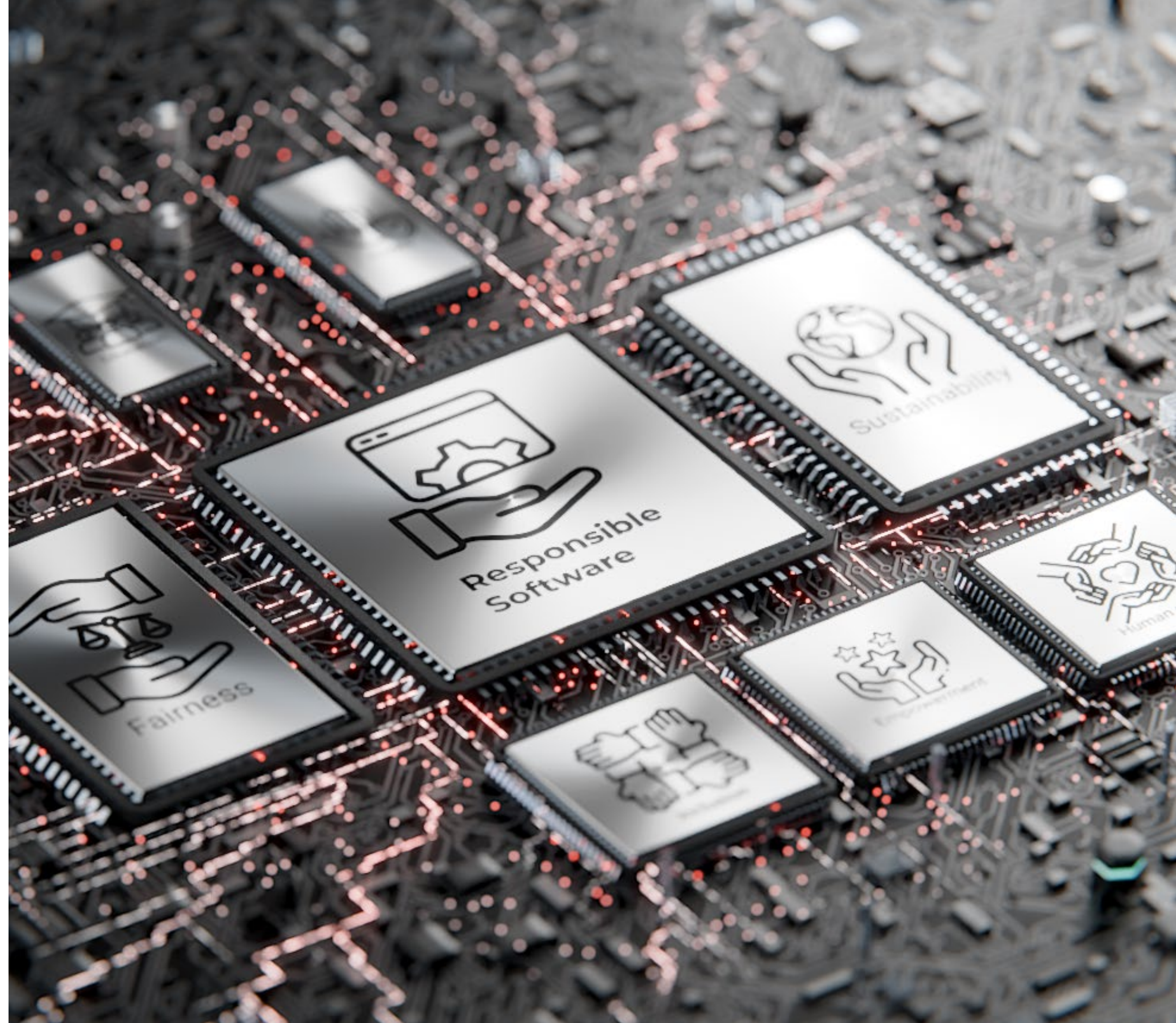


EPFL

**Getting
Started
8 sept.**

Cécile Hardebolle

**Responsible
Software**



Agenda for today

1. Course format and logistics: important information!
2. Learning with videos
3. Working with Jupyter Notebooks
4. Getting started with case studies

Your instructor

Education:

- Engineer spe. Electrical Engineering and Computer Science
- PhD in Computer Science

Experience:

- Assistant Professor in Computer Science (6 years, Supélec, FR)
 - + Teaching at Polytechnique Paris and Centrale Paris
- Pedagogical Advisor and Learning Scientist (11 years, EPFL)

Passionate about **computer science, ethics and learning!**

The Responsible Software Team

Have prepared all the material
+ will assist for the course:

- Noa Trojman
- Eugène Bergeron

- *Mattéo Berthet*
- *Florian Dufour*
- *Rose Ndanga Nya*
- *Athina Papageorgiou Koufidou*
- *Maxime Lelièvre*

Doctoral assistants:

- Yingxuan You
- Ayush Kumar Tarun

Student assistants:

- Camille Lannoye
- Mamoun Imghi
- Florian Kolly
- Elyes Trabelsi
- Elise Hueber
- Othmane Housni

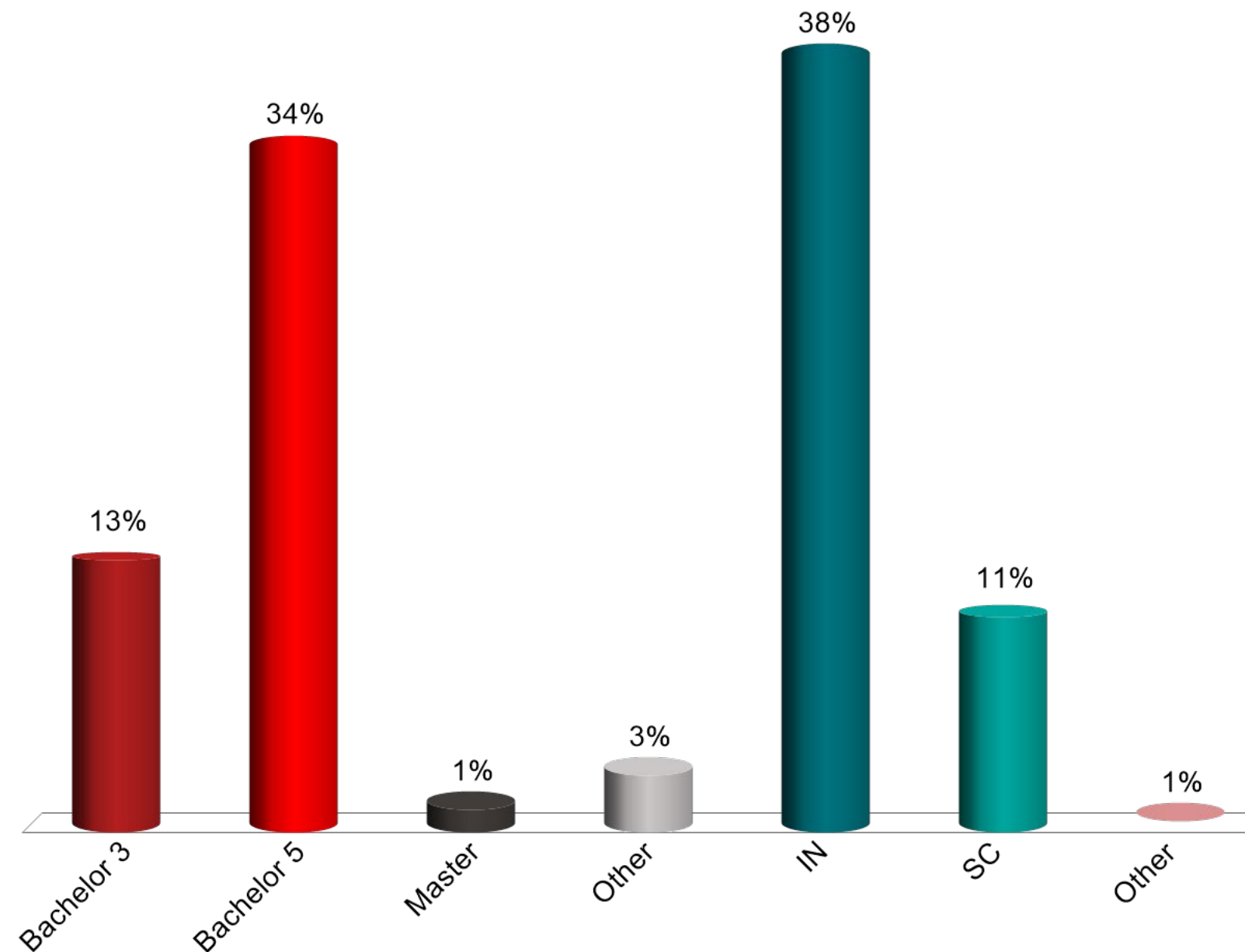
You are in:

URL: ttpoll.eu
Session ID: cs290

Select your **level** and your **section** (click on 2 options)

