Lycée Hoche
ECG1 A

2025-2026
Mathématiques

TP de Python numéro 13 : Bases de données avec Pandas

*Semaine du **jeudi 11 juin 2026**.*

Le but de ce TP est de se familiariser avec la bibliothèque `pandas` de Python. Celle-ci permet de manipuler des **bases de données**. Les fonctionnalités de pandas que nous rencontrerons sont les suivantes :

- Introduction d'un nouveau type d'objet : le type `pandas.DataFrame` (dit `DataFrame` dans la suite).
- Introduction de nouvelles commandes pour définir des objets de type `Dataframe` et pour les manipuler,
- Introduction d'outils permettant une analyse statistique élémentaire sur ces objets.

I. Importation de bases de données en .csv avec pandas

1. Téléchargement de bases de données

Une base de données, comme son nom l'indique, est un ensemble de données organisées (généralement collectées suites à une étude, un recensement, un sondage etc.).

Nous allons ici travailler avec 2 bases de données récoltées sur le site de `data.gouv.fr` (un site gouvernemental mettant à disposition des bases de données qui proviennent de sources publiques). De nombreuses bases de données sont également disponibles sur le site de l'INSEE.

Les données consultables sur ces sites sont très variées : données socio-économiques, environnementales, liées à des secteurs stratégiques (énergie)...

Exercice 1. Nous allons commencer par télécharger les bases de données utilisées en exemple pour ce TP.

1. Se rendre sur la page `data.gouv.fr`, et rendez-vous sur le moteur de recherche de ce site permettant de trier les données téléchargeables : *Découvrez les jeux de données.*
2. Téléchargez la première base de donnée, relative aux indices de position sociale des lycées en France. Les étapes sont en annexe.
3. Téléchargez la seconde base de donnée, relative au prix du carburant en France (étapes en annexe).

Vous avez maintenant téléchargé les bases de données avec lesquelles nous allons travailler. Pour des raisons décrites plus bas, nous allons travailler dans un dossier comme indiqué en annexe.

Ouvrez Pyzo, créez un nouveau fichier Python et enregistrez-le dans ce dossier (nommez-le TP13.py).

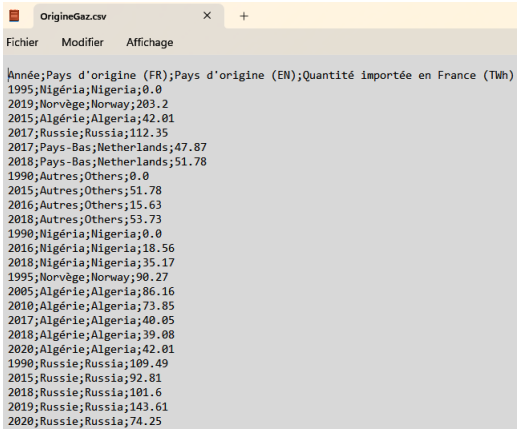
Vous devriez avoir, à ce stade, dans le dossier TP13, 3 fichiers : les deux bases de données (IPSLycee.csv et PrixCarburant.csv) et votre fichier Python de travail vierge (TP13.py).

2. Qu'est-ce qu'une base de données en .csv?

Voyons maintenant ce que nous avons téléchargé.

Exercice 2. Ouvrez le fichier "PrixCarburant.csv" avec un éditeur de texte ***brut*** : ce sont les éditeurs de texte sans mise en forme. Sur windows, le Bloc-Notes est installé par défaut. Sur Mac, c'est TextEdit. Sur Linux, ça dépend de votre distribution (c'est généralement Text Editor). Pour ouvrir le fichier avec un programme choisi, utilisez clique droit > "ouvrir avec" sur Windows.

À droite, ce qui devrait s'afficher.



Format de fichier ".csv"

L'extension ".csv" abrège "comma-separated values". Comme vous pouvez le constater, un fichier .csv est construit de la manière suivante :

1. La première ligne contient une liste de **descripteurs** séparés par un caractère, appelé le **séparateur** du fichier.
   - Les descripteurs décrivent les données récoltées.
   - Le séparateur est un caractère spécial, qui sert à séparer les données présentes sur les lignes.
2. Les lignes suivantes contiennent les séries de données collectées (les "entrées"), séparées par un séparateur. Chaque ligne est une série d'entrées différente.
3. Ces entrées peuvent être du texte (La deuxième entrée de chaque ligne est un nom de pays, ou "Autres" de temps en temps) ou des nombres (la première entrée de chaque ligne est une date). Un détail sera important : **la virgule numérique est ici un point**, comme on le voit dans les dernières entrées.

