UNIVERSITÉ DE LYON

ENTPE

L'école de l'aménagement durable des territoires

Data Science : Principles and Applications

Class 2: Essential Tools and Practices for Programming Projects in AI

Speaker : Rim SLAMA SALMI, Assistant Professor (ENTPE, Univ. Eiffel, LICIT-ECO7)
contact : rim.slamasalmi@entpe.fr
Github account: Rimons

Install Environment

- **Logiciels à installer**

✅ **Obligatoires**

- **Visual Studio Code (VS Code)** → code.visualstudio.com/Download
- **Git** → git-scm.com/downloads
- **Anaconda** (inclut Python) → anaconda.com/download

- **Extensions VS Code**

(à rechercher dans Extensions ou Ctrl+Shift+X)

✅ **Obligatoires**

- **Python** (Microsoft) → Python Extension
- **Jupyter** (Microsoft) → Jupyter Extension

⚙ **Optionnelles**

- **Conda Wingman (DJSaunders1997)** → gérer facilement les environnements Conda (création, activation, export) directement depuis la palette de commandes VS Code.-> Conda Wingman
- **Markdown All in One** (Yu Zhang) → Markdown All in One
  → *aide à écrire et structurer les fichiers README en Markdown.*
- **GitHub Pull Requests and Issues** (GitHub) → GitHub PRs and Issues
  → *gérer issues, branches et PR directement depuis VS Code.*
- **GitHub Copilot** (GitHub) → GitHub Copilot
  → *autocomplétion intelligente pour écrire du code plus rapidement.*
- **GitHub Copilot Chat** (GitHub) → GitHub Copilot Chat
  → *chat intégré dans VS Code : explication du code, génération de tests, aide à écrire des messages de commit clairs.*

What and why good practices?

- **Why?** Without good practices:
  - even the **smartest code** becomes **useless**
  - your classmates can't understand it, you can't reuse it or share it,
  - your professor can't trust it.

- So, **what** are these good practices**?**

📁 **Organize the project & write readable code** → You get clear, structured work that's easy to follow.

📝 **Explain your code** → You get classmates (and your future self) who can understand and reuse it.

🧪 **Check and fix** → You get fewer bugs and you save time in the long run.

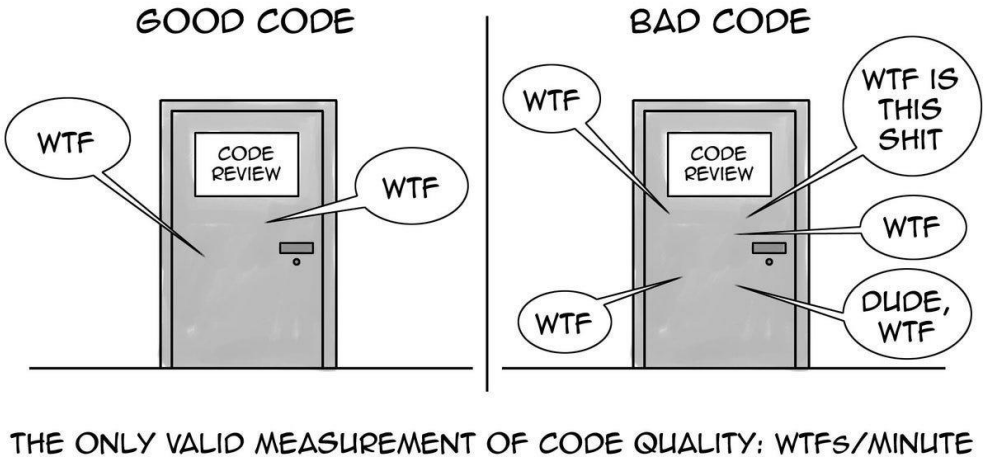
🔁 **Make your program reproducible** → You get a project that others can rerun and validate.

👥 **Work as a team** → You get smoother collaboration and better version tracking.

🧭 **Reflect on AI and ethics** → because AI projects should not only work technically, but also respect ethical principles and have a positive impact.

How Good Practices Should Be?

- **Organize the project**
  - Keep files and folders tidy
  - Separate data, code, and results
- **Write readable code**
  - Use clear names for files and functions
  - Follow a consistent style
- **Check and fix**
  - Test your code step by step
  - Handle errors instead of ignoring them
- **Explain your work**
  - Add short, useful comments
  - Write a simple guide (README)
- **Ensure reproducibility**
  - Make it easy for others to run your project
  - Avoid mixing tools and libraries between projects and Give each project its own space
- **Work as a Team**
  - **Share** work regularly
  - **Track** who does what
  - **Keep history** of all versions
- **Reflect on AI and Ethics**
  - **Follow an ethical approach during the project**
  - **Respect universal principles and rules**
  - **Apply an "ethical filter" to decisions and results**



Following these rules means less chaos, fewer headaches, and no more 'who wrote this mess?'
Just clean, shareable, and trustworthy projects.

Essential Tools and Practices for Programming Projects

- Practical tips and the tools available to help you:
  - **Virtual Environments**

  → ensure reproducibility | Tools: *venv, conda*

  - **Version Control & Collaboration**

  → track & share | Tools: *Git, GitHub*

  - **AI & Ethics**

  → responsible choices | Tools: Reflection Process , *checklists, documentation*

ESSENTIAL BEST PRACTICES IN PROGRAMMING PROJECTS

Essential Best Practices in Programming Projects:

Organize the project

❌ Messy project structure

```
ml_project/
  final2.ipynb
  test1.ipynb
  draft_v1.ipynb
  results_v3.ipynb
  data.csv
  data_clean.csv
  code.py
  script_v3.py
  helpers.py
  model1.pkl
  res.json
  fig1.png
  readme.txt
  req.txt
```

✅ Slightly better

```
ml_project/
  01_exploration.ipynb
  02_preprocessing.ipynb
  03_training.ipynb
  04_evaluation.ipynb
  raw_data.csv
  processed_data.csv
  train_model.py
  evaluate_model.py
  utils.py
  model.pkl
  metrics.json
  confusion_matrix.png
  README.md
  requirements.txt
```

Good practice

```
ml_project/
  data/
    raw_data.csv
    processed_data.csv
  src/
    train_model.py
    evaluate_model.py
    utils.py
  results/
    model.pkl
    metrics.json
    confusion_matrix.png
  notebooks/
    01_exploration.ipynb
    02_preprocessing.ipynb
    03_training.ipynb
    04_evaluation.ipynb
  README.md
  requirements.txt
```

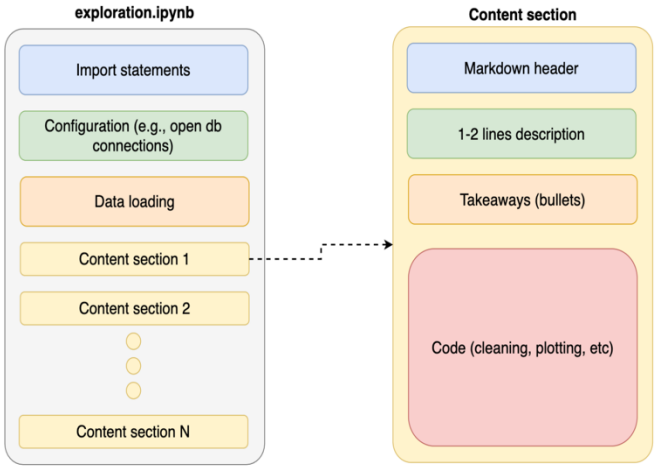
What's wrong?

- ❌ **File names are not meaningful** → you don't know what each file contains.
- ❌ **Files are not grouped by functionality** → everything is mixed, nothing is easy to find.
- ❌ **Project looks messy** → hard for others to understand or contribute.
- ❌ **You forget later what's what** → even you waste time figuring out your own work.

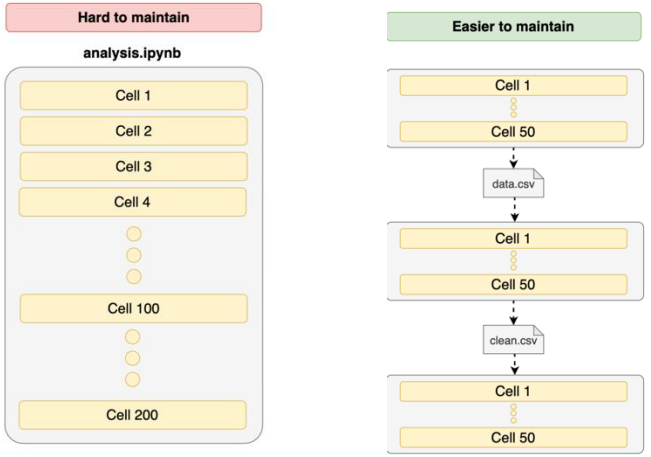
Essential Best Practices in Programming Projects:

writing clean Jupyter notebooks

Organize in sections



Write shorter notebooks



https://ploomber.io/blog/clean-nbs/

Essential Best Practices in Programming Projects:

Declare paths and configuration parameters in a single file

❌ Worst case : hardcoded directly when called

```
# Directly hardcoded in the function call
df = pd.read_csv("/Users/alice/Desktop/project/data/raw/customers.csv")

df.to_csv("/Users/bob/Documents/tmp/results/output.csv", index=False)
```

❌ Problem: paths are buried inside the code
→ hard to find, impossible to reuse.

✅ Slightly better : variables at the top of the script

```
# Centralized at the beginning of the script
RAW_DATA = data/raw/customers.csv"
RESULTS_PATH = "results/output.csv"

df = pd.read_csv(RAW_DATA)
df.to_csv(RESULTS_PATH, index=False)
```

⚠ Better: at least all paths are in one place.

🌟 Good practice — external config file (YAML)

```
yaml
data:
  raw_path: data/raw/customers.csv
  results_path: results/output.csv
```

```
import pandas as pd, yaml

# Load config from file
with open("config.yaml") as f:
    cfg = yaml.safe_load(f)

raw_path = cfg["data"]["raw_path"]
results_path = cfg["data"]["results_path"]

df = pd.read_csv(raw_path)
df.to_csv(results_path, index=False)
```

Essential Best Practices in Programming Projects:

Write readable code

Code should not only work!
It should also be easy to read and reuse!

❌ Less readable code

```
x = [8, 10, 12]
y = 0
for i in x:
    y += i
z = y / 3
print(z)
```

Problems:

- Variable names are unclear (`x`, `y`, `z`).
- No explanation of what the code does.
- Hard to reuse for other cases.

✅ More readable code

```
scores = [8, 10, 12]
total = 0

for score in scores:
    total += score

average_score = total / len(scores)
print("Average score:", average_score)
```

Improvements:

- Clear variable names (`scores`, `average_score`).
- Easier to understand at first glance.

Best practice: reusable function

```
def compute_average(numbers):
    """Return the average of a list of numbers."""
    return sum(numbers) / len(numbers)

grades = [15, 18, 12, 16]
average_grade = compute_average(grades)

print("Average grade:", average_grade)
```

Advantages:

- Self-explanatory function with a docstring.
- Reusable in different projects.
- Clean, consistent style.

📌 **Rule of thumb for snake_case**:
Write everything in **lowercase** and replace every space between words with an **underscore (_)**.
Example:

- `Student Name` → `student_name`
- `Total Sum` → `total_sum`

Essential Best Practices in Programming Projects:

Write readable code

Write Modular, Reusable Code

```python
# preprocessing.py

def clean_data(df):
    """
    Clean a DataFrame: drop rows with NaNs and lowercase column names.

    Parameters
    ----------
    df : pandas.DataFrame
        Input DataFrame.

    Returns
    -------
    pandas.DataFrame
        Cleaned DataFrame.
    """
    return df.dropna().rename(columns=str.lower)
```

- A **docstring** is a documentation string in Python.
- Written inside triple quotes """ ... """.
- Placed right after a function, class, or module definition.
- Describes what it does, its parameters, and return value.
- Can be read with `help()` or `.__doc__`.

```
src/
 preprocessing.py
 model.py
notebooks/
    exploration.ipynb
```

```python
# model.py
def train_model(X_train, y_train, params):
    model = RandomForestRegressor(**params)
    model.fit(X_train, y_train)
    return model
```

Essential Best Practices in Programming Projects:

Delete dead and unreachable code

```
def process_data(data):
    stats = compute_statistics(data)
    pivoted = pivot(data)
    # we are not using the stats variable!
    return transform(pivoted)
```

`compute_statistics` is dead code since we're never using the output (`stats`)

```
def process_data(data):
    pivoted = pivot(data)
    return transform(pivoted)
```

Essential Best Practices in Programming Projects:
Error handling

- **Example 1:** Importing an external library

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

```
ModuleNotFoundError: No
module named 'sklearn'
```

- **Example 2:** Reading a file

```
import pandas as pd

df = pd.read_csv("data.csv")
print("Lecture réussie :", df.head())
```

```
Error: the file cannot be
opened.
```

Essential Best Practices in Programming Projects:
Error handling

`try`/`except` helps write **robust and readable code**:

- **Prevent crash** → program doesn't stop with a cryptic error.
- **Clear message** → user understands what went wrong.
- **Control** → you decide: stop gracefully, retry, or use a fallback.

