**Panda and CSV File**

Numpy est un élément très puissant pour ce qui est des données numériques. Mais dans un projet data scienec, nous avons souvent affaire à beaucoup de données : catégorie, label, dates… Cela devient vite fastidient de gérer toutes ces données avec numpy. Panda est intéressant parce qu’elle permet de faire beaucoup d’opérations complexes sur les données comme de l’algèbre relationnel. Pandas fournit des structures de données puissantes et simples à utiliser, ainsi que les moyens d'opérer rapidement des opérations sur ces structures.

On peut importer un fichier csv ou importer un fichier texte puis le transformer en DataFrame via la commande famille\_panda\_df = pd.DataFrame(famille\_panda\_numpy, index=[‘maman’, ‘bebe’, ‘papa’], columns=[‘pattes’, ‘poil’, ‘queue’, ‘ventre’]).

* Famille\_panda\_df.ventre pour accéder à une colonne du tableau
* Famille\_ panda\_df.loc[« papa »] pour accéder à une ligne
* Famille\_ panda\_df.iloc[2] pour accéder à une ligne en utilisant la position (si on a des label qui sont des nombres on ne risque pas d’avoir une anbiguité)
* Famille\_ panda\_df [Famille\_ panda\_df .ventre==80] pour filtrer que les pandas dont la taille de ventre est égal à 80
* Tous\_les\_pandas = Famille\_ panda\_df .append(autres\_pandas) pour concaténer des dataframe
* Tous\_les\_pandas.drop\_duplicates() pour supprimer les doublons
* Data = pd.read\_csv(« pandas.csv”) pour charger un fichier csv et le transformer en un DataFrame

Pour avoir accès à une colonne d’un fichier csv qui ne contient pas de nom :

* Dataframe.iloc[0]
* Dataframe.head(10) pour avoir accès aux 10 premiers éléments du DataSet
* Dataframe.tail(10) pour avoir accès aux 10 derniers éléments du DataSet
* sorted\_hills = dataframe.sort\_values(by=['Height'], ascending=False) pour trier dans l’ordre décroissant par rapport à ‘Height’
* Dataframe.describe() pour avoir une description du DataSet (moyenne, écart-type, …)
* Dataframe.dropna() pour enlever toutes les données Nan

<https://ourcodingclub.github.io/tutorials/pandas-python-intro/>

**Découvrez les librairies Python pour la data science**

R (très puissant) est un logiciel destiné à la base aux statisticiens. Python est devenu par accident une référence en matière de data science grâce à ses librairies spécialisées (numpy, …). SAS surtout utilisé par les statisticiens et Matlab par les ingénieurs et les chercheurs.

La simplicité vs la puissance :

Limites des tableurs :

1. Ils ne peuvent pas faire face à l'énorme quantité de données que nous manipulons régulièrement. Par exemple, le tableur de LibreOffice ne peut gérer que 1024 colonnes et un peu plus d'un million de lignes.
2. Leur puissance de calcul est très insuffisante.
3. Ils ne permettent pas de faire tourner les algorithmes à la pointe.

Python est simple, interactif (on peut lancer un script en Python sans étape de compilation), écosystème (numpy et scipy pour les calculs, Matplotlib et Seaborn pour la visualisation, Scikit-learn pour les algorithmes, Pandas pour les gérer les données (les charger, appliquer des opérations d'algèbre relationnelle, etc.), Tensorflow et PyTorch pour le deep learning).

Anaconda est une distribution, c’est-à-dire un langage de programmation + certaines librairies et autres fonctionnalités.

Questions :

* Nombre de CPU utilisé puissance de 2 ?
* Python est interactif (on peut lancer un script en Python sans étape de compilation) ? Performance ?
* Serveur AWS ?
* Quelle différence dans VS Code .py avec un U ou sans ?