

Projet Python :

Les arbres de Grenoble



L'emblématique arbre de Venon

Source : francebleu.fr

Objectifs du module

L'objectif de ce projet est de vous plonger, tête la première dans de l'analyse de données, sans aucun pré-requis technique. A l'issue de ce projet, vous aurez vu les bases de la **manipulation de données grâce à Python**.

Nous sommes conscients qu'il existe parmi vous une disparité de connaissances relatives au monde informatique, à la programmation et à l'analyse de données. Certains d'entre vous ont probablement déjà codé (régulièrement ou non) au cours de leur vie personnelle ou professionnelle (python, matlab, R ou autre) alors que d'autres découvrent ce monde aujourd'hui.

Il est donc tout à fait normal que le projet à suivre paraisse facile à certaines personnes et difficile pour d'autres. Tant pis, et tant mieux ! Nous vous encourageons autant que possible à vous entraider, à transmettre, et à collaborer. Prenez votre temps au Campus Numérique comme un **espace disponible vous permettant de vous auto-former et d'expérimenter à plusieurs**. N'hésitez pas à approfondir vos connaissances et à aller chercher plus loin que ce qui est strictement attendu de vous si vous en sentez le besoin ou le désir.

Le rôle des formateurs au cours de ce projet (et de la formation de manière plus générale) sera celui d'**accompagnateur méthodologique**. Nous n'aurons probablement pas la réponse à toutes vos questions.

Modalités

- Durée du projet : 5 jours.
- Ce projet sera réalisé en autonomie
- L'entraide est (très) fortement encouragée !

Description du projet

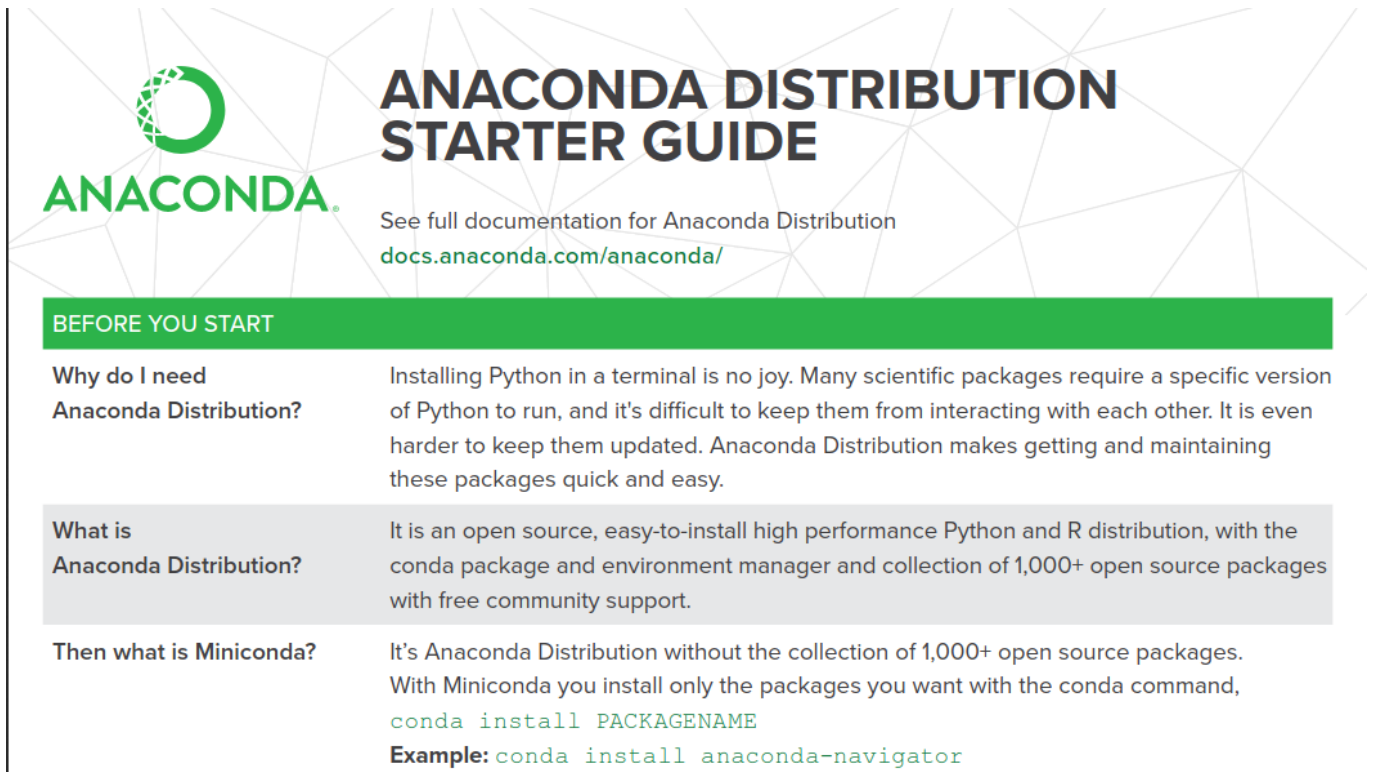
- Ce projet s'insère dans le cadre de l'**Open Data**, et en particulier des données libérées par la Ville de Grenoble. Ce jeu de données contient des informations relatives aux arbres situés sur le territoire de la ville : numéro d'identification, date de plantation, genre botanique, espèce, etc. Ce jeu de données est disponible à cette adresse : <http://data.metropolegrenoble.fr/ckan/dataset/les-arbres-de-grenoble>
- Vous explorerez les données via un **Jupyter notebook**, particulièrement adapté pour le calcul scientifique et la Data Science. Jupyter est un projet open-source et gratuit.



