

TD 1j – Installer un projet en lisant son README (facultatif)

Guide détaillé, étape par étape

Le projet trajet produit une vidéo commentée du trajet à pied entre la gare de Noisy-Champs et l'école, à partir d'un fond de carte et d'un fichier d'étapes. Le TD consiste à installer ce projet, qu'on n'a pas écrit, et à le lancer, en suivant uniquement sa documentation, puis à lire l'un après l'autre les fichiers que la commande produit. Le TD dure un quart d'heure, et il est facultatif : il se fait en séance si le temps le permet, ou seul ensuite.

Le TD nécessite l'invite de commandes d'Anaconda, où se tapent les commandes, un éditeur de texte (VSCode ou Notepad++) et un lecteur vidéo. L'installation télécharge Python, ffmpeg et ImageMagick, et nécessite une session réseau ouverte. Le guide suit la section « Installation » du README du projet ; le TD demande de suivre cette section soi-même, et le guide ne sert qu'en cas de blocage.

Étape	Objectif	Ce qu'on fait
1	trouver dans le README ce qu'il faut pour installer	copier le projet et lire son README
2	installer un projet par sa documentation	installer le projet par les commandes du README
3	relier la commande à ce qu'elle produit	lancer la commande, et lire ce qu'elle produit
4	exécuter une à une les commandes que le projet enchaîne	en option : les commandes magick et ffmpeg une à une, dans un notebook

Chaque étape commence par un encadré qui la résume. Ce que chaque étape fait constater est expliqué à la fin du guide, dans « Ce que le TD fait constater » : faire l'étape d'abord, et noter ce qu'on observe, avant de lire l'explication.

1 · Copier le projet et lire son README

À faire : copier `depart\trajet\` dans `travail\` ; lire le README du projet, et repérer sa section « Installation ».

À obtenir : `travail\trajet\` contient le projet, et on sait quelles commandes taper, dans quel ordre.

Les fichiers du TD

Ouvrir `info01\cours1\lj_trajet\`. Le dossier contient :

lj_trajet\	
├─ depart\	
│ └─ trajet\	le projet, tel qu'il est fourni
│ └─ README.md	sa documentation
│ └─ pyproject.toml	ce qu'est le projet, et ce dont il dépend
│ └─ environment.yml	l'environnement conda dans lequel il s'exécute
│ └─ data\	
│ └─ carte.png	le fond de carte
│ └─ carte.json	l'emprise et la taille de la carte
│ └─ etapes.csv	une ligne par point du trajet
│ └─ src\trajet\	le code, en quatre modules
├─ travail\	vide
├─ outils_video.ipynb	facultatif : les commandes une à une (étape 4)
├─ carte.py	le programme qui a fabriqué le fond de carte
├─ td_lj_trajet.pdf	la feuille du TD
└─ README.md	

`depart\` contient le projet fourni, et ne se modifie pas. `travail\` reçoit la copie du projet, qu'on installe et qu'on modifie. `carte.py` a été lancé une fois par l'enseignant, avant la séance, pour télécharger le fond de carte depuis OpenStreetMap : il n'est pas à relancer.

Copier le projet

Dans l'explorateur : sélectionner le dossier `depart\trajet`, Ctrl + C, ouvrir `travail\`, puis Ctrl + V.

Vérification : `travail\trajet\` contient `README.md`, `pyproject.toml`, `environment.yml`, `data\` et `src\`.

Lire le README

Ouvrir `travail\trajet\README.md` dans VSCode, et afficher son aperçu (Ctrl + Maj + V), ou l'ouvrir dans Notepad++. Le lire en entier avant de taper quoi que ce soit : il décrit le projet, les fichiers qu'il produit, puis son installation et son utilisation.

À noter : le nombre de commandes de la section « Installation », et ce qu'elle indique sur ffmpeg et ImageMagick.

2 · Installer le projet

À faire : taper les trois commandes de la section « Installation », dans l'ordre, depuis le dossier travail\trajet\ ; vérifier l'installation.

À obtenir : l'invite commence par (trajet_ensg), et magick --version, ffmpeg -version et trajet --help répondent.

Se placer dans le dossier du projet

Menu Démarrer, taper anaconda, choisir « Anaconda Prompt ». Puis :

```
cd C:\Users\eleve\Desktop\info01\cours1\lj_trajet\travail\trajet
dir
```

dir doit lister environment.yml et pyproject.toml : la première commande lit environment.yml, et la troisième pyproject.toml, dans le dossier courant.

Les trois commandes

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

1. conda env create -f environment.yml lit le nom de l'environnement, trajet_ensg, et la liste de ses paquets dans le fichier, puis les installe. Le téléchargement prend plusieurs minutes. Contrairement à conda install, la commande ne demande pas de confirmation.
2. conda activate trajet_ensg : l'invite passe de (base) à (trajet_ensg).
3. python -m pip install -e . installe le projet lui-même. Le point désigne le dossier courant, travail\trajet, et -e installe le projet là où il est, sans le copier : une modification de ses fichiers prend effet sans réinstallation.