Le séparateur et la virgule numérique sont deux caractères "réservés" dans un fichier .csv : ils ne peuvent être utilisés autrement que pour séparer des données ou marquer une virgule numérique.
Une description correcte des descripteurs est généralement fournie avec les données (voir sur le site de téléchargement).

Exercice 3. Ouvrir les fichiers téléchargés avec un éditeur de texte brut. Répondez aux questions en observant ce fichier.

1. Quelles sont les séparateurs de ces fichiers ? Quels sont les caractères utilisés en virgule numérique ?
2. Quels descripteurs de ces fichiers sont clairement identifiables ? Lesquels ne comprenez-vous pas ?
3. Assurez vous d'avoir compris l'intérêt de la première ligne, et sa correspondance avec les autres.

Pour y voir plus clair, ces données sont en fait naturellement organisées sous la forme d'un tableau. Les sauts de lignes séparent les lignes, les séparateurs séparent les colonnes. Vous pouvez ouvrir ces fichiers avec un tableur comme OpenOffice pour avoir une visualisation satisfaisante. Il existe d'autres formats de base de données (.ods, .xls, .xlsx...), le format .csv est très utilisé car assez universel.

3. Se faire une espace de travail

Nous allons utiliser le module `pandas` de Python pour visualiser et surtout manipuler ces bases de données. Pour cela, il sera pratique d'avoir le fichier python sur lequel on travaille dans le même dossier que nos données.

Exercice 4. Ouvrez, avec Pyzo, votre fichier de travail. Inscrivez en première ligne :

```
import pandas as pd
```

Dorénavant, on travaillera sur ce fichier.

4. Importation de fichiers .csv avec pandas

Le module `pandas` de `Python` nous permet de manipuler des bases de données avec Python, à des fins d'analyse par exemple.

Pour cela, il faut indiquer à Python l'endroit où se trouve la base de donnée qu'on veut charger (ici, nos fichiers .csv). "L'endroit" où se trouve un fichier sur l'ordinateur s'appelle l'adresse de ce fichier.

a) Pour indiquer une adresse à Python

Nous avons trois possibilités :

- On peut fournir une adresse **absolue**. Il s'agit de l'adresse de votre fichier à partir du nom du disque dur de votre ordinateur.

  Exemple :

  ```
  C:\Users\Luke\Desktop\TP13\IPSLycee.csv
  ```

  Pour donner cette adresse à Python, on remplacera les antislash (caractère réservé en Python) par des slash, et on la donnera sous la forme d'une chaine de caractère.

  Exemple :

`"C:/Users/Luke/Desktop/TP13/IPSLycee.csv"`

C'est un peu pénible (ça dépend du système d'exploitation), et nous avons un moyen plus pratique de charger ces fichiers dans Python.

- On peut fournir une adresse **relative** à Python, c'est-à-dire l'adresse prise à partir du dossier dans lequel on travaille. Par exemple, notre fichier Python étant dans le dossier "TP13", Python interprétera `"PrixCarburant.csv"` comme une adresse désignant un fichier PrixCarburant.csv du dossier dans lequel se trouve notre fichier de travail (TP13.py).
- Enfin, on peut fournir une adresse web, avec les mêmes règles que pour l'adresse absolue.

b) Import de fichier avec `pd.read_csv`

La bibliothèque `pandas` est chargée avec le raccourci `pd` grâce à la première ligne de votre fichier Python.

pd.read_csv

La commande Python `pd.read_csv()` permet d'importer un fichier .csv sous la forme d'un objet de type `pd.DataFrame`.
Argument obligatoire
Le premier argument pris par `pd.read_csv()` est l'adresse du fichier à importer.

Arguments supplémentaires remarquables
L'argument `sep="separateur"` spécifie le séparateur utilisé par le fichier. Par exemple, `sep=";"` spécifie que les entrées sont séparées par un point-virgule, et `sep=","` spécifie qu'elles sont séparés par une virgule.
L'argument `decimal="virguledecimale"` spécifie la virgule décimale utilisée par le fichier. Par exemple, `decimal="."` spécifie que c'est un point, et `decimal=","` spécifie que c'est une virgule.
L'argument `header=None` indique que le fichier .csv utilisé ne contient pas de lignes de descripteurs (nous ne nous en servirons pas).

