Data analysis and vizualisation with Python libraries

*

Objectives

- To have an overview of the scientific ecosystem of Python
- To know the essential scientific libraries for data science
- To be able to manipulate data with Python
- To understand the importance of data visualisation, to know how to visualise data with Python

Programme

01

The modern scientific Python ecosystem

The modern scientific Python ecosystem Knowing where to find new libraries…

PyPI **(Python Package Index)**	The main repository for Python libraries.	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	Git server. Many Python libraries are hosted on GitHub. It’s an excellent platform for discovering open-source projects and seeing contributions in real time.	https://awesomepython.org/
Forums	The r/Python subreddit regularly features discussions about new Python libraries, with feedback and real-world experience shared by the community. Stack Overflow: answers to development questions often include recommendations for popular or emerging libraries. It is also a good indicator of the size and activity of a library’s community.	https://www.reddit.com/r/Python https://stackoverflow.com/
Newsletters	The (many) newsletters share discoveries of new Python libraries—mostly on a weekly basis—along with discussions about trending tools.	

The modern scientific Python ecosystem … and how to assess their longevity

- **Popularity/adoption** (number of downloads in the 'Downloads' section on PyPI, number of forks and stars on GitHub, mentions in articles and usage in popular projects/established companies)
- **Update frequency** (date of the last update, commit history, regular version cycles, documented changes)
- **Community support and responsiveness** (issues, maintainer response times)
- **Documentation** (completeness, accuracy, clarity, structure…)
- **Dependencies and compatibility** (minimal and well-managed dependencies, tracking recent Python versions)

And, even though it's of a different nature, don't forget to inquire about the licence and terms of use!

The modern scientific Python ecosystem - Beyond the scientific ecosystem, the technical ecosystem for data science

IDE: PC jupyter jupyterlab spyder

Data manipulation and processing: pandas dask PySpark (Vaex) and many others

Data visualisation: (matplotlib) seaborn plotly bokeh tableau Power BI

Machine Learning and Deep Learning: scikit learn XGBoost LightGBM K Keras

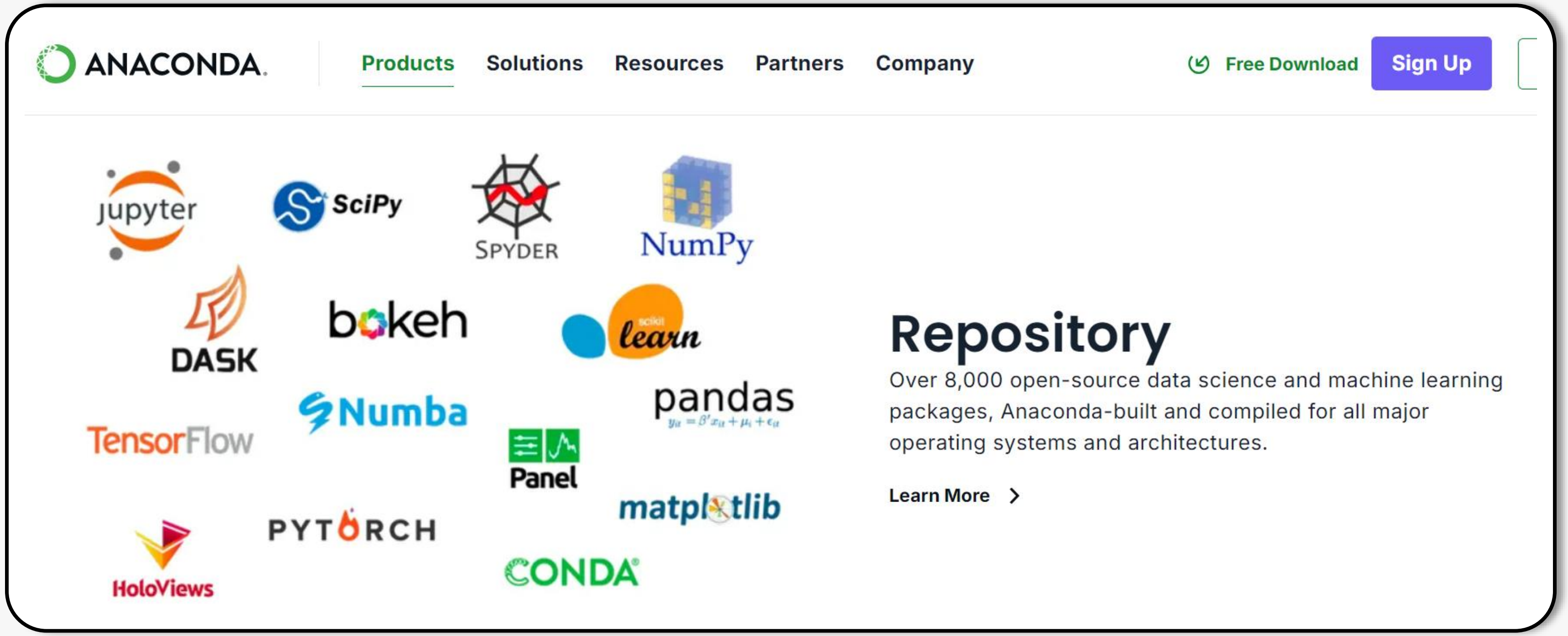
Database management: MySQL PostgreSQL MongoDB SQLite

