

# Python

## Pour l'analyse de données

MARS 2025



# Objectifs

- Posséder une vue d'ensemble de l'écosystème scientifique de Python
- Connaître les librairies scientifiques incontournables pour la science des données
- Être capable de manipuler des données volumineuses avec Python
- Comprendre l'intérêt de la data visualisation
- Savoir visualiser des données avec Python



# Présentations (s'il y a des nouveaux !)

- Ce qui vous semble pertinent (rôle, expérience, attentes...).
- Votre (éventuel) rapport à Python dans votre travail.
- Votre background (pour orienter vers certains axes le discours et les échanges).



# Programme



- 01** Introduction - l'écosystème scientifique de Python
- 02** La scipy stack
- 03** Les librairies de visualisation et la data visualisation
- 04** Les formats de fichiers scientifiques et la manipulation de données volumineuses

# Dépôt Github distant



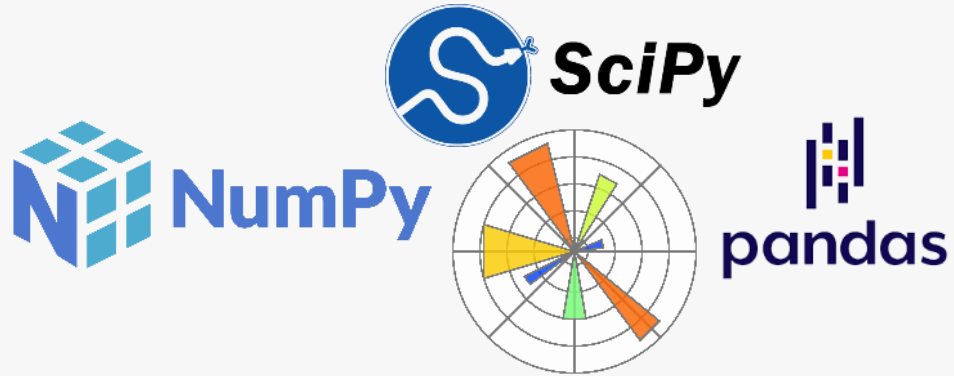
<https://github.com/BEESPE/caceis-data-analysis>

# 01

## Introduction - l'écosystème scientifique de Python

# L'écosystème scientifique de Python

les librairies incontournables, en particulier pour la Data Science



La scipy stack

(une partie dédiée de la formation y sera consacrée)

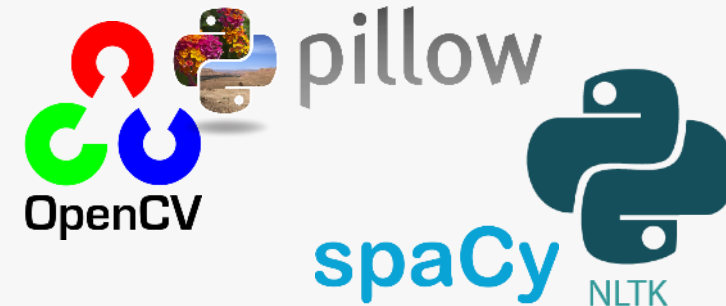


Scikit-learn



Frameworks de calcul tensoriel

(pour le Deep Learning)



Librairies pour le traitement de données spécifiques

(images, textes...)

# L'écosystème scientifique de Python

## Savoir où trouver de nouvelles librairies...

<b>PyPI</b> (Python Package Index)	Le principal dépôt pour les bibliothèques Python.	<a href="https://pypi.org">https://pypi.org</a>
<b>Github</b>	Server Git. De nombreuses bibliothèques Python sont hébergées sur GitHub. C'est une excellente plateforme pour découvrir des projets open source et voir les contributions en temps réel	<a href="https://github.com/search">https://github.com/search</a>
<b>Awesome Python</b>	Une liste collaborative et bien entretenue des meilleures bibliothèques Python. C'est un excellent point de départ pour trouver des outils spécifiques et éprouvés.	<a href="https://awesome-python.com">https://awesome-python.com</a>
<b>Forums</b>	<p>Le subreddit r/Python propose régulièrement des discussions autour de nouvelles bibliothèques Python, avec des retours d'expérience de la communauté</p> <p>Stack Overflow : les réponses aux questions de développement incluent souvent des recommandations pour des bibliothèques populaires ou émergentes. C'est aussi un bon indicateur de la communauté autour d'une bibliothèque</p>	<p><a href="https://www.reddit.com/r/Python">https://www.reddit.com/r/Python</a></p> <p><a href="https://stackoverflow.com/">https://stackoverflow.com/</a></p>
<b>Newsletters</b>	Les (multiples) newsletters partagent des découvertes de nouvelles bibliothèques Python, majoritairement à fréquence hebdomadaire, ainsi que des discussions autour des outils tendances.	



# L'écosystème scientifique de Python





## ... et comment juger de leur pérennité

- **Popularité/adoption** (nombre de téléchargements dans la section "Downloads" sur PyPI, nombre de forks et d'étoiles sur GitHub, mentions dans des articles et utilisation dans des projets populaires/des entreprises bien établies)
- **Fréquence des mises à jour** (date de la dernière mise à jour, historique des commits, cycle de versions régulières, changements documenté)
- **Support de la communauté et réactivité** (issues, temps de réponse des mainteneurs)
- **Documentation** (complétude, exactitude, clarté, structure...) Tests et intégration continue (couverture de tests automatisé, badges d'intégration)
- **Dépendances et compatibilité** (dépendances minimales et bien gérées, suivi des versions récentes de Python)

**Et, même si c'est de nature différente, ne pas oublier de se renseigner sur la licence et les conditions d'utilisation !**

# L'écosystème scientifique de Python

Au-delà de l'écosystème scientifique, l'écosystème technique pour la data science

IDE :  (Visual Studio Code)  (PyCharm)  / Jupyterlab  spyder




Manipulation et traitement de données :  pandas  dask  PySpark  (Vaex) et bien d'autres

Visualisation de données :  (matplotlib)  seaborn  plotly  bokeh  + a b l e a u  Power BI

Machine Learning et Deep Learning :  scikit-learn  XGBoost  LightGBM  TensorFlow  K Keras 