Remarque. Si on ne spécifie pas d'argument supplémentaire, le séparateur par défaut est la virgule, et la virgule numérique par défaut est le point.

Dans votre fichier `Python`, ajoutez les lignes :

```
PrixCar=pd.read_csv("PrixCarburant.csv", sep=";")
```

Puis, exécutez ce fichier avec "Executer > Executer le script" et tapez "IPS" dans l'interpréteur. Que remarquez-vous ?

Après cette commande effectuée, le contenu de "PrixCarburant.csv" est enregistré dans la variable `PrixCar`, qui est de type `pd.DataFrame`.

Remarque. Pour la lisibilité, Python indique une numérotation des lignes.

Exercice 5. Rajoutez les ligne permettant d'importer la base de données relative aux IPS des lycées, dans une variable Python nommée `IPS`.

Remarque. Avant d'importer un fichier .csv, il est de bon goût de l'ouvrir avec un éditeur de texte brut pour connaitre la nature du séparateur et de la virgule décimale.

II. Manipulation des bases de données

À ce stade, nous avons donc téléchargé des bases de données, que nous avons importées dans Python.

Rajoutez en première ligne l'import `import numpy as np`.

1. Premiers outils

Premiers outils

Soit `L` une base de données de type `np.DataFrame` et `'Desc'` l'un des descripteur de `L`.

- La commande `L.shape` renvoie le couple `(l,c)` formé du nombre `l` de lignes de `L`, et du nombre `c` de ses colonnes.
- La commande `L.head()` renvoie les 5 premières lignes de `L`.
- La commande `L.tail()` renvoie les 5 dernières lignes de `L`.
- La commande `L.columns` renvoie (sous un format particulier) la liste des descripteurs de `L`.
- La commande `L['Desc']` renvoie la base de données obtenue à partir de `L` en ne conservant que la colonne de descripteur `'Desc'`.
- La commande `L['Desc'][i]` renvoie l'entrée de la `i` ième ligne correspondant au descripteur `'Desc'`.
- La commande `L['Desc'].values` renvoie la colonne de descripteur `'Desc'` sous forme de tableau numérique `np.ndarray`.
- La commande `L.values` renvoie les données de `L` sous forme d'une liste `np.ndarray` des listes donnant les entrées de chaque lignes.
- La commande `L.index` renvoie la liste (sous un format particulier) des indices des lignes de `L`.

Exemple 6. Lancer `PrixCar.head()` dans l'interpréteur et constater.

Remarque. La commande `.head()` est très utile pour vérifier que l'importation s'est bien passée, sans problème de séparateur ou de virgule décimale.

Exercice 7. Tester ces commandes sur les bases de données introduites auparavant. Remarquer que les descripteurs sont systématiquement de type `string`.

2. Extraction de lignes et de colonnes

Souvent, les bases de données trouvées contiennent des informations dont nous ne nous soucions pas pour notre étude. On veut alors, ne serait-ce que pour des questions de lisibilité, ne garder que les informations qui nos intéressent.

Extraction de colonnes

Soit `L` une base de données, et `D` une liste de descripteurs de `L`.
Alors, `L[D]` renvoie la base de données obtenue à partir de `L` en ne conservant que les colonnes dont le descripteur est dans `D`.

Exemple 8. La base de données PrixCar contient beaucoup de données de localisation temporelle et spatiale. Simplifions cela. Tapez, à la suite, dans votre code Python ci-dessous puis (après exécution) demander à python d'afficher la variable `CarbSimp`. Comparer avec `PrixCar`.

```
CarbSimp=PrixCar[["Ville","Prix Gazole","Prix SP98", "code_departement"]]
```

Exercice 9. Créer une variable `IPSlisible` contenant la base de données obtenue à partir de `IPS` en ne conservant que les colonnes donnant : L'année, le code du département, le nom de l'établissement, le nom de la commune, l'IPS pour la voie générale et technologique, et l'IPS pour la voie professionnelle.

Extraction de lignes

Soit `L` une base de données, et `n,p` des entiers de type `int`.
Alors,

- `L[n:p]` renvoie la base de données obtenue à partir de `L` en ne gardant que les lignes de la `n` ième à la `p-1` ième.
- `L.head(n)` renvoie les `n` premières lignes de `L`.