Big Data: hadoop PySpark dask kafka

Version control and collaboration: git GitLab GitHub

The modern scientific Python ecosystem - Using a scientific distribution like Anaconda

- There are Python distributions that simplify installation, package management, and virtual environments for Data Science and machine learning projects.
- For a scientist, it may be beneficial to use such a distribution.
- Anaconda is one of those distributions (it is also a distribution of R). It is open-source.



Repository

Over 8,000 open-source data science and machine learning packages, Anaconda-built and compiled for all major operating systems and architectures.

The modern Python scientific ecosystem

The modern Python scientific ecosystem

SciPy

NumPy

pandas

The modern Python scientific ecosystem

The modern Python scientific ecosystem - pandas

- Pandas is a data manipulation and analysis library in Python. It provides efficient and easy-to-use data structures, such as DataFrames, which allow for working with tabular data.
- Launched in 2008 by Wes McKinney, Pandas quickly established itself as an essential tool for data science. It is used to import, clean, transform, and analyse heterogeneous datasets.
- Pandas is optimised for handling large datasets thanks to its integration capabilities with NumPy and other Python libraries.

The modern Python scientific ecosystem

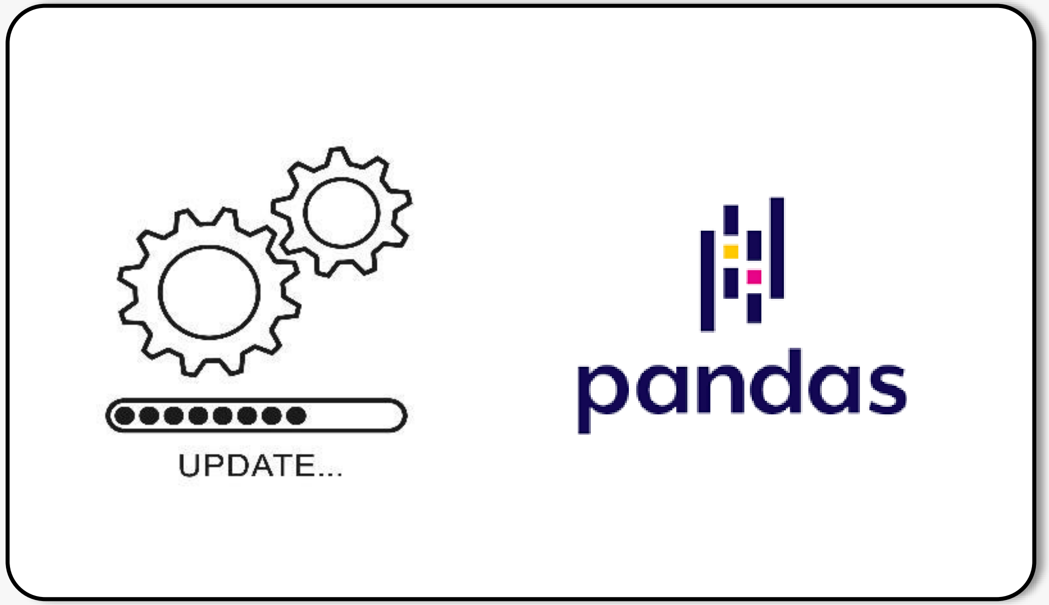
The modern Python scientific ecosystem - pandas

- Pandas DataFrames allow you to manipulate data organised in rows and columns, similar to Excel spreadsheets. Pandas also supports indexing, making data access, sorting, and selection easier.