Gestion des bases de données :  MySQL  PostgreSQL  MongoDB  SQLite

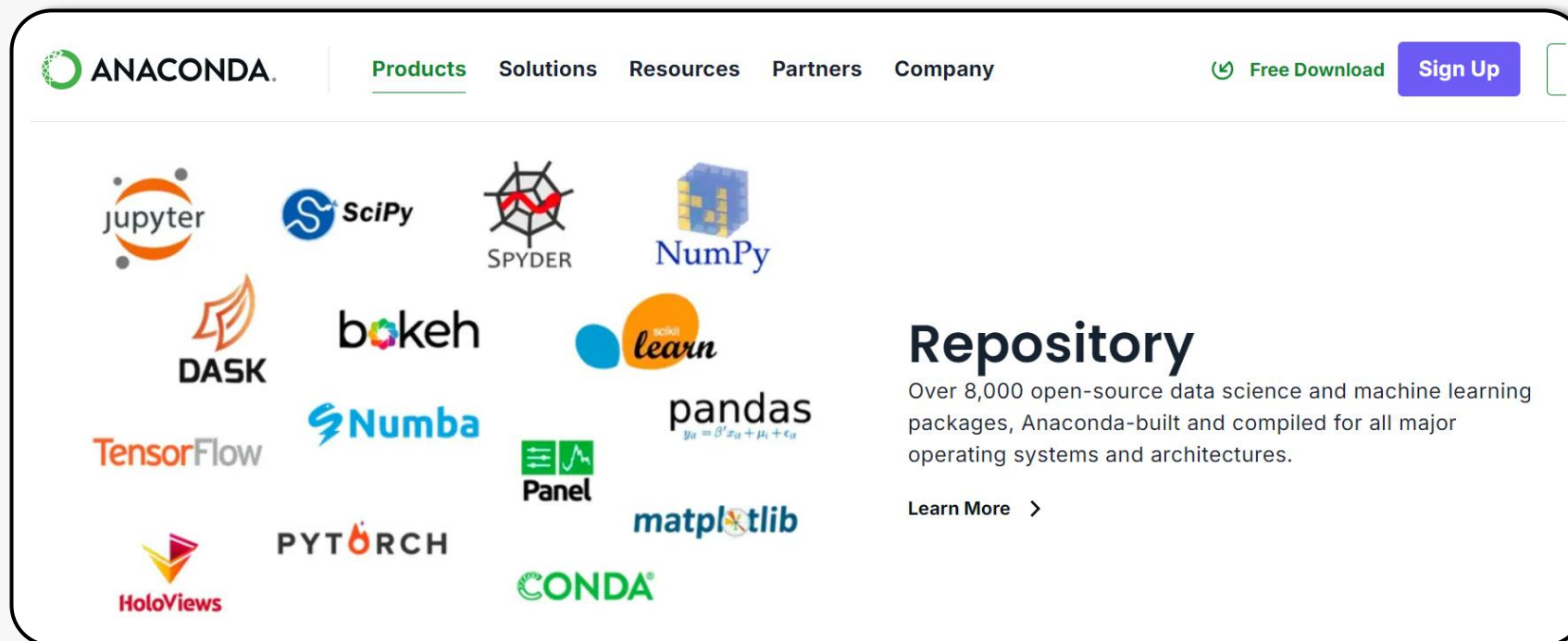
Big Data :  hadoop  PySpark  dask  kafka

Gestion des versions et collaboration :  git  GitLab  GitHub

# L'écosystème scientifique de Python

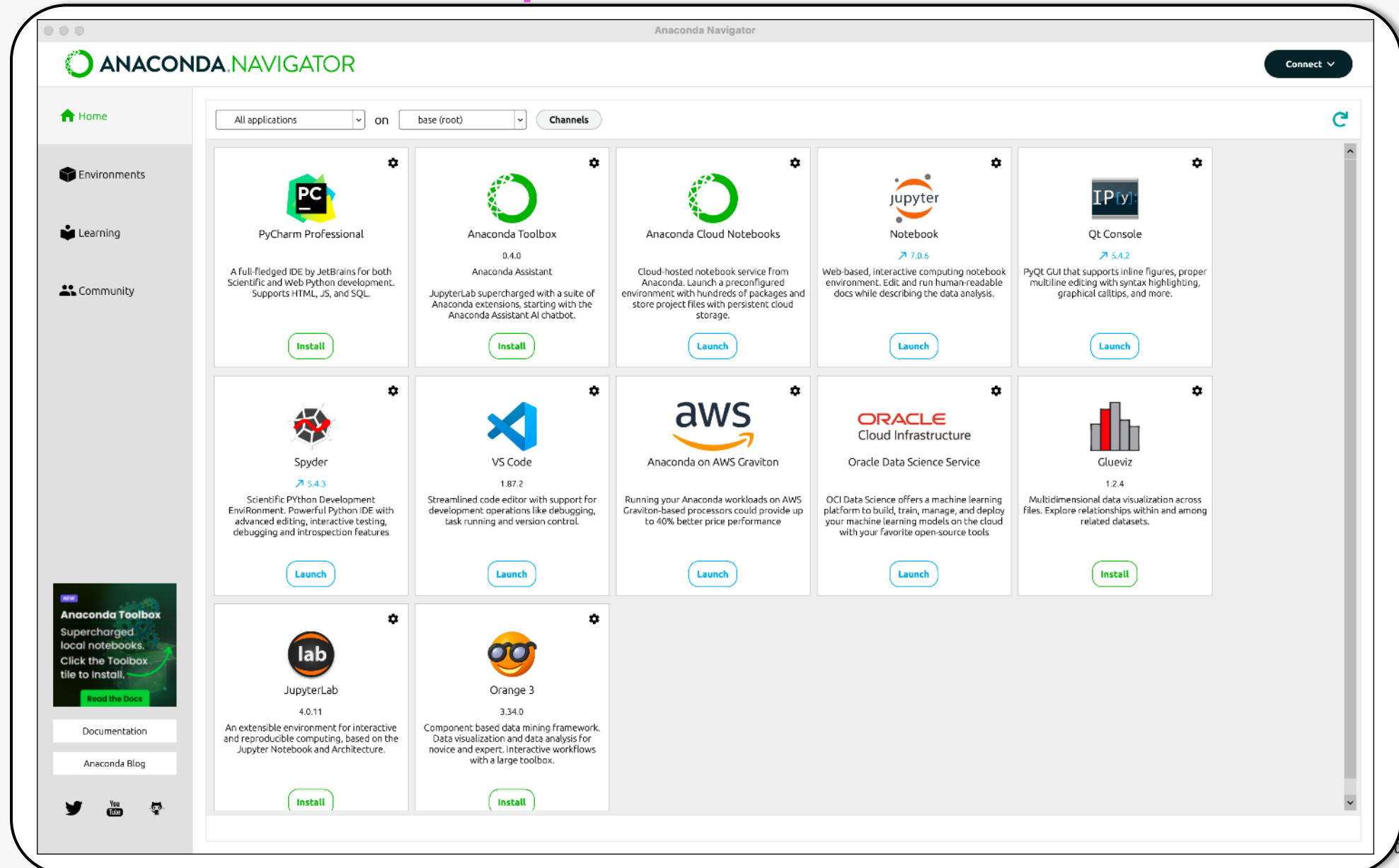
## Utiliser une distribution scientifique comme Anaconda

- Il existe des distributions de Python qui simplifient l'installation, la gestion de packages et les environnements virtuels pour les projets de Data Science et d'apprentissage automatique.
- Pour un scientifique, il peut être intéressant d'utiliser une telle distribution.
- Anaconda est une de ces distributions (c'est aussi une distribution de R). Elle est open-source.