Ouvrir environment.yml et pyproject.toml dans l'éditeur pendant que la première commande s'exécute. Les deux fichiers commencent par un commentaire, qui explique ce qu'ils contiennent.

À noter : ce qu'environnement.yml installe, et ce que pyproject.toml déclare dans dependencies.

Vérifier l'installation

Le README donne trois commandes de vérification :

```
magick --version
ffmpeg -version
trajet --help
```

magick est la commande d'ImageMagick. ffmpeg prend un seul tiret devant -version, magick en prend deux. trajet --help affiche les options de la commande du projet : --carte, --etapes et --sortie.

Vérification : chacune des trois commandes affiche du texte, et aucune ne répond « n'est pas reconnu ».

À noter : pour magick et ffmpeg, ce qu'ils sont, et dans quel fichier du projet ils sont déclarés.

En cas d'erreur

- 'conda' n'est pas reconnu en tant que commande interne ou externe : la commande a été tapée dans un cmd ou un PowerShell ordinaire. La taper dans l'invite de commandes d'Anaconda.
- EnvironmentFileNotFound ou un fichier introuvable à la première commande : le dossier courant n'est pas travail\trajet. Refaire le cd.
- CondaToSNonInteractiveError: Terms of Service have not been accepted, ou une question Do you accept the Terms of Service (ToS) ... [(a)ccept/(r)ect/(v)iew] : répondre a.

- `CondaHTTPError: HTTP 000 CONNECTION FAILED` : la session réseau n'est pas ouverte. L'ouvrir, et relancer la commande. Sans réseau, l'environnement ne peut pas être créé : lire le README et le code, et regarder la vidéo d'un voisin.
- « Solving environment » qui dure plusieurs minutes : la première commande qui emploie `conda-forge` télécharge la liste de ses paquets. Attendre.
- `'magick'` n'est pas reconnu ou `'ffmpeg'` n'est pas reconnu, alors que l'installation a réussi : l'environnement n'est pas actif dans cette fenêtre. Taper `conda activate trajet_eng`.
- `'trajet'` n'est pas reconnu : la troisième commande n'a pas été tapée, ou pas avec `trajet_eng` actif. Activer l'environnement, se placer dans `travail\trajet`, et la taper.

3 · Lancer la commande, et lire ce qu'elle produit

À faire : lancer `trajet` ; regarder `trajet.mp4` ; ouvrir les fichiers intermédiaires ; changer un texte d'`etapes.csv` et relancer.

À obtenir : `travail\trajet\` contient `trajet.mp4`, et une seconde vidéo, avec le texte changé, est produite par la même commande.

Lancer trajet

Dans l'invite de commandes d'Anaconda, toujours dans `travail\trajet` et avec `trajet_engs` actif :

```
trajet
```

La commande s'exécute pendant quelques secondes, puis affiche une ligne qui donne le chemin de la vidéo, le nombre d'étapes, la durée en secondes et la taille en octets. Ouvrir `travail\trajet\trajet.mp4` par double-clic, dans le lecteur vidéo du poste.

À noter : le nombre d'étapes, la durée de la vidéo, et où apparaissent les textes.

Les fichiers que la commande a écrits

Afficher le contenu de `travail\trajet\` dans l'explorateur. La commande y a écrit huit fichiers à côté du projet : cinq images, `etape_01.png` à `etape_05.png`, `trajet.srt`, `montage.txt` et `trajet.mp4`.

1. Ouvrir `data\etapes.csv` dans l'éditeur. La première ligne nomme les colonnes : `numero`, `duree`, `x`, `y`, `texte`.
2. Ouvrir `etape_01.png`, puis `etape_05.png`, dans la visionneuse d'images.
3. Ouvrir `trajet.srt` et `montage.txt` dans l'éditeur.

Relever, pour chaque étape du traitement, le fichier lu et le fichier écrit :

Étape	Qui la fait	Entrée	Sortie
lecture	<code>etapes.py</code>	<code>data\etapes.csv</code>	
tracé	<code>images.py</code> , par <code>magick</code>	<code>data\carte.png</code> et les étapes	
commentaires	<code>montage.py</code>	les étapes et leurs durées	
montage	<code>montage.py</code> , par <code>ffmpeg</code>	les images et le <code>.srt</code>	

Les modules sont dans `src\trajet\`, et `__main__.py` les appelle dans cet ordre : sa fonction `main` compte une trentaine de lignes, et se lit sans difficulté.

À noter : la colonne « Sortie » ; la forme des lignes de `trajet.srt` et de `montage.txt`, et le rapport entre les nombres qu'elles contiennent et la colonne `duree` d'`etapes.csv`.

