" mode, 3 kernels, and the "main" workspace.

Shut down the kernels you don't use anymore

Save Resources!



The image shows a JupyterLab web interface. The browser address bar displays the URL `noto.epfl.ch/user/ch-epfl-249277/lab/tree/cs-290-responsible-software-2025`. The 'File' menu is open, showing options such as 'Close Tab', 'Save', and 'Log Out'. A red circle highlights the 'File' menu, and a red arrow points to the 'Log Out' option at the bottom of the menu. A red speech bubble with the text 'Log out to free up the resources you used!' is overlaid on the interface. The main content area shows a 'Launcher' view with various environment icons like Python3, Bash, C++ 23, ContentMod, Octave, p3_codecarbon, p3_krip, p3_shap, R, and SoS.

File Edit View Run Kernel Git Nbgrader Tabs Settings Help

- Close Tab Alt+W
- Close and Shut Down Ctrl+Shift+Q
- Close All Tabs
- Save Ctrl+S
- Save As... Ctrl+Shift+S
- Save All
- Reload from Disk
- Revert to Checkpoint...
- Rename...
- Duplicate
- Download
- Save and Export Notebook
- Workspaces
- Launch Dask Dashboard L
- Print...
- Hub Control Panel
- Log Out

Launcher

cs-290-responsible-software-2025/Notebook-0-Introduction

Notebook

Python3 Bash C++ 23 ContentMod

Octave p3_codecarbon p3_krip

p3_shap R SoS

Launcher 0

Log out to free up the resources you used!

Configuration for a Specific Course

Libraries

■ Centrally installed:

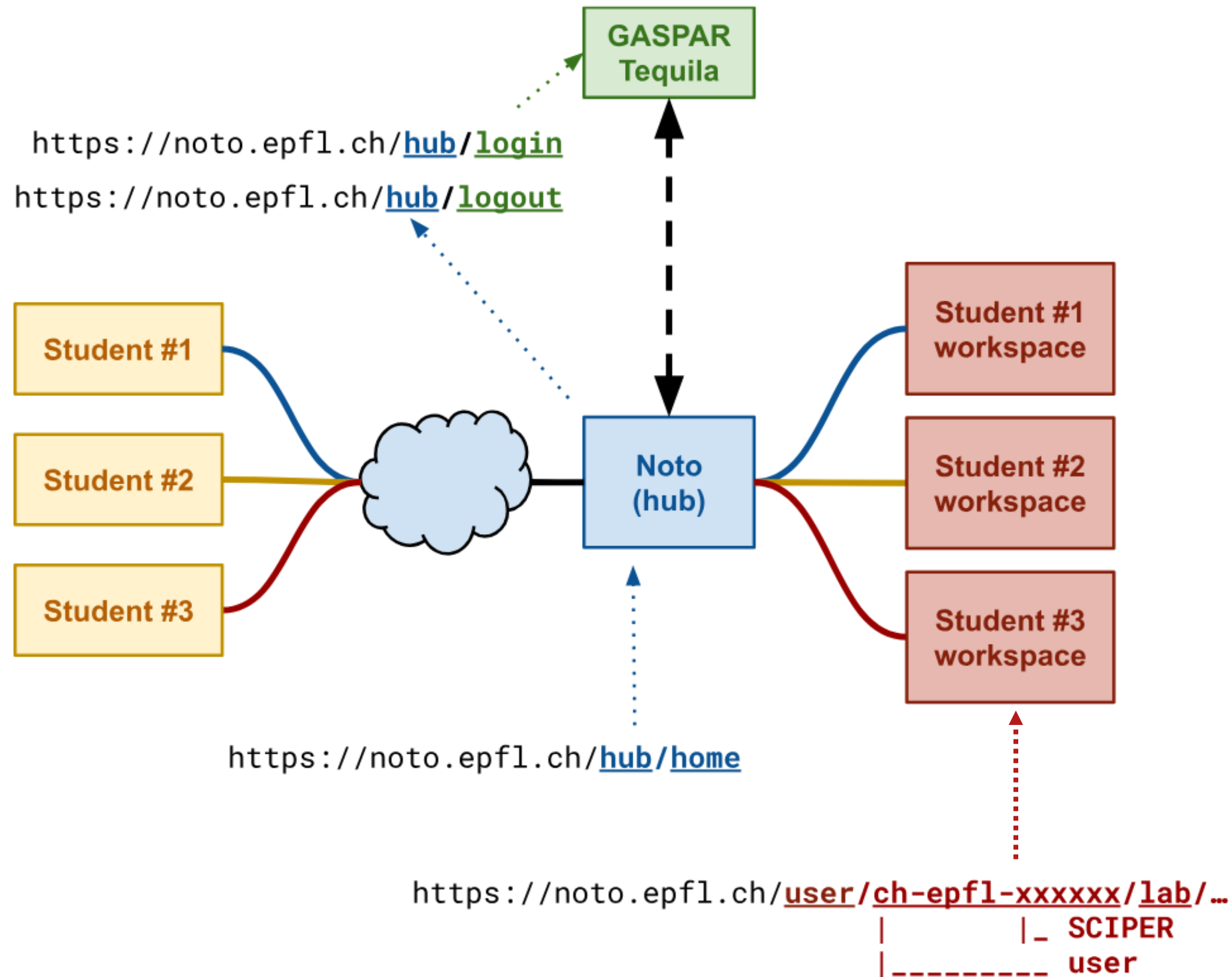
- We install the libraries; students don't need to worry about it!
 - 👉 Contact us with the list of libraries you need 2 weeks before the start of semester