- **Example 1:** Importing an external library

```
try:
    from sklearn.preprocessing
import StandardScaler
    from sklearn.decomposition
import PCA
except Exception:
    print("Scikit-learn
introuvable. "
          "Installez-le avec :
pip install scikit-learn")
    PCA = None
```

- **Example 2:** Reading a file

```
import pandas as pd

try:
    df = pd.read_csv("data.csv",
encoding="utf-8")
    print("Lecture réussie :", df.head())
except Exception:
    print("Erreur : le fichier ne peut pas
être ouvert.")
```

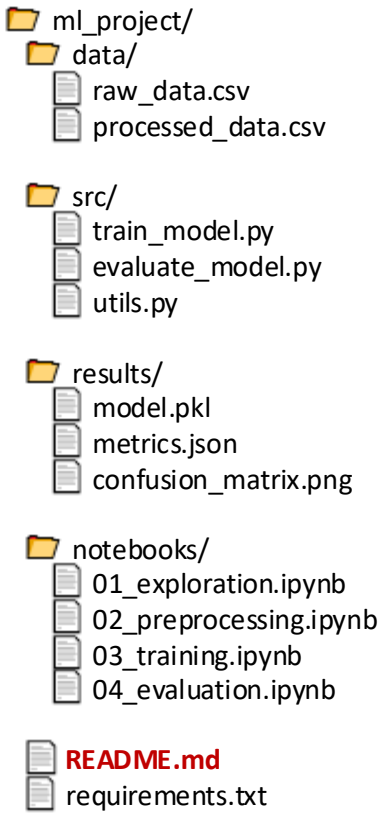
Essential Best Practices in Programming Projects:

Explain and share your work

- **Write ReadMe.md file Main information:**
  - **Online tutorial : link**
  - **Project Title & Description** – what the project does
  - **Author(s)** – who made it and contact
  - **Installation** – how to set it up
  - **Usage** – how to run it (examples)
  - **Results** – main outputs or metrics (optional)
  - **Project Structure** – short folder overview (optional but recommended)
  - **Citation/License** – if academic or open-source

- **Tutorial to write with Markdown** A simple guide to get started: https://www.markdownguide.org/basic-syntax/

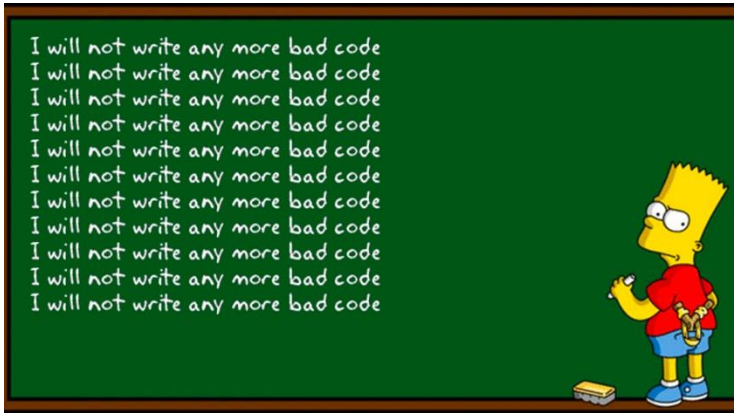
- **Examples**
  - **https://github.com/Rimons/DSPAP_Lab2-Teacher**
  - **https://github.com/SuperMedIntel/one-prompt?utm_source=chatgpt.com**
  - **https://github.com/autonomousvision/mip-splatting**



Essential Tools and Practices for Programming Projects

The Data Science Checklist: Best Practices for Data Science Projects

- Store data, artifacts, and code in different locations
- Declare paths and configuration parameters in a single file
- Store credentials securely
- Keep your repository clean by using a .gitignore file
- Automate the workflow end-to-end
- Declare all software dependencies
- Split production and development dependencies
- Have an informative and concise README
- Document code
- Organize your repository in a hierarchical structure
- Package your code
- Keep your Jupyter notebooks simple
- Use the logging module (do not use print)
- Test your code
- Take care of code quality
- Delete dead and unreachable code



Why Good Practices Matter

- **Organize the project**
  - Keep files and folders tidy
  - Separate data, code, and results
- **Write readable code**
  - Use clear names for files and functions
  - Follow a consistent style
- **Check and fix**
  - Test your code step by step
  - Handle errors instead of ignoring them
- **Explain your work**
  - Add short, useful comments
  - Write a simple guide (README)
- **Ensure reproducibility**
  - Make it easy for others to run your project
  - Avoid mixing tools and libraries between projects and Give each project its own space
- **Work as a Team**
  - **Share** work regularly
  - **Track** who does what
  - **Keep history** of all versions
- **Reflect on AI and Ethics**
  - **Follow an ethical approach during the project**
  - **Respect universal principles and rules**
  - **Apply an "ethical filter" to decisions and results**

VIRTUAL ENVIRONMENTS IN PYTHON

Virtual Environments

How a perfect Python environment looks like!

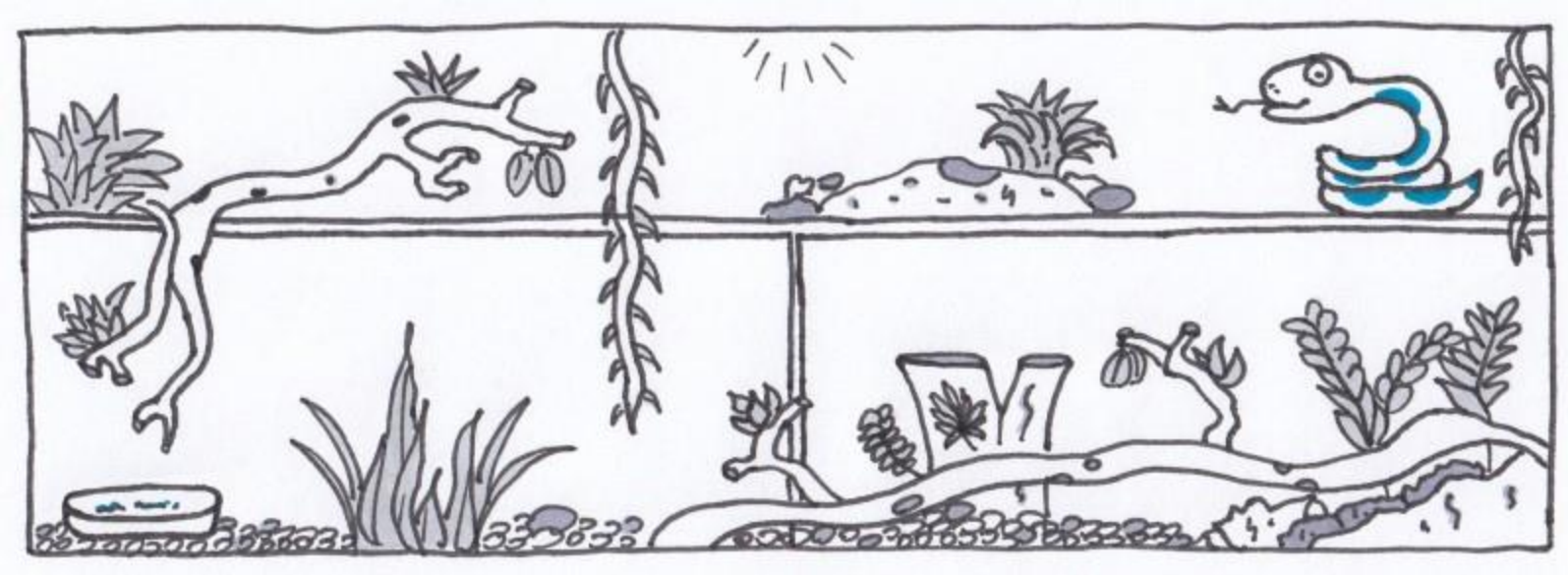
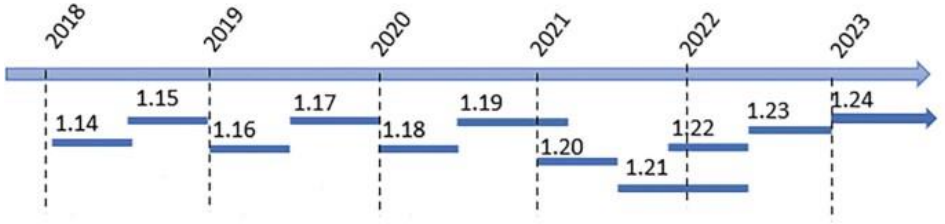


Image credit: https://medium.freecodecamp.org/why-you-need-python-environments-and-how-to-manage-them-with-conda-85f155f4353c

Virtual Environments

How library installation works in Python?



Success! Library 1.22 Installed

- Installing a new/different version of **NumPy (1.24)** will **replace the old one**.
- You can only have **one version of NumPy at a time** in the global Python installation.

By default, Python uses the **base environment** (base).

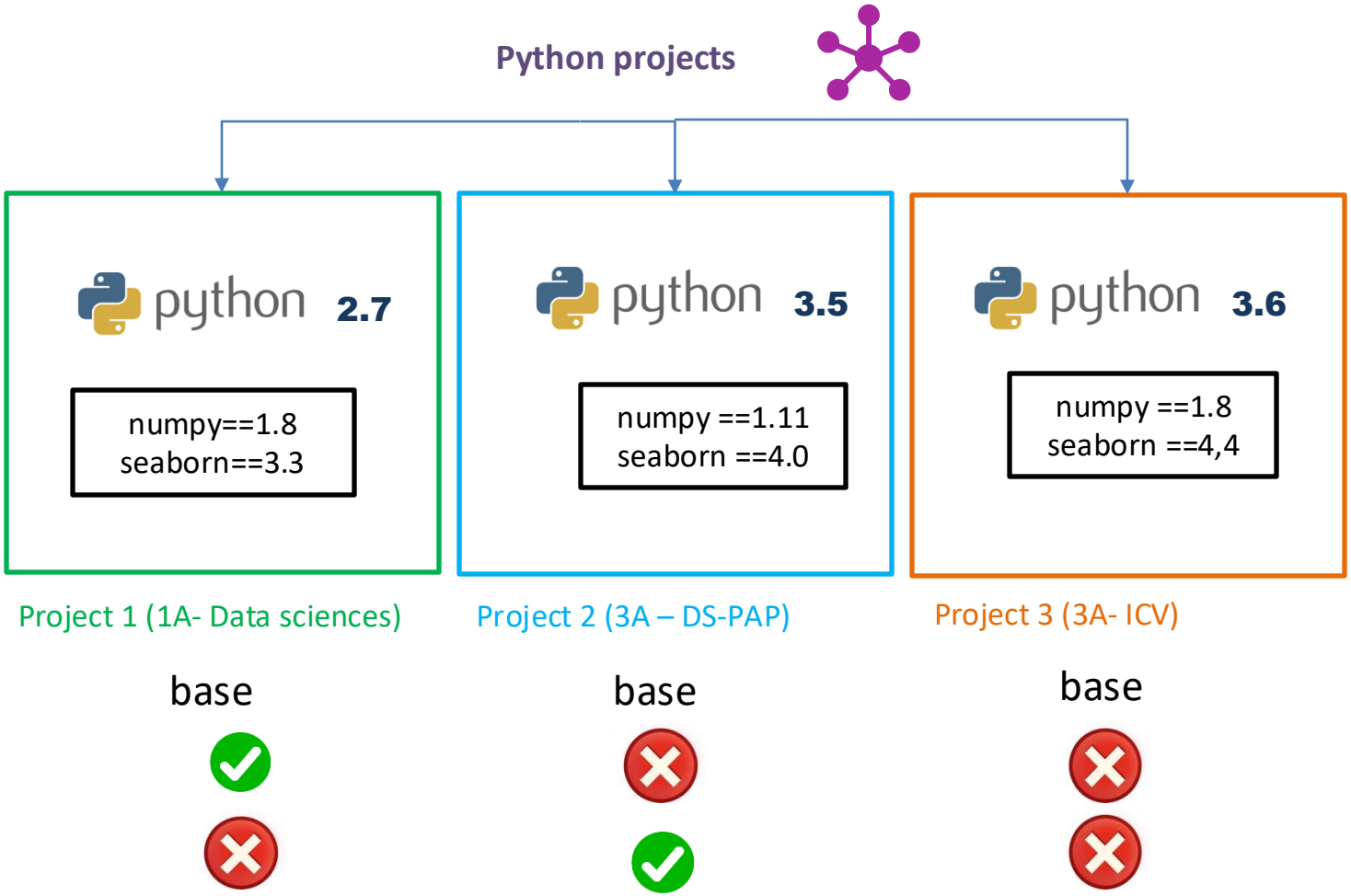
- If you don't create a virtual environment → all packages are installed in **base**.

```
# conda environments:
#
base                  * /opt/anaconda3
```

- What if you have different projects with different libraries versions ?