# L'écosystème scientifique de Python

## Utiliser une distribution scientifique comme Anaconda



# L'écosystème scientifique de Python

## Pourquoi utiliser une distribution scientifique, sur l'exemple d'Anaconda

- Une distribution comme Anaconda va **faciliter l'installation des bibliothèques scientifiques** complexes qui, autrement, nécessiteraient plusieurs dépendances difficiles à gérer. Lors de l'installation d'Anaconda, il précharge plus de 1 500 packages scientifiques populaires (NumPy, Pandas, Scikit-learn, Matplotlib, Jupyter, SciPy...) Cela réduit les efforts pour configurer un environnement scientifique complet (gain de temps, résolution automatique des dépendances).
- Anaconda inclut un **gestionnaire de paquets et d'environnements** appelé Conda. Il permet de créer facilement des environnements virtuels isolés avec différentes versions de Python et des packages spécifiques à chaque projet.
- Anaconda inclut par défaut **Jupyter Notebook** et **JupyterLab**, des outils interactifs très populaires en data science. Ils permettent d'exécuter du code, de visualiser des graphiques, et de documenter des résultats dans des notebooks réutilisables.
- Anaconda fonctionne sur **Windows, macOS, et Linux**, ce qui rend le développement et le partage de projets entre différentes plateformes simple.
- Contrairement à pip, qui ne gère que les packages Python, Conda peut gérer des **packages non Python** (comme des bibliothèques C ou des compilateurs), ce qui est peut parfois être utile pour des outils dépendant de composants en C++, CUDA...

# L'écosystème scientifique de Python

TP



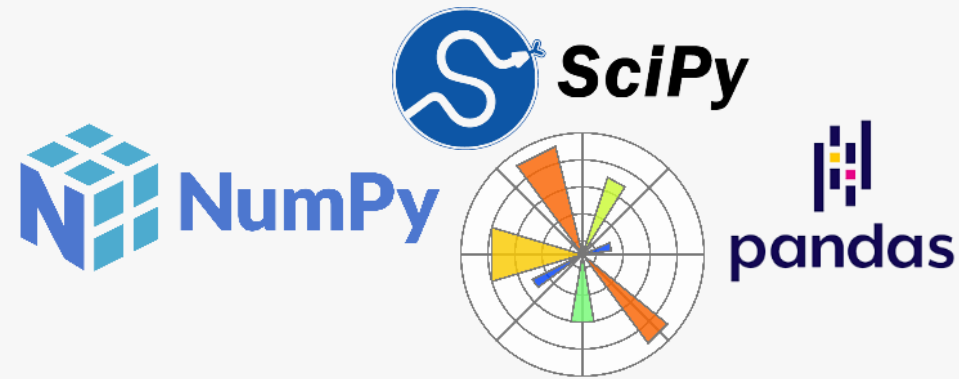
**Mise en place de  
l'environnement de  
développement**

# 02

## La scipy stack

# La scipy stack

Le socle de librairies scientifiques sur lequel sont basées beaucoup d'autres



La scipy stack



# La scipy stack

## Pandas : l'analyse de données tabulaires (CSV, Excel, etc.), statistiques, filtres...

- Pandas est une bibliothèque de **manipulation** et d'**analyse de données** en Python. Elle fournit des structures de données performantes et faciles à utiliser, comme les DataFrames, qui permettent de travailler avec des données tabulaires.
- Lancée en 2008 par Wes McKinney, Pandas s'est rapidement imposée comme un outil incontournable pour la science des données. Elle est utilisée pour importer, nettoyer, transformer et analyser des ensembles de données hétérogènes.
- Pandas est optimisée pour la **manipulation de grands ensembles de données** grâce à ses capacités d'intégration avec NumPy et d'autres bibliothèques Python.

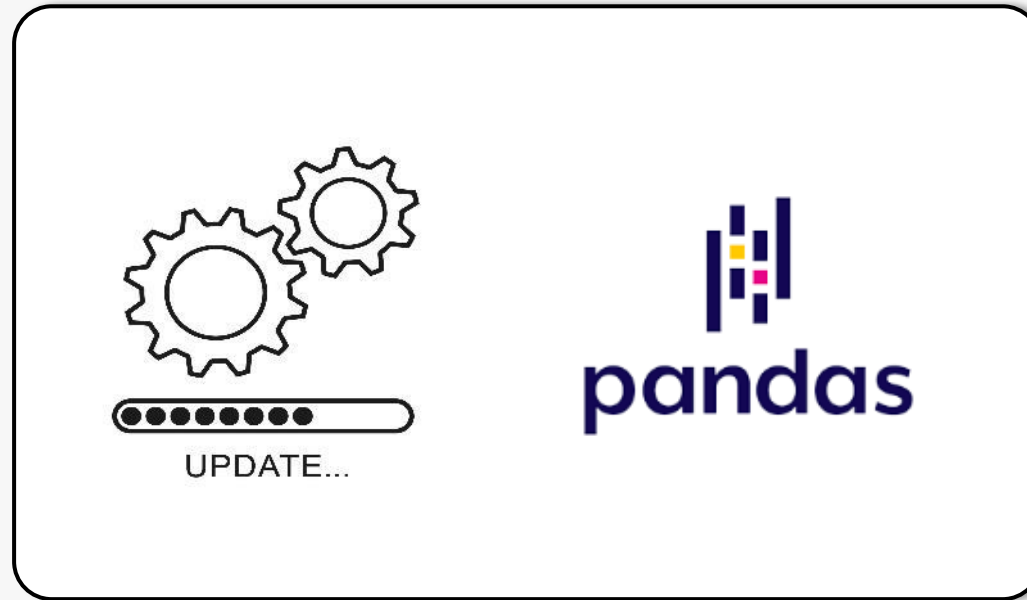
# La scipy stack

## Pandas : l'analyse de données tabulaires (CSV, Excel, etc.), statistiques, filtres...

- Les **DataFrames** de Pandas permettent de manipuler des données organisées en lignes et colonnes, semblables à des feuilles de calcul Excel. Pandas prend également en charge des index, ce qui facilite l'accès, le tri, et la sélection des données.
- La bibliothèque est idéale pour les tâches de **nettoyage de données**, de **fusion d'ensembles de données**, et de **calcul de statistiques descriptives**. Elle est souvent utilisée en tandem avec d'autres outils comme Matplotlib pour la visualisation ou scikit-learn pour l'apprentissage automatique.