Installation de l'environnement Anaconda

Objectifs de l'activité

- Mettre en place l'environnement de développement scientifique avec Anaconda



The graphic is a 'Starter Guide' for Anaconda Distribution. It features the Anaconda logo (a green snake head) and the text 'ANACONDA DISTRIBUTION STARTER GUIDE'. Below this, it says 'See full documentation for Anaconda Distribution' and provides the URL 'docs.anaconda.com/anaconda/'. A green bar with the text 'BEFORE YOU START' is followed by a table of questions and answers. The questions are: 'Why do I need Anaconda Distribution?', 'What is Anaconda Distribution?', and 'Then what is Miniconda?'. The answers explain that Anaconda Distribution simplifies the installation and management of Python and R packages, and that Miniconda is a lighter version without the default package collection. An example command is provided: `conda install anaconda-navigator`.

ANACONDA DISTRIBUTION STARTER GUIDE

See full documentation for Anaconda Distribution
docs.anaconda.com/anaconda/

BEFORE YOU START

Why do I need Anaconda Distribution?	Installing Python in a terminal is no joy. Many scientific packages require a specific version of Python to run, and it's difficult to keep them from interacting with each other. It is even harder to keep them updated. Anaconda Distribution makes getting and maintaining these packages quick and easy.
What is Anaconda Distribution?	It is an open source, easy-to-install high performance Python and R distribution, with the conda package and environment manager and collection of 1,000+ open source packages with free community support.
Then what is Miniconda?	It's Anaconda Distribution without the collection of 1,000+ open source packages. With Miniconda you install only the packages you want with the conda command, <code>conda install PACKAGENAME</code> Example: <code>conda install anaconda-navigator</code>

Consignes

- Pour Linux :
 - Télécharger <https://www.anaconda.com/distribution/#linux>
 - Choisir Python 3.7 => 64-bit (x86) installer (506MB)
 - Consignes d'installation :
<https://docs.anaconda.com/anaconda/install/linux/>
 - Pour ouvrir un terminal à l'endroit où vous avez téléchargé Anaconda
 - Click droit => Open Terminal Here
 - Vous pouvez alors exécuter les instructions de la doc
- Pour Windows
 - Idem mais avec l'interface graphique !