■ Possibility for each user to install libraries in virtual environments

- ⚠ Risk of breaking the main Jupyter environment of the user
 - 👉 Imperative to use **virtual environments**

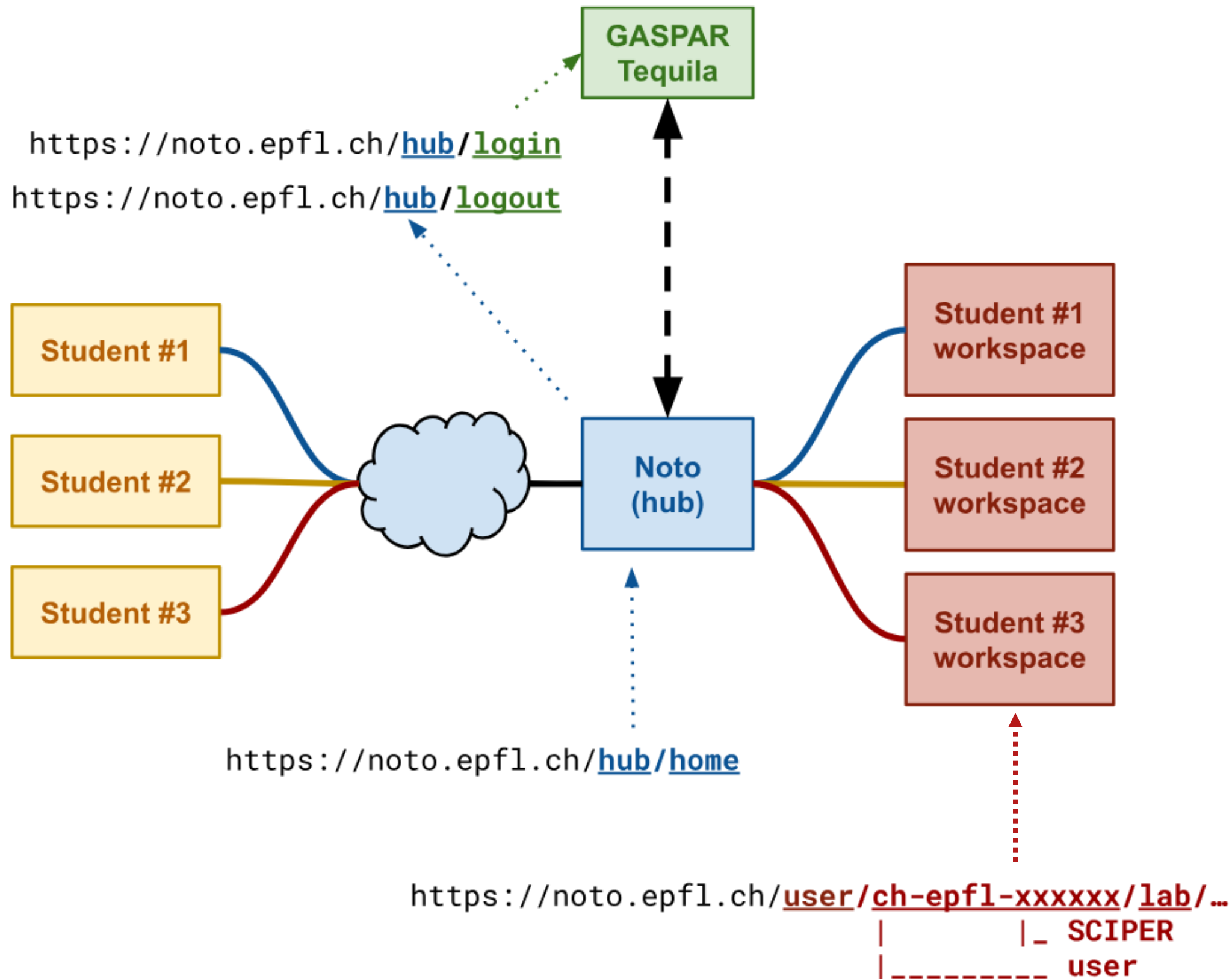
Troubleshooting

Architecture of noto



- **Client** (web browser): renders HTML/JavaScript
- **Hub**: manages users, authentication, and spawns individual Jupyter server instances for each user
- **Personal server**: Jupyter server that runs the notebooks and kernels, individual to each user but resources are shared (run on virtual machine)

Architecture of noto



⚠ session in the web client
≠ session on the hub
≠ session on the personal server

👉 Closing the web session doesn't close the session on noto (hub + personal server)