# La scipy stack

Pandas : l'analyse de données tabulaires (CSV, Excel, etc.), statistiques, filtres...



```
pip install --upgrade pandas
```

# La scipy stack

Pandas : l'analyse de données tabulaires (CSV, Excel, etc.), statistiques, filtres...

`pandas.DataFrame()` crée une table de données.

`df.head()` affiche les premières lignes, `df.tail()` affiche les dernières lignes.

`df.describe()` fournit un résumé statistique des colonnes numériques.

`df.groupby()` regroupe les données pour appliquer des opérations agrégées.

`df.merge()` fusionne plusieurs DataFrames.

`df.isna()` Identifie les valeurs manquantes, `df.fillna()` les remplit.

Pandas est compatible avec divers formats, comme CSV, Excel, JSON, SQL, ou encore HDF5.

La documentation de pandas est disponible à l'adresse suivante :

<https://pandas.pydata.org/docs/>

# La scipy stack

## TP pandas #1



**Utiliser pandas pour  
manipuler des données  
tabulaires de trajets en  
Velib, comme on le ferait  
avec des requêtes SQL**

# La scipy stack

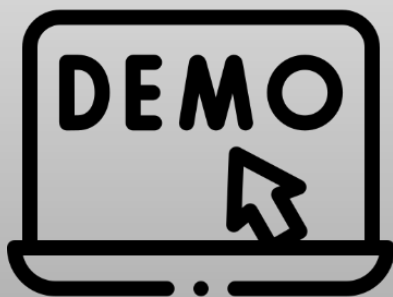
## TP pandas #2



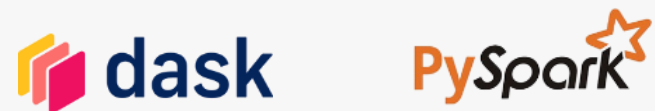
**Identifier des données  
aberrantes dans un jeu de  
données de compositions  
de produits alimentaires**

# Ouverture

## Démo



Traitement équivalent  
avec dask et pyspark



# La scipy stack

## TP bilan pandas



## Calcul de deltas dans un portefeuille



# La scipy stack

## NumPy : calcul numérique et algèbre linéaire

- NumPy, abréviation de "Numerical Python", est une bibliothèque fondamentale en Python pour le **calcul numérique** et la **manipulation de tableaux multidimensionnels**. Elle est largement utilisée dans divers domaines, y compris le traitement d'images.
- Elle a été créée en 2005 par Travis Oliphant en se basant sur des travaux précédents. NumPy est open source et a connu une adoption rapide parmi les communautés scientifiques et d'ingénierie. Elle a posé les bases pour de nombreuses autres bibliothèques de calcul scientifique en Python (scipy, matplotlib, scikit-learn...).

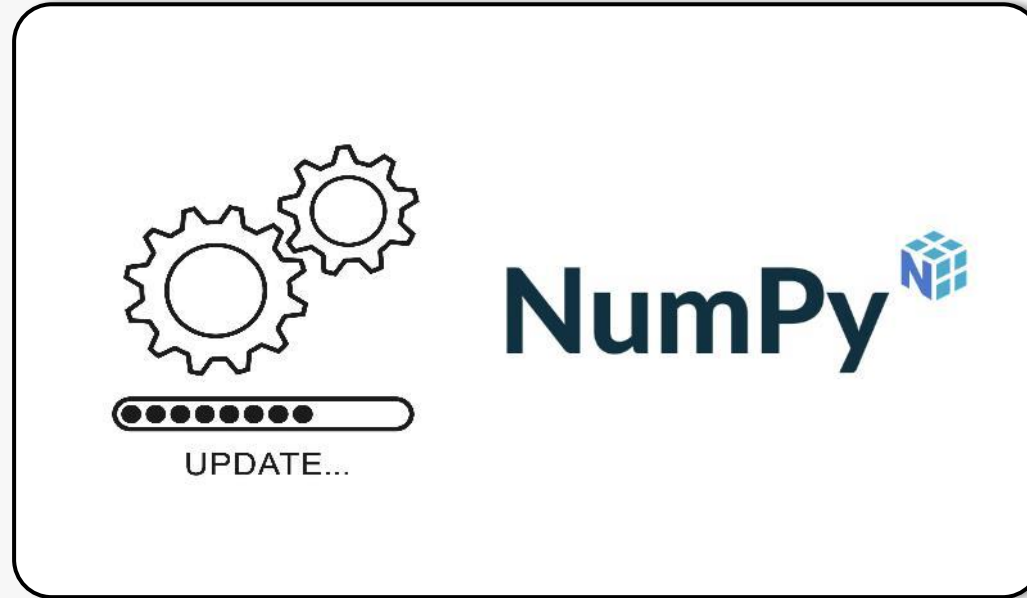
# La scipy stack

## NumPy : calcul numérique et algèbre linéaire

- NumPy est construite autour du concept de tableaux NumPy (ndarrays), qui sont des structures de données multidimensionnelles homogènes et efficaces pour stocker et manipuler des données numériques.
- Les **tableaux NumPy** (arrays) sont similaires aux listes Python, mais ils sont optimisés pour les opérations numériques. Ils peuvent avoir des dimensions multiples et prennent en charge des opérations vectorielles, ce qui les rend idéaux pour le calcul numérique.
- NumPy offre des fonctions pour effectuer des **opérations vectorielles** sur des tableaux, ce qui permet d'effectuer des calculs efficaces et parallèles sur de grandes quantités de données.
- NumPy est couramment utilisée pour **traiter et analyser des données**, notamment dans les domaines de la science des données, de l'apprentissage automatique et du traitement d'images.

# La scipy stack

NumPy : calcul numérique et algèbre linéaire



```
pip install --upgrade numpy
```

# La scipy stack

## NumPy : calcul numérique et algèbre linéaire

`numpy.array()` crée un tableau NumPy à partir d'une séquence de données.

`numpy.zeros()`, `numpy.ones()`, `numpy.empty()` créent des tableaux remplis de zéros, de uns ou de valeurs vides. Il faut en préciser la taille.

`numpy.arange()`, `numpy.linspace()` génèrent des séquences de nombres.

`numpy.shape()`, `numpy.reshape()`, `numpy.dim()` permettent de manipuler la forme et les dimensions des tableaux.

`numpy.mean()`, `numpy.max()`, `numpy.min()`, `numpy.std()` calculent des statistiques sur les données.

Il est possible d'accéder aux et de manipuler les éléments d'un tableau en utilisant des **indices** et des opérations de découpage (**slicing**).

La documentation de numpy est disponible à l'adresse suivante :

<https://numpy.org/doc/>

# La scipy stack

SciPy, basée sur NumPy pour les statistiques, les analyses fonctionnelles...

