Les librairies Python pour l'analyse de données

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 avec Python
- Comprendre l'intérêt de la data visualisation, savoir visualiser des données avec Python

Programme

01 **L’écosystème scientifique moderne Python**

02 **La DB API**

03 **La data visualisation**

01

L’écosystème scientifique moderne Python

L'écosystème scientifique moderne Python

Savoir où trouver de nouvelles librairies…

PyPI (Python Package Index)	Le principal dépôt pour les bibliothèques Python.	https://pypi.org
Github	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	https://github.com/search
Awesome Python	Une liste collaborative et bien entretenue des meilleures bibliothèques Python. C'est un excellent point de départ pour trouver des outils spécifiques et éprouvés.	https://awesomepython.org/
Forums	Le subreddit r/Python propose régulièrement des discussions autour de nouvelles bibliothèques Python, avec des retours d'expérience de la communauté 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	https://www.reddit.com/r/Python https://stackoverflow.com/
Newsletters	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 moderne 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 !

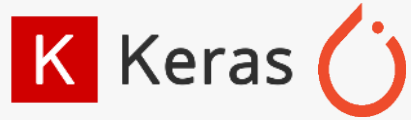
L'écosystème scientifique moderne Python

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 :     (Vaex) et bien d'autres

Visualisation de données :  (matplotlib)  plotly bokeh  Power BI 

Machine Learning et Deep Learning :  XGBoost   

Gestion des bases de données :    

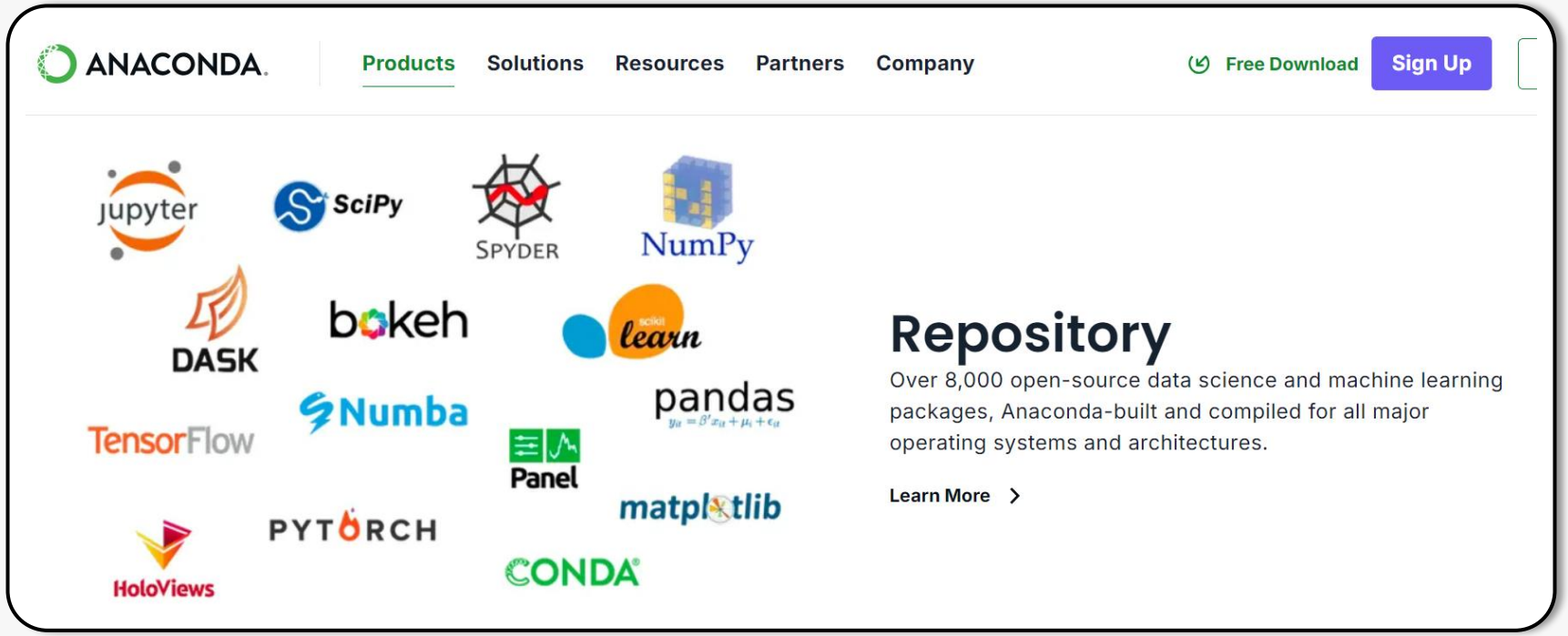
Big Data :    