- a. Bachelor 3
- b. Bachelor 5
- c. Master
- d. Other

- e. IN
- f. SC
- g. Other



Course format **& logistics**

Your prior experience with flipped classes:

21% a. I have attended several flipped classes

24% b. I have attended one flipped class

48% c. I have heard about it but never attended one

7% d. Never heard about it

URL: ttpoll.eu

Session ID: cs290

A specific pedagogical approach

- You study the **theory** at home
- The **practice** takes place in class
- 👉 Each week:
 - Starts with *practice* = programming exercises (with TAs)
 - Ends with *practice* = case studies (with me)

A typical "week"

Tuesday 10h15-12h: programming exercise

- ◆ Solution provided in the evening

By the next Monday:

- ◆ **Watch videos of theory & strategies + quizzes**
- ◆ Finish case studies from the previous week
- ◆ Finish programming exercises

Monday 15h15-17h: case studies

- ◆ Interactive polling session
- ◆ Group activities & questions
- ◆ Solution provided in the evening

Average of
**5 videos /
week**

Experience



Knowledge



Habits



A typical "week"

Date	Week	Lecture (Monday 15h15-17h) in STCC Cloud C	Exercise session (Tuesday 10h15-12h)	Independent study (due before the following Monday)
08/09	1	Getting started	Introduction notebook	Introduction videos and quizzes
15/09	2	Introduction cases (in CO3)	Safety 1 notebook	Safety 1 videos and quizzes
22/09	3	public holiday	Safety 2 notebook	Safety 2 videos and quizzes
29/09	4	Safety 2 cases	Fairness 1 notebook	Fairness 1 videos and quizzes
06/10	5	Fairness 1 cases (in CO3)	Fairness 2 notebook	Fairness 2 videos and quizzes
13/10	6	Fairness 2 cases	Graded notebook 1	-
20/10			Autumn break	
27/10	7	Graded 1 debriefing	Mock test	-
03/11	8	Mock test debriefing (in CO3)	Sustainability 1 notebook	Sustainability 1 videos and quizzes
10/11	9	Sustainability 1 cases	Sustainability 2 notebook	Sustainability 2 videos and quizzes
17/11	10	Sustainability 2 cases	Empowerment 1 notebook	Empowerment 1 videos and quizzes
24/11	11	Empowerment 1 cases	Graded notebook 2	-
01/12	12	Graded 2 debriefing	Empowerment 2 notebook	Empowerment 2 videos and quizzes
08/12	13	Empowerment 2 cases	Graded case	Conclusion videos and quizzes
15/12	14	Conclusion cases	Conclusion review	-
Revisions				
TBD	Exams	Written exam		

Overall schedule

Date	Week	Lecture (Monday 15h15-17h) in STCC Cloud C	Exercise session (Tuesday 10h15-12h)	Independent study (due before the following Monday)
08/09	1	Getting started	Introduction	Introduction videos and quizzes
15/09	2	Introduction cases (in CO3)	Safety 1 notebook	Safety 1 videos and quizzes
22/09	3	public holiday	Safety 2 notebook	Safety 2 videos and quizzes
29/09	4	Safety 2 cases	Fairness 1 notebook	Fairness 1 videos and quizzes
06/10	5	Fairness 1 cases (in CO3)	Fairness 2 notebook	Fairness 2 videos and quizzes
13/10	6	Fairness 2 cases	Graded notebook 1	-
20/10			Autumn break	
27/10	7	Graded 1 debriefing	Mock test	-
03/11	8	Mock test debriefing (in CO3)	Sustainability 1 notebook	Sustainability 1 videos and quizzes
10/11	9	Sustainability 1 cases	Sustainability 2 notebook	Sustainability 2 videos and quizzes
17/11	10	Sustainability 2 cases	Empowerment 1 notebook	Empowerment 1 videos and quizzes
24/11	11	Empowerment 1 cases	Graded notebook 2	
01/12	12	Graded 2 debriefing	Empowerment 2 notebook	
08/12	13	Empowerment 2 cases	Graded case	
15/12	14	Conclusion cases	Conclusion review	
Revisions				
TBD	Exams	Written exam		

Graded assignments
during the semester

Final assessment
in exam session

Evaluation: during the semester

2 Graded notebooks: 8% each (16% total)

- At “home”
- Programming + open questions
- Individual – no group work allowed!
- Open book – no GenAI tool allowed!

Submission dates

- 14 October
- 25 November

1 Graded case study: 4%

- At “home”
- Open questions
- In small groups
- Open book – no GenAI tool allowed!

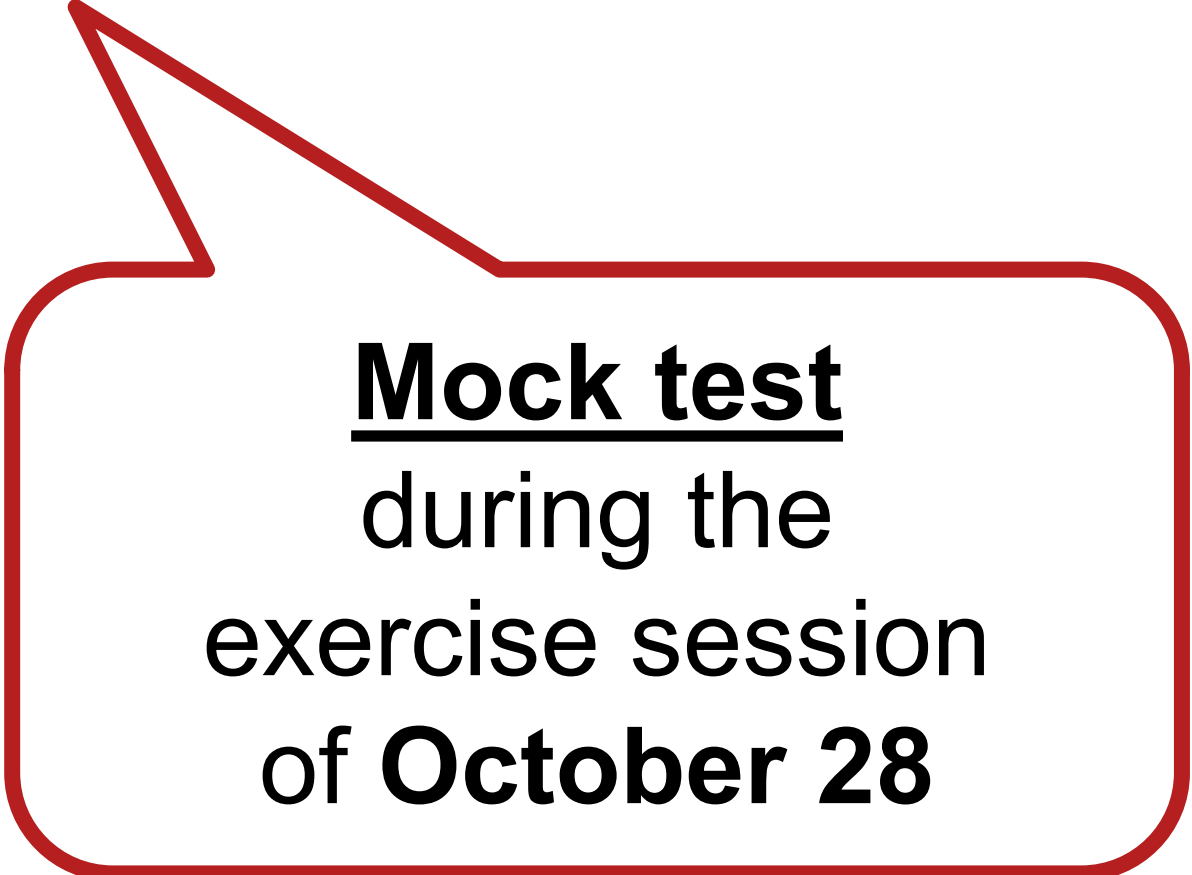
Submission date

- 9 December

Evaluation: final exam

Final exam: 80%

- In the **Winter exam session** (date TBD)
- Supervised in lecture hall, individual
- Pen and paper (QCM + case studies)
- Closed book, one A4 paper sheet allowed



Mock test
during the
exercise session
of **October 28**

Changes compared to last year

- 5 ECTS credits (was 4 previously)
[without any increase of the content or workload!]
- Exam 80% + assignments 20% (was 60% + 40%)
- Assignments at “home” (were done in class)
- Exam in the Winter exam session (was end of the semester)

Double Deck - Difficulties with rooms

As you know, there is intense work on campus generating a lot of noise 🙌 huge reduction in the number of usable rooms!

■ Interactive lecture:

- Mainly **STCC Cloud C**
- ⚠ In **CO3** on 15/09, 06/10 and 03/11

Rules for the STCC Cloud C auditorium

- Exhibitions happen in the same building (which is privately owned)
👉 Only use **entrances and facilities indicated “Cours EPFL”**
- Do **NOT eat or drink** in the auditorium
and do **NOT leave any waste**
- In case you forget some belongings:
“lost & found” located at the entrance
of the auditorium



Double Deck - Difficulties with rooms

As you know, there is intense work on campus generating a lot of noise 🙌 huge reduction in the number of usable rooms!