- SciPy, abréviation de "Scientific Python", est une bibliothèque conçue pour le **calcul scientifique et technique**. Elle repose sur NumPy et fournit une large gamme d'algorithmes pour des tâches telles que l'intégration, l'optimisation, l'interpolation, l'algèbre linéaire et le traitement du signal.
- Créée en 2001 par Travis Oliphant, Eric Jones, et Pearu Peterson, SciPy a joué un rôle crucial dans l'essor de Python en tant que langage phare pour la science des données et la recherche. Elle est open source et bénéficie d'une communauté active de développeurs et d'utilisateurs.
- SciPy est conçue pour être modulaire, avec plusieurs sous-modules spécialisés comme `scipy.integrate` (intégration), `scipy.optimize` (optimisation), ou `scipy.spatial` (géométrie et distances).

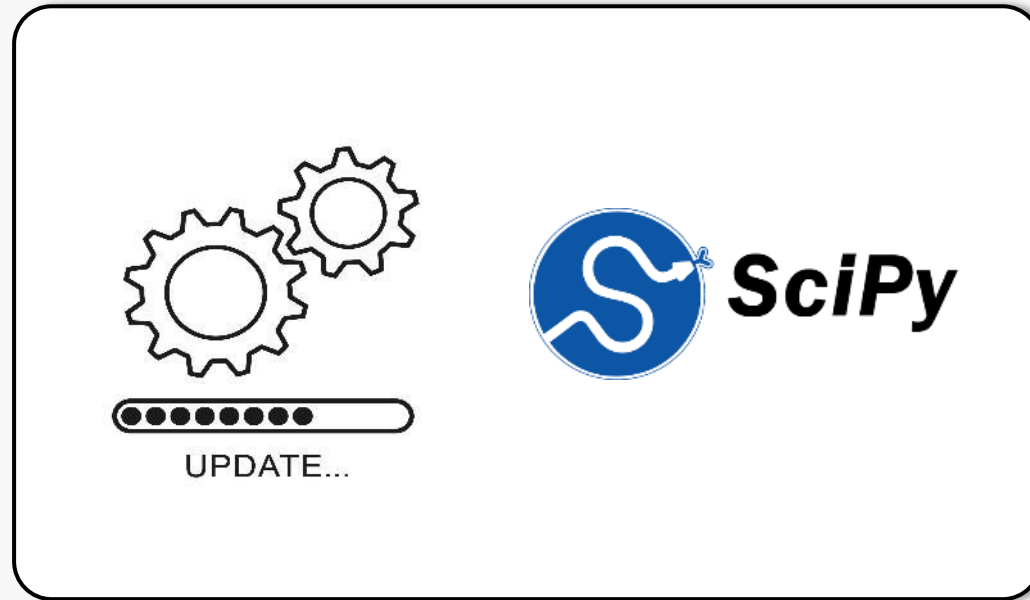
# La scipy stack

SciPy, basée sur NumPy pour les statistiques, les analyses fonctionnelles...

- SciPy s'appuie sur NumPy pour manipuler les données et se concentre sur les outils d'analyse avancés. Elle est couramment utilisée dans des domaines variés comme l'analyse des données expérimentales, la physique, la bioinformatique et l'ingénierie.
- Ses modules permettent de résoudre des équations différentielles, d'effectuer des décompositions matricielles avancées, ou encore d'analyser des distributions statistiques. Par exemple, la fonction `scipy.stats` fournit des outils pour réaliser des tests d'hypothèse ou ajuster des distributions.
- Avec ses performances optimisées, SciPy permet de résoudre des problèmes complexes de manière efficace.

# La scipy stack

SciPy, basée sur NumPy pour les statistiques, les analyses fonctionnelles...



```
pip install --upgrade scipy
```

# La scipy stack

SciPy, basée sur NumPy pour les statistiques, les analyses fonctionnelles...

`scipy.optimize.minimize()` résout des problèmes d'optimisation.

`scipy.integrate.quad()` calcule l'intégrale d'une fonction.

`scipy.interpolate.interp1d()` interpole des points de données.

`scipy.linalg.eig()` calcule les valeurs propres d'une matrice.

`scipy.stats.norm.pdf()` donne la densité de probabilité d'une loi normale.

SciPy permet également de manipuler des structures de données comme les matrices creuses via `scipy.sparse`.

La documentation de scipy est disponible à l'adresse suivante :

<https://docs.scipy.org/doc/scipy/>



# Matplotlib

## Matplotlib : la librairie de visualisation de données incontournable

- **Matplotlib** est une bibliothèque de **visualisation** de données flexible pour Python. Elle est utilisée pour créer une grande variété de graphiques, de tracés et de visualisations, que ce soit pour des analyses de données, des présentations ou des rapports scientifiques. Elle est également capable d'afficher et de traiter des images de base, ce qui en fait un outil polyvalent pour la visualisation et le traitement d'image.
- Elle a été créée par John D. Hunter en 2003. Il a développé cette bibliothèque pour aider les scientifiques et les ingénieurs à visualiser leurs données de manière efficace et esthétique.