Gestion des versions et collaboration :   

L'écosystème scientifique moderne Python

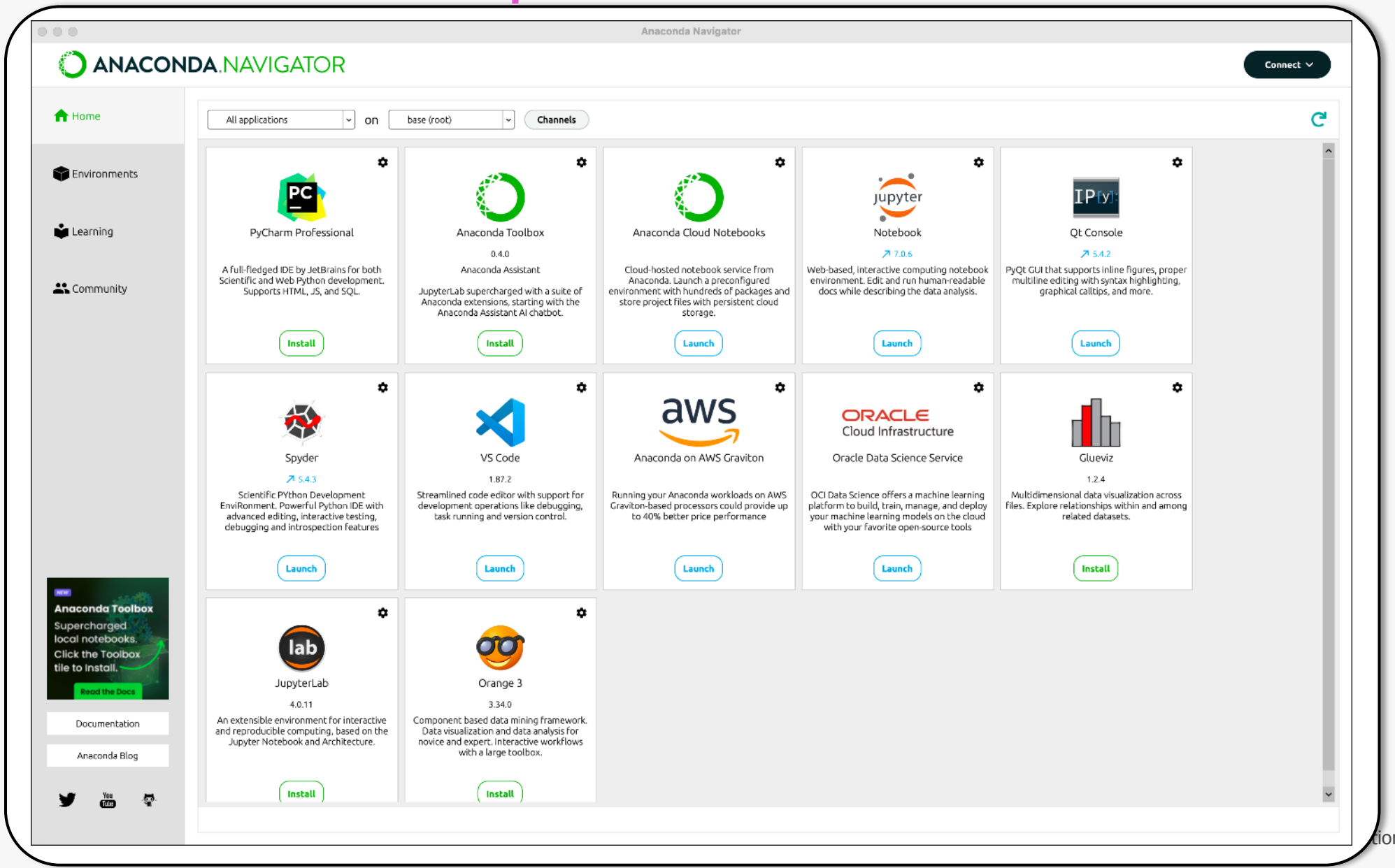
Utiliser une distribution scientifique comme Anaconda