■ Interactive lecture:

- Mainly **STCC Cloud C**
- ⚠ In **CO3** on 15/09, 06/10 and 03/11

■ Exercise sessions:

- Rooms without computers: INF 1 (\approx 100 seats)
- Rooms with computers (but risk of noise): CO4, CO5, CO6 (\approx 110 seats)
- Rooms with computers (but hard to find): GRB001, GRC002 (\approx 75 seats)

For the exercise sessions (Tue. 10h15-12h)

I intend to join the following room (select one option):

47 a. INF1 (no computers)

11 b. CO4

4 c. CO5

3 d. CO6

4 e. I would prefer the rooms in GR

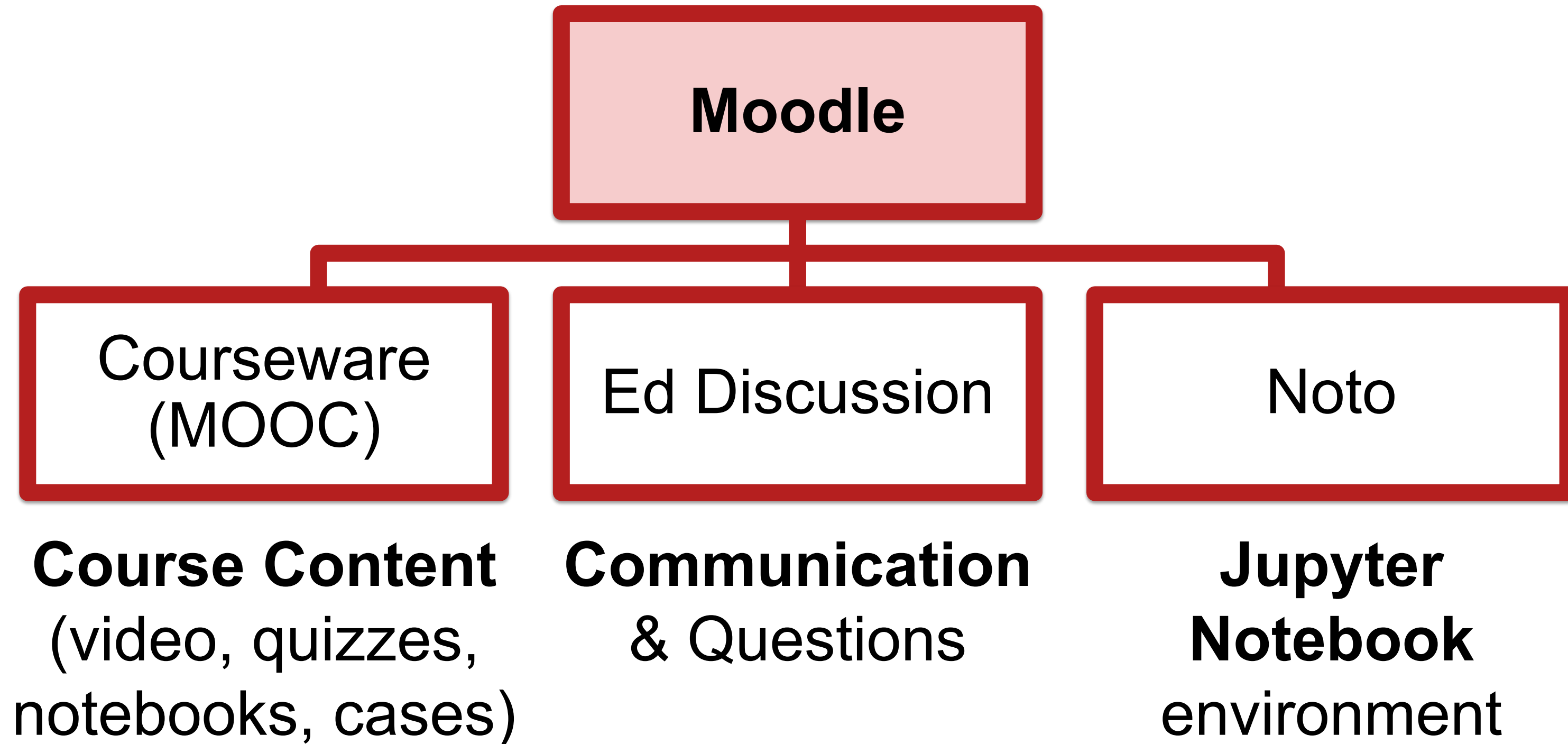
16 f. I will not come (schedule conflict...)

URL: ttpoll.eu

Session ID: cs290

The tools we use

<https://go.epfl.ch/CS-290>



Learning with videos

Role of videos in the course

- Introduce the **theory**

= technical and ethical *knowledge*

👉 practice = **quizzes**

- Introduce “**strategies**”

= *methodologies* for responsible software design

👉 practice = **case studies**

When watching the videos I will:



42% a. Take handwritten notes (paper, tablet)

10% b. Take typed notes on my laptop

47% c. Only watch and listen

1% d. Other

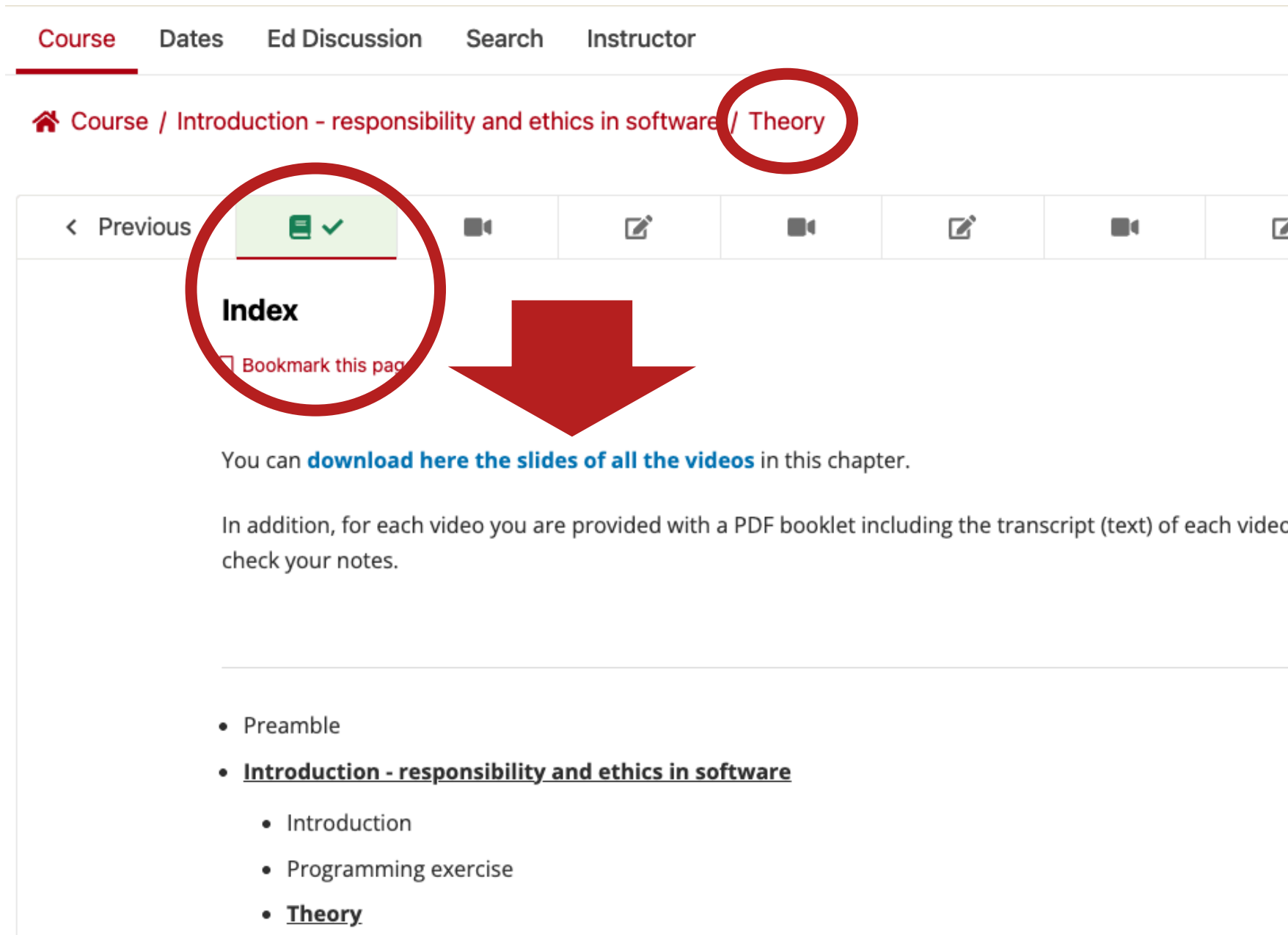
URL: ttpoll.eu

Session ID: cs290

Taking notes with videos

■ Note taking support:

- Blank page
- Slides from the video



The screenshot shows a course page with a navigation bar at the top containing 'Course', 'Dates', 'Ed Discussion', 'Search', and 'Instructor'. Below this is a breadcrumb trail: 'Course / Introduction - responsibility and ethics in software / Theory', where 'Theory' is circled in red. A video player interface is visible, with a 'Previous' button and a 'Next' button (both with video camera icons). The 'Next' button is highlighted with a green checkmark and circled in red. Below the video player, the text 'Index' is circled in red, with a large red arrow pointing down to the text 'You can [download here the slides of all the videos](#) in this chapter.' Below this, there is a paragraph: 'In addition, for each video you are provided with a PDF booklet including the transcript (text) of each video check your notes.' At the bottom, there is a list of items: 'Preamble', 'Introduction - responsibility and ethics in software' (which is bolded), and 'Theory' (which is bolded and underlined).