Extraction de colonnes et de lignes

Soit `L` une base de données, `D` une liste de descripteurs de `L` et `n,p` des entiers.
Alors, `L[D][n:p]` renvoie la base de données obtenue à partir de `L` en ne conservant que les colonnes dont le descripteur est dans `D`, et les lignes de la `n` ième à la `p-1` ième.

3. Filtrage des lignes

On peut appliquer des filtres pour ne garder que certaines lignes.

Filtre simple

Soit `L` une base de données, et `'Desc'` un descripteur de `L`.
Alors,

- La commande `L[L['Desc']==n]` renvoie la base de données obtenue à partir de `L` en ne gardant que les lignes dont l'entrée de la colonne `'Desc'` vaut `n`.
- La commande `L[L['Desc']>n]` renvoie la base de données obtenue à partir de `L` en ne gardant que les lignes dont l'entrée de la colonne `'Desc'` est strictement supérieure à `n` (données numériques uniquement).
- Le même principe marche avec les autres opérateurs de comparaison `<, <=, >=, !=`.

Exemple 10. Exécuter la commande `CarbSimp[CarbSimp["code_departement"]=='78']` dans l'interpréteur et constater le résultat.

Exercice 11. À partir de la base de donnée IPS :

1. Afficher la liste des lycées avec un IPS en voie GT supérieur à 110 .
2. Afficher la liste des villes où le prix du gazole dépasse 2.20 euros

Remarque. Quand on construit une nouvelle base de donnée en filtrant les lignes, celles-ci ne sont pas renumérotées. Les indices des lignes (qui permettent de désigner ces lignes) deviennent alors lacunaires, d'où la commande `L.index` pour retrouver ces indices.

Filtre multiple

Soit `L` une base de données. Soient `filtre1,filtre2` deux filtres construits selon l'encadré ci-dessus. Par exemple, on pourrait avoir `L['Desc1']>n` pour `filtre1` et `L['Desc2']==n` pour `filtre2`.
Alors, `L[np.logical_and(filtre1,filtre2)]` renvoie la base de données obtenue en ne gardant que les lignes satisfaisant les deux filtres.
De même, `L[np.logical_or(filtre1,filtre2)]` renvoie la base de données obtenue en ne gardant que les lignes satisfaisant l'un des deux filtres.

Exercice 12.

1. Afficher la liste des villes du Bas-Rhin (67) où le prix du gazole dépasse 2.30 euros.
2. Afficher la liste des lycées des Yvelines (78) dont l'IPS en voie GT est inférieur à 120.

4. Méthode de tri

Le module pandas inclut une fonctionnalité de tri selon les valeurs d'une colonne numérique.

Tri d'une base de données

Soit `L` une base de données et `'Desc'` un descripteur de `L` dont la colonne correspondante est à entrées numériques.
Alors, `L.sort_values('Desc')` renvoie la base de données obtenue en triant les lignes de `L` par valeurs croissantes de la colonne `'Desc'`.

Exemple 13. Dans l'interpréteur, exécuter `PrixCroissant=PrixCar.sort_values("Prix Gazole")`. Afficher `PrixCroissant`, et à l'aide de `PrixCroissant.tail()` et `PrixCroissant.head()`, donner les records de prix (ville).

Exercice 14. 1. Donner les 5 lycées ayant l'IPS en voie GT le plus élevé, puis le plus faible.

2. Donner les 5 lycées d'Haute-Savoie (74) ayant l'IPS en voie GT le plus élevé, puis le plus faible.

5. Indicateurs statistiques

Pandas fournit des indicateurs statistiques.

Commandes à connaître

Soit `L` une base de données.

- La commande `L.describe()` renvoie une analyse statistique des colonnes numériques de `L`, contenant, dans l'ordre, l'effectif total, la moyenne, l'écart-type, le minimum, les 3 quartiles, et le maximum.
- `L.mean()` renvoie les moyennes des colonnes numériques de `L`. La commande `L.mean()` évite le message d'erreur quand `L` contient des données non numériques.
- `L.std()` renvoie les écarts-types des colonnes numériques de `L`.
- `L.median()` renvoie la médiane des colonnes numériques de `L`.
- `L.count()` renvoie l'effectif de chaque colonne de `L`.