- The library is ideal for data cleaning tasks, merging datasets, and calculating descriptive statistics. It is often used in conjunction with other tools such as Matplotlib for visualisation or scikit-learn for machine learning.

The modern scientific Python ecosystem

The modern Python scientific ecosystem - pandas



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

The modern scientific Python ecosystem

The modern scientific Python ecosystem - pandas

df = pandas.DataFrame() creates a data table.

df = pandas.read_csv(), df = pandas.read_excel(), df = pandas.read_sql(), etc. allow Pandas to interact with external data sources.

df.head() displays the first rows, df.tail() displays the last rows.

df.describe() provides a statistical summary of the numeric columns.

df.groupby() groups the data to apply aggregate operations.

df.merge() merges multiple DataFrames.

df.isna() identifies missing values, df.fillna() fills them in.

Pandas is compatible with various formats (CSV, Excel, JSON, SQL, HDF5…).

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

The modern scientific Python ecosystem - pandas lab #1

Using pandas to manipulate time-series tabular data

The modern scientific Python ecosystem - pandas lab #1

Identifying outliers

The modern scientific Python ecosystem – distributing pandas type data processings

DEMO

Equivalent processing with dask and pyspark

dask PySpark

The modern scientific Python ecosystem - pandas lab #3: balance sheet

Calculating deltas in a stock portfolio

The modern Python scientific ecosystem

The modern Python scientific ecosystem - NumPy

- NumPy, short for 'Numerical Python', is a fundamental library in Python for numerical computation and the manipulation of multidimensional arrays. It is widely used in various fields, including image processing.

- It was created in 2005 by Travis Oliphant, based on previous work. NumPy is open source and has seen rapid adoption among scientific and engineering communities. It laid the groundwork for many other scientific computing libraries in Python (scipy, matplotlib, scikit-learn…).

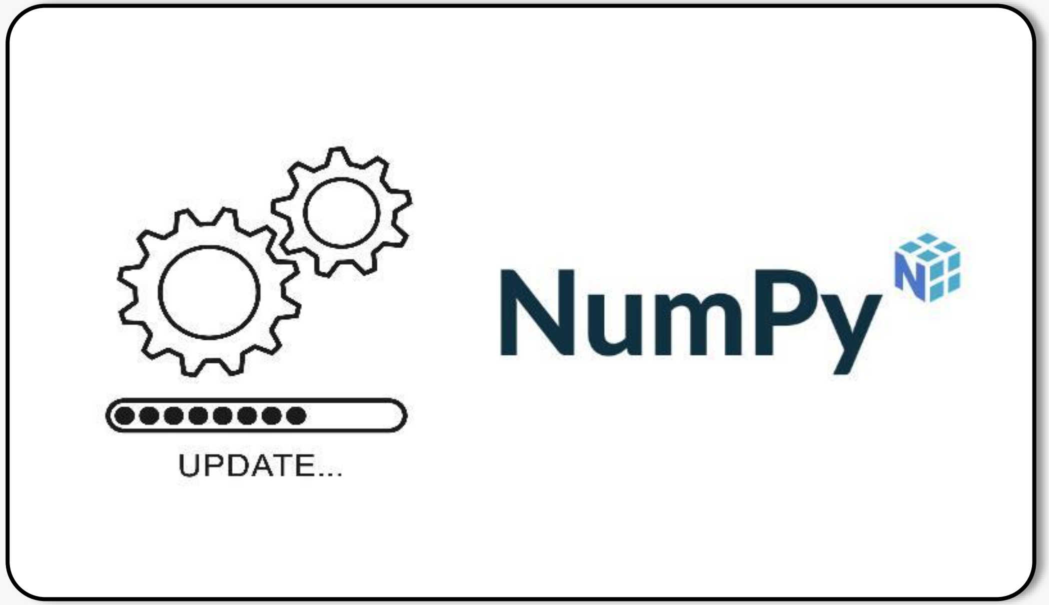
The modern Python scientific ecosystem

The modern Python scientific ecosystem - NumPy

- NumPy is built around the concept of NumPy arrays (ndarrays), which are homogeneous multidimensional data structures that are efficient for storing and manipulating numerical data.
- NumPy arrays are similar to Python lists, but they are optimised for numerical operations. They can have multiple dimensions and support vectorised operations, making them ideal for numerical computing.
- NumPy provides functions for performing vectorised operations on arrays, allowing efficient and parallel calculations on large amounts of data.
- NumPy is commonly used for processing and analysing data, particularly in the fields of data science, machine learning, and image processing.