■ Playing speed:

- Adjust the speed to be able to take notes without pausing the video (as in a lecture)
- Or regularly pause the video to note down important elements

Checking your notes

0.1 What does responsible software mean?

[Bookmark this page](#)

Watch:

EPFL

What does responsible software mean?

Cécile Hardebolle

Responsible Software

Start of transcript. Skip to the end.

What does the term "responsible software" mean to you?

Pause this video for a second, write down your answer on a piece of paper, and then start again.

At the end of this video you should be able to explain what we mean by "responsible software" in this course. You should also be able to describe the core approach of the course

Video

- Download video file

Transcripts

- Download SubRip (.srt) file
- Download Text (.txt) file

Download here the [PDF document](#) corresponding to the above video.

■ BOOC = slides + transcript

Responsible software?

(van de Poel, 2011)

Focus on:

- Human responsibility
- Active (forward-looking) responsibility

It is important to clarify the approach that we will use in this course. First, we will consider that software cannot be considered responsible in itself, and so we will focus on the responsibility of people, and more specifically, on the responsibility of software engineers.

notes

summary

1m 1s

page 5/24 - 0.1 What does responsible software mean

■ Transcript

Reviewing content

■ Interactive quizzes

- With feedback
- Once attempted:
possibility to reset
+ show the answer

Question 0.1.4

0 points possible (ungraded)

Which of the following best describes active responsibility as presented in the course?

- Assigning blame after a software failure has occurred
- Following legal requirements during software development
- Reacting to problems after they have been identified
- Proactively anticipating and preventing negative impacts before they occur

Reset | Show answer

Envoyer

■ Search tool for video content

Course Dates Ed Discussion Recherche

🏠 Course / Introduction - responsibility and ethics in software / Theory

Search for a video sequence by inserting a keyword in the bar below! Then click on the contextual links to access the corresponding video sequences.

Search...



dilemma



Introduction - responsibility and ethics in software

Theory

0.3 Ethical dilemmas:

[What is a dilemma for you?](#) [Could you give an example of a dilemma of ethical dilemma and give an example. in ethical problems, including dilemmas. An ethical dilemma is a specific type that real or genuine ethical dilemmas are dilemmas in which none](#) [Such dilemmas are also called](#) [What does such a dilemma look like?](#) [In real life, ethical dilemmas of ethical issues and ethical dilemmas and ethical dilemmas in the course, to any ethical dilemma. Dilemmas often present](#) [First, we have seen that ethical dilemmas of an ethical dilemma.](#)

Regarding the moment I will watch the videos:



26% a. I will schedule a fixed weekly time slot

58% b. I will adapt each week depending on my workload

15% c. I don't know yet

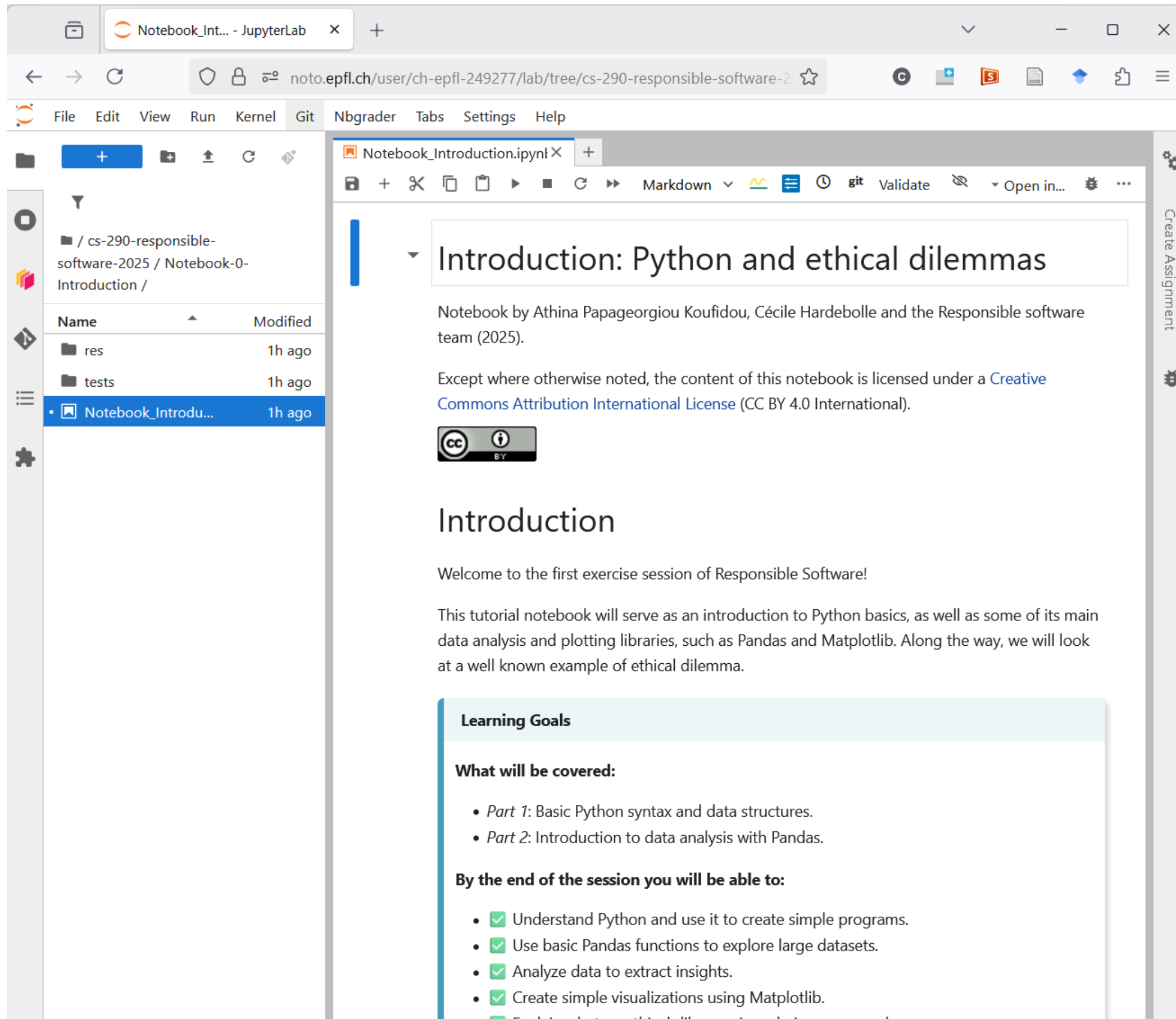
0% d. Other

URL: ttpoll.eu

Session ID: cs290

Working with Jupyter Notebooks

Jupyter notebooks



The screenshot shows a JupyterLab interface in a browser. The browser address bar shows the URL: noto.epfl.ch/user/ch-epfl-249277/lab/tree/cs-290-responsible-software-2. The notebook title is "Introduction: Python and ethical dilemmas". The content includes a Creative Commons Attribution International License (CC BY 4.0 International) logo and the following text:

Introduction

Welcome to the first exercise session of Responsible Software!

This tutorial notebook will serve as an introduction to Python basics, as well as some of its main data analysis and plotting libraries, such as Pandas and Matplotlib. Along the way, we will look at a well known example of ethical dilemma.

Learning Goals

What will be covered:

- *Part 1:* Basic Python syntax and data structures.
- *Part 2:* Introduction to data analysis with Pandas.