Changer un texte, et relancer

1. Dans `travail\trajet\data\etapes.csv`, changer le texte d'une étape, en fin de ligne, sans toucher aux virgules ni aux nombres. Enregistrer.
2. Relancer `trajet` dans l'invite de commandes d'Anaconda.
3. Rouvrir `trajet.mp4`, puis `trajet.srt`.

Le fichier est un CSV : les virgules séparent les colonnes. Un texte qui contient lui-même une virgule doit être entouré de guillemets droits, "comme, ceci".

À noter : ce qu'il a fallu refaire pour obtenir la nouvelle vidéo, et ce qu'il aurait fallu refaire pour obtenir la même chose avec un logiciel de montage à la souris.

4 · En option : les commandes, une à une

À faire : ouvrir `outils_video.ipynb` avec un noyau de `trajet_engsg` ; exécuter ses cellules.

À obtenir : les images produites par `magick` et la durée de la vidéo produite par `ffmpeg`, affichées sous les cellules qui les ont produites.

Le notebook `outils_video.ipynb` reprend les commandes `magick` et `ffmpeg` du projet, isolées, avec leur résultat affiché sous chacune. Il ne s'exécute pas dans le navigateur, avec JupyterLite : il lance des programmes par subprocess, ce qu'un onglet de navigateur ne peut pas faire.

Ouvrir le notebook

`trajet_engsg` ne contient ni client ni noyau Jupyter : son `environment.yml` n'installe que Python, `ffmpeg` et `ImageMagick`. Le notebook peut recevoir un noyau de deux façons, celles du TD 1i :

- dans VSCode : ouvrir `cours1\1j_trajet\outils_video.ipynb` (menu File, Open File...), « Select Kernel », « Python Environments... », `trajet_engsg`. À l'exécution de la première cellule, VSCode propose d'installer `ipykernel` : accepter ;
- dans JupyterLab : installer le client dans l'environnement, puis le lancer depuis le dossier du TD :

```
conda install -n trajet_engsg -c conda-forge jupyterlab
conda activate trajet_engsg
cd C:\Users\eleve\Desktop\info01\cours1\1j_trajet
jupyter lab
```

Dans les deux cas, le noyau est celui de `trajet_engsg`, dont l'environnement est actif : les commandes `magick`, `ffmpeg` et `trajet` y sont trouvées.

Régler le chemin du projet

La première cellule désigne le projet par `PROJET = Path("depart/trajet")`, un chemin relatif au dossier du notebook. Le projet installé, et la vidéo de l'étape 3, sont dans `travail\trajet` : remplacer la ligne par

```
PROJET = Path("travail/trajet")
```

avant d'exécuter. Avec `depart/trajet`, la cellule `ffprobe` répond que `trajet.mp4` n'existe pas encore, et la dernière consigne du notebook fait modifier le fichier d'étapes de `depart\`, que la commande `trajet` ne lit pas.

Exécuter les cellules

Exécuter les cellules une à une (Maj + Entrée). La fonction `lancer` affiche chaque commande, précédée de \$, avant de la lancer ; les images et la vidéo produites sont enregistrées dans `travail\essai\`, que la première cellule crée.

Ce que le notebook montre	Ce qu'on y voit
<code>subprocess.run</code>	la commande donnée en liste, un élément par argument
<code>magick identify, -resize, -draw</code>	l'image produite, affichée dans la cellule qui l'a produite
<code>ffprobe</code> , puis <code>ffmpeg -f concat</code>	la liste de montage écrite à la main, et la vidéo qu'elle produit

La dernière cellule lance la commande `trajet` elle-même, avec les options `--carte` et `--sortie`. Taper `trajet --help` dans l'invite de commandes d'Anaconda pour lire les trois chemins que la commande accepte.

À noter : pour la commande de la cellule `-draw`, ce que représente chacun des éléments de la liste, et à quel endroit du fichier `src\trajet\images.py` la même commande est construite.

Ce que le TD fait constater

Cette section se lit après avoir fait les étapes.

Étapes 1 et 2 : ce dont le projet dépend

	Ce qu'on fait	Ce qu'on constate
1	copier <code>depart\trajet\</code> dans <code>travail\</code> , lire son README	une section « Installation », trois commandes
2	les taper, dans l'ordre	l'environnement <code>trajet_engs</code> , puis le projet installé
3	<code>magick --version</code> et <code>ffmpeg -version</code>	deux programmes, qui ne sont pas des bibliothèques Python
4	<code>trajet</code> , puis ouvrir <code>trajet.mp4</code>	cinq étapes, vingt et une secondes, sous-titres incrustés

Trois commandes et trois vérifications suffisent à installer le projet, parce que quelqu'un les a écrites et essayées. Le TD 1h a fait écrire ce type de documentation pour le projet recette ; ce TD en montre l'usage, en installant, à partir de sa seule documentation, un projet écrit par quelqu'un d'autre.