- 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 moderne Python

Utiliser une distribution scientifique comme Anaconda



tion préalable.

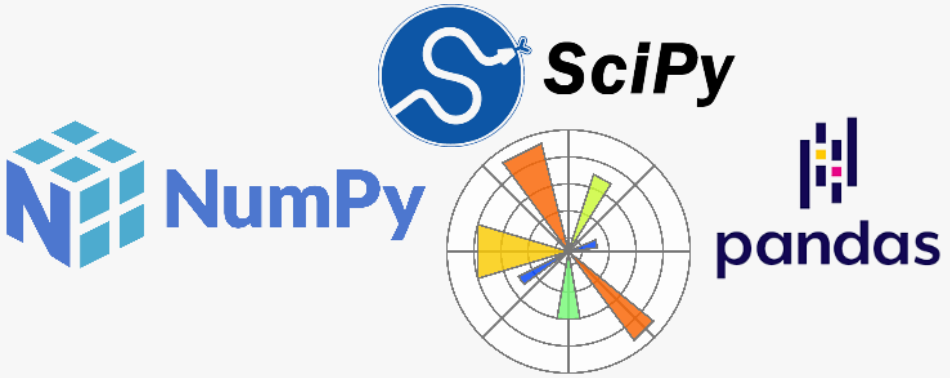
L'écosystème scientifique moderne Python

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 moderne Python

L’écosystème scientifique Python moderne



L'écosystème scientifique moderne Python

L’écosystème scientifique Python moderne - pandas

- 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.

L'écosystème scientifique moderne Python

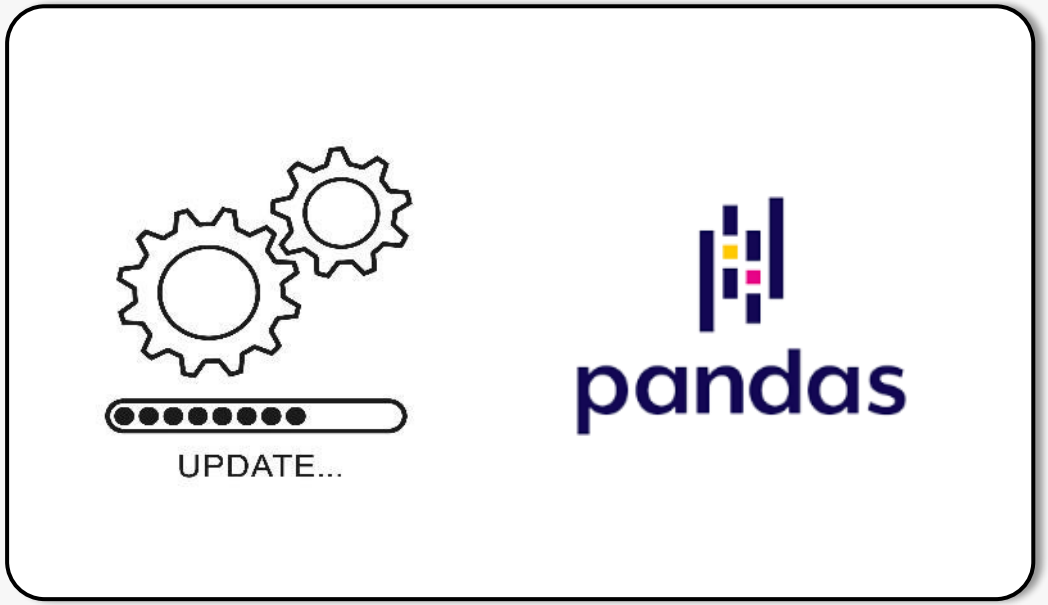
L’écosystème scientifique Python moderne - pandas

- 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.