By the end of the session you will be able to:

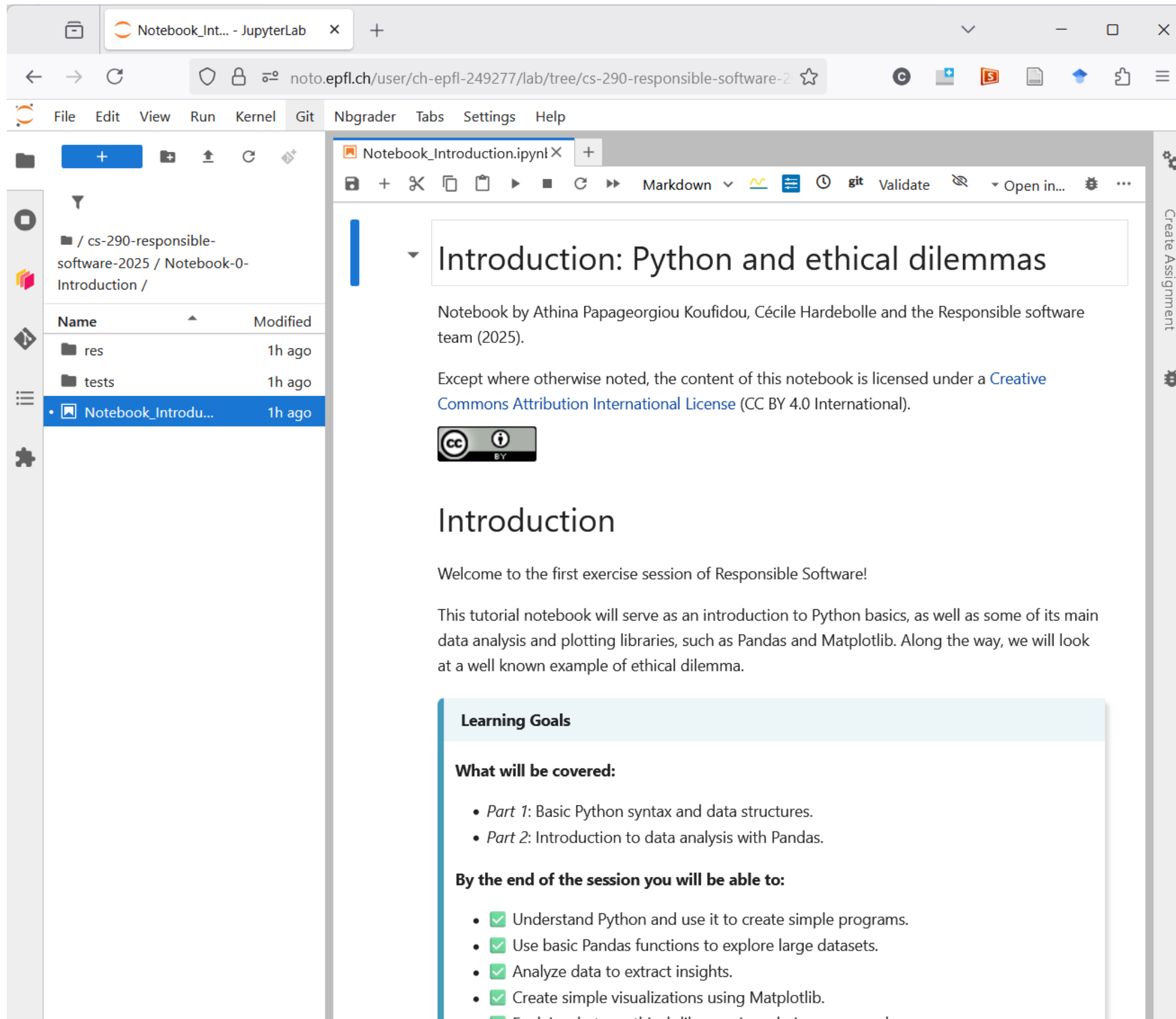
- Understand Python and use it to create simple programs.
- Use basic Pandas functions to explore large datasets.
- Analyze data to extract insights.
- Create simple visualizations using Matplotlib.

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

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

(you can use VS Code, but we don't provide support for it)

Weekly exercises with notebooks



The screenshot shows a JupyterLab interface with a file explorer on the left and a notebook titled "Introduction: Python and ethical dilemmas" in the main area. The notebook content includes a title, author information, a license notice (Creative Commons Attribution International License), and a "Learning Goals" section with bullet points.

Learning Goals

What will be covered:

- *Part 1:* Basic Python syntax and data structures.
- *Part 2:* Introduction to data analysis with Pandas.

By the end of the session you will be able to:

- Understand Python and use it to create simple programs.
- Use basic Pandas functions to explore large datasets.
- Analyze data to extract insights.
- Create simple visualizations using Matplotlib.

■ Best feature of the course according to last year's students!

■ But:

- Python programming
- Good amount of reading
- Can be long (that's one of the reasons why we have added 1 ECTS to the course)

Getting started with noto

The screenshot shows the JupyterLab interface. On the left, the file browser displays the workspace structure:

- Workspace: / cs-290-responsible-software-2025 /
- Sub-directory: Notebook-0-Introduction /
- Files and folders:
 - res (3 days ago)
 - tests (3 days ago)
 - Notebook_Introduction.ipynb (3 days ago)

The main 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!

At the bottom of the interface, the status bar shows: Mode: Command | Ln 1, Col 1 | Notebook_Introduction.ipynb | 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`. Below the browser is a menu bar with options: File, Edit, View, Run, Kernel, Git, Nbgrader, Tabs, Settings, Help. On the left is a sidebar with a file explorer showing a tree structure for `NOTEBOOK_INTRODUCTION.IPYNB`. The tree includes sections like "Part 1: Python basics" and "Part 2: Data analysis with Python and Pandas". A red circle highlights the hamburger menu icon in the sidebar. A red callout bubble with the text "Interactive outline" points to this icon. The main area shows 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.

```
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, the status bar shows "Simple" mode, "main" environment, "Python3 | Idle", and "Mode: Command".

Executing code

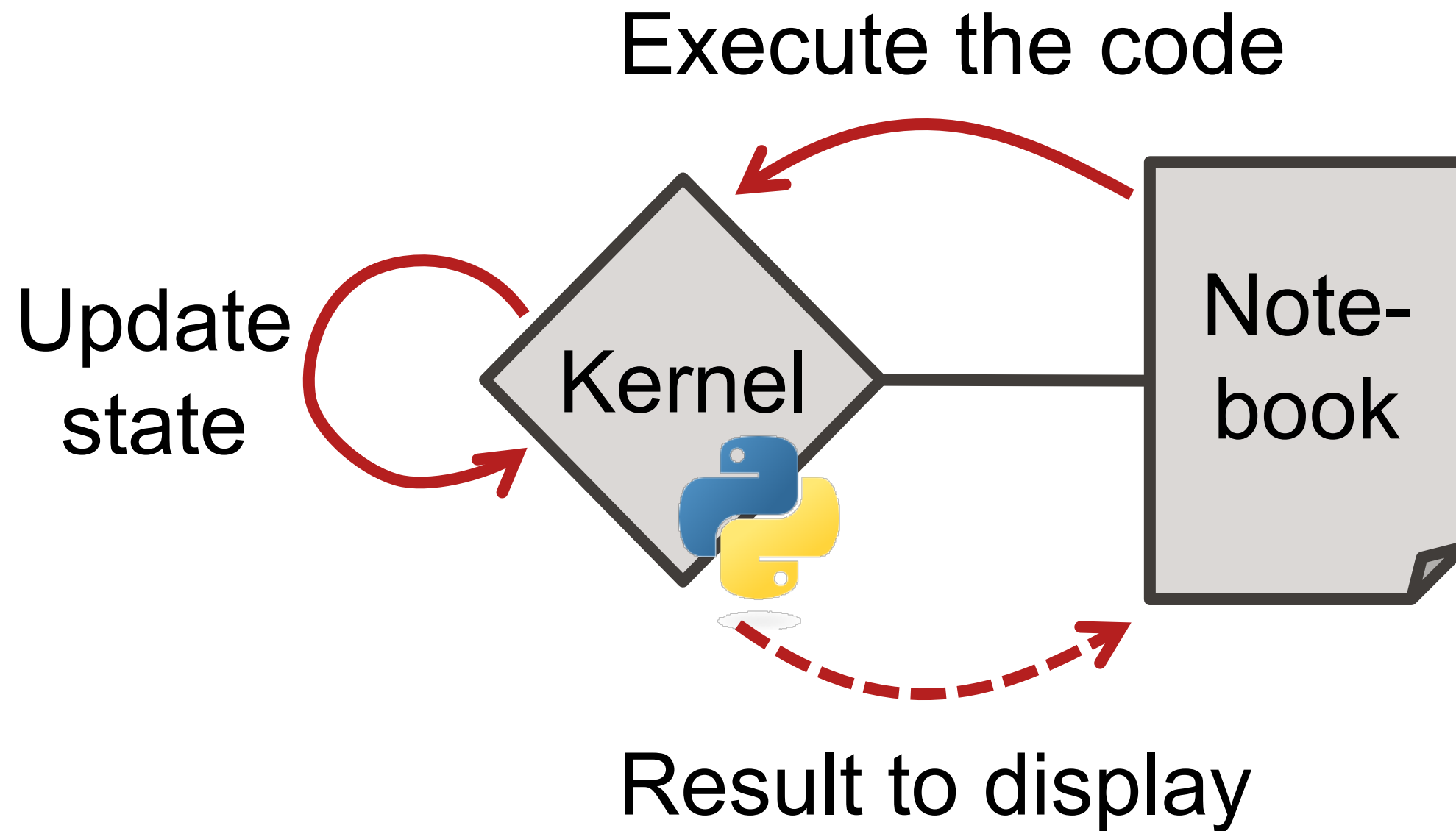
The image shows a JupyterLab notebook window with several annotations:

- 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 in the notebook.
- A red speech bubble at the bottom left contains the text "Execution number". A blue arrow points from this bubble to the "[1]:" label of the first code cell.

The notebook content includes:

- Text: "We will start by creating a string variable to represent the scenario."
- Code cell [1]: `scenario = "A self-driving car with brake failure is heading towards five ped"`
- Text: "You can use the `print` function to display the content of a variable and the `type` function to display the type of a variable."
- Text: "Instruction: Execute the cell below to see the output."
- Code cell [2]: `print(scenario)`
`print("The type of the scenario variable is", type(scenario))`
- Output of cell [2]:
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'>

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 screenshot of the JupyterLab web interface. The browser address bar shows the URL `noto.epfl.ch/user/ch-epfl-249277/lab/tree/git_Noto/responsible-soft`. The JupyterLab menu bar includes File, Edit, View, Run, Kernel, Git, Nbgrader, Tabs, Settings, and Help. On the left sidebar, the 'Kernels' panel is highlighted with a red circle, and a red arrow points to the 'Python3' kernel. Below it, the active notebook 'Notebook_Introduction.ipynb' is listed. A red speech bubble with the text 'List of active kernels' points to the Kernels panel. The main notebook area shows a code cell with the following content:

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

The cell contains text instructions: 'We will start by creating a string variable to represent the scenario.' and 'You can use the print function to display the content of a variable and the type function to display its type.' Below this is an 'Instructions' box: 'Execute the cell below to see the output.' The code cell contains:

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

The output of the 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 status bar at the bottom 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)
```