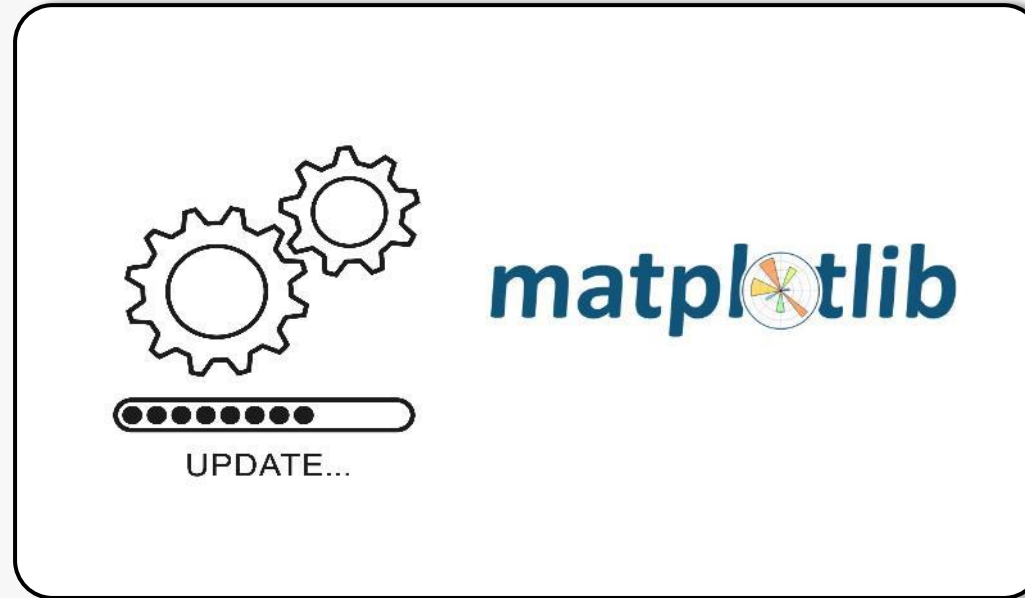
# Matplotlib

## Matplotlib : la librairie de visualisation de données incontournable

- Matplotlib fonctionne en fournissant une **interface orientée objet** pour la création de graphiques et de tracés en Python. Le sous-module `pyp1ot` fournit une interface simplifiée pour construire des visualisations rapidement.
- Matplotlib permet aux utilisateurs de créer des **figures** et des **axes** qui peuvent être personnalisés de manière approfondie. Les utilisateurs peuvent ajouter des éléments tels que des lignes, des points, des légendes, des titres... aux graphiques de manière « programmatique ».
- Matplotlib prend en charge une **grande variété de types de graphiques**, notamment les graphiques en nuages de points, les histogrammes, les graphiques à barres, les graphiques en boîte, les graphiques en courbes, les graphiques de dispersion, les graphiques en secteurs, etc.
- Matplotlib permet une **personnalisation avancée** des graphiques, avec la possibilité de contrôler chaque aspect visuel, y compris les couleurs, les styles de ligne, les polices, les étiquettes, les axes et les échelles.
- Elle est souvent utilisée avec Pandas pour tracer des données tabulaires ou avec NumPy pour visualiser des données numériques brutes.

# Matplotlib

Matplotlib : la librairie de visualisation de données incontournable



```
pip install --upgrade matplotlib
```

# Matplotlib

## Matplotlib : la librairie de visualisation de données incontournable

`plt.figure()`, `plt.subplot()` permettent de créer des figures et des sous-graphiques.

`plt.plot()`, `plt.scatter()`, `plt.bar()` créent différents types de graphiques.

`plt.xlabel()`, `plt.ylabel()`, `plt.title()`, `plt.legend()` ajoutent des éléments de légende/titres à la visualisation

`plt.savefig()` sauvegarde le graphique dans un fichier image.

La documentation de matplotlib est disponible à l'adresse suivante :  
<https://matplotlib.org/stable/index.html>

# La scipy stack

## TP



## Manipulation d'images avec la scipy stack

# La scipy stack

## Exemple d'une autre bibliothèque basée sur la scipy stack : scikit-learn

**scikit-learn** est une (excellente) bibliothèque Python pour le Machine Learning, offrant un ensemble d'outils **standardisés** pour l'apprentissage supervisé et non supervisé. Elle est couramment utilisée comme **catalogue** d'algorithmes implémentés, et plus généralement pour **construire des modèles**, transformer les données et **évaluer** les performances des algorithmes de Machine Learning.



### Prétraitement des données

scikit-learn inclut des méthodes pour normaliser, transformer et manipuler les données afin de les préparer pour le modèle

### Algorithmes de Machine Learning

scikit-learn propose un (large) éventail de modèles pour des tâches de classification, régression, clustering, réduction de dimensionnalité, et sélection de modèles

### Évaluation des modèles

scikit-learn fournit des fonctions pour évaluer la performance des modèles grâce à des métriques standard et des techniques de validation croisée

### Optimisation de modèles

scikit-learn permet de rechercher les meilleurs hyperparamètres pour un modèle à l'aide de techniques telles que GridSearchCV et RandomizedSearchCV

# La scipy stack

## Exemple d'une autre bibliothèque basée sur la scipy stack : scikit-learn

### Estimator:

The base object, implements a `fit` method to learn from data, either:

```
estimator = estimator.fit(data, targets)
```

or:

```
estimator = estimator.fit(data)
```

### Predictor:

For supervised learning, or some unsupervised problems, implements:

```
prediction = predictor.predict(data)
```

Classification algorithms usually also offer a way to quantify certainty of a prediction, either using `decision_function` or `predict_proba`:

```
probability = predictor.predict_proba(data)
```

### Transformer:

For filtering or modifying the data, in a supervised or unsupervised way, implements:

```
new_data = transformer.transform(data)
```

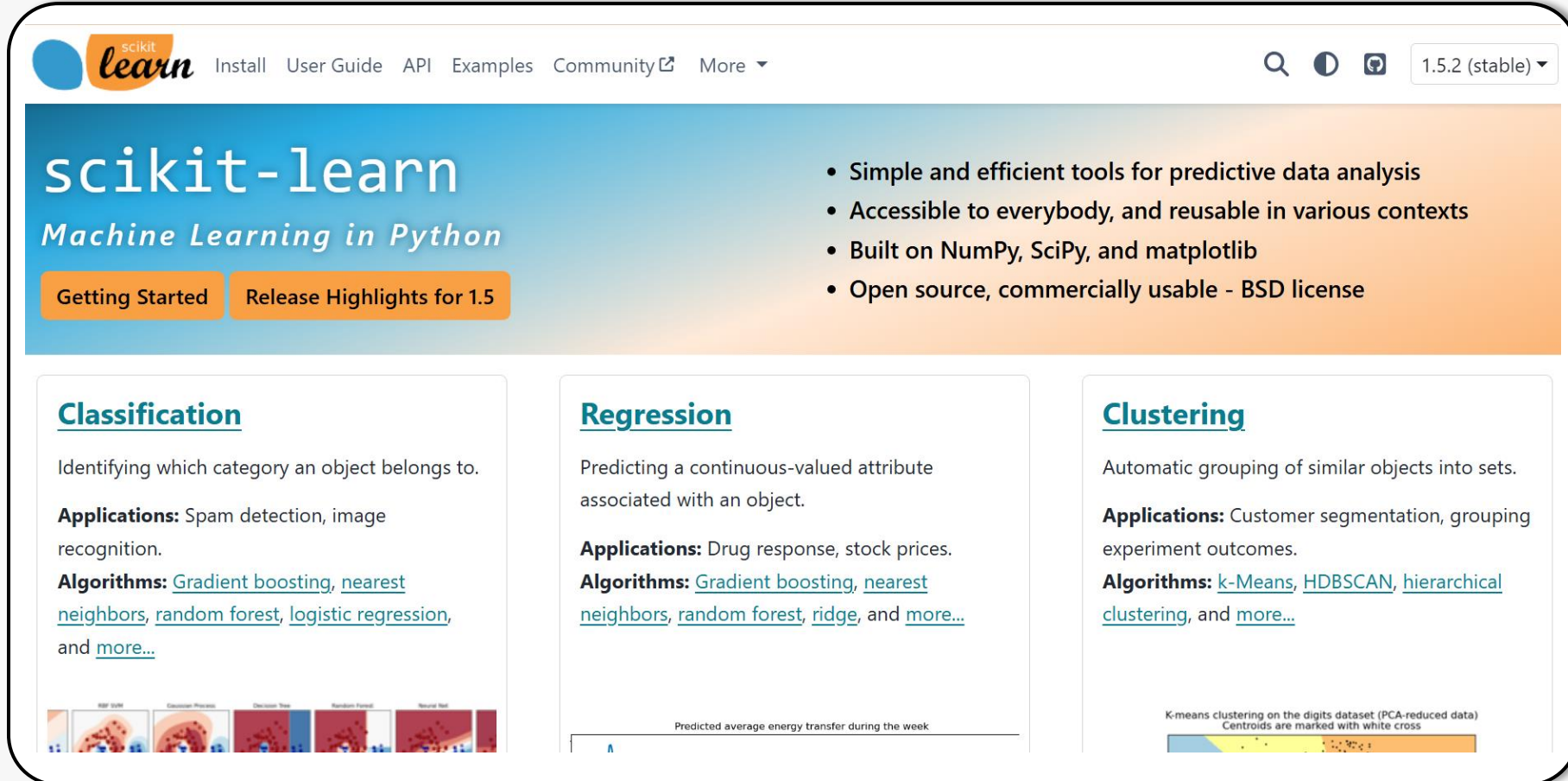
When fitting and transforming can be performed much more efficiently together than separately, implements:

```
new_data = transformer.fit_transform(data)
```