L'écosystème scientifique moderne Python

L’écosystème scientifique Python moderne - pandas




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

L'écosystème scientifique moderne Python

L’écosystème scientifique Python moderne - pandas

df = *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 (CSV, Excel, JSON, SQL, HDF5…).

La documentation de pandas est disponible à l’adresse suivante :
https://pandas.pydata.org/docs/

L'écosystème scientifique moderne Python

TP pandas #1

Utiliser pandas pour manipuler des données tabulaires temporelles (données Vélib)

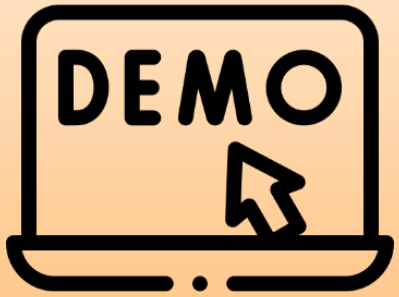
L'écosystème scientifique moderne Python

TP pandas #2

Identifier des données aberrantes

L'écosystème scientifique moderne Python

Distribuer des traitements type « pandas »



Traitement équivalent avec dask et pyspark

dask PySpark

L'écosystème scientifique moderne Python

TP bilan pandas

Calcul de deltas dans un portefeuille boursier

L'écosystème scientifique moderne Python

L’écosystème scientifique Python moderne - NumPy

- 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…).

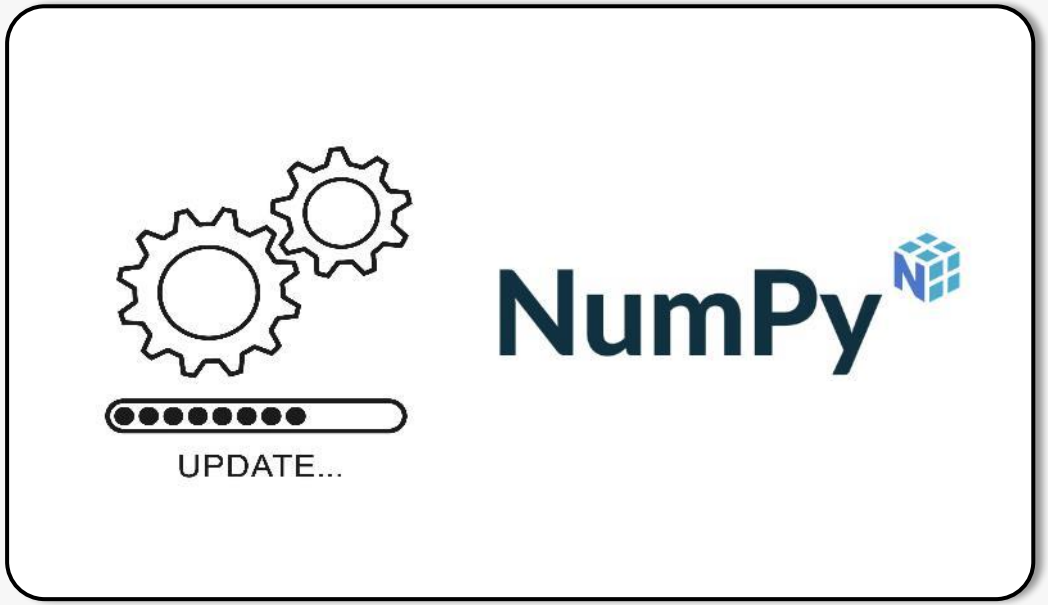
L'écosystème scientifique moderne Python

L’écosystème scientifique Python moderne - NumPy

- 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.