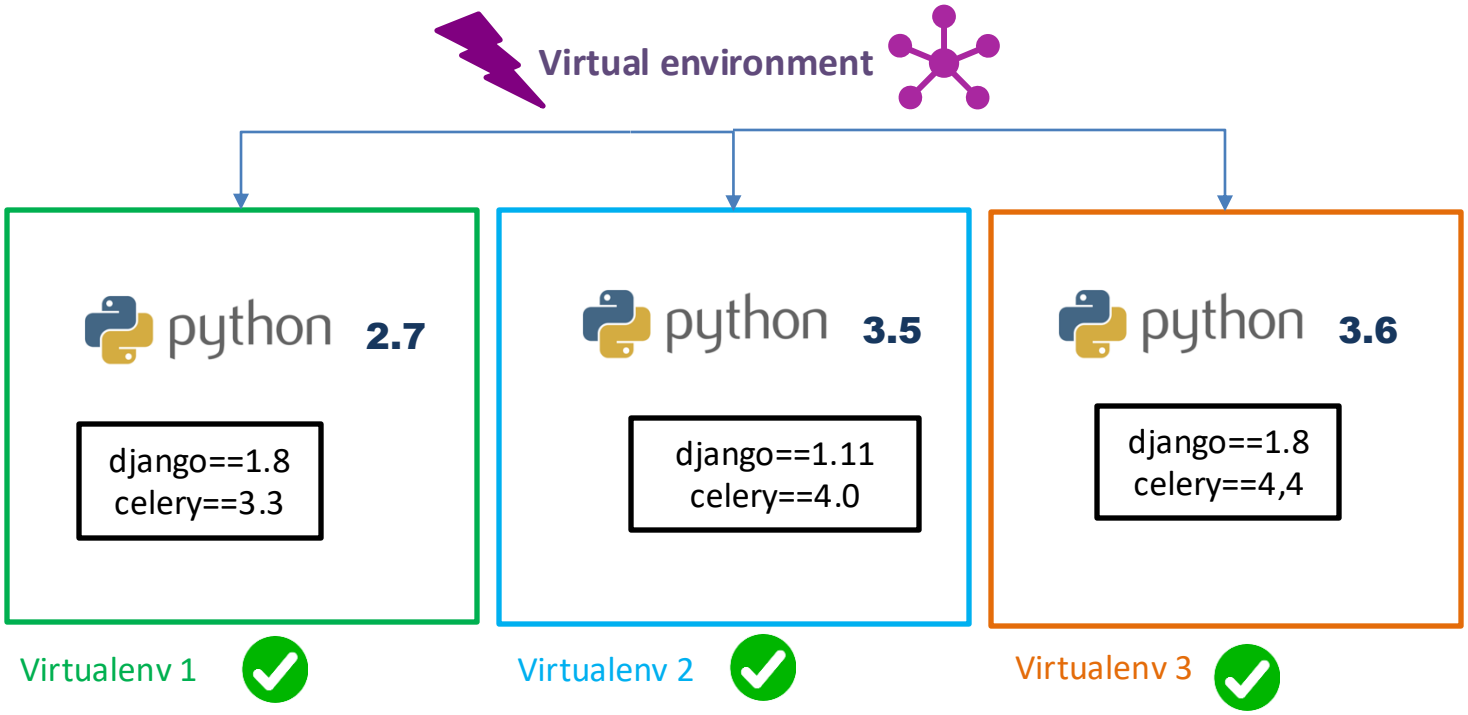
Virtual Environments

Why you need multiple Python environments



Virtual Environments

Why you need to create a new environment



What is a virtual environment : isolated workspace where you can install and use specific versions of Python and its libraries without affecting other projects or the system installation.

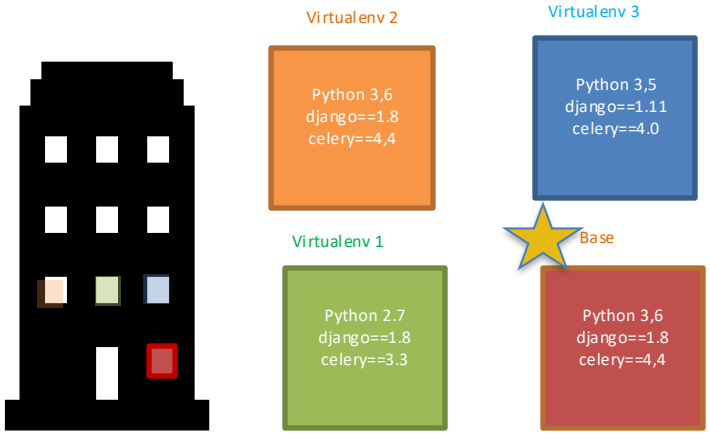
Rule of thumb: *One project → one conda environment*

Virtual Environments

Why you need to create a new environment

- **Isolation of Dependencies**

Think of your computer as a building, and a virtual environment as one room inside it. In that room, you can install the libraries you need. When you enter the room, everything is available. When you leave, it's as if nothing exists outside that room.

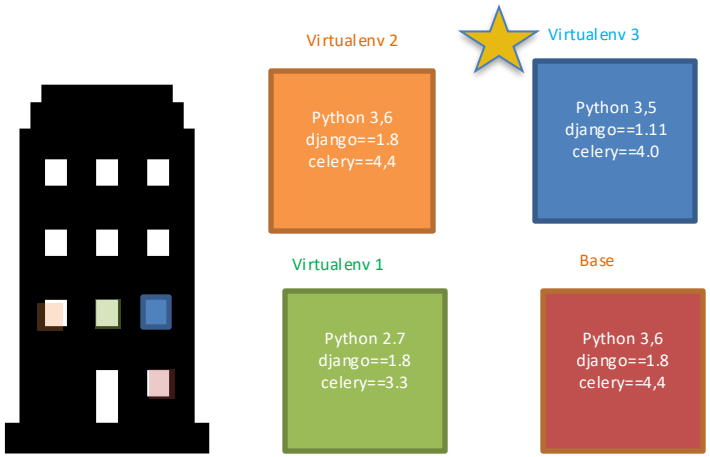


Virtual Environments

Why you need to create a new environment

- **Isolation of Dependencies**

Think of your computer as a building, and a virtual environment as one room inside it.
In that room, you can install the libraries you need. When you enter the room, everything is available. When you leave, it's as if nothing exists outside that room.

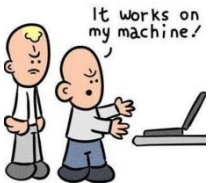


- **Reproducibility**

Environments also act like a **recipe** for your project.
They make sure that if your code runs today, it will still run the same way tomorrow or on someone else's computer.
Example: You list all the packages (and their exact versions) in a file, such as `requirements.txt` or **`environment.yml.`**

•A teammate (or your future self!) can recreate **the exact same setup** by following this recipe.

This avoids the classic "It works on my machine but not on yours!" problem.




```yaml
# environment.yml

name: Lab2
channels:
  - conda-forge
  - defaults
  - https://repo.anaconda.com/pkgs/main
  - https://repo.anaconda.com/pkgs/r
dependencies:
  - appnope=0.1.4=pyhd8ed1ab_1
  - asttokens=3.0.0=pyhd8ed1ab_1
  - brotli=1.1.0=h6caf38d_4
  - brotli-bin=1.1.0=h6caf38d_4
  - bzip2=1.0.8=hd037594_8
  - ca-certificates=2025.8.3=hbd8a1cb_0
  - comm=0.2.3=pyhe01879c_0
  - contourpy=1.3.3=py311h57a9ea7_2
  - cycler=0.12.1=pyhd8ed1ab_1
  - debugpy=1.8.16=py311ha59bd64_1
  - decorator=5.2.1=pyhd8ed1ab_0
  - exceptiongroup=1.3.0=pyhd8ed1ab_0
  - executing=2.2.1=pyhd8ed1ab_0
  - fonttools=4.59.2=py311h2fe624c_0
  - freetype=2.14.0=hce30654_1
```

Virtual Environments: How it works?

- Basic Anaconda Commands for a local project : [Documentation for anaconda](#).

Environments

```
# Create a new environment with Python 3.10
conda create -n myenv python=3.10

# Activate environment
conda activate myenv

# Deactivate environment
conda deactivate

# List all environments
conda env list

# Remove an environment
conda remove -n myenv --all
```

Packages

```
# Install a package
conda install numpy

# Install multiple packages
conda install numpy pandas scikit-learn

# Update a package
conda update numpy

# Remove a package
conda remove numpy

# List packages in current environment
conda list
```

Sharing / Reproducibility

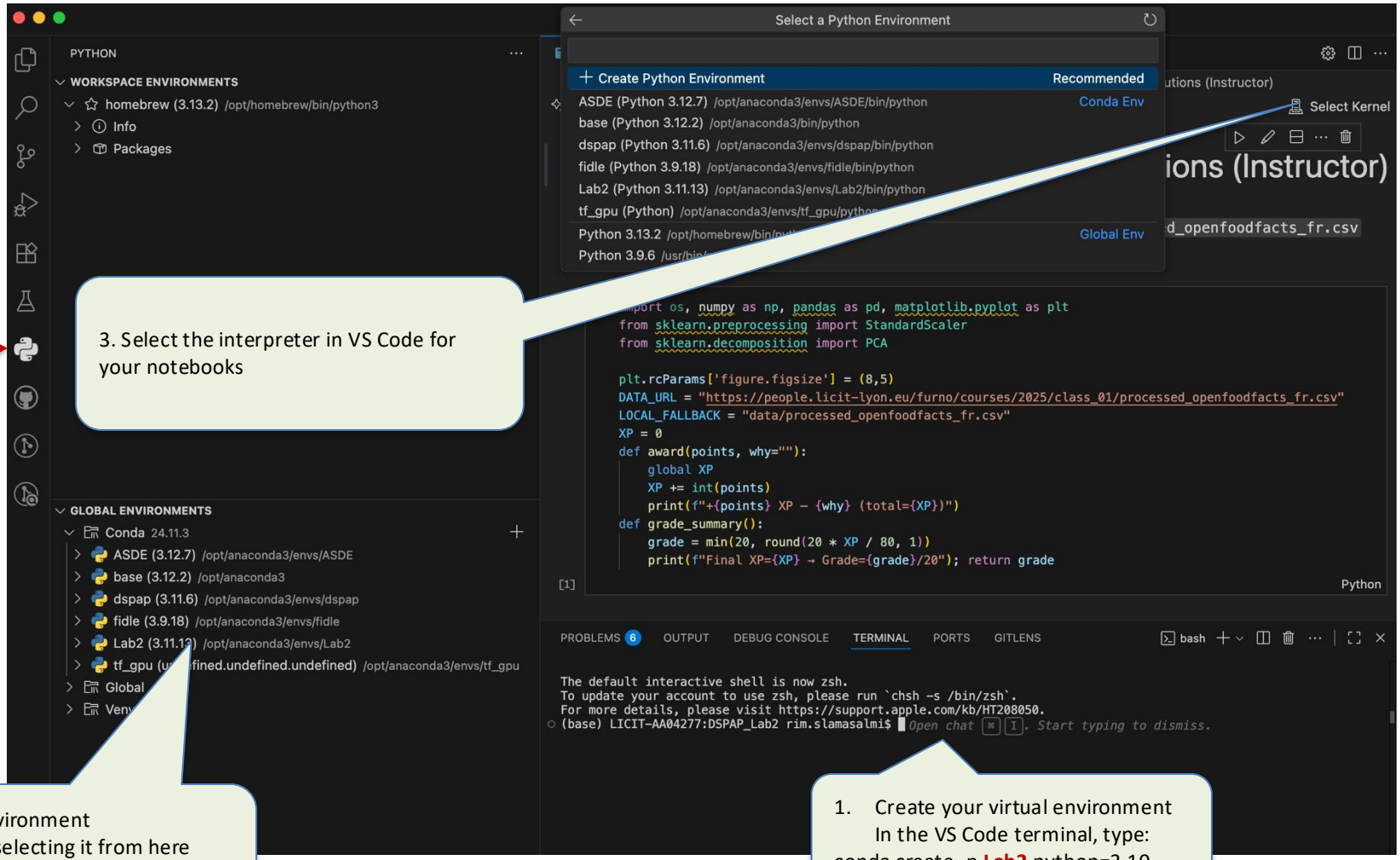
```
# Export environment to a file (flexible)
conda env export > environment.yml
# Create environment from a file
conda env create -f environment.yml

#  Export current environment to .txt (exact version)
conda list --explicit > spec-file.txt
#  Create a new environment from .txt
conda create --name myenv --file spec-file.txt
```

More commands

```
# Clone env
conda create --name myenv_copy --clone myenv


# Add conda-forge channel
conda config --add channels conda-forge


# Install package from specific channel
conda install -c conda-forge matplotlib
conda install -c pytorch pytorch torchvision torchaudio

# Check environment details
conda info

# Update conda itself
conda update conda
```

Virtual Environments

How it works in VS code?