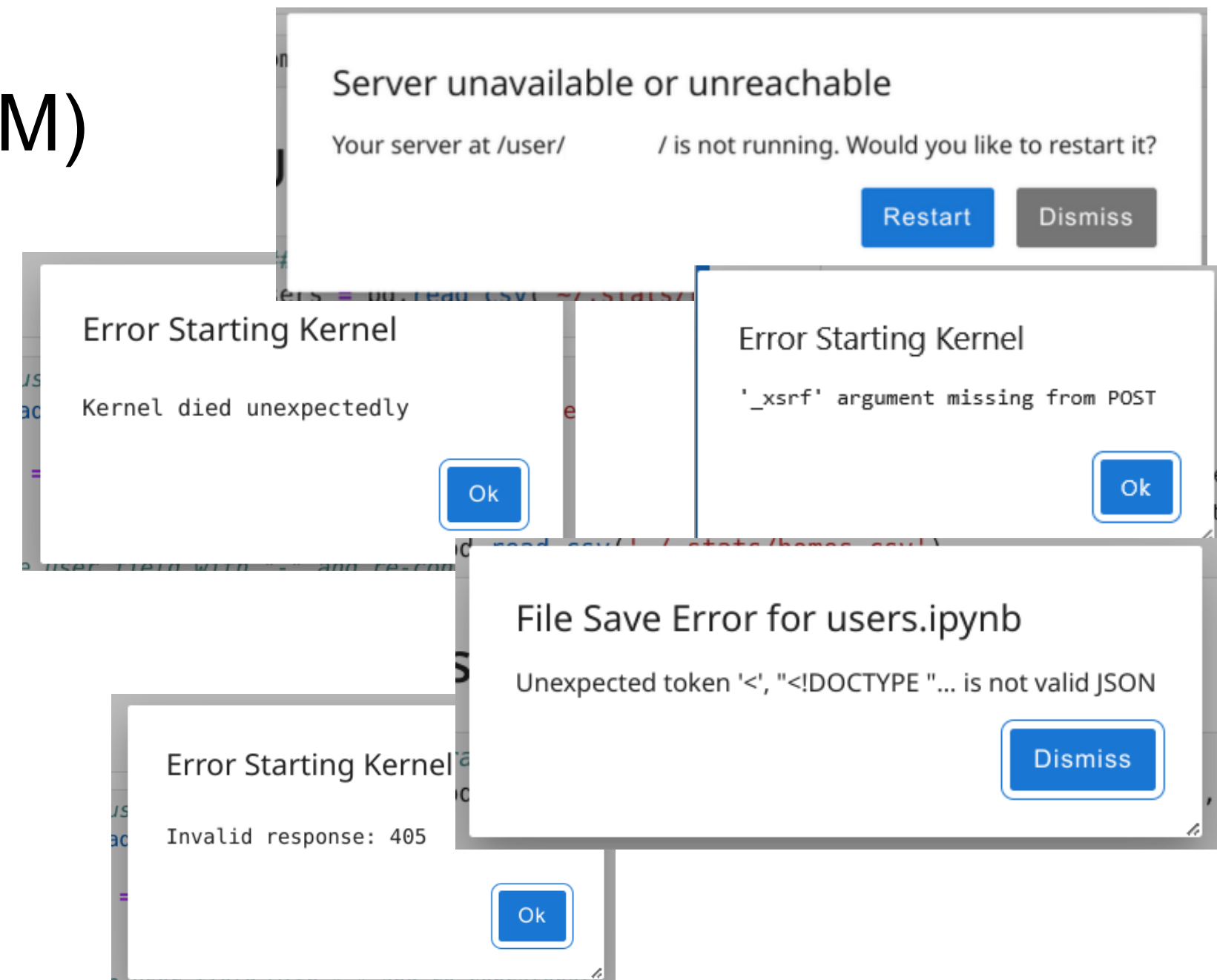
👉 The hub can work without any session on the personal server