Itération 1

Exploration préliminaire du dataset “Arbres”

Objectifs de l’activité

- Installer Anaconda
- Nous nous Intéresserons dans un premier temps à quelques **structures de données internes à Python** - *built-in data structures* - sans ajout de librairie spécialisée (type *pandas*) et aux opérations de bases (**boucles et conditions**)
- Ouverture et lecture du contenu d’un fichier “.csv”
- Exploration préliminaire des données.

Compétences

- Ouvrir et lire le contenu d’un fichier .csv
- Utiliser des boucles for
- Utiliser des structures conditionnelles
- Manipuler des listes et tuples

Avertissement préliminaire



Les métiers de Data Analyst et Data Scientist visent à recueillir des données, les traiter et s’en servir pour extraire une information “métier” pertinente.

Le projet sur lequel nous travaillons a un intérêt en termes pédagogiques (python), cependant il est important de garder en tête que sans les compétences métiers associées au projet - dans notre cas celles des urbanistes, pépiniéristes, pédologues et biologistes - les analyses effectuées n’auront que peu d’intérêt réel.

Consignes

1. Avant d’explorer les données, pensez à **ranger votre arborescence de fichier**. Vous allez très rapidement accumuler de nombreux dossiers et fichiers, pas question de tout mettre en **b@r#** sur le bureau ! Vous y perdriez beaucoup de temps et d’énergie assez rapidement. Au sein de votre répertoire de travail, créez un dossier spécifique à ce projet ayant une structure :

```
Projet_arbre
|_ data
    |_ arbres.csv
|_ notebooks
    |_ arbres.nbpy
```

2. Votre fichier contenant les données est un csv (**comma separated values** : https://fr.wikipedia.org/wiki/Comma-separated_values)
3. Avant même de commencer à coder, il est très important d'observer les données. Ouvrez donc vos données dans un éditeur de texte (vscode, sublime text, etc.) ou même dans le *bash* (*cat* ou *nano* par exemple). Ou même dans Excel 2000 ! Regardez les données à traiter droit dans les yeux !
4. Nous allons maintenant écrire du code dans un notebook pour commencer à analyser les données (*voir ressources*). Vous utiliserez pour le moment uniquement le module csv de la librairie standard de python dans un premier temps.

Note sur les modules python :

Un module est un fichier python contenant des définitions et des instructions. Dans notre cas, le module **csv** va contenir un ensemble de fonctions pré-codées nous permettant de facilement lire des fichiers csv.

Les modules de la librairie standard sont déjà installés, vous n'avez rien d'autres à faire que de dire à Python que vous en avez besoin grâce au mot clé **import** :

```
>>> import csv
```

Vous êtes ensuite capables d'utiliser les fonctions du module, comme par exemple la fonction permettant de lire un fichier csv en utilisant `csv.reader()`

5. Le but des prochaines étapes est de gagner en agilité dans la manipulation des données en Python :
 - ❑ Extraire l'ensemble des lignes (*rows*) et stocker ces informations dans une variable de type *List*
 - *Doc officielle sur les csv* :
<https://docs.python.org/3/library/csv.html>
 - *Doc officielle sur les list (tutoriel)* :
<https://docs.python.org/fr/3/tutorial/introduction.html#lists>
 - La réponse à cette première question est donnée en fin de document (annexe 1) pour ne pas rester coincé ici si la question est trop rude.
 - ❑ Afficher les deux premières lignes de ce csv. Que représentent elles ?
 - ❑ Afficher la 2ème colonne de chacune de ces deux lignes.
 - ❑ Extraire l'information concernant l'année de plantation pour l'ensemble des lignes (*rows*) et stocker ces informations dans une autre variable de type *List*
 - Vous pouvez essayer avec une **boucle for** :
https://www.w3schools.com/python/python_for_loops.asp