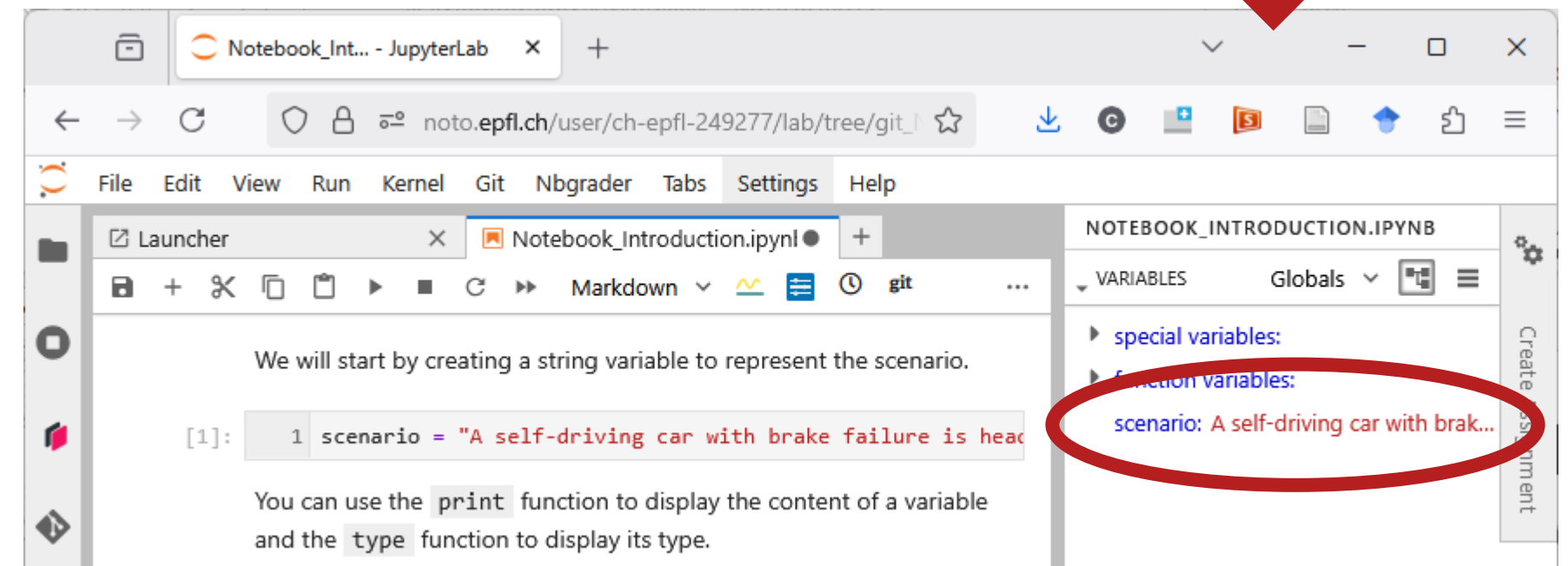
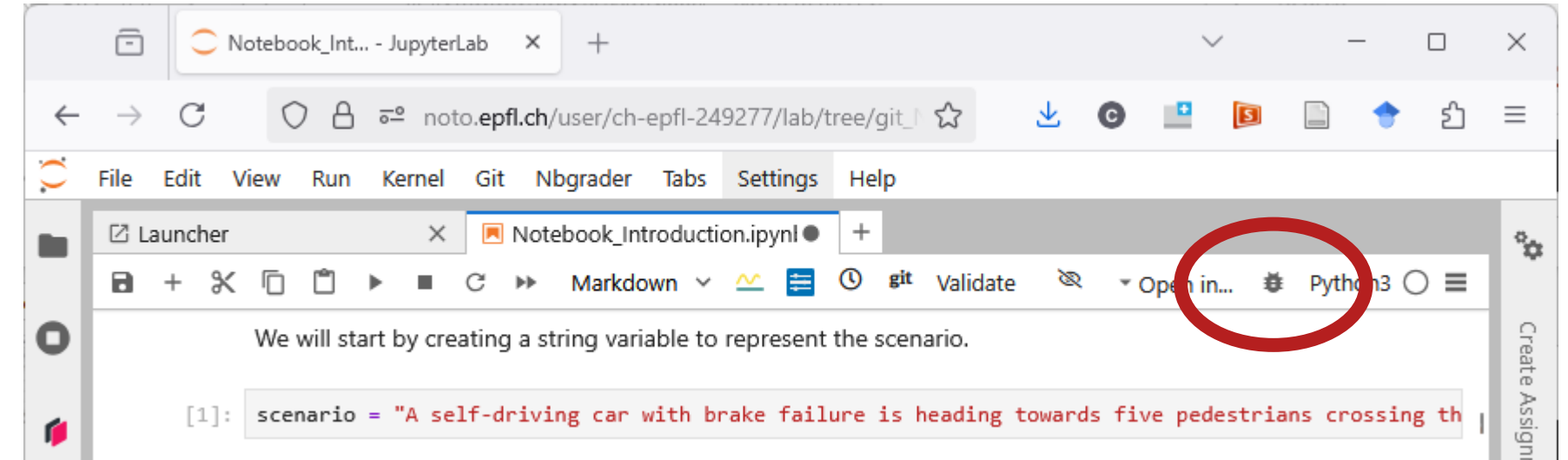
A self-driving car with brake failure is heading towards five pedestrians crossing the street. The car can swerve to the other lane, hitting one pedestrian instead. What should the autopilot do?

■ 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 cross  
ing the street. The car can swerve to the other lane, hitting one pedestrian ins  
tead. 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.ipynb'. The notebook contains two code cells. The first cell, labeled '[]:', contains a Python assignment: `scenario = "A self-driving car with brake failure is heading towards five ped"`. The second cell, labeled '[1]:', contains two lines of Python code: `print(scenario)` and `print("The type of the scenario variable is", type(scenario))`. The output of the second cell shows a `NameError` with the message `NameError: name 'scenario' is not defined`. A red arrow points from the error message back to the first cell, indicating that the variable `scenario` was not defined before it was used in the second cell. The error message is circled in red, and the first cell is also circled in red.

Launcher Notebook_Introduction.ipynb

```
[ ]: 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.