3. Select the interpreter in VS Code for your notebooks

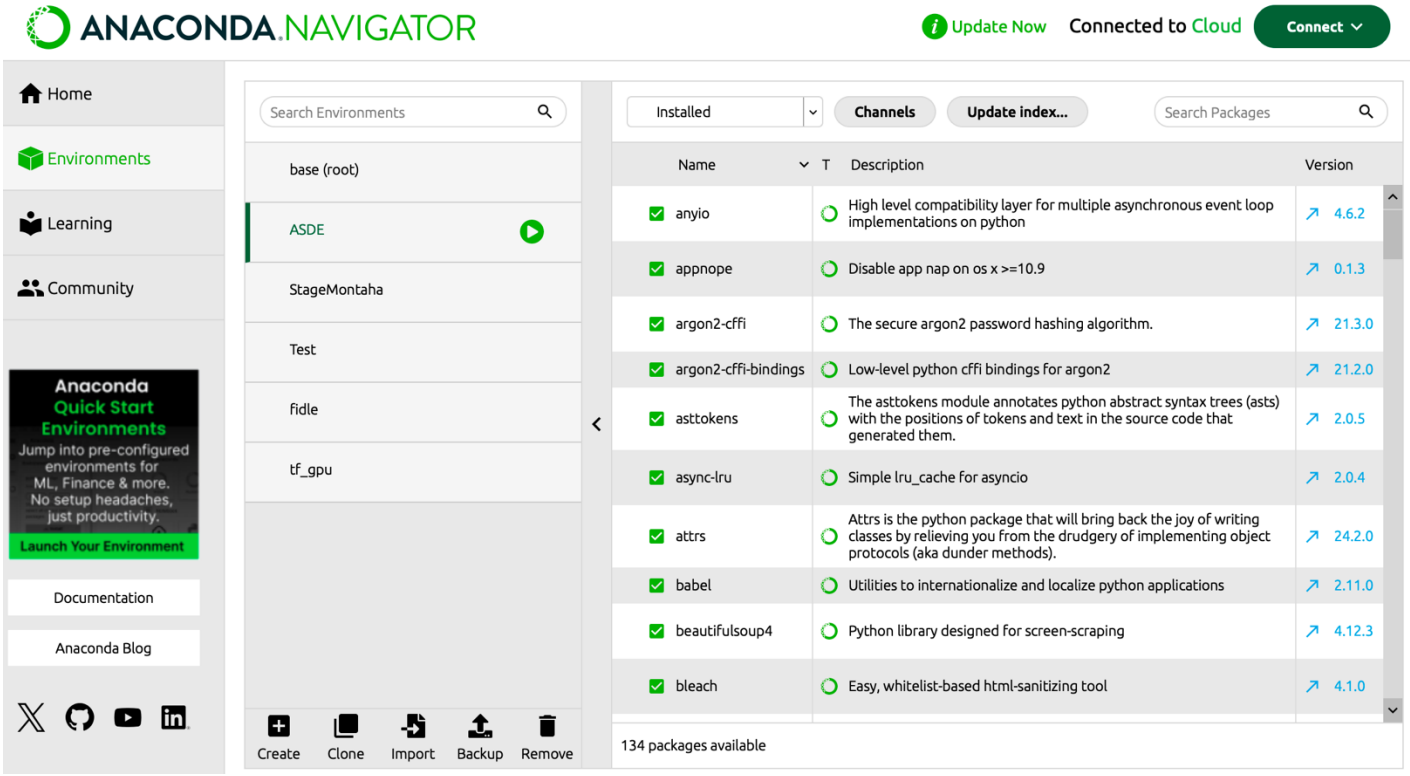
2. Activate the environment
Once created, by selecting it from here (or in with command line : conda activate Lab2)

1. Create your virtual environment
In the VS Code terminal, type:
conda create -n **Lab2** python=3.10

Virtual Environments

How it works in Anaconda navigator?

- Using the **graphical interface** with **Anaconda Navigator**



Virtual Environments
How it works Across Different Platforms?

In JupyterHub (ENTPE): JupyterHub

- Teachers/admins usually provide one or more **preconfigured environments**
- If permissions are granted by the admin, students may also **create their own virtual environments** in their home directory.

In Google Colab : https://colab.research.google.com/

- A separate environment is already provided for each notebook.
- The machine is reset every time you reconnect → installed packages are temporary.

👉 **Note**

Besides **Conda**, it is also possible to create a virtual environment with **`venv`**, which comes built into Python.

Create a virtual environment: python -m venv myenv

Activate it:

- On **Windows**:myenv\Scripts\activate
- On **macOS/Linux**: source myenv/bin/activate

To go further with venv: https://docs.python.org/3/tutorial/venv.html?utm_source=chatgpt.com

DEMO: Python Environment in VS Code with Anaconda

1. Open Terminal in VS Code

•Go to **View** → **Terminal**.

2. Write the conda commands to:

- Create a virtual environment named **Lab2_yourName** with **Python 3.11** and the libraries: numpy, pandas, matplotlib, pyyaml, and scikit-learn.
- Create another environment named **Lab2_test** with **Python 3.13** and **NumPy 2.3.3**.
- Verify the installed library versions in each environment.
- Switch from one environement to another
- Finally, remove the **Lab2_test** environment.

```
conda create -n Lab2_yourName python=3.11 numpy pandas matplotlib pyyaml scikit-learn ipykernel
conda create -n Lab2_test -c conda-forge python=3.13 numpy=2.3.3 ipykernel
conda remove -n Lab2_test --all
```

4. Save Environment of Lab2 _yourName as .yml and .txt and explore them to explain the difference

conda env export > Lab2.yml
conda list --explicit > Lab2.txt

5. If not yet done for the next courses : Create a Virtual Environment dspap from t

conda env create -f dspap.yml

dspap.yml

```
#create with conda env create -f dspap_advanced.yml
name: dspap
channels:
  - conda-forge
  - defaults
dependencies:
  - python=3.13
  - ipykernel
  - nodejs
  - jupyterlab
  - numpy
  - scipy
  - statsmodels
  - notebook
  - matplotlib
  - seaborn
  - pandas
  - pandas-datareader
  - scikit-learn
  - python-graphviz
  - graphviz
```

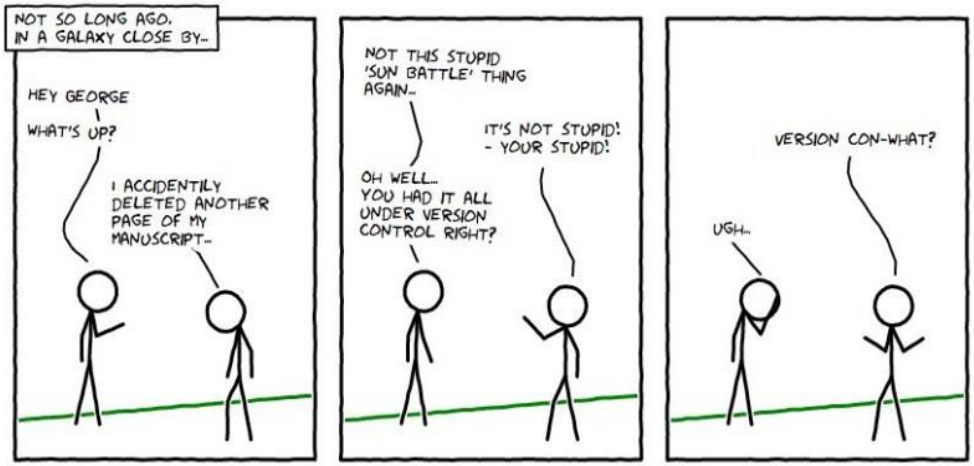
- The name of the environment will be the one specified **inside the yml**
- ⚠ If you want to override the name directly from the command line, you can do:
  - conda env create -f demo_env.yml -n newname

VERSION CONTROL AND COLLABORATION

Why Good Practices Matter

- **Organize the project**
  - Keep files and folders tidy
  - Separate data, code, and results
- **Write readable code**
  - Use clear names for files and functions
  - Follow a consistent style
- **Check and fix**
  - Test your code step by step
  - Handle errors instead of ignoring them
- **Explain your work**
  - Add short, useful comments
  - Write a simple guide (README)
- **Ensure reproducibility**
  - Make it easy for others to run your project
  - Avoid mixing tools and libraries between projects and Give each project its own space
- **Work as a Team and share your work**
  - **Share** work regularly
  - **Track** who does what
  - **Keep history** of all versions
- **Reflect on AI and Ethics**
  - **Follow an ethical approach during the project**
  - **Respect universal principles and rules**
  - **Apply an "ethical filter" to decisions and results**

Aaah… Computers



Solution?

•Regular backups with a history of changes

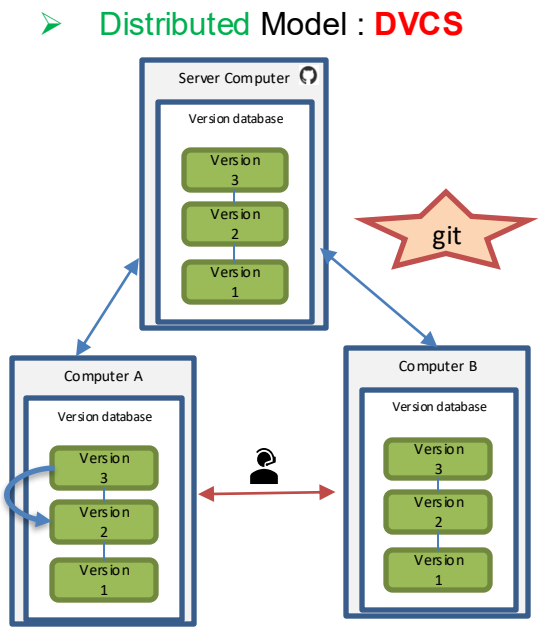
- project_final
- **Project_final_final**
- project_final_final_reallyfinal_final
- project_final_final_reallyfinal_final 09-25
- **…**

- Spending all your time sending (and resending) emails!

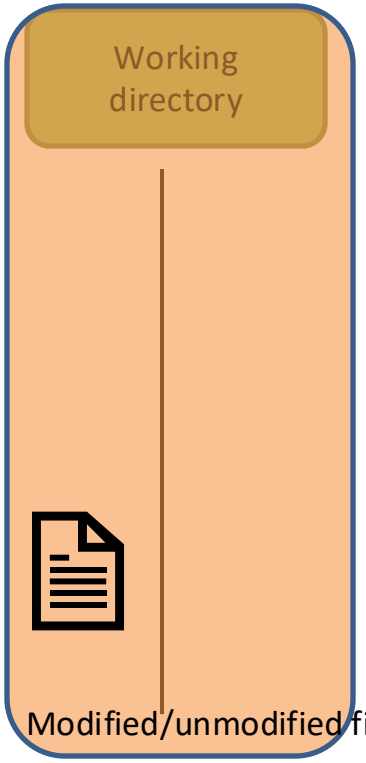
- *What happens if you are working on a project and…*
  - you lose your laptop but need your work?
  - you want to go back to an older version of your code/report?
  - two people edit the same file at the same time?
  - You want to know who added what and when?

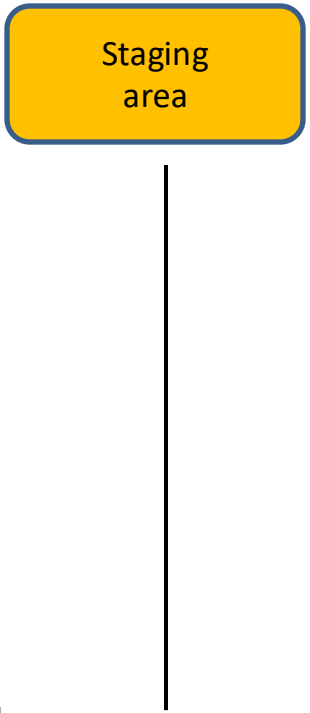
Version Control and Collaboration: Solution

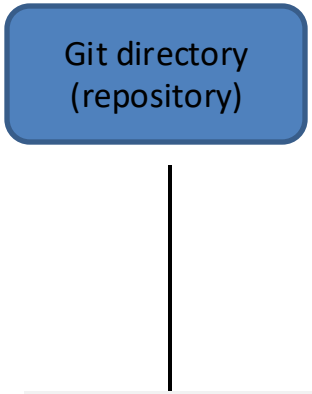
- *Git is a distributed version-control system*
  DVCS
- **Key features**
  - Keeps a history (who and when) of modifications without duplicating backup folders
  - Saves data (with its history) on remote computers
  - "Automatic" merging of files edited by multiple people
- **Git**
  - 2005: created for the development of the Linux kernel by Linus Torvalds
    - >20 million lines of code
    - ≈14,000 developers



Git workflow: The three states







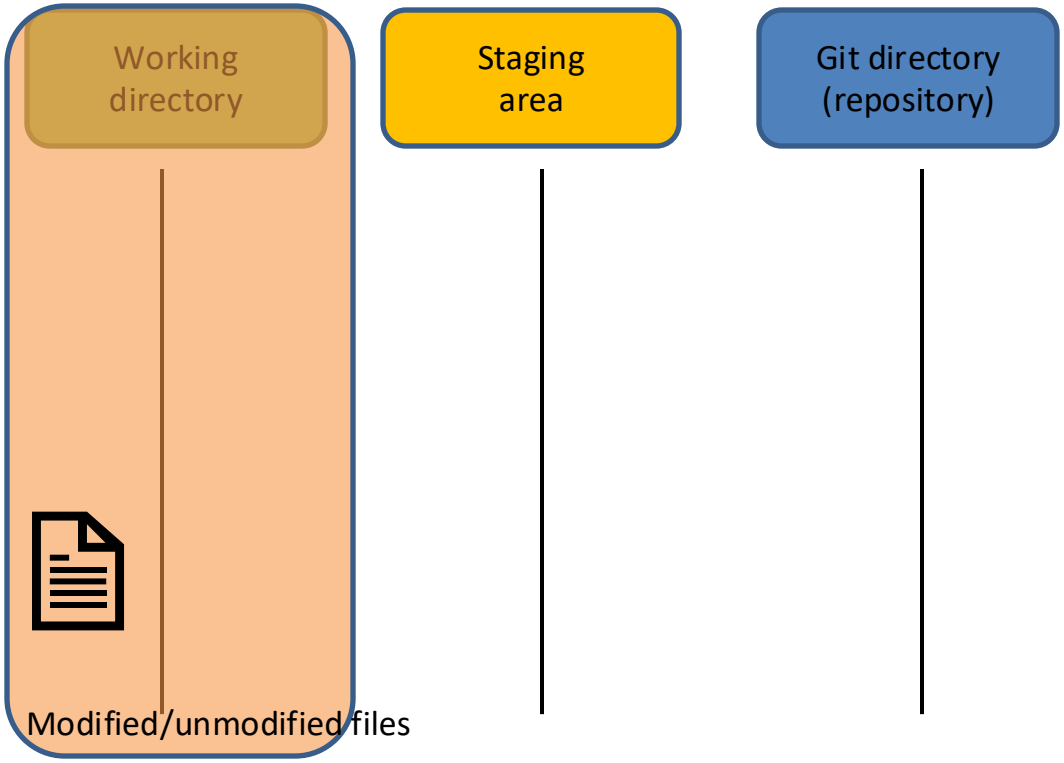
In a Git repository your file can reside in three main states:

- Modified
- Staged
- Committed

What does this mean?