Loss of Personal Server/Kernel

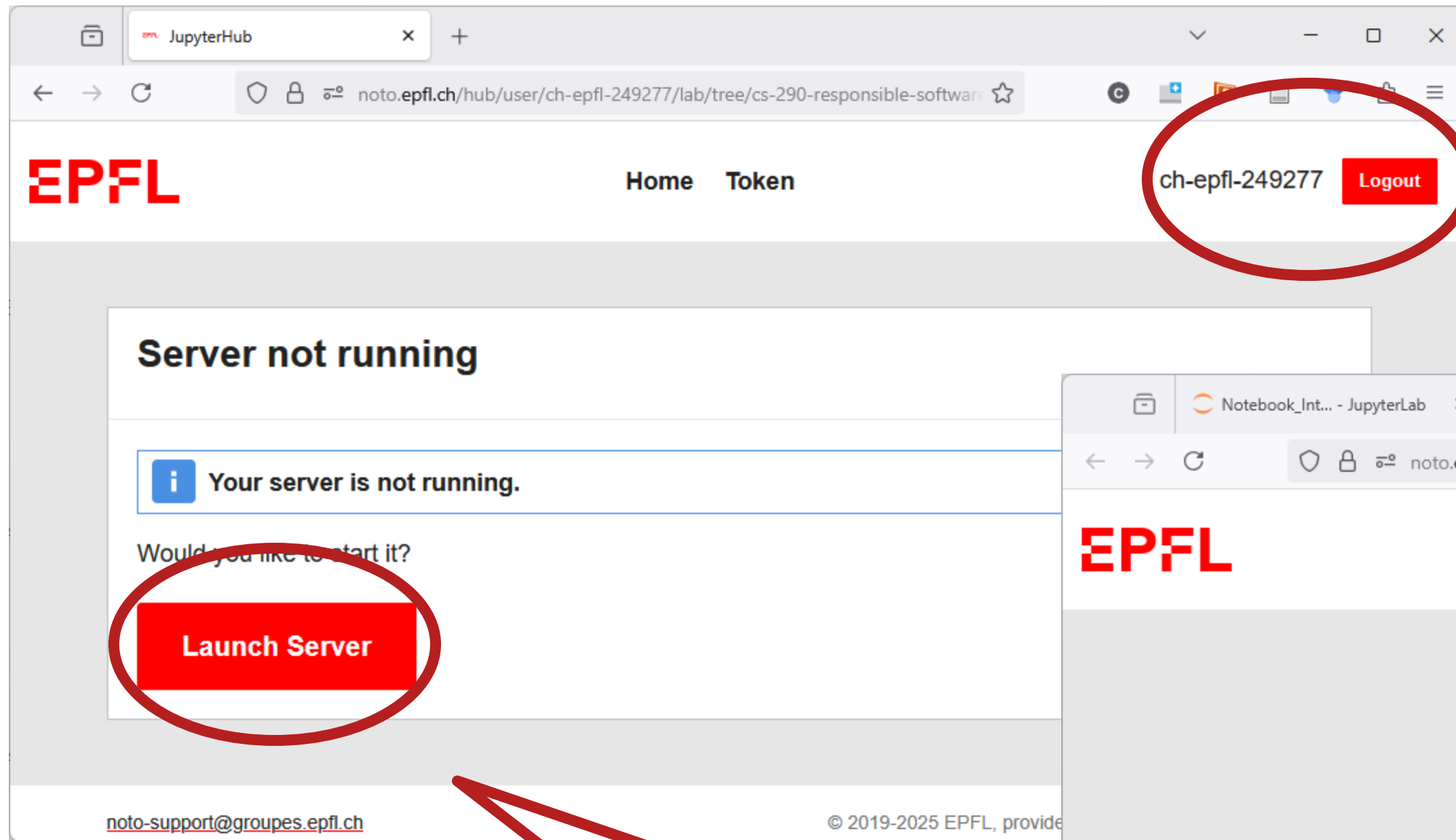
- **Popups with error messages usually mean that**
 - ⚠ **the personal server is no longer available** ⚠
 - Timeout after 20 minutes of inactivity
 - Too many resources used (> 2 GB RAM)
- ⚠ The session on the hub may still be active so the interface looks like it works but it does NOT!