- ❑ Afficher les 50 premières lignes et les 50 dernières lignes de cette liste.
 - List slicing : <https://www.geeksforgeeks.org/python-list-slicing/>
- ❑ Combien d'arbres sont recensés dans ce jeu de données ?
 - Fonction len : https://www.w3schools.com/python/ref_func_len.asp
- ❑ Pour combien d'arbres manque-t-il l'information concernant la date de plantation ? (Vous pouvez essayer de le faire d'au moins deux manières différentes.)
 - Fonction count : <https://www.programiz.com/python-programming/methods/list/count>
- ❑ Combien d'arbres ont été plantés l'année de votre naissance ?
- ❑ Quelle est la plus ancienne année de plantation recensée dans ce dataset ? La plus récente ?
- ❑ Combien d'arbres ont été plantés année par année (ex : 1987 : 771, 1988 : 266, etc...) ?
 - Fonction range : https://www.w3schools.com/python/ref_func_range.asp
- ❑ Combien d'arbres ont été plantés en moyenne chaque année ?
 - Moyenne d'une liste de deux manières : <https://www.geeksforgeeks.org/find-average-list-python/>
- ❑ Stocker conjointement l'année de plantation et le nombre d'arbres plantés dans un tuple. Les tuples seront stockés dans une liste (ex : [('1987', 771), ('1988', 266),])
 - Les tuples en python : <https://courspython.com/tuple.html>
- ❑ Quel Maire a planté le plus d'arbres à Grenoble ?
- ❑ Récupérez maintenant l'information concernant le genre botanique et la stocker conjointement avec l'année de plantation dans un tuple. Les tuples seront stockés dans une liste (ex : [('1987', 'Acer'), ('1988', 'Acerifolia),])
- ❑ Pour combien d'arbres manque-t-il l'information concernant le genre botanique ?
- ❑ Utilisez Matplotlib pour tracer l'histogramme représentant le nombre d'arbres plantés par année.

- Exemple :
<https://www.kite.com/python/answers/how-to-plot-a-histogram-from-a-list-in-matplotlib-in-python>
 - Doc officielle :
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html
- ❑ (plus dur) Triez les tuples (année, genre_botanique) en fonction de l'année de plantation par ordre décroissant.

Note sur les ressources :

Les ressources que vous trouverez sur Internet concernant Python (et plus largement sur les langages de programmation) sont de plusieurs types : **tutoriels** (écrits, vidéos, livres), **how-to** (orientés sur un but précis) et **forums** (stackoverflow par exemple) et **documentation officielle**. Vous serez régulièrement amenés à jongler entre toutes ces sources d'information. Ne négligez pas la documentation officielle! Elle est souvent aride, mais elle est exhaustive. Il est important de s'y confronter régulièrement pour finir par la dompter.

Ressources

- Notebooks
 - <https://www.codecademy.com/articles/how-to-use-jupyter-notebooks>
- Documentation officielle Python 3
 - <https://docs.python.org/3/library/csv.html>
 - <https://docs.python.org/3/howto/sorting.html>
 - <https://docs.python.org/3/tutorial/introduction.html#lists>
- Documentation W3C
 - https://www.w3schools.com/python/python_datatypes.asp
- Tutoriel par Google :
 - <https://developers.google.com/edu/python>
- Documentation Matplotlib :
 - <https://matplotlib.org/tutorials/index.html>
- Un peu de biologie (désolé pour les vrais biologistes...) :
 - [https://fr.wikipedia.org/wiki/Genre_\(biologie\)](https://fr.wikipedia.org/wiki/Genre_(biologie))

Annexe 1

Une manière possible de lire un fichier `trees.csv` (rangé dans un dossier `data`).

```
In [1]: import csv
        with open('data/trees.csv', newline='') as f:
            reader = csv.reader(f)
            data = list(reader)
```

La variable `data` contient désormais les informations du fichier `csv`.