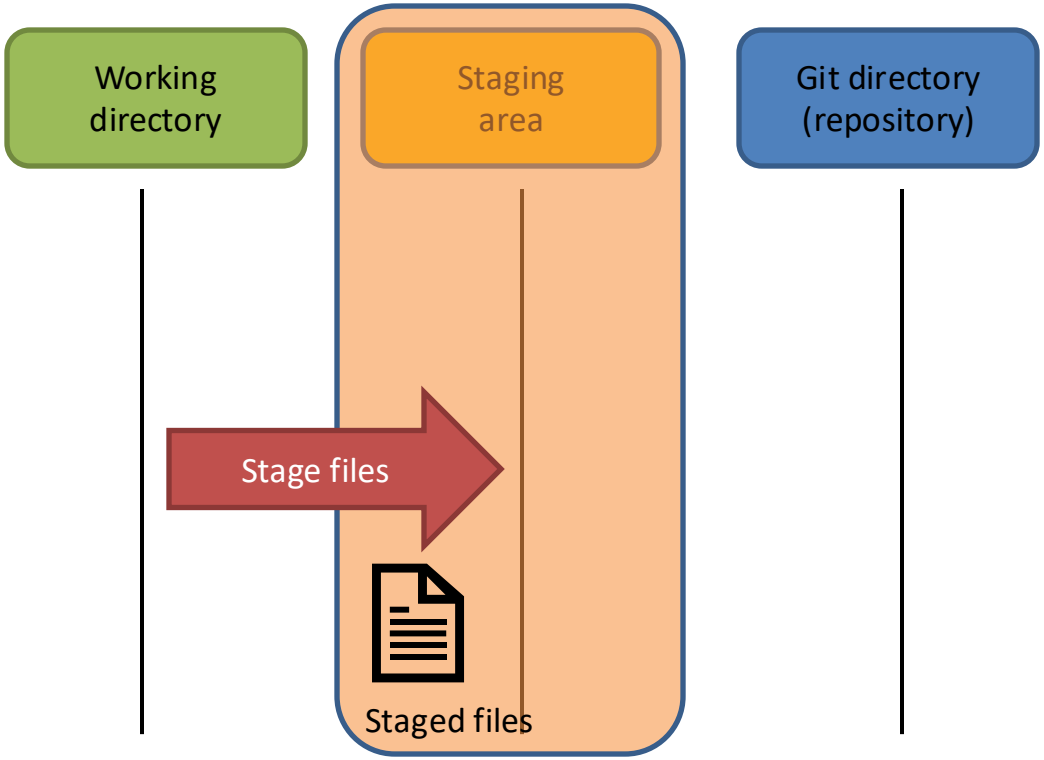
- You modify files in your working directory

Git workflow: The three states



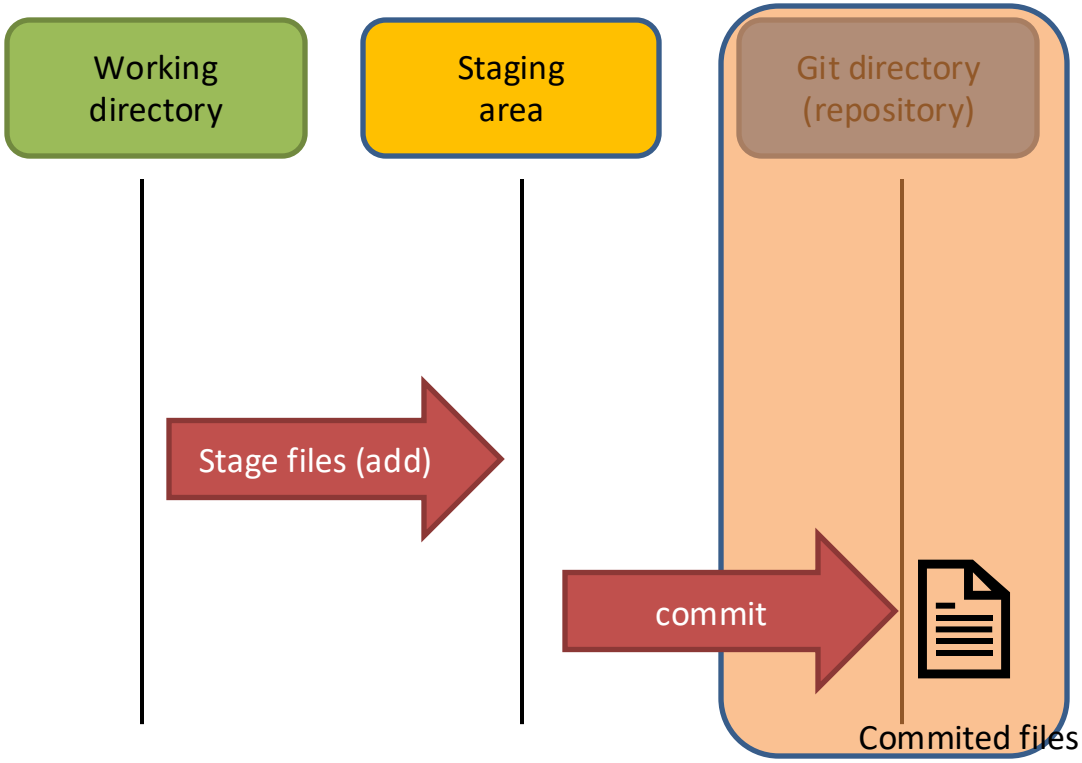
- You modify files in your working directory

Git workflow: The three states



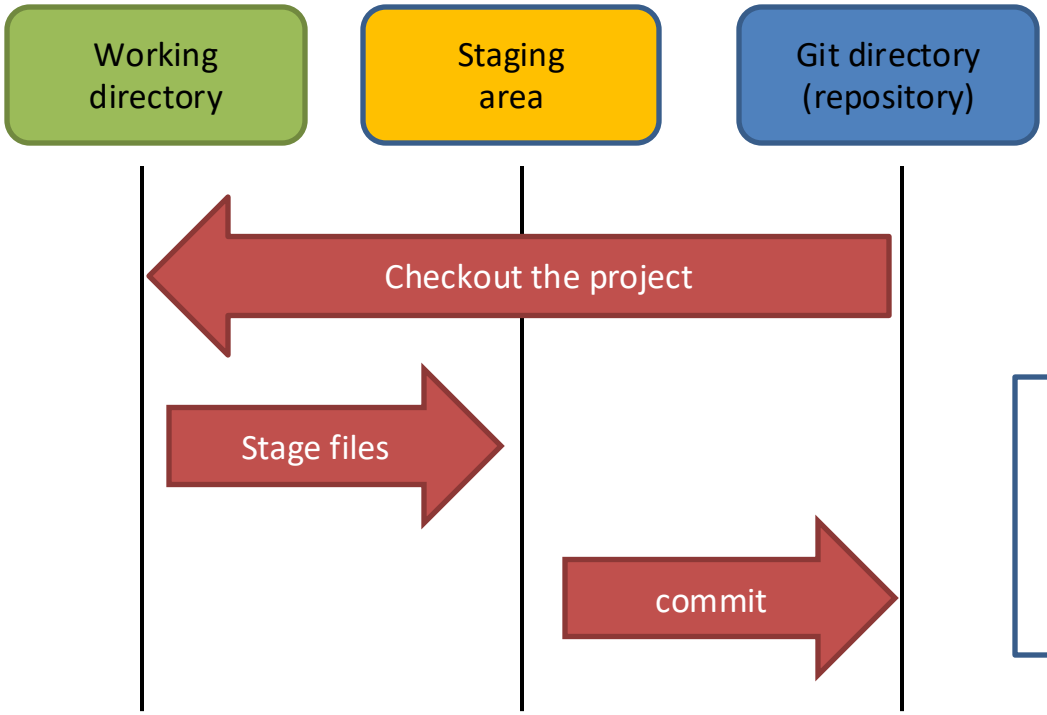
- You stage the files, adding snapshots of them to your staging area

