

Séance 4 : projet d'application 1

TD d'application des cours 1 à 3 : un script Python
construit pas à pas, versionné avec git

1re année géomatique

Contenu de la séance

Un TD d'application des cours 1 à 3, sur un seul projet : un script Python qui calcule les quantités d'une recette.

Parcours	Ce qu'on fait	Durée
Tous	présentation de la séance	5 min
Standard	TD 4a, partie A : créer le dossier du projet dans le terminal, l'ouvrir dans VS Code, créer le dépôt git	30 min
	TD 4a, partie B : construire un script Python, avec un commit git à la fin de chaque étape	75 min
Avancé	TD 4a, parties A et B, en 50 minutes	50 min
	exposé : découper un script en modules, décrire son environnement, l'installer comme une commande	10 min
	TD 4b : transformer le script en programme installable avec pip	55 min

Deux parcours

Chaque élève garde le parcours choisi au cours 3. Les deux parcours écrivent le même programme.

	Parcours standard	Parcours avancé
Au cours 3	TD 3e : une recette en Markdown	TD 3f : le programme <code>recette.py</code>
Au projet 4	TD 4a, en suivant le guide détaillé	TD 4a en 50 minutes, puis TD 4b
Dossier	<code>cours4/4a_recette/</code>	<code>cours4/4a_recette/</code> , puis les fichiers à compléter de <code>cours4/4b_paquet/</code>

Garder le dossier `travail/recette/` après la séance : les séances suivantes le reprennent.

TRAVAUX DIRIGÉS

TD 4a Un script Python, pas à pas
cours4/4a_recette/

≈ 30' + 75'

Parcours avancé · Un projet Python

Un programme réparti en modules, un environnement décrit par un fichier, une commande installée par pip

Un module, un import

Un fichier Python est un module. `import` exécute le module une fois, et ses noms sont ensuite utilisables dans le fichier qui l'importe.

`quantites.py`

```
import csv

VERS_US = {"g": ("oz", 1 / 28.3495), ...}

def adapter(ingredients, personnes_recette,
            personnes):
    ...

def convertir(ingredients, table):
    ...
```

`recette.py`

```
from quantites import VERS_US, adapter, convertir

def main():
    ...
    ingredients = adapter(ingredients, 4, 6)
    ingredients = convertir(ingredients, VERS_US)
```

Python cherche un module dans le dossier du fichier lancé, puis dans la bibliothèque standard (`csv`, `pathlib`) et dans les paquets de l'environnement actif.

Un environnement conda

Un environnement est un dossier qui contient un Python et des paquets. `conda activate` place ses dossiers en tête de `PATH`, et `python` désigne alors le Python de l'environnement.

Commande	Invite (base)	Invite (recette)
<code>which python</code>	le Python d'Anaconda, dans <code>anaconda3/</code>	le Python de l'environnement, dans <code>envs/recette/</code>
<code>which pandoc</code>	le pandoc d'Anaconda	le pandoc de l'environnement
<code>pip install ...</code>	installe dans <code>base</code>	installe dans <code>recette</code> seulement

Une commande désigne un exécutable cherché dans les dossiers de `PATH`, dans l'ordre (cours 1). Deux projets, deux environnements : les versions installées pour l'un ne changent pas l'autre.

environment.yml, conda et pip

`environment.yml` décrit l'environnement du projet. Versionné avec le code, il permet de recréer l'environnement sur un autre poste.

environment.yml

```
name: recette
channels:
  - conda-forge
dependencies:
  - python=3.12
  - pandoc
  - pip
  - setuptools
```

conda installe Python et des programmes qui ne sont pas écrits en Python, comme pandoc. pip installe des paquets Python : ici, le projet lui-même.

Dans le terminal, dans le projet

```
conda env create -f environment.yml → crée recette
```

```
conda activate recette → (recette)
```

```
pip install -e . --no-build-isolation → installe  
le projet
```

pyproject.toml : du fichier à la commande

`pyproject.toml` est la fiche du projet, lue par `pip`. `[project.scripts]` y déclare les commandes à installer.

<code>[project]</code>	la fiche du projet
<code>name = "recette"</code>	le nom du paquet, pour <code>pip</code>
<code>[project.scripts]</code>	les commandes à installer
<code>recette = "recette:main"</code>	la commande <code>recette</code> appelle <code>main</code> du module <code>recette</code>
<code>[tool.setuptools]</code>	où sont les modules
<code>package-dir = {"" = "src"}</code>	dans le dossier <code>src/</code>
<code>py-modules = ["recette", "quantites"]</code>	deux modules, un fichier chacun

`pip install -e .` installe le projet du dossier courant en mode modifiable : la commande appelle le code de `src/`, et une modification de ce code vaut sans réinstaller.

TRAVAUX DIRIGÉS

TD 4b Un programme installable — parcours avancé
cours4/4b_paquet/

≈ 55'

Ce qu'on rend

Le dossier du projet, avec son historique git.

Commande	Parcours standard	Parcours avancé
<code>git log --oneline</code>	huit commits, neuf avec le bonus	douze commits au moins
<code>git status</code>	« rien à valider » ; <code>sortie/</code> n'est pas listé	« rien à valider » ; <code>sortie/</code> n'est pas listé
l'aide	<code>python recette.py --help</code>	<code>recette --help</code>
le résultat	<code>sortie/crepes_6_US.csv</code>	le même, et <code>environment.yml</code> , <code>pyproject.toml</code>