```
[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

Simple 2 main Python3 | Idle Mode: Command Ln 1, Col 1 Notebook_Introduction.ipynb 0

Actions on the kernel

The screenshot displays the JupyterLab interface. The 'Kernel' menu is open, showing options such as 'Interrupt Kernel', 'Restart Kernel...', 'Restart Kernel and Clear Outputs of All Cells...', 'Restart Kernel and Run up to Selected Cell...', 'Restart Kernel and Run All Cells...', 'Restart Kernel and Debug...', 'Reconnect to Kernel', 'Shut Down Kernel', 'Shut Down All Kernels...', and 'Change Kernel...'. A red arrow points to the 'Restart Kernel...' 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.

```
[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 your work

The image shows a JupyterLab notebook interface. A red callout box with the text "Ctrl + S" points to the save icon (a floppy disk) in the notebook's toolbar. The notebook contains two code cells. The first cell, labeled [2]:, contains the following Python code:

```
scenario = "A self-driving car with brake failure is heading towards five ped |
```

The second cell, labeled [3]:, contains the following Python code:

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

The output of the second 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 interface also shows a file browser on the left 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 status bar at the bottom indicates: Simple 2 main Python3 | Idle Mode: Command Ln 1, Col 1 Notebook_Introduction.ipynb 0

Opening multiple notebooks

The image shows a JupyterLab interface with two browser tabs open. The first tab, titled "Notebook_Int... (2) - JupyterLab", is circled in red. A red callout bubble points to this tab with the text: "⚠️ Avoid opening noto in multiple browser tabs!". The JupyterLab interface displays a file browser on the left with a list of files and folders, including "Sample_HTML-PDF", "Sample_Octave", and "Sample_Py3.ipynb". The main area shows the content of "Notebook_Introduction.ipynb", which includes a title "Introduction: Python and ethical dilemmas" and a Creative Commons Attribution International License (CC BY 4.0 International) logo. The status bar at the bottom indicates "Mode: Command" and "Ln 1, Col 1".

Name	Modified
Sample_HTML-PDF	yesterday
Sample_Octave	yesterday
Sample_Bash.ipynb	yesterday
Sample_C_CPP.ipynb	yesterday
Sample_dask.ipynb	yesterday
Sample_Font.ipynb	yesterday
Sample_panel.ipynb	yesterday
Sample_Py3_cartopy.ipynb	yesterday
Sample_Py3_R.ipynb	yesterday
• Sample_Py3.ipynb	next yr.
Sample_R.ipynb	yesterday
Sample_SoS.ipynb	yesterday

Opening multiple notebooks

The screenshot shows the JupyterLab interface with three notebooks open in the top bar: Notebook_Introduction.ipyn, 30_Using_git.ipynb, and Sample_Py3.ipynb. A red oval highlights these tabs. The main content area shows the 'Python3 Notebook' with the following text:

Python3 Notebook

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 *L^AT_EX* 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'.

Instead, open the notebooks inside noto

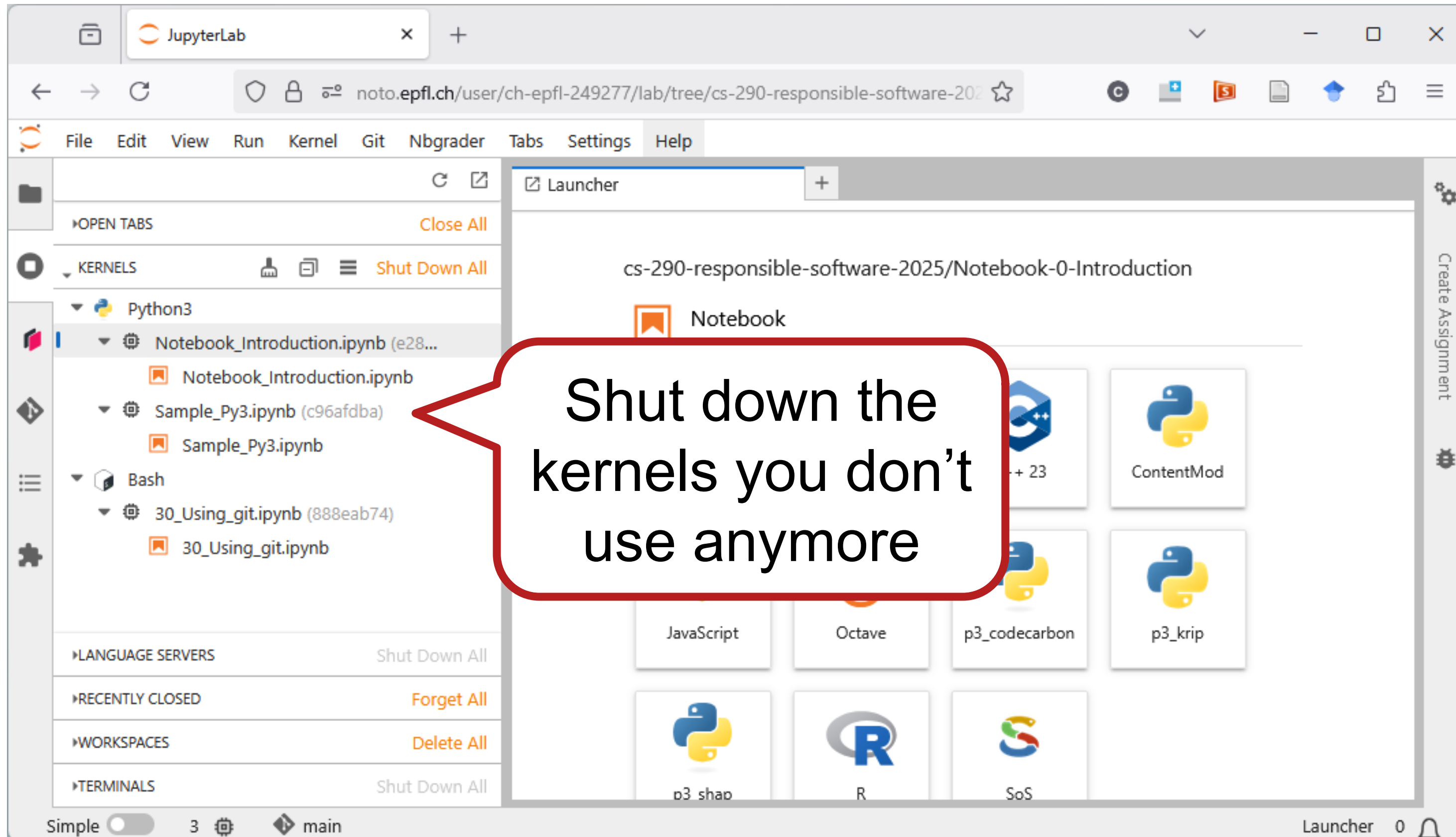
Opening multiple notebooks

You can collapse the side bar

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

The screenshot displays the JupyterLab web interface. At the top, a browser window shows the URL `noto.epfl.ch/user/ch-epfl-249277/lab/tree/Documentation/Examples/Samp`. Below the browser, the JupyterLab application has a menu bar with options: File, Edit, View, Run, Kernel, Git, Nbgrader, Tabs, Settings, and Help. Two notebook panels are open side-by-side. The left panel, titled 'Notebook_Introduction.ipynl', shows a document with the heading 'Introduction: Python and al dilemmas' and a Creative Commons license icon. The right panel, titled 'Sample_Py3.ipynb', shows a document with the heading 'Python3 Notebook' and the text 'First example'. The interface includes a sidebar on the left for file navigation and a vertical toolbar on the right with icons for 'Create Assignment' and a gear icon. The bottom status bar shows 'Simple' mode, '3' tabs, 'main Python3 | Idle', 'Mode: Command', 'Ln 1, Col 1', and the active notebook 'Sample_Py3.ipynb'.