`pyproject.toml` déclare une liste de dépendances vide, `dependencies = []`, et un commentaire précise que ce n'est pas un oubli : le code n'importe que la bibliothèque standard de Python. Le projet ne fonctionne pourtant pas sans `environment.yml`, qui installe `ffmpeg` et `imagemagick`. Ce sont des programmes, que le code lance par `subprocess` comme on les lancerait au terminal, et non des paquets Python : `pip` ne sait pas les installer, alors que `conda` le peut. Les deux fichiers ne décrivent donc pas la même chose : `pyproject.toml` décrit le projet Python, `environment.yml` l'environnement complet dans lequel il s'exécute. Comme les programmes sont appelés depuis le code Python, et non depuis un script propre à un système, le projet fonctionne sous Windows, macOS et Linux.

La troisième commande, `python -m pip install -e .`, est celle du TD 1h : le code étant sous `src\`, `import trajet` échoue tant que le projet n'est pas installé. Elle crée aussi la commande `trajet`, que `pyproject.toml` déclare dans sa section `[project.scripts]` ; `python -m trajet` fait la même chose.

Étape 3 : ce que la commande enchaîne

Étape	Qui la fait	Entrée	Sortie
lecture	<code>etapes.py</code>	<code>data\etapes.csv</code>	cinq étapes, en mémoire
tracé	<code>images.py</code> , par <code>magick</code>	<code>data\carte.png</code> et les étapes	<code>etape_01.png</code> à <code>etape_05.png</code>
commentaires	<code>montage.py</code>	les étapes et leurs durées	<code>trajet.srt</code> et <code>montage.txt</code>
montage	<code>montage.py</code> , par <code>ffmpeg</code>	les images et le <code>.srt</code>	<code>trajet.mp4</code> , 21 secondes

Chaque étape produit un fichier que la suivante lit. `etapes.csv` a six lignes de données : le point de départ, de durée nulle, puis cinq étapes de 4, 4, 4, 4 et 5 secondes, soit 21 secondes et cinq images. Chaque image montre en bleu le chemin déjà parcouru, et en orange le segment de l'étape.

Les deux fichiers du milieu sont du texte. `trajet.srt` est un format de sous-titres : pour chaque étape, un numéro, un intervalle de temps et le texte, par exemple `00:00:16,000 --> 00:00:21,000` pour la cinquième. `montage.txt` est la liste que `ffmpeg` lit en mode concat : une ligne `file` et une ligne `duration` par image, et la dernière image répétée, sans quoi `ffmpeg` la coupe. Ces fichiers sont écrits par un programme et lus par un autre, et restent lisibles dans un éditeur. `etapes.csv` est du texte de la même façon, comme le `content.xml` d'un `.odt` au TD 1b : le contenu utile de ces fichiers est du texte, que deux programmes s'échangent.

Changer un texte d'étape a nécessité de modifier une ligne d'étapes.csv et de relancer une commande. La même vidéo montée à la souris, dans un logiciel de montage, serait à refaire en entier. La ligne de commande demande plus de travail pour produire la première vidéo, et presque aucun pour les suivantes.

Le fond de carte, carte.png, est livré avec le projet. Le serveur de tuiles d'OpenStreetMap est un service bénévole, dont les conditions d'usage interdisent qu'une promotion entière lui envoie des requêtes en même temps : carte.py a téléchargé le fond de carte une fois, avant la séance. La mention « © les contributeurs d'OpenStreetMap », incrustée en bas de l'image, est exigée par la licence des données, l'ODbL.

Étape 4 : les commandes dans un notebook

subprocess.run reçoit la commande sous forme de liste, un élément par argument : Python ne découpe pas la commande aux espaces, et un nom de fichier qui contient un espace n'a pas à être protégé par des guillemets. L'instruction de dessin line 398,248 410,360, qui contient des espaces, est un seul argument de -draw. Dans la cellule -draw, CARTE est l'image d'entrée, -fill, -stroke et -strokewidth règlent le trait pour les -draw qui suivent, chaque -draw trace un segment, et le dernier élément est l'image de sortie. images.py construit la même liste dans sa fonction _commande, un -draw par segment du trajet.

Le notebook ne s'exécute pas dans JupyterLite : un onglet de navigateur ne peut pas lancer de programme. Cette limite est la contrepartie de l'exécution dans l'onglet, qui rendait JupyterLite commode au TD 1g, où rien n'était à installer. Une bibliothèque écrite uniquement en Python s'y installe par %pip install, mais pas un programme comme ffmpeg. Les outils du notebook, ImageMagick et ffmpeg, sont aussi ceux du projet de la séance 4, le studio d'animation.