En rajoutant l'argument `numeric_only=True` dans les fonctions ci-dessus, cela évite les messages d'erreur en cas de colonnes non numériques.
Pour chaque commande ci-dessus, l'argument optionnel `axis=1` effectue le calcul statistique sur les lignes de `L` à la place des colonnes. Par exemple, `L.std(axis=1)` affiche l'écart type des entrées des lignes numériques de `L` (ce qui peut facilement n'avoir aucun sens).

Exemple 15. Afficher, à l'aide de `describe`, les indicateurs statistiques des IPS des lycées d'abord sur toute la France, puis exclusivement sur l'Ile-de-France (75-77-78-91-92-93-94-95).

III. Exercices

On considère les variables globales `CarbSimp` et `IPS` affectées selon le déroulé du TP. Cet encadré peut être utile.

Les entrées sont "itérables"

Soit `L` une base de données et `'Desc'` un descripteur de `L`. Alors, `for val in L['Desc']:` initie une boucle dans laquelle la variable (locale) `val` parcourt les entrées de la colonne `'Desc'` de `L`.
De plus, `for i in L.index :` initie une boucle dans laquelle la variable (locale) `i` parcourt les indices des lignes de `L`.

Exercice 16. Travail sur la base de données relative au prix du gazole. On supposera que ces données d'importation sont exhaustives. Les questions sont posées assez directement et nécessitent généralement plusieurs étapes.

1. Afficher l'analyse statistique élémentaire de la base de données relative aux importations de gaz (effectif total, moyenne, médiane, etc.).
2. Déterminer la liste des départements concernés par cette base de données. Construire, en Python, la liste de ces départements (on pourra penser à la commande `np.unique`).
3. En une ligne de code, calculer la moyenne des prix du gazole pour le département du Rhône (69). On pourra utiliser la commande `np.mean`.
4. Tracer la moyenne du prix du gazole par département pour les départements numérotés de 21 à 35.
5. Déterminer le département où le prix de l'essence SP98 est le plus élevé. Le plus faible ? Le plus proche de la moyenne française ?

Exercice 17. Travail sur la base de données relative aux IPS des lycées. On ne gardera pas les colonnes (inutiles ici, Python sait le faire) donnant les écarts-types des IPS.

1. Consulter les données présentes pour votre lycée d'origine.
2. Déterminer la liste des années scolaires pour lesquelles des données sont présentes.

3. Définir une base de données (stockée dans une variable Python) ne gardant que les données d'une même année (de votre choix). Dans la suite de cet exercice, on travaillera sur cette base de données (on restreint notre étude à l'année scolaire choisie).
4. Déterminer l'IPS moyen des lycées français en voie générale et technologique, puis en voie professionnelle.
5. Écrire une fonction prenant en entrée le numéro d'un département, et renvoyant en sortie la moyenne et l'écart-type des IPS en voie GT de ce département (sous la forme d'un couple).
6. Déterminer les lycées ayant les cinq plus grands IPS en voie GT.
7. Déterminer le département ayant l'IPS en voie générale et technologique moyen le plus élevé, puis le plus faible.
8. Déterminer le département dont l'écart type des IPS en voie GT est maximale, puis minimale.

IV. Annexe

Téléchargement des bases de données

Avant tout, créez un nouveau dossier sur le bureau de votre ordinateur, nommez-le TP13 par exemple (ne le nommez pas "pandas"). Des explications seront données plus tard.

Téléchargez la première base de données, relative aux indices de position sociale des lycées, en suivant les instructions ci-dessous.

1. Cherchez «*Indice de position sociale des lycées*» et cliquez sur le premier lien.

2. Dans l'onglet "Fichiers" (bas de page), cliquez sur le premier lien permettant de télécharger la base de données au format `csv`.

3. Déplacez le fichier téléchargé dans le dossier TP13. Renommez ce fichier avec un nom simple : `IPSLycee.csv`.

Téléchargez la seconde base de données, relative au prix du carburant en France.

1. Retournez sur le moteur de recherche du site *data.gouv.fr*. Cherchez « *Prix carburants*» et cliquez sur le lien "Prix des carburants en France - Flux instantané - v2".
2. Dans l'onglet "Fichiers", cliquez sur le premier lien permettant de télécharger la base de données au format `csv`.
3. Déplacez le fichier téléchargé dans le dossier TP13. Renommez ce fichier : `PrixCarburant.csv`.