<https://scikit-learn.org/stable/developers/develop.html>

# La scipy stack

## Exemple d'une autre bibliothèque basée sur la scipy stack : scikit-learn



The screenshot shows the scikit-learn website homepage. At the top, there is a navigation bar with links for 'Install', 'User Guide', 'API', 'Examples', 'Community', and 'More'. The main header features the 'scikit-learn' logo and the tagline 'Machine Learning in Python'. Below this, there are two orange buttons: 'Getting Started' and 'Release Highlights for 1.5'. To the right, a list of bullet points highlights key features: 'Simple and efficient tools for predictive data analysis', 'Accessible to everybody, and reusable in various contexts', 'Built on NumPy, SciPy, and matplotlib', and 'Open source, commercially usable - BSD license'. The page is divided into three main sections: 'Classification', 'Regression', and 'Clustering'. Each section provides a brief description, applications, and algorithms. The 'Classification' section includes a row of small images representing different models. The 'Regression' section includes a line plot showing 'Predicted average energy transfer during the week'. The 'Clustering' section includes a scatter plot showing 'K-means clustering on the digits dataset (PCA-reduced data)' with centroids marked by white crosses.

**scikit-learn**  
*Machine Learning in Python*

Getting Started Release Highlights for 1.5

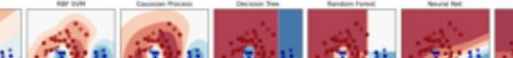
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

### Classification

Identifying which category an object belongs to.

**Applications:** Spam detection, image recognition.

**Algorithms:** [Gradient boosting](#), [nearest neighbors](#), [random forest](#), [logistic regression](#), and [more...](#)

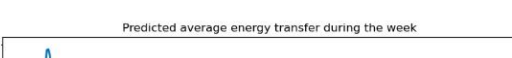


### Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, stock prices.

**Algorithms:** [Gradient boosting](#), [nearest neighbors](#), [random forest](#), [ridge](#), and [more...](#)

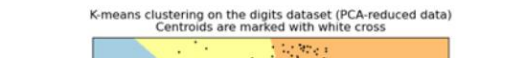


### Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, grouping experiment outcomes.

**Algorithms:** [k-Means](#), [HDBSCAN](#), [hierarchical clustering](#), and [more...](#)



<https://scikit-learn.org/stable/index.html>



# 03

## **Les librairies de visualisation et la data visualisation**

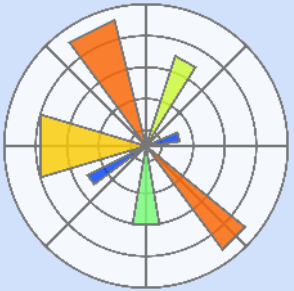
# Les librairies de visualisation et la data visualisation

## Panorama des librairies de visualisation de Python

### Visualisation 2D/3D



#### Matplotlib



C'est la librairie de visualisation la plus populaire pour les graphiques 2D en Python. Elle permet de créer des graphiques classiques tels que des histogrammes, des graphiques en courbes, des diagrammes en boîte, etc.

Ex. : Graphiques linéaires, scatter plots, histogrammes.



#### Seaborn



Construite sur Matplotlib, Seaborn simplifie la création de visualisations statistiques avancées. Elle est particulièrement adaptée aux visualisations basées sur des ensembles de données complexes.

Ex. : Cartes thermiques, boxplots, pairplots, distribution de données.

# Les bibliothèques de visualisation et la data visualisation

## Panorama des bibliothèques de visualisation de Python

### Visualisation 2D/3D



#### Plotly



Offre des graphiques interactifs et des visualisations en 3D. Permet de créer des graphiques très esthétiques et interactifs qui peuvent être intégrés à des sites web.

Ex. : Graphiques interactifs en 2D/3D, cartes, visualisations de séries temporelles.



#### Bokeh



Permet de créer des visualisations interactives en 2D et en 3D, souvent utilisées pour des dashboards. Prend en charge les web apps en temps réel.

Ex. : Visualisation de séries temporelles, diagrammes en barres interactifs.

# Les bibliothèques de visualisation et la data visualisation

## Panorama des bibliothèques de visualisation de Python

### Visualisation 2D/3D



#### Altair



Librairie déclarative pour la visualisation de données. Utilise un langage basé sur JSON pour décrire les visualisations, ce qui la rend plus intuitive

Ex. : Graphiques interactifs, visualisations statistiques simples.

# Les bibliothèques de visualisation et la data visualisation

## Panorama des bibliothèques de visualisation de Python

### Visualisation web (interactive et dashboards)



#### Dash (Plotly)



Permet de créer des applications web interactives. Très populaire pour les dashboards et la visualisation en temps réel des données.

Ex. : Dashboards interactifs, visualisations en temps réel.



#### Panel (HoloViz)



Librairie basée sur Bokeh pour créer des dashboards interactifs. Elle offre une grande flexibilité pour intégrer des widgets et des contrôles interactifs.

Ex. : Interface pour ajuster des paramètres de visualisation en temps réel.

# Les bibliothèques de visualisation et la data visualisation

## Panorama des bibliothèques de visualisation de Python

### Visualisation de données statistiques



#### ggplot (ggplot2)



Un port de la fameuse librairie ggplot2 de R. Elle suit une approche déclarative pour créer des visualisations basées sur la grammaire des graphiques.

Ex. : Graphiques statistiques élégants.



#### Statsmodels



Bien que principalement une librairie pour l'analyse statistique, Statsmodels propose également des visualisations pour analyser des régressions et des séries temporelles

Ex. : Graphiques de résidus, autocorrélation, analyse de séries temporelles.

