

# Exercise sessions

Welcome to the course “*Introduction to Optimization and Operations Research*”.

At each exercise session, you will be expected to have a computer available, and you will solve exercises using Jupyter notebooks. Jupyter notebook files will be available at the github link [https://github.com/transp-or/optimization\\_2024/tree/main](https://github.com/transp-or/optimization_2024/tree/main). Each session will have one notebook with the questions and another notebook with solutions. We encourage you to try to solve the questions without looking at the solutions and to use Chatgpt or your preferred LLM to assist you with coding questions.

There are different software tools that you can use to run Jupyter notebooks (e.g., conda) but EPFL offers a tool called “noto” which requires no installation, and you can access with your gaspar account.

You can run notebook files with noto.

## Option 1:

To get started click on this link [NOTO exercise 1](#)

Or copy and paste the link on your browser: [https://noto.epfl.ch/hub/user-redirect/git-pull?repo=https%3A%2F%2Fgithub.com%2Ftransp-or%2Foptimization\\_2024&urlpath=lab%2Ftree%2Foptimization\\_2024%2F%2Fweek\\_01%2Fflab01\\_coola\\_questions.ipynb%3Fautodecode&branch=main](https://noto.epfl.ch/hub/user-redirect/git-pull?repo=https%3A%2F%2Fgithub.com%2Ftransp-or%2Foptimization_2024&urlpath=lab%2Ftree%2Foptimization_2024%2F%2Fweek_01%2Fflab01_coola_questions.ipynb%3Fautodecode&branch=main)

Once the link opens noto, you should select the optimization kernel.

## Option 2:

Download files from github and upload them on noto or conda.

Noto:

1. Download exercise files to your computer from [https://github.com/transp-or/optimization\\_2024/tree/main](https://github.com/transp-or/optimization_2024/tree/main).
2. Launch Noto <https://go.epfl.ch/noto-share>
3. Upload exercise files to noto.
4. Open exercise files with the kernel “optimization”

The screenshot shows a JupyterLab interface with a file browser on the left, a notebook titled "Simplex tableau algorithm" in the center, and a "Start Preferred Kernel" dialog box open in the foreground. The dialog box has a "Start Preferred Kernel" section with a dropdown menu showing "Python3". Below this, there are sections for "Use No Kernel", "Start Other Kernel", "Use Kernel from Preferred Session", and "Use Kernel from Other Session". The "Use Kernel from Preferred Session" section has a dropdown menu showing "Python3". The "Use Kernel from Other Session" section has a list of sessions: "00\_Welcome.ipynb", "10\_Envs\_and\_kernels.ipynb", and "11\_Tutorial\_Envs.ipynb". The "Use Kernel from Preferred Session" section has a checkbox labeled "Always start the preferred kernel" which is unchecked. The "Use Kernel from Other Session" section has a "No Kernel" button and a "Select" button. A red arrow points to the "Optimization" kernel in the "Start Other Kernel" list, and another red arrow points to the "No Kernel" button in the dialog box.

File Edit View Run Kernel Git Nbgrader Tabs Settings Help

Launcher lab05\_pivoting\_responses.ipynb lab06\_simplex\_t...

Filter files by name

/ Documentation / ORcourse /

Name	Modified
Introduction	47m ago
linear optimization	12m ago
simplex	12m ago
lab01_enumerat...	8m ago
lab01_enumerat...	8m ago
lab02_graphical_...	8m ago
lab02_graphical_...	5m ago
lab03_simplex_q...	8m ago
lab03_simplex_f...	8m ago
lab04_tableau_q...	8m ago
lab04_tableau_f...	8m ago
lab05_pivoting_...	8m ago
lab05_pivoting_f...	3m ago
lab06_simplex_t...	8m ago
lab07_phase_on...	8m ago
lab07_phase_on...	8m ago
lab08_multiple_...	8m ago
lab08_multiple_...	8m ago

## Simplex tableau algorithm

Introduction to optimization and o

Michel Bierlaire

```
[ ]:
import sys
from enum import Enum, auto

import numpy as np
from teaching_optimization.tableau import Simplex
```

Consider the following linear optimization problem

subject to

The objective of this exercise is to implement the simplex

First, we write the problem in standard form:

$$\min -4x_0 + 3x_1$$
$$2x_0 + x_1 + x_2 = 6,$$
$$x_0 - x_1 + x_2 = 2,$$
$$x_0, x_1, x_2 \geq 0.$$

Open in... No Kernel

Create Assignment

# Example of ChatGPT answer:



How do I merge two lists together in a numpy array in python ?



To merge two lists together into a NumPy array in Python, you can follow these steps:

1. **Import the NumPy library:** Make sure you have NumPy installed and then import it.
2. **Convert the lists to NumPy arrays:** If the lists are not already NumPy arrays, you can convert them.
3. **Use NumPy's concatenation function:** You can use `numpy.concatenate` to merge the arrays.

Here's a simple example:

python

Copy code

```
import numpy as np

# Example lists
list1 = [1, 2, 3]
list2 = [4, 5, 6]

# Convert lists to numpy arrays
array1 = np.array(list1)
array2 = np.array(list2)

# Merge the arrays
merged_array = np.concatenate((array1, array2))

print(merged_array)
```

This will output:

csharp

Copy code

```
[1 2 3 4 5 6]
```

## Explanation:

- `np.array(list1)` and `np.array(list2)` convert the Python lists into NumPy arrays.
- `np.concatenate((array1, array2))` merges the two arrays into a single NumPy array.

If you want to merge them along a different axis (e.g., stacking them vertically or horizontally), you can use `np.vstack`, `np.hstack`, or similar functions depending on your needs.



Message ChatGPT