L'écosystème scientifique moderne Python

L’écosystème scientifique Python moderne - NumPy




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

L'écosystème scientifique moderne Python

L'écosystème scientifique Python moderne - NumPy

`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/

L'écosystème scientifique moderne Python

L’écosystème scientifique Python moderne - SciPy

- 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).

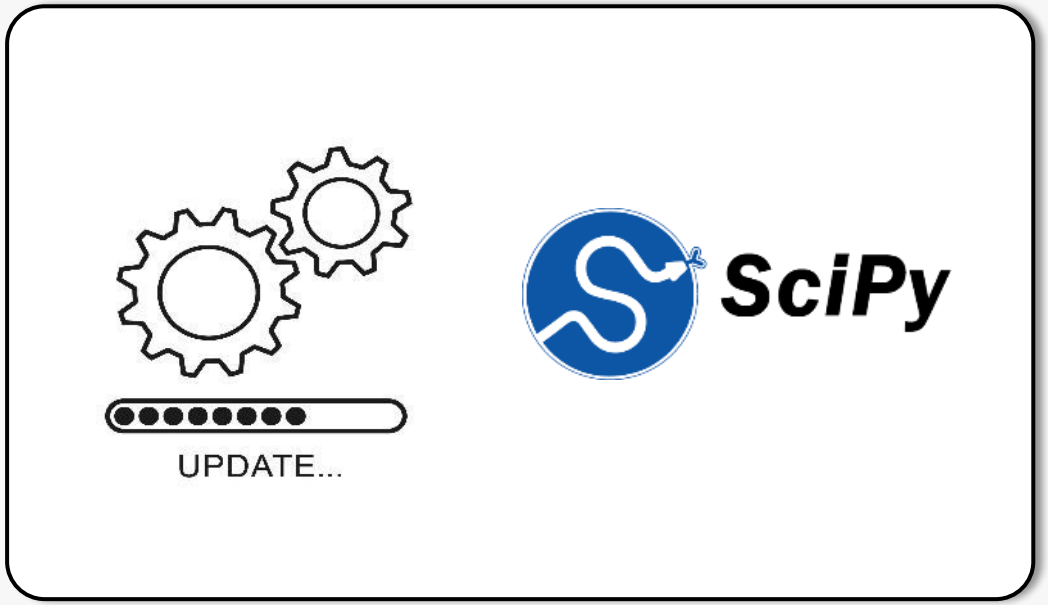
L'écosystème scientifique moderne Python

L’écosystème scientifique Python moderne - SciPy

- 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.

L'écosystème scientifique moderne Python

L’écosystème scientifique Python moderne - SciPy




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

L'écosystème scientifique moderne Python

L'écosystème scientifique Python moderne - SciPy

`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/

L'écosystème scientifique moderne Python

L’écosystème scientifique Python moderne - Matplotlib

- **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.

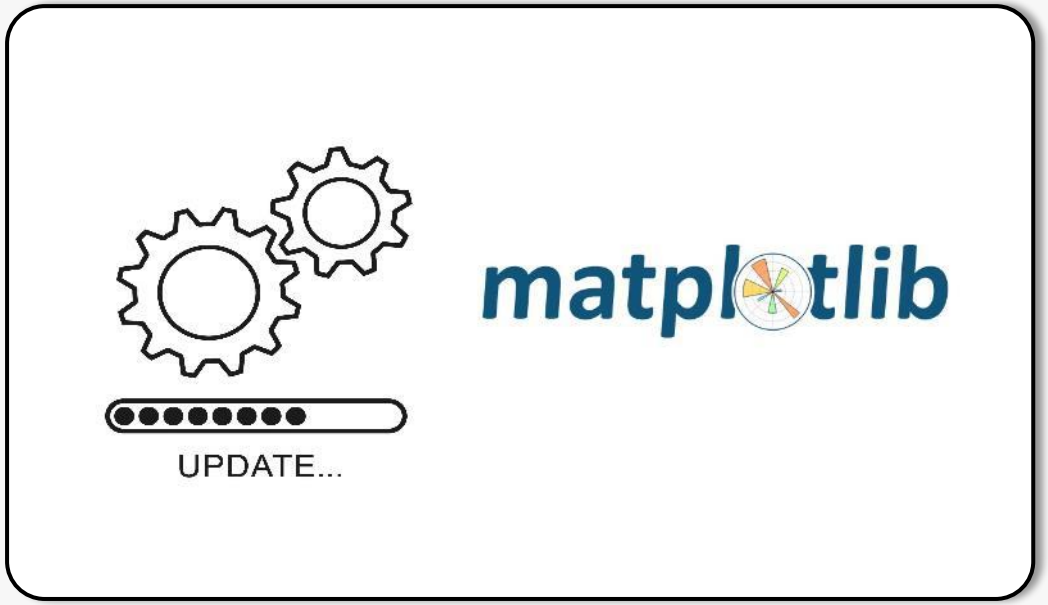
L'écosystème scientifique moderne Python