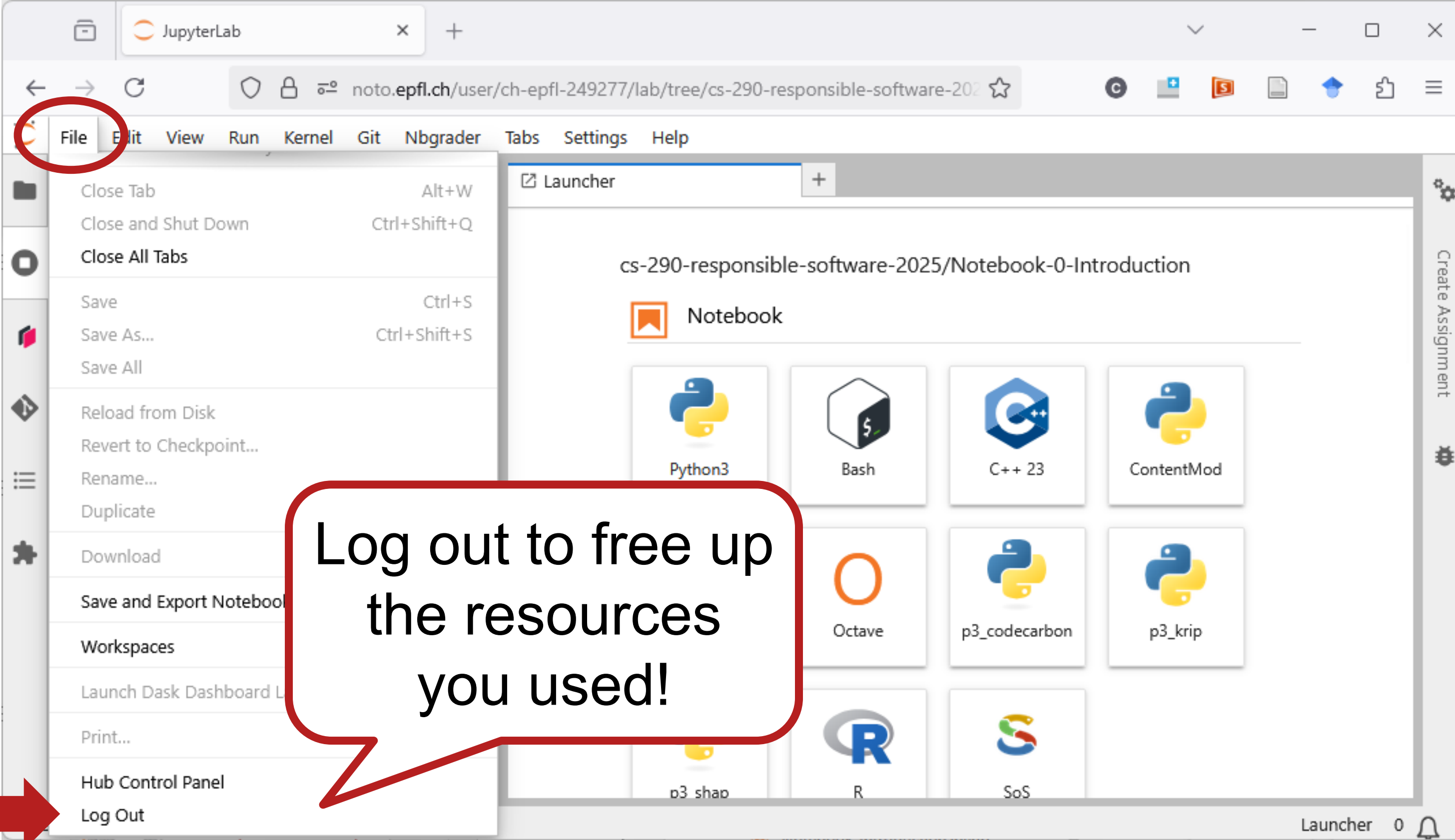
Save resources!



The image shows a screenshot of the 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 side, 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 a red border is overlaid on the interface, containing the text: "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" and a grid of available kernels including ContentMod, p3_krip, JavaScript, Octave, p3_codecarbon, 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-202`. 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

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

Notebook

Python3 Bash C++ 23 ContentMod

Octave p3_codecarbon p3_krip

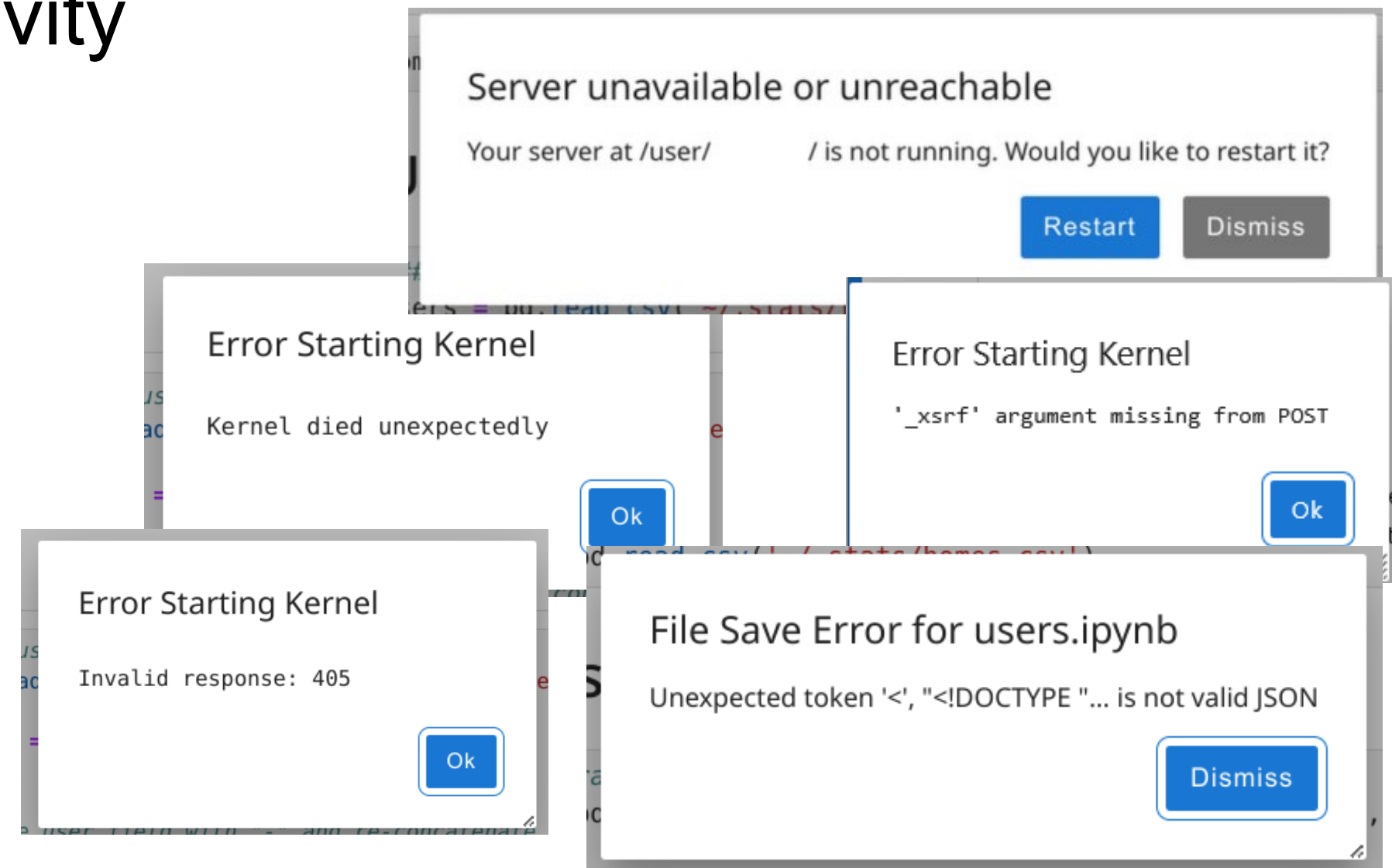
p3_shap R SoS

Launcher 0

Log out to free up the resources you used!

Error messages

- If you start getting **popups with error messages**, it means that
 - ⚠ your personal server is no longer available ⚠
 - Timeout after \approx 30 minutes of inactivity
 - Too many resources used
- Do not simply dismiss or click ok!
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>



Equipment for the programming exercises

■ Computer

- Your own laptop
- Machines in the computer rooms:
 - ◆ Use the default OS
 - ◆ Or start a windows virtual machine: IC-Windows-Cours2

■ Recent web browser

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

Getting started with case studies

Case studies

- Cases = situations which are realistic (in general) but simplified
- Studying a case =
 - Putting you in the shoes of the software designer
 - Having you practice methodologies for responsible design
- Doing it in class together 🙌 figure out the “gray zones”

A “getting started” case

See posts on [SpeakUp](#)

- You are designing a **form for customer contact details** on a retail website.
- You decide to apply the following rules :
 - People have one first name and one last name
 - People’s names do not contain special characters such as apostrophes
 - People’s names cannot change
- What **ethical issues** could you have with these choices?

Post your ideas:

<https://speakup.epfl.ch>

Room key: **77412**



When is it an “ethical” issue?

- How can we know when an issue is an “ethical” issue?

When is it an “ethical” issue?

- Ethical = “the right thing to do”
(≠ from legal obligation)

👉 **How do we know what is “right”?**

- Moral frameworks:

- Respect the rights of people
- Treat people fairly
- Do good and avoid harm
- Serve the community,
protect the common good
- Be a good person
- Care for people
- ...

- In software engineering?

- Intersection of technical possibility, business interests, public policy, and human well-being
- Regulatory frameworks
- Multiple stakeholders

What's next?

To do as soon as possible

Check that you can access the **course material** on Courseware:

- Go to moodle
- Click on the link to **Courseware**
 - 👉 In case of issue, post on Ed Discussion in “technical issue”

Check that you can access the **notebook** for tomorrow:

- In Courseware, find the programming exercise for the Introduction chapter
- Click on the link to the notebook, which should launch **noto** and open the Introduction notebook
 - 👉 In case of issue, post on Ed Discussion in “technical issue”

Tomorrow: exercise session

Room	Student assistants	Doctoral assistants
INF 1 (\approx 100 seats, no computers)	Eugène Bergeron Camille Lannoye Florian Kolly	Yingxuan You
CO 4 (\approx 35 seats)	Mamoun Imghi	Ayush Kumar Tarun
CO 5 (\approx 40 seats)	Othmane Housni + Elyes Trabelsi	
CO 6 (\approx 40 seats)	Noa Trojman	

By next Monday:

- Watch **videos 0.1 to 0.5** + do the **quizzes**
- Finish the programming exercise

- On Monday 15: **case studies in class**
 - We will work on interactive questions on the theory part
 - We will work on the case studies together
i.e. the “lecture” is more like an exercise session, where I will be your TA