The modern Python scientific ecosystem

The modern Python scientific ecosystem - NumPy



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

The modern Python scientific ecosystem

The modern Python scientific ecosystem - NumPy

numpy.array() creates a NumPy array from a sequence of data.

numpy.zeros(), numpy.ones(), numpy.empty() create arrays filled with zeros, ones, or empty values. The size needs to be specified.

numpy.arange(), numpy.linspace() generate sequences of numbers.

numpy.shape(), numpy.reshape(), numpy.dim() allow manipulation of the shape and dimensions of arrays.

numpy.mean(), numpy.max(), numpy.min(), numpy.std() compute statistics on the data.

It is possible to access and manipulate the elements of an array using indices and slicing operations.

La documentation de numpy est disponible à l'adresse suivante :
https://numpy.org/doc/

The modern Python scientific ecosystem

The modern Python scientific ecosystem - SciPy

- SciPy, short for 'Scientific Python', is a library designed for scientific and technical computing. It is built on NumPy and provides a wide range of algorithms for tasks such as integration, optimisation, interpolation, linear algebra, and signal processing.
- Created in 2001 by Travis Oliphant, Eric Jones, and Pearu Peterson, SciPy has played a crucial role in the rise of Python as a leading language for data science and research. It is open source and benefits from an active community of developers and users.
- SciPy is designed to be modular, with several specialised sub-modules such as scipy.integrate (integration), scipy.optimize (optimisation), or scipy.spatial (geometry and distances).

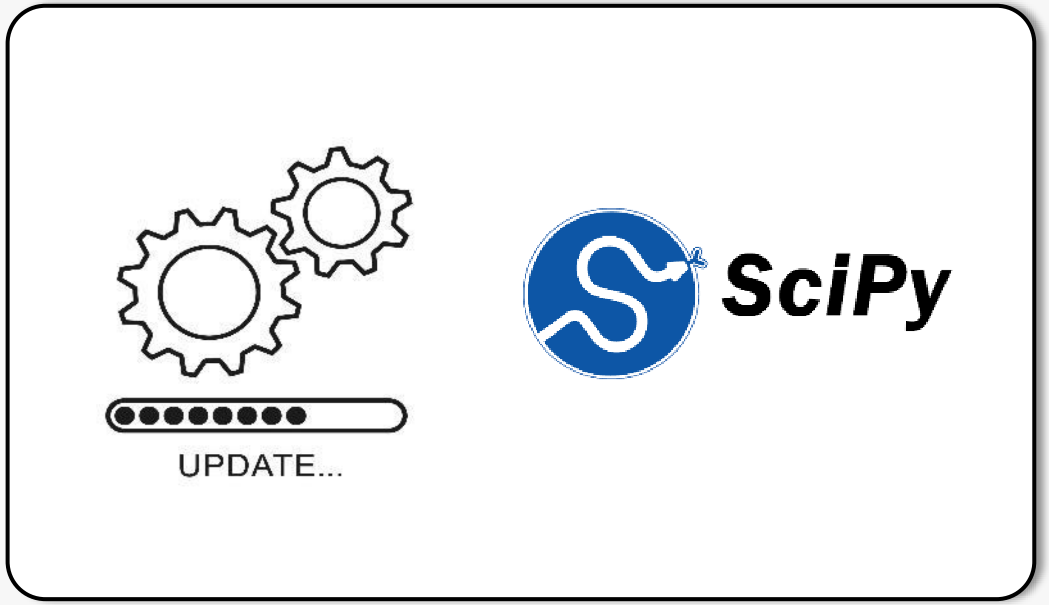
The modern Python scientific ecosystem

The modern Python scientific ecosystem - SciPy

- SciPy relies on NumPy for data manipulation and focuses on advanced analysis tools. It is commonly used in various fields such as experimental data analysis, physics, bioinformatics, and engineering.
- Its modules allow for solving differential equations, performing advanced matrix decompositions, and analysing statistical distributions. For example, the function scipy.stats provides tools for conducting hypothesis tests or fitting distributions.
- With its optimised performance, SciPy allows for the efficient solving of complex problems.