L’écosystème scientifique Python moderne - Matplotlib

- Matplotlib fonctionne en fournissant une **interface orientée objet** pour la création de graphiques et de tracés en Python. Le sous-module `pyplot` 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.

L'écosystème scientifique moderne Python

L’écosystème scientifique Python moderne - Matplotlib




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

L'écosystème scientifique moderne Python

L'écosystème scientifique Python moderne - Matplotlib

`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

L'écosystème scientifique moderne Python

Le stack scientifique plus au complet

Où est Charlie ?

L'écosystème scientifique moderne Python

Le stack scientifique plus au complet

Apple stock price data

L'écosystème scientifique moderne Python

L'écosystème scientifique Python moderne - 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	Algorithmes de Machine Learning	Évaluation des modèles	Optimisation de modèles
scikit-learn inclut des méthodes pour normaliser, transformer et manipuler les données afin de les préparer pour le modèle	**scikit-learn propose un (large) éventail de modèles pour des tâches de classification, régression, clustering, réduction de dimensionnalité, et sélection de modèles**	**scikit-learn fournit des fonctions pour évaluer la performance des modèles grâce à des métriques standard et des techniques de validation croisée**	**scikit-learn permet de rechercher les meilleurs hyperparamètres pour un modèle à l'aide de techniques telles que GridSearchCV et RandomizedSearchCV**

02

La DB API

La DB API

Les accès aux bases de données relationnelles, le fonctionnement de la DB API

DB API 2.0 (PEP 249) : norme Python qui définit une interface commune pour interagir avec **toutes** les bases relationnelles (SQLite, PostgreSQL, MySQL, Oracle…).

Objectif : écrire du code Python indépendant du SGBD autant que possible.

Avantages :

- **Uniformité** : même interface pour SQLite, PostgreSQL, MySQL…
- **Sécurité** : placeholders pour éviter les injections SQL
- **Contrôle des transactions** : commit() et rollback()
- **Facilité d'usage** avec curseurs et méthodes fetch

SGBD	Module Python
SQLite	sqlite3
PostgreSQL	psycopg2
MySQL	mysql-connector-python
Oracle	cx_Oracle

La DB API

Les accès aux bases de données relationnelles, le fonctionnement de la DB API

Étapes générales :

1. Importer le module Python correspondant au SGBD (SQLite, PostgreSQL, MySQL, etc.).
2. Créer une connexion à la base de données.
3. Créer un curseur à partir de la connexion.
4. Exécuter des commandes SQL via le curseur (SELECT, INSERT, UPDATE, DELETE, etc.).
5. Récupérer les résultats des requêtes si nécessaire (fetchone(), fetchall()).
6. Valider les modifications avec commit() pour les commandes modifiant la base.
7. Gérer les erreurs et éventuellement effectuer un rollback() en cas d'exception.
8. Fermer le curseur et la connexion à la base.

La DB API

Les accès aux bases de données relationnelles, le fonctionnement de la DB API

Concept	Description
Connection	Objet représentant la connexion à la base. Permet de créer des curseurs et de gérer les transactions.
Cursor	Objet pour exécuter des requêtes SQL et récupérer les résultats.
Placeholders	Permettent d'éviter l'injection SQL. Syntaxe : ? pour SQLite, %s pour PostgreSQL/MySQL.
commit()	Valide les changements pour INSERT/UPDATE/DELETE.
rollback()	Annule les changements non commités en cas d'erreur.
fetchone() / fetchall()	Méthodes pour récupérer les résultats d'un SELECT.