- **Log out and then log in again:**

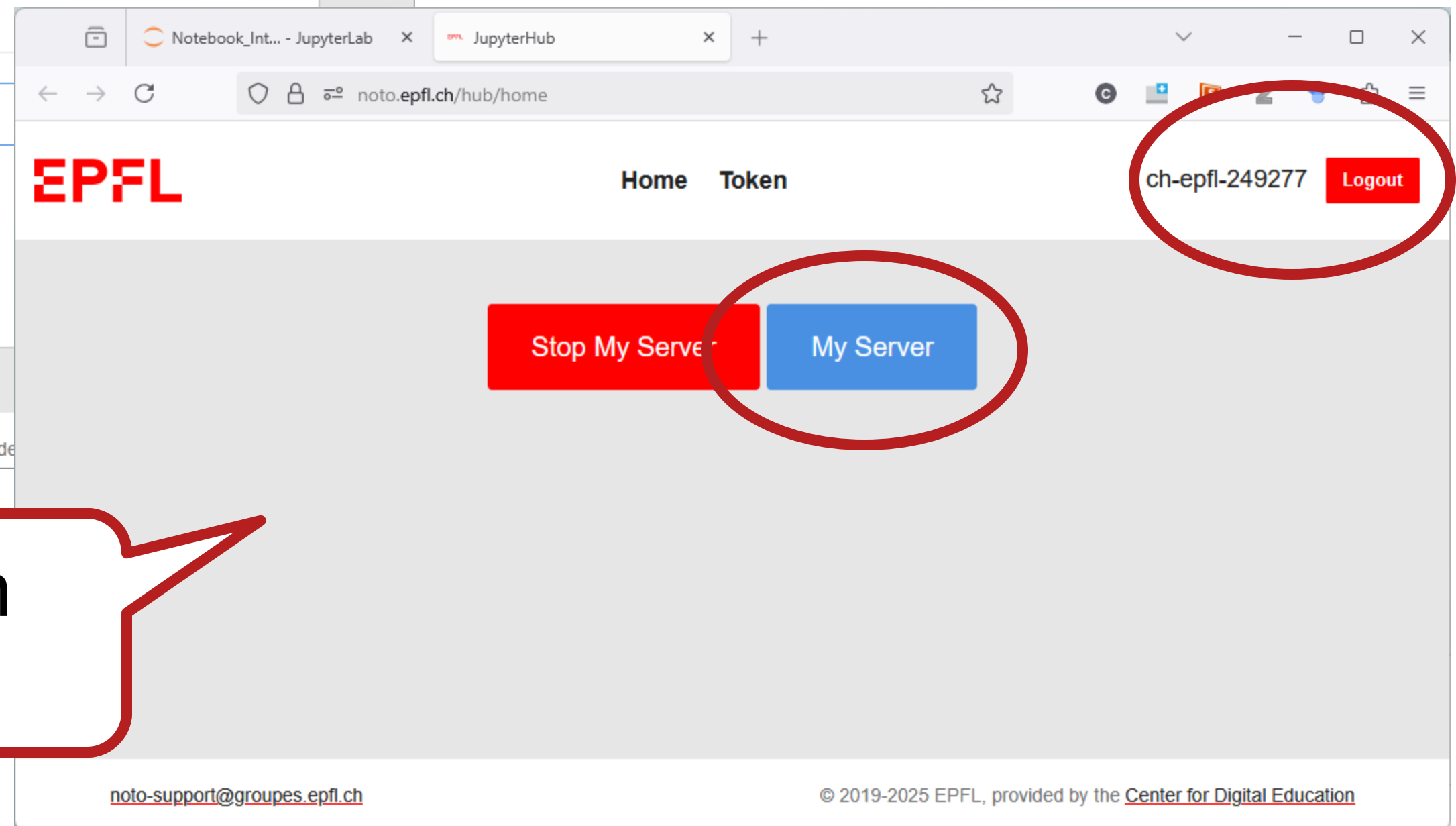
- File > Logout
- <https://noto.epfl.ch/hub/logout>



Server Not Running



In case of issue, log out then log in again



Try to launch the server

Other Typical Issues

- Spawn failed / Error 503
 - Issue with load on noto or issue with the personal server itself
 - Troubleshoot: logout, wait for 30 sec, login
- White page, interface taking time to load
 - Issue with load on noto
 - Troubleshoot: wait for 1 or 2 mins, if it doesn't load logout/login
- Cells not running
 - Issue with the kernel
 - Troubleshoot: restart the kernel, check the resources used (top)
- Cells running slowly
 - Too many kernels running, too many resources used
 - Troubleshoot: restart the kernel, check the resources used (top)

Technical Support

- First level: teaching assistants
 - Check for typical programming issues (infinite loops, etc.)
 - Check the resources used (RAM, CPU, GPU)
 - Apply troubleshooting steps: restart kernel, logout/login, etc.
- Second level: noto-support@groupes.epfl.ch
 - Include:
 - ◆ Screen captures
 - ◆ Error messages
 - ◆ SCIPER