The modern scientific ecosystem of Python

The modern Python scientific ecosystem - SciPy



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

The modern scientific ecosystem of Python

The modern scientific Python ecosystem - SciPy

`scipy.optimize.minimize()` solves optimisation problems.

`scipy.integrate.quad()` calculates the integral of a function.

`scipy.interpolate.interp1d()` interpolates data points.

`scipy.linalg.eig()` calculates the eigenvalues of a matrix.

`scipy.stats.norm.pdf()` provides the probability density of a normal distribution.

SciPy also allows manipulation of data structures like sparse matrices via scipy.sparse.

La documentation de scipy est disponible à l'adresse suivante :
https://docs.scipy.org/doc/scipy/

The modern scientific Python ecosystem

The modern scientific Python ecosystem - Matplotlib

- **Matplotlib is a flexible data visualisation library for Python. It is used to create a wide variety of graphs, plots, and visualisations, whether for data analysis, presentations, or scientific reports. It is also capable of displaying and processing basic images, making it a versatile tool for visualisation and image processing.**
- It was created by John D. Hunter in 2003. He developed this library to help scientists and engineers visualise their data effectively and aesthetically.

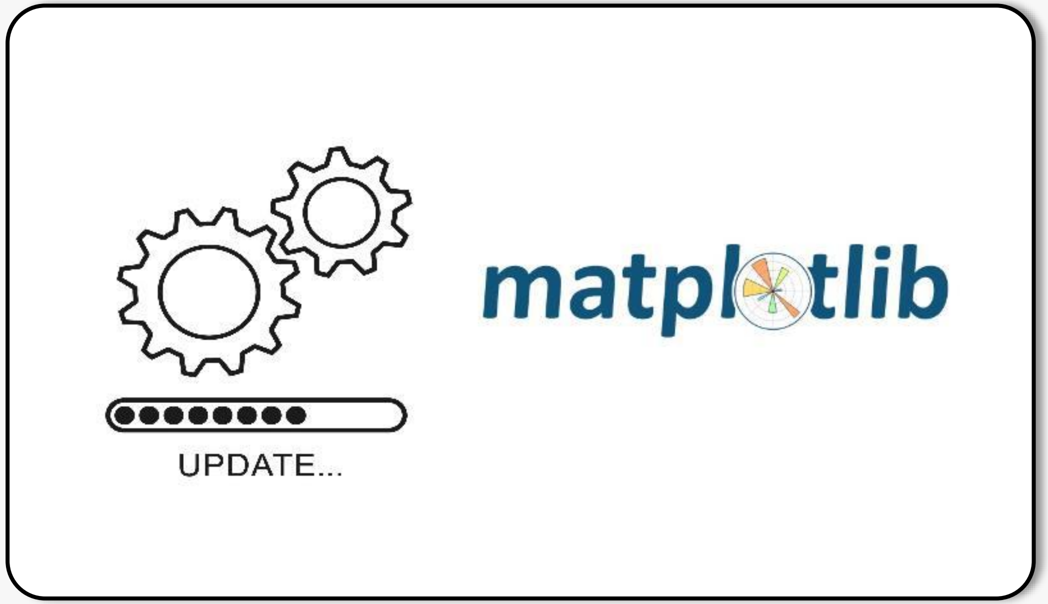
The modern Python scientific ecosystem

The modern Python scientific ecosystem - Matplotlib

- Matplotlib works by providing an object-oriented interface for creating graphs and plots in Python. The pyplot submodule provides a simplified interface for quickly building visualisations.
- Matplotlib allows users to create figures and axes that can be extensively customised. Users can add elements such as lines, points, legends, titles, etc., to graphs in a 'programmatic' manner.
- Matplotlib supports a wide variety of chart types, including scatter plots, histograms, bar charts, box plots, line graphs, scattergraphs, pie charts, and more.
- Matplotlib allows for advanced customisation of graphs, with the ability to control every visual aspect, including colours, line styles, fonts, labels, axes, and scales.
- It is often used with Pandas for plotting tabular data or with NumPy to visualise raw numerical data.

The modern Python scientific ecosystem

The modern Python scientific ecosystem - Matplotlib



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

The modern Python scientific ecosystem

The modern scientific Python ecosystem - Matplotlib

`plt.figure(), plt.subplot()` allow you to create figures and subplots.