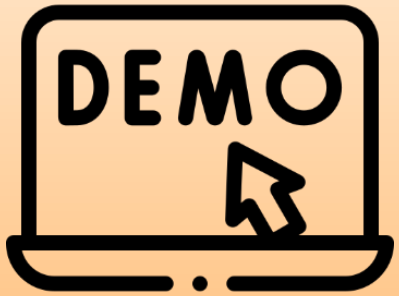
La DB API

TP

sqlite3 pour SQLite

La DB API

Démo



Psycopg2 pour POSTGRESQL

03

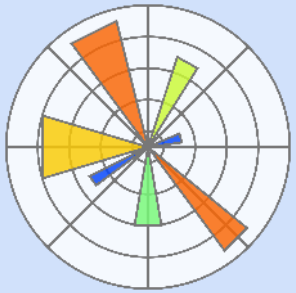
La data visualisation

La data visualisation

Les librairies de visualisation

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

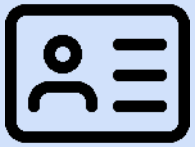
Construite sur Matplotlib, Seaborn simplifie la création de visualisations statistiques avancées. Elle est particulièrement adaptée aux visualisations basées sur des ensembles de données complexes.

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

La data visualisation

Les librairies de visualisation

Visualisation 2D/3D

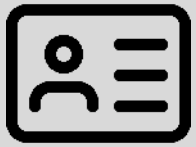


Plotly

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

bokeh

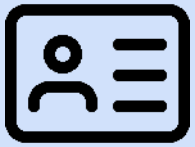
Permet de créer des visualisations interactives en 2D et en 3D, souvent utilisées pour des dashboards. Prend en charge les web apps en temps réel.

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

La data visualisation

Les librairies de visualisation

Visualisation 2D/3D



Altair

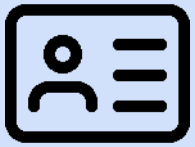
Librairie déclarative pour la visualisation de données. Utilise un langage basé sur JSON pour décrire les visualisations, ce qui la rend plus intuitive

Ex. : Graphiques interactifs, visualisations statistiques simples.

La data visualisation

Les librairies de visualisation

Visualisation web (interactive et dashboards)

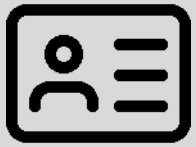


Dash (Plotly)

Dash by 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)

Panel

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.

La data visualisation

Les librairies de visualisation

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

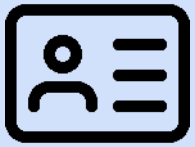
Bien que principalement une librairie pour l'analyse statistique, Statsmodels propose également des visualisations pour analyser des régressions et des séries temporelles

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

La data visualisation

Les librairies de visualisation

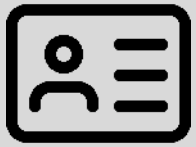
Cartographie et visualisation géospatiale



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 librairie 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.

La data visualisation

Les librairies de visualisation

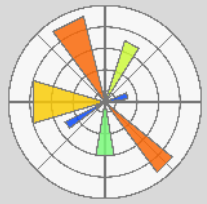
Cartographie et visualisation géospatiale

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.

La data visualisation

Les librairies de visualisation

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)

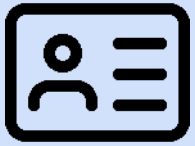
Conçue pour visualiser de très grands ensembles de données (Big Data), HoloViews se base sur des abstractions de haut niveau qui permettent une visualisation rapide.

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

La data visualisation

Les librairies de visualisation

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.

La data visualisation

Les librairies de visualisation

Autres librairies et outils utiles pour la visualisation

Pyvis

PyViz

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.

La data visualisation

TP

Représentation de l’emplacement des aéroports français et américains sur une carte

Fin de cette partie

Merci !