# Les bibliothèques de visualisation et la data visualisation

## Panorama des bibliothèques de visualisation de Python

### Cartographie et visualisation géospatiale



#### Geopandas



Extension de Pandas, Geopandas permet de travailler avec des données géospatiales (formes géométriques et géographiques) et de les visualiser facilement.

Ex. : Cartes, cartes choroplèthes, visualisation de zones géographiques.



#### Folium



Basée sur la bibliothèque JavaScript Leaflet, Folium permet de créer des cartes interactives dans Python. Idéale pour la visualisation de données géospatiales interactives.

Ex. : Cartes interactives, données géospatiales en temps réel.

# Les librairies de visualisation et la data visualisation

## Panorama des librairies de visualisation de Python

### Cartographie et visualisation géospatiale



#### Cartopy

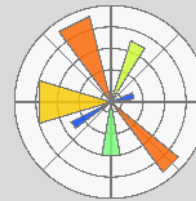


Librairie pour la cartographie géospatiale et la visualisation des données climatiques et environnementales.

Ex. : Cartes des données climatiques, géospatiales.



#### Basemap



Extension de Matplotlib pour créer des cartes géographiques et des visualisations de données spatiales.

Ex. : Cartes topographiques, cartes des températures, cartes de trajectoires.



# Les bibliothèques de visualisation et la data visualisation

## Panorama des bibliothèques de visualisation de Python

### Big Data et visualisation de données volumineuses



#### Datashader (HoloViz)



Librairie conçue pour la visualisation de très grandes quantités de données. Elle permet de créer des graphiques en temps réel pour des ensembles de données massifs.

Ex. : Visualisation de données massives, comme les données spatiales ou temporelles en temps réel.



#### HoloViews (HoloViz)



Conçue pour visualiser de très grands ensembles de données (Big Data), HoloViews se base sur des abstractions de haut niveau qui permettent une visualisation rapide.

Ex. : Graphiques interactifs pour données massives, visualisation de séries temporelles.

# Les bibliothèques de visualisation et la data visualisation

## Panorama des bibliothèques de visualisation de Python

### Big Data et visualisation de données volumineuses



#### Vaex



Librairie spécialisée dans la visualisation et l'analyse de très grands ensembles de données. Elle est optimisée pour travailler sur des jeux de données volumineux sans nécessiter beaucoup de mémoire.

Ex. : Visualisation de grands ensembles de données dans des graphiques interactifs.

# Les bibliothèques de visualisation et la data visualisation

## Panorama des bibliothèques de visualisation de Python

### Autres bibliothèques et outils utiles pour la visualisation



#### Pyvis



Librairie permettant de visualiser des graphes (réseaux) interactifs. Elle est utilisée pour représenter des graphes de manière intuitive et interactive.

Ex. : Visualisation de réseaux sociaux, graphes relationnels.



#### WordCloud



Spécialement conçue pour créer des nuages de mots (word clouds), une représentation visuelle simple mais percutante des fréquences de mots dans un texte.

Ex. : Analyse de texte, visualisation de mots-clés dans des corpus.

# 04

## **Les formats de fichiers scientifiques et la manipulation de données volumineuses**

# Les formats de fichiers scientifiques et la manipulation de données volumineuses

## Panorama des principaux formats de fichiers scientifiques

Ces formats sont optimisés pour stocker, échanger et analyser des données massives ou complexes.

Format	Description	Domaines d'application	Avantages
NetCDF (Network Common Data Form)	Conçu pour la manipulation de données multidimensionnelles (variables, temps, espace).	Climatologie, océanographie, modélisation atmosphérique.	Compact, largement utilisé, compatible avec Xarray.
HDF5 (Hierarchical Data Format v5)	Format hiérarchique conçu pour les données volumineuses et complexes.	Simulations numériques, bioinformatique, machine learning.	Flexible, rapide, compatible avec plusieurs langages.
GRIB (GRIdded Binary)	Format binaire spécialisé pour les données météorologiques.	Prévisions météorologiques, climatologie.	Compact, optimisé pour les grilles spatiales.
MATLAB (.mat)	Format natif pour stocker les données MATLAB.	Recherche scientifique, ingénierie.	Pour le partage entre utilisateurs MATLAB.
CGNS (CFD General Notation System)	Format destiné à la dynamique des fluides numériques.	Simulation en mécanique des fluides.	Standardisé, structuré.
JSON (JavaScript Object Notation)	Format léger pour représenter des objets de données.	APIs, échanges de données, stockage léger.	Lisible par l'humain, très populaire.
PARQUET	Format binaire orienté colonnes pour les données analytiques.	Big Data, machine learning.	Optimisé pour la compression et les requêtes rapides.

# Les formats de fichiers scientifiques et la manipulation de données volumineuses

## Manipuler des données volumineuses

- Les bibliothèques modernes permettent de traiter efficacement des datasets volumineux sans charger entièrement les données en mémoire.
- Nous avons déjà vu Spark pour réaliser ces manipulations !
- Quelques autres outils :



Vaex



# Les formats de fichiers scientifiques et la manipulation de données volumineuses

## Manipuler des données volumineuses



- **Dask** : bibliothèque Python permettant de travailler avec des données volumineuses grâce à une parallélisation simple.
- Utilise une API similaire à Pandas et NumPy, mais traite les données en morceaux pour réduire l'utilisation de la mémoire.
- Compatible avec Pandas, NumPy et scikit-learn.
- Supporte des workflows sur des clusters de calcul.

```
import dask.dataframe as dd

df = dd.read_csv('large_file.csv')
mean_value = df['column'].mean().compute()
print(mean_value)
```

# Les formats de fichiers scientifiques et la manipulation de données volumineuses

## Manipuler des données volumineuses



- **Vaex** : conçu pour analyser rapidement des datasets massifs (plusieurs téraoctets) sans les charger en mémoire.
- Support pour les fichiers HDF5, CSV, et Parquet.
- Calcul rapide grâce à l'utilisation de techniques de mémoire partagée et différée

```
import vaex

df = vaex.open('large_file.hdf5')
print(df.mean('column'))
```



# Les formats de fichiers scientifiques et la manipulation de données volumineuses

## Manipuler des données volumineuses



- **Xarray** : Xarray est conçu pour manipuler des données multidimensionnelles comme celles issues des fichiers NetCDF ou HDF5.
- Idéal pour les grilles régulières (données spatiales ou temporelles).\*
- Intégration avec Dask pour le traitement distribué.

```
import xarray as xr

dataset = xr.open_dataset('data.nc')
temperature = dataset['temperature'].mean(dim='time')
print(temperature)
```

# 05

## Projet final

# Projet final



**Préparer des données  
de consommation  
énergétique de  
bâtiments pour un futur  
modèle prédictif de  
machine learning**

# **Fin de la formation**

## **Merci !**