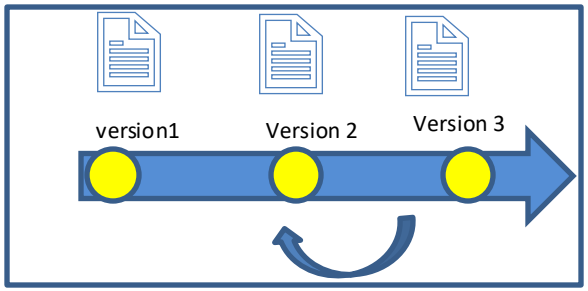
Git workflow: The three states



- You do a commit that stores snapshots permanently to your Git directory

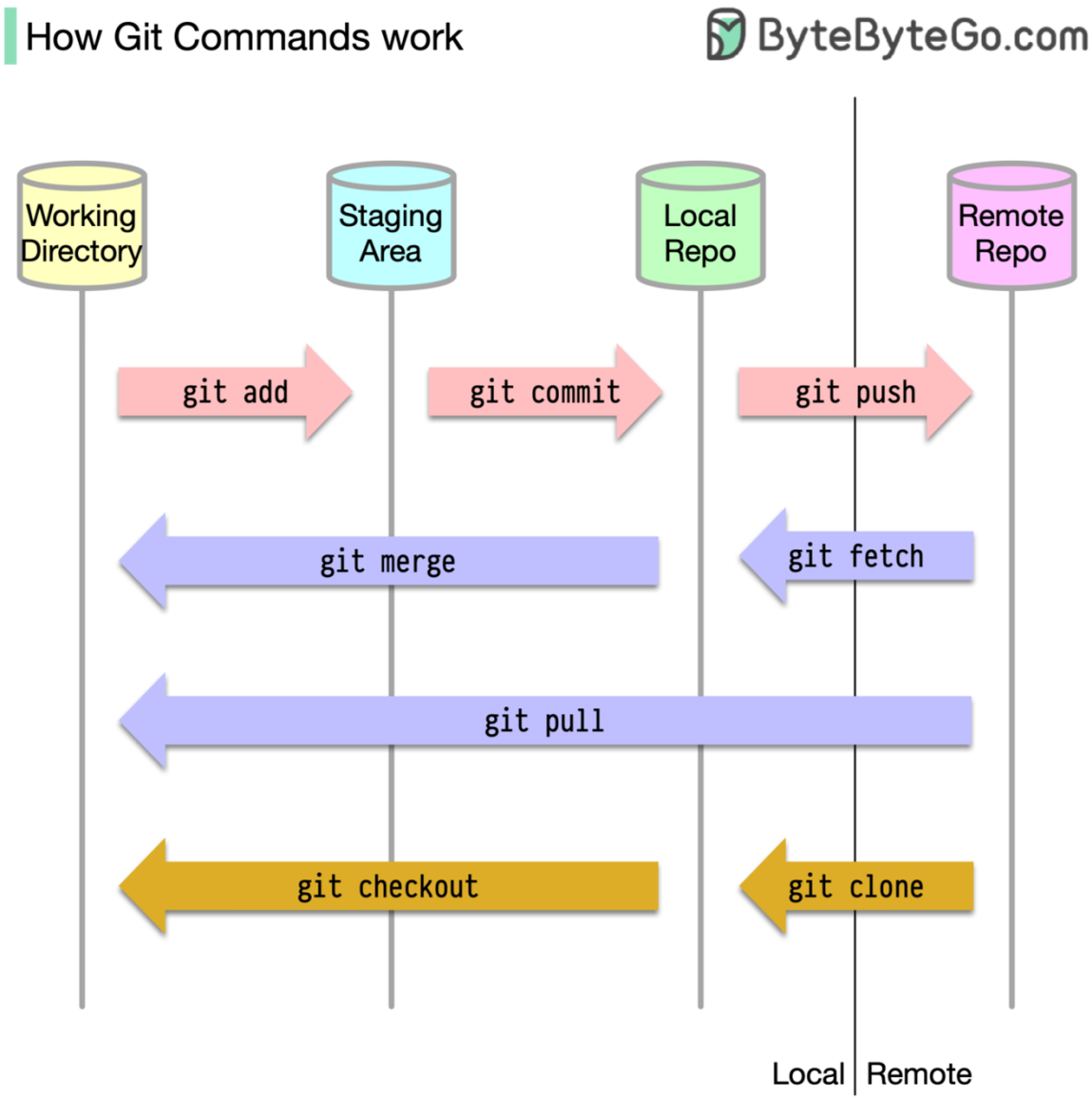
Git workflow: The three states





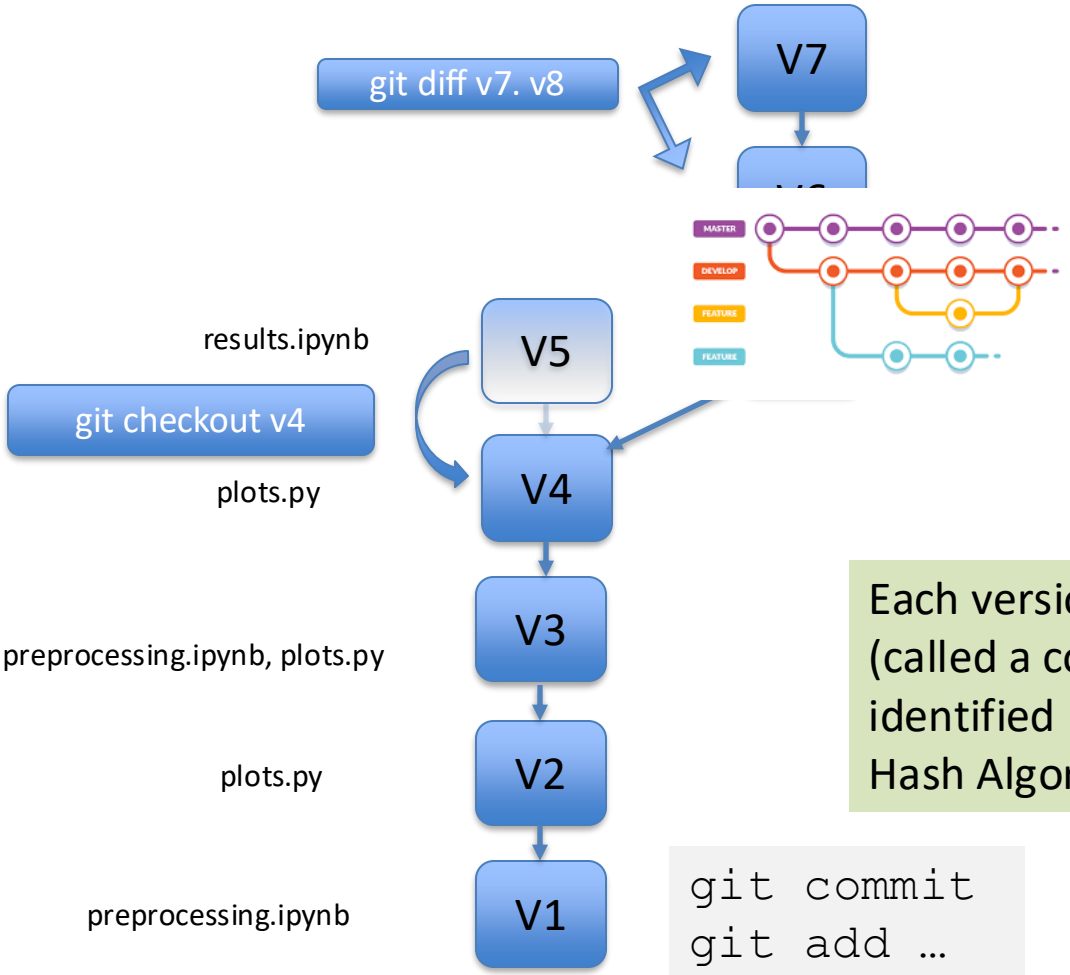
- Then, you can checkout any existing version, make changes, stage them and commit

How git works



⚠ GitHub refuses any file larger than **100 MB** in a standard repository.

Version Control and Collaboration: A simple story



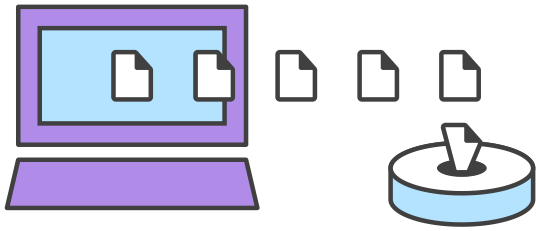
A history at the granularity of the ~~file~~ project file

Each version of the project (called a commit or revision) is identified by a SHA1 (Secure Hash Algorithm)

```
git commit
git add …
```

Ignore ?

- If you have some **folders or files you don't want to track** in your project (because they are temporary, generated, or large data that should not go into version control):
  - > you can tell Git to **ignore them**.



⚠ GitHub refuse **tout fichier >100 Mo** dans un dépôt classique.

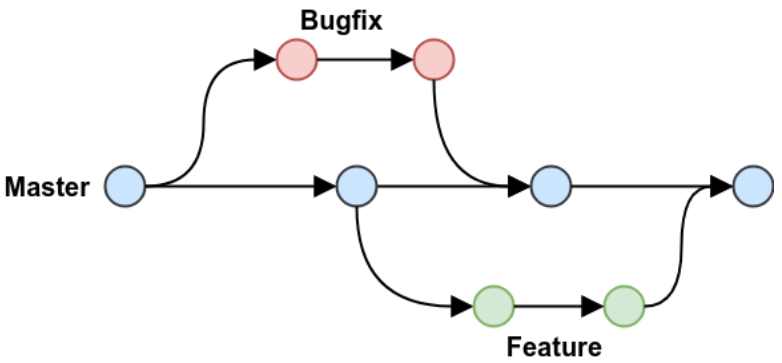
- To do this, you create a special file called .gitignore in the root of your repository.

- Inside .gitignore:
  - you list the patterns of files/folders you want Git to skip.

```
#.gitignore
# Ignore all log files
*.log

# Ignore a specific folder
/temp/

# Ignore Python cache
__pycache__/
```

Branching

- When several people collaborate:
  - Each person can **create their own branch** to work independently.
  - No one touches the stable main branch until their work is tested and merged.
  - This avoids conflicts and keeps the main project clean.
- Tuto:https://code.visualstudio.com/docs/sourcecontrol/overview#_branches-and-tags



Common Reasons to Create a Branch

✨ Developing a **new feature**

🐞 Fixing a **bug**

🧪 **Experimenting** with ideas safely

👥 **Collaborating** with teammates without interfering with each other

Merge conflicts

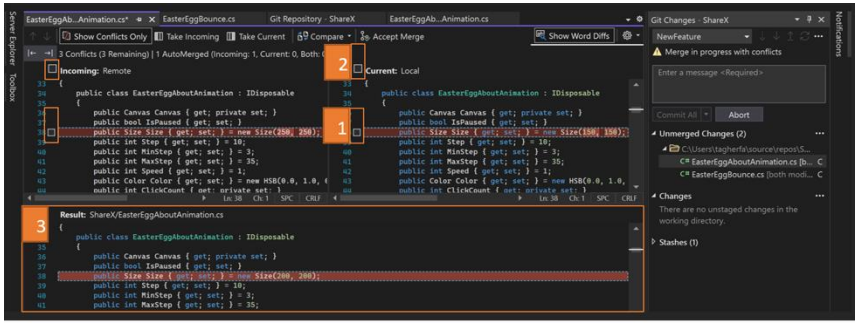
⚠ Merge Conflicts in Git

- When two branches change the **same part of a file**, Git can't decide which version to keep.
- **How Git shows conflicts**

```
<<<<<<< HEAD:index.html
<div id="footer">todo: message here</div>
=======
<div id="footer">
  thanks for visiting our site
</div>
>>>>>>> SpecialBranch:index.html
```

branch 1's version

branch 2's version



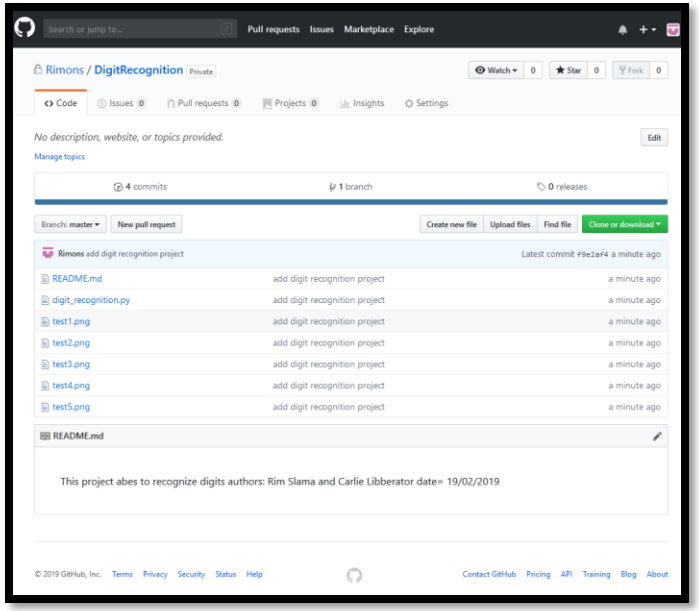
- 🔧 How to resolve:
- Tuto: https://code.visualstudio.com/docs/sourcecontrol/overview#_branches-and-tags
  - **Find all conflict markers** (<<<<<<<, =======, >>>>>>>).
  - **Decide what to keep**: your version,the other version, or a combination of both.
  - **Remove the conflict markers** after editing.
  - Save the file and run:

Interaction with remote repo

- ☁ **Online Storage (Remote Repositories)**
- **Tuto:** https://code.visualstudio.com/docs/sourcecontrol/github

  - Online storage means keeping a copy of your Git repository on a remote server (e.g., **GitHub, GitLab, Bitbucket**).
  - It acts as a **backup** and a **collaboration hub**.
  - Developers can **push** their changes to share them, and **pull** updates from others.

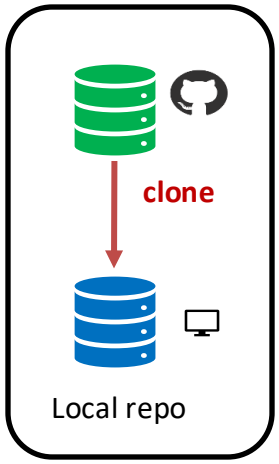
GitHub



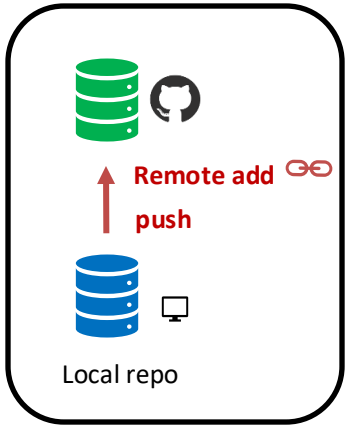
Github https://github.com/

Pulling and pushing

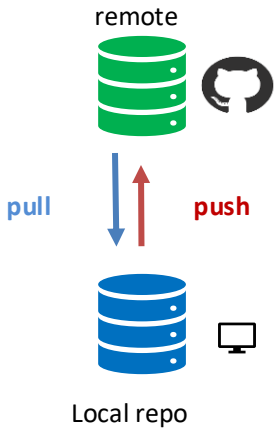
- Clone from remotes



- Push existing local repo to remote

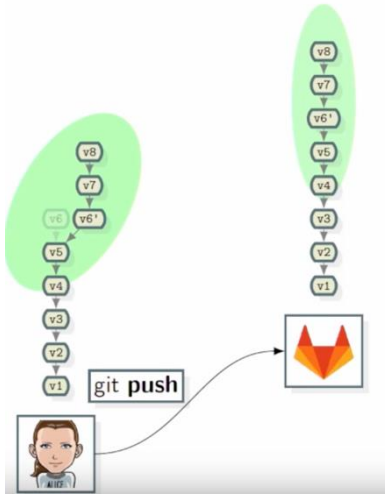


- Pulling and pushing



you can see changes remotely:
https://github.com/MunGell/awesome-for-beginners?utm_source=chatgpt.com

Stop tracking a file :
git rm --cached config.yaml



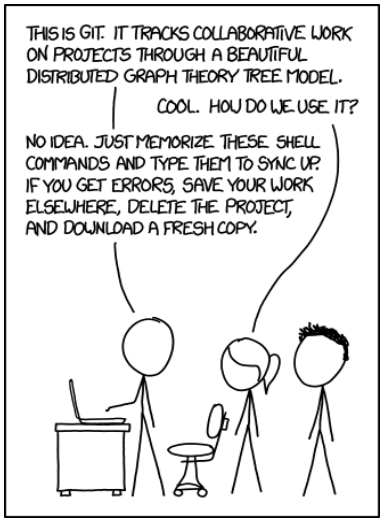
How to use git ?

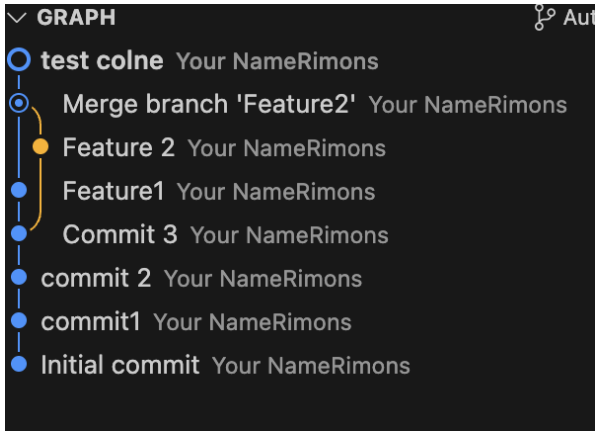
Command Line

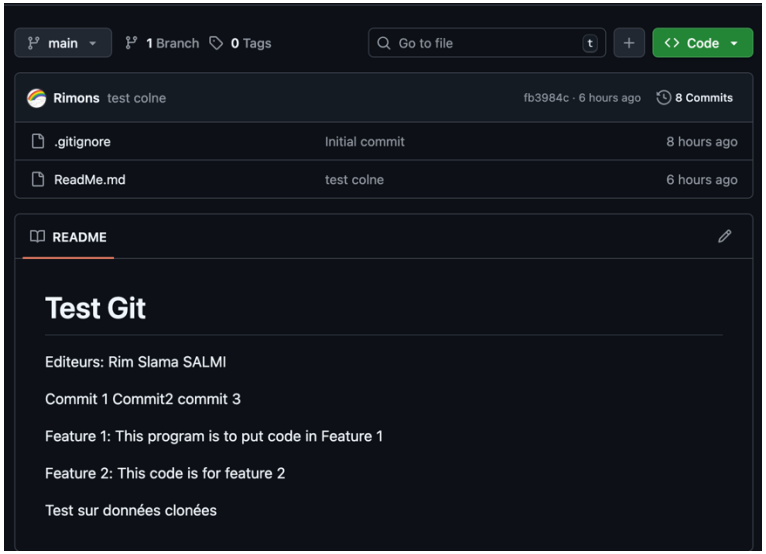
- Type commands directly (git status, git commit, git push).
- Most powerful, shows what's really happening.
- All commands and examples can be found here: w3schools

Interfaces/GUI Tools

- Built into tool or/and *plugins* (VS Code, PyCharm, JupyterLab, etc.)
- apps (GitHub Desktop, Sourcetree).







DEMO

Mini Git Demo Lab

1. Setup

- Create folder TestGit
- Add ReadMe.md → title: # Demo for git use

2. Commits

- Add 'Change 1' → Commit 1
- Add 'Change 2' → Commit 2
- Add 'Change 3' → Commit 3

3. **Undo**: Undo last commit (revert/reset)

4. Remote

- Connect to GitHub repo
- Push main branch

5. Branching

- Create: Feature1, Feature2, Feature3
- Delete Feature3

6. Feature Work

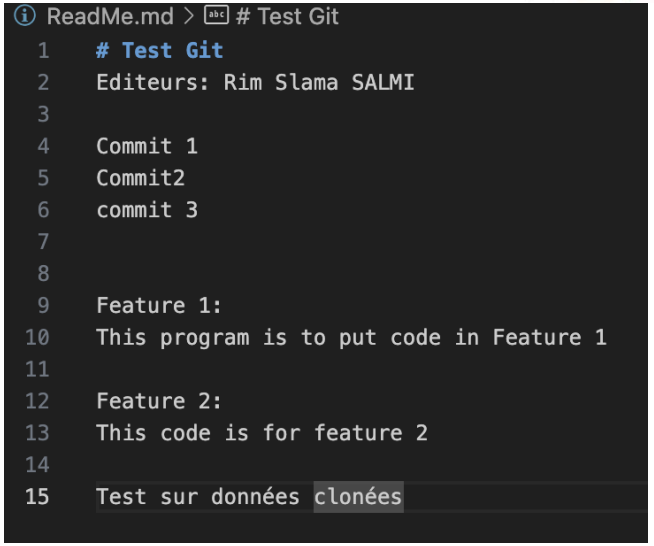
- Switch → Feature1 → add 'Feature 1' → commit
- Switch → Feature2 → add 'Feature 2' → commit

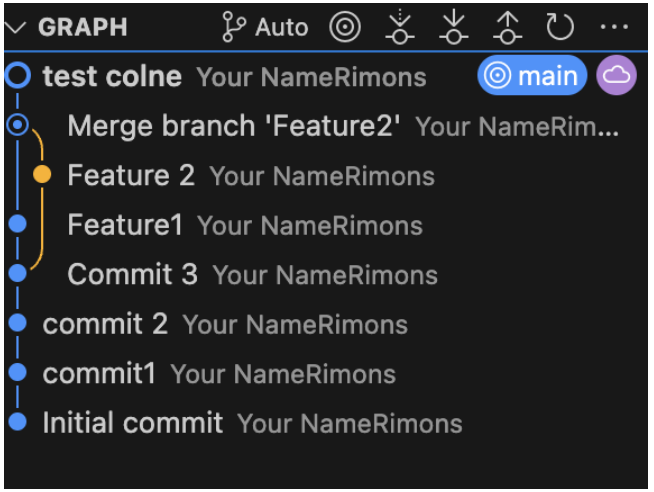
7. Merge & Conflict

- On main: merge Feature1
- On main: merge Feature2 → ⚠ cause conflict
- Resolve in editor → choose change(s) → commit

8. Clone & Contribute

- Clone repo elsewhere
- Make a new commit

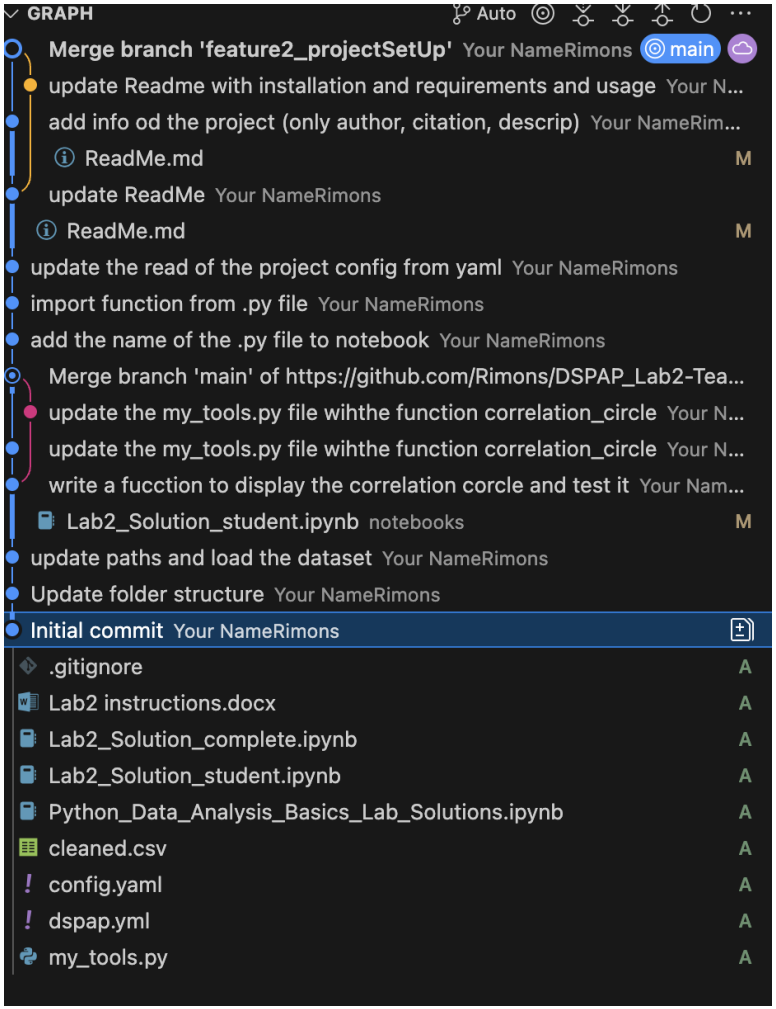




Practice

Instructions:

- Go to **Moodle**, class **2**, and download the ZIP file: **Lab Material: DSPAP_Lab2 Student**.
- Unzip the file and place it in a folder of your choice (e.g., Documents/, DSPAP_Labs/, or another working directory).
- **Rename the folder** by adding your name, for example: DSPAP_Lab2_Student_JohnDoe
- Open the folder in **VS Code**.
- Work on the notebook **Lab2_student.ipynb** inside VS Code.



Other ressources

- Other ressources: https://learngitbranching.js.org/
- Installation : https://git-scm.com/downloads
- conda install git
- https://www.atlassian.com/git/tutorials/setting-up-a-repository
- command lines: https://www.w3schools.com/git/git_branch.asp?remote=github

AI AND ETHICS: REFLECTIONS FOR A PROJECT

Why Good Practices Matter

- **Organize the project**
  - Keep files and folders tidy
  - Separate data, code, and results
- **Write readable code**
  - Use clear names for files and functions
  - Follow a consistent style
- **Check and fix**
  - Test your code step by step
  - Handle errors instead of ignoring them
- **Explain your work**
  - Add short, useful comments
  - Write a simple guide (README)
- **Ensure reproducibility**
  - Make it easy for others to run your project
  - Avoid mixing tools and libraries between projects and Give each project its own space
- **Work as a Team**
  - **Share** work regularly
  - **Track** who does what
  - **Keep history** of all versions
- **Reflect on AI and Ethics**
  - **Follow an ethical approach during the project**
  - **Respect universal principles and rules**
  - **Apply an “ethical filter” to decisions and results**

AI Ethics

- **What is Ethics?**

  Wikipedia:
  "Ethics is a philosophical discipline dealing with moral judgments, and its concept is therefore very close to that of morality. It is a fundamental reflection of every people in order to establish its norms, its limits, and its duties."

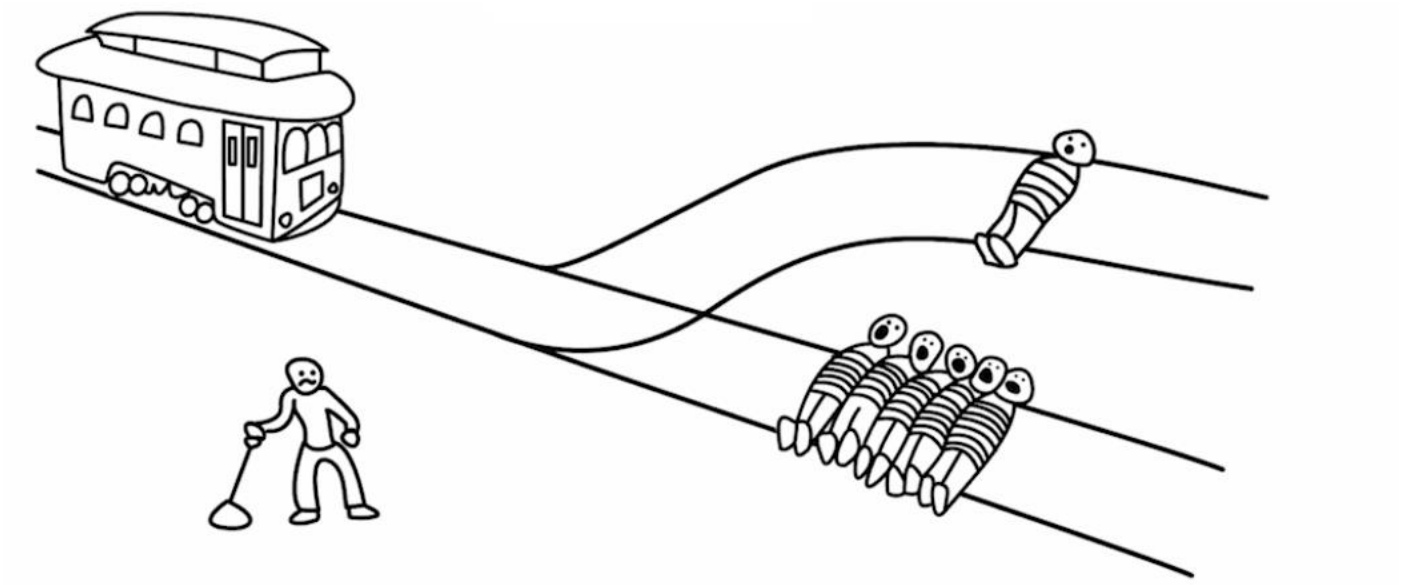
- **Ethical conduct:**
  - Respectful behavior
  - Altruistic behavior

AI Ethics

- **What is Ethics?**
  - Ethics is a discipline concerned with moral principles, values, and judgments that guide human behavior.
- **What is an Ethical Dilemma?**
  - An ethical dilemma occurs when there is a conflict between two or more important ethical values.
- **Example:**
  A doctor must decide whether to disclose confidential medical information to a patient's family or respect the patient's privacy.

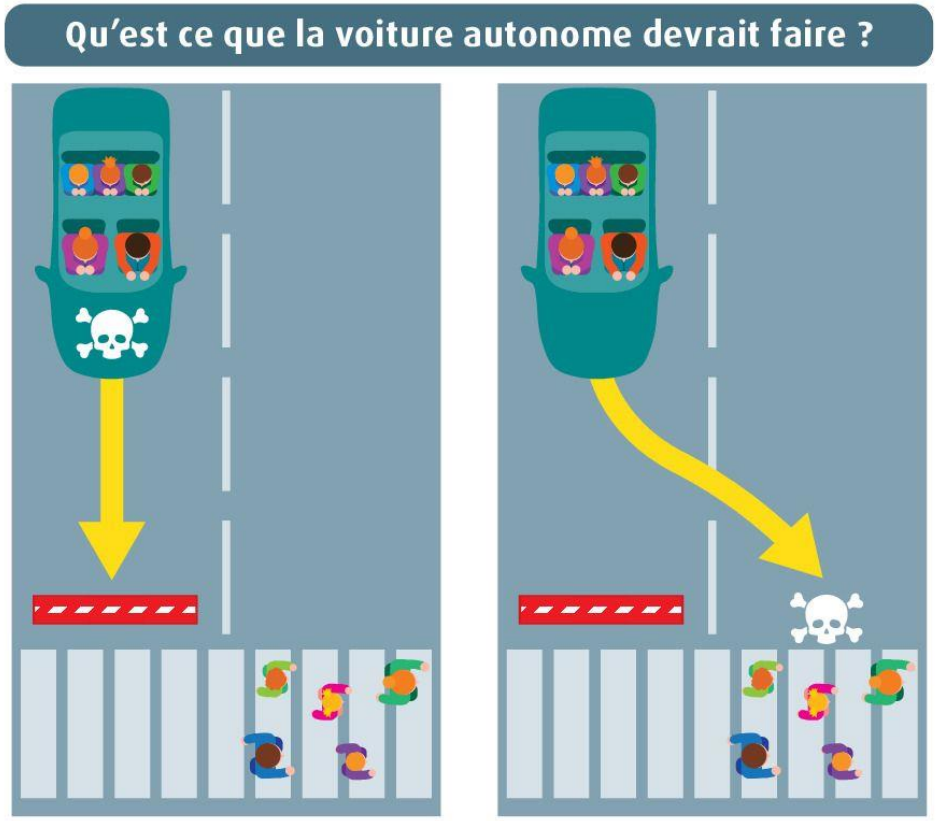
AI Decides on Absurd Trolley Problems

Watch here: https://www.youtube.com/watch?v=1boxiCcpZ-w



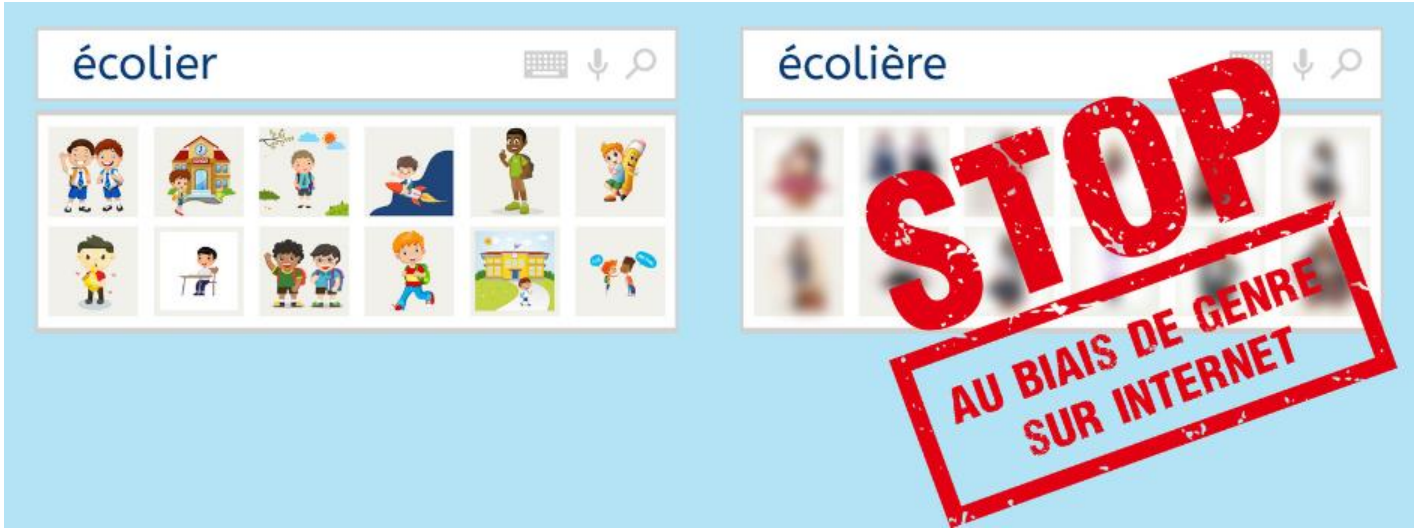
AI Ethics

- Autonomous Car



AI Ethics

- *AI Bias: Search for “schoolboy” and “schoolgirl” on Google*



AI and Art

- **346 Rembrandt paintings** were analyzed pixel by pixel and scaled using deep learning algorithms to create a unique database.
- Every detail of Rembrandt's artistic identity was captured and used as the foundation for an algorithm capable of producing an unprecedented masterpiece.
- To bring the painting to life, a 3D printer recreated the brushstroke texture and paint layers on the canvas, resulting in a stunning outcome that could fool even the best art experts.
- **But who is the author?**
  The company behind the project, the engineers, the algorithm… or Rembrandt himself?

- **Jason Allen** won **first place** at the Colorado State Fair with an AI-generated image.
  In response, many artists voiced their anger.
  Vanity Fair article

Case Study – Reflection Process for an AI Project

- 🔍 **Understand the Situation**
  - **Context & Objective**: Define the setting, goals, and constra
  - **Key Facts**: Identify essential parameters shaping the problem.
  - **Problem Reformulation**: Restate the issue clearly to ensure shared understanding.
  - **Data Integrity**: Verify provenance, representativeness, absence of bias, and ethical collection.
  - **Accountability**: Clarify ownership of the model and responsibility for outcomes.

- **Generate Solutions**
  - Brainstorm all possible options (without immediate judgment)

- **Identify Stakeholders**
  - List all individuals or groups affected by each solution

- **Assess Impacts**
  - For each solution: analyze consequences for stakeholders
  - Weigh pros and cons, and propose mitigation for drawbacks

- **Apply the Ethical Filter**
  - Does the solution respect people, laws, and shared values?
  - Is it altruistic, fair, transparent, and free of harm?
  - Apply ALTAI as a practical ethical filter.

- **Make a Decision**
  - Go / No Go decision based on collective discussion
  - Ensure inclusiveness, open debate, and genu consensus

- **Final Ethical Check**
  - Am I at peace with this decision?
  - Are consequences balanced, responsible, and empathetic?
  - Would I accept this decision if I were in others' place?
  - Would I be proud to justify it publicly or to future generations?

Example Case Study – Predictive Maintenance for Public Buses

Context: A city introduces an **AI system to predict bus breakdowns** by analyzing engine data, GPS logs, and sensor readings. The goal is to improve reliability and reduce costs.

1. Understand the Situation

Context & Objective: Improve service reliability, reduce downtime and maintenance costs.

Key Facts: AI predicts failures based on sensor and historical data.

Problem Reformulation: How to ensure predictive maintenance **benefits all users fairly** and doesn't compromise safety?

Data Integrity: Sensor data may be noisy, biased toward certain bus types.

Accountability: Who is responsible if AI misses a failure → the operator, the vendor, or the city?

2. Generate Solutions

Solution	Role of AI	Role of Human
Do nothing	No role	Plans and executes manually
Decision support	Analyzes, alerts, advises	Decides and schedules, executes repair
Hybrid	Proposes and schedules automatically	Validates or rejects before execution
Fully automate	Analyzes, decides, and directly schedules repairs	Executes only the scheduled work

3. Identify Stakeholders

Bus operators and drivers.; Maintenance staff.; Passengers.; City transport; authority.; AI vendor.

4. Assess Impacts

Solution	Stakeholder impacts	(+) Pros	(–) Cons	(*) Mitigation
Baseline	Operators overloaded, passengers delayed, authority high costs	Simple, known process	High downtime, costly, low safety	None
Decision support	Staff aided, passengers ↑ reliability, authority saves, vendor trusted	Better planning, ↑ Safety, some transparency	Human bias, alerts ignored	Staff training, audit trails
Hybrid	Staff + AI share responsibility, passengers safer, authority ↑ accountability, vendor credibility	Efficiency + oversight, High Safety, ↑ Fairness	Slower, role conflicts	Clear roles, logs, streamlined process
Fully automate	Staff bypassed, passengers faster service but safety risk, authority accountable, vendor liable	Fast, consistent scheduling	↓ Safety/Fairness if errors, unclear accountability	Fail-safe fallback, monitoring, liability rules

5. Apply the Ethical Filter

Safety first: never compromise passenger security.
Transparency: explain how AI decisions are made.
Fairness: ensure all bus types and districts get equal attention.

6. Make a Decision

Adopt **hybrid system**: AI assists, humans decide final scheduling.
Collective discussion with unions, passengers, and city authorities.

7. Final Ethical Check

Peace of mind? ✔ Yes, because safety is prioritized.
Balanced consequences? ✔ Yes, workers retrained instead of replaced
Would I defend it publicly? ✔ Yes, reliability improves and risks are managed.

Ethical actions a developer could take:

1. Data Responsibility

- **Minimize data collection**: only gather sensor and operational data that is relevant to maintenance (avoid unnecessary personal data, e.g., passenger info).
- **Ensure data privacy**: if any personal or location data is involved (e.g., GPS tracking of buses), anonymize or aggregate it before analysis.
- **Transparency in data use**: document clearly what data is collected, how it's used, and who has access.

2. Fairness & Non-Discrimination

- Check that algorithms do not indirectly disadvantage certain groups (e.g., if maintenance prioritization leads to poorer service in less affluent neighborhoods).
- Ensure **fair resource allocation** so predictive maintenance does not systematically favor certain routes or fleets.

3. Accountability

- Developers should provide clear **explainability** (e.g., dashboards showing which sensor triggered the alert).

4. Safety & Reliability

- Test thoroughly to prevent **false negatives** (failing to predict a breakdown that leads to passenger harm).

Case Study – Pedestrian Detection with Computer Vision

Context: A city installs **AI-powered cameras** at intersections to detect pedestrians and adjust traffic lights automatically. Objective: improve pedestrian safety and reduce accidents.

Task (20 minutes**):**

Work in groups of 3 and apply the **7-step ethical reflection framework** to this case.

1.Understand the Situation

1. Define context, objectives, key facts
2. Reformulate the problem clearly

2.Generate Solutions

1. Brainstorm at least 3 alternatives (e.g. full AI, operator, pedestrian control, hybrid).

3.Identify Stakeholders

1. List all groups affected (directly or indirectly).

4.Assess Impacts

1. Discuss positive and negative impacts for each solution.

5.Apply the Ethical Filter

1. Use fairness, safety, privacy, and transparency as lenses.

6.Make a Decision

1. Select one solution as your group's choice.

7.Final Ethical Check

1. Would you feel safe as a pedestrian?
2. Would you defend this decision publicly?

Case Study – Reflection Process for an AI Project

Catégorie	Ressource	Lien
Cadres institutionnels	ALTAI – Assessment List for Trustworthy AI (UE)	digital-strategy.ec.europa.eu
	NIST AI Risk Management Framework (US)	nist.gov
	UNESCO – Recommendation on the Ethics of AI	unesco.org
Outils pratiques	Microsoft Responsible AI Toolbox	responsibleaitoolbox.ai
	Documentation Toolbox (PDF)	microsoft.com
Évaluations d'impact	AI Now Institute – Algorithmic Impact Assessments	ainowinstitute.org
Guides IA générative	The Generative AI Ethics Playbook (2024)	arxiv.org/abs/2501.10383
Réflexions critiques & académiques	Atlas of AI – Kate Crawford (2021)	yalebooks.yale.edu
	AI Ethics & Governance in Practice (2024)	arxiv.org/abs/2403.15403
	Ethical Considerations in AI Courses (2017)	arxiv.org/abs/1701.07769
	Practical Ethics of Generative AI in Creative Processes (2024)	arxiv.org/abs/2412.03579

AI Ethics in transportation

Theme	Key Issues	Source / Link
Safety & Accountability	Who is responsible in case of an accident? End-to-end AI driving models often lack ethical reasoning and can behave unpredictably.	HiveData – AI drivers: death and taxes
Privacy & Data Governance	Autonomous vehicles collect massive amounts of data (video, lidar, GPS). How can we ensure privacy, security, and proper governance?	Numalis – Ethics of AI in Transportation
Moral Decision-Making	How should a vehicle be programmed to act in ethical dilemmas (e.g., protect passenger vs. pedestrian)?	Tech-Stack – AI in Transportation
Transparency & Public Trust	AI algorithms are often black boxes. How do we explain decisions so the public can trust them?	Tech-Stack – AI in Transportation
Surveillance & Fairness	Smart transportation systems monitor, track, and optimize flows. There is a risk of excessive surveillance or unequal service allocation depending on socio-economic areas.	Tech-Stack – AI in Transportation
Public Transport Privacy	AI systems collect sensitive passenger data (routes, payments, biometrics). How can we protect commuters from misuse?	Tech-Stack – AI in Transportation

Extra Resource – Video: Ethics and Risks of AI in Transportation (YouTube)

THANK YOU FOR YOUR ATTENTION

Contact: rim.slamasalmi@entpe.fr