

Installer un projet Python, et décrire son installation

Objectif : compléter un projet auquel il manque le fichier qui décrit son environnement, et la section de son README qui dit comment l'installer

`cours1/1h_recette/` `≈ 20'`

Le projet, dossier par dossier

Copier `depart/recette/` dans `travail/`, l'ouvrir dans l'éditeur, et dire à quoi sert chaque fichier avant d'en lancer un seul.

Ce que le projet contient

`pyproject.toml`

`data/`

`src/recette/`

`recette_a_la_main.py`, `recette_avec_tabulate.py`

`README.md`

Deux choses manquent, et les écrire est l'objet du TD : le fichier qui décrit l'environnement, et la section « Installation » du `README`.

Deux sortes de dépendances

Le programme importe trois noms que Python ne fournit pas. Deux viennent d'un dépôt public, le troisième est le projet lui-même.

Publiées, dans un dépôt

```
import markdown
from tabulate import tabulate
```

Déclarées dans `dependencies`, et posées par `conda install` depuis `conda-forge`.

Une bibliothèque qu'on écrit soi-même n'est nulle part tant que personne ne l'a publiée. C'est le cas ici, et c'est le cas ordinaire.

Le projet lui-même

```
from recette.calculs import adapter
from recette.tableau import GABARIT
```

Le code est sous `src/`, il n'est sur aucun dépôt, et s'installe depuis le dossier.

Un environnement d'essai

On part d'un environnement qui ne contient que Python, pour voir ce qui s'y trouve d'origine et ce qui n'y est pas.

	Ce qu'il faut faire	Ce que vous constatez
1	<code>conda create -n recette -c conda-forge python=3.12</code>	
2	<code>conda activate recette, puis conda list</code>	
3	<code>cd travail/recette, puis python recette_a_la_main.py</code>	

L'environnement actif se lit à deux endroits : en tête de l'invite du terminal, (`recette`), et en bas à droite dans la barre d'état. Pour en changer sans taper de commande, palette puis « Python: Select Interpreter ».

Résoudre les deux dépendances

Les bibliothèques publiées d'abord, le projet ensuite. Deux commandes, et le même message d'erreur entre les deux, pour deux causes différentes.

	Ce qu'il faut faire
4	<code>conda install -c conda-forge markdown tabulate</code> , puis <code>y</code>
5	<code>python recette_a_la_main.py</code>
6	<code>python -m pip install -e .</code>
7	<code>python recette_a_la_main.py</code> , puis ouvrir <code>recette.html</code>

Les deux messages sont le même, et les causes ne le sont pas : `markdown` manquait à l'environnement, `recette` est le code du projet, que rien n'avait encore rendu importable.

Écrire ce qui manque

Quatre commandes tapées ne se retrouvent pas. Écrites dans le projet, elles refont le même environnement sur une autre machine.

`environment.yml`, à créer

```
name: recette
channels:
  - conda-forge
dependencies:
  - python=3.12
  - markdown
  - tabulate
```

`README.md`, section à ajouter

```
## Installation

```bash
conda env create -f environment.yml
conda activate recette
python -m pip install -e .
```
```

Le fichier dit ce dont la machine a besoin, le `README` dit dans quel ordre s'en servir. Aucun des deux n'installe quoi que ce soit.

Refaire l'environnement à partir du fichier

Rien ne prouve qu'une documentation est juste tant que personne ne l'a suivie. On efface l'environnement, et on le refait en lisant la sienne.

| | Ce qu'il faut faire |
|----|--|
| 8 | <code>conda deactivate, puis conda env remove -n recette</code> |
| 9 | <code>conda env create -f environment.yml</code> |
| 10 | <code>conda activate recette, puis python -m pip install -e .</code> |
| 11 | <code>python recette_a_la_main.py</code> |

Les étapes 9 et 10 sont celles que vous venez d'écrire dans le `README`. S'il en manque une ici, c'est qu'elle manque aussi là-bas.