`plt.plot(), plt.scatter(), plt.bar()` create different types of charts.

`plt.xlabel(), plt.ylabel(), plt.title(), plt.legend()` add legend/title elements to the visualisation

`plt.savefig()` saves the chart to an image file.

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

The modern scientific Python ecosystem – NumPy lab

Where's Wally?

The modern scientific Python ecosystem – another lab

Apple stock price data

The modern Python scientific ecosystem

The modern Python scientific ecosystem - scikit-learn

scikit-learn is an (excellent) Python library for Machine Learning, providing a set of standardised tools for supervised and unsupervised learning. It is commonly used as a catalogue of implemented algorithms and, more generally, for building models, transforming data, and evaluating the performance of Machine Learning algorithms.

Data Preprocessing

scikit-learn includes methods for normalising, transforming, and manipulating data to prepare it for the model

Machine Learning Algorithms

scikit-learn offers a (wide) range of models for tasks such as classification, regression, clustering, dimensionality reduction, and model selection

Model Evaluation

scikit-learn provides functions to evaluate model performance using standard metrics and cross-validation techniques

Model optimisation

scikit-learn allows for the search of the best hyperparameters for a model using techniques such as GridSearchCV and RandomizedSearchCV

02

The DB API

The DB API

Access to relational databases, the operation of the DB API

DB API 2.0 (PEP 249): a Python standard that defines a common interface for interacting with all relational databases (SQLite, PostgreSQL, MySQL, Oracle…).

Objective: to write Python code that is as database-independent as possible.

Advantages:

- Uniformity: same interface for SQLite, PostgreSQL, MySQL…
- Security: placeholders to prevent SQL injections
- Transaction control: commit() and rollback()
- Ease of use with cursors and fetch methods

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

The DB API

Access to relational databases, the functioning of the DB API

General steps:

1. Import the corresponding Python module for the database (SQLite, PostgreSQL, MySQL, etc.).
2. Create a connection to the database.
3. Create a cursor from the connection.
4. Execute SQL commands via the cursor (SELECT, INSERT, UPDATE, DELETE, etc.).
5. Retrieve the results of queries if necessary (fetchone(), fetchall()).
6. Commit changes with commit() for commands that modify the database.
7. Handle errors and optionally perform a rollback() in case of an exception.
8. Close the cursor and the connection to the database.

The DB API

Access to relational databases, the functioning of the DB API

Concept	Description
Connection	Object representing the database connection. It allows the creation of cursors and the management of transactions.
Cursor	An object used to execute SQL queries and retrieve results.
Placeholders	They help prevent SQL injection. Syntax: ? for SQLite, %s for PostgreSQL/MySQL.
commit()	Validates and commits changes made by INSERT, UPDATE, and DELETE operations.
rollback()	Reverts any uncommitted changes if an error occurs.
fetchone() / fetchall()	Methods for retrieving the results of a SELECT query.

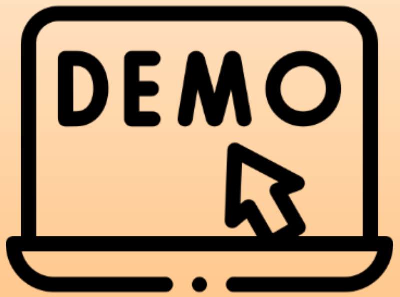
The DB API – Working with SQL databases

Lab

sqlite3 for SQLite

The DB API

Demo



Psycopg2 for POSTGRESQL

03

Data visualisation

Data visualisation

Visualisation libraries

2D/3D Visualisation

Matplotlib

It is the most popular visualisation library for 2D graphs in Python. It allows for the creation of classic graphs such as histograms, line graphs, box plots, and so on.

E.g.: Line graphs, scatter plots, histograms.

Seaborn

Built on Matplotlib, Seaborn simplifies the creation of advanced statistical visualisations. It is particularly suited for visualisations based on complex data sets.

E.g.: Heat maps, box plots, pair plots, data distributions.

Data visualisation

Visualisation libraries

2D/3D Visualisation

Plotly

Offers interactive graphs and 3D visualisations. It allows for the creation of highly aesthetic and interactive graphs that can be embedded in websites.

E.g.: Interactive graphs in 2D/3D, maps, time series visualisations.

Bokeh

Allows the creation of interactive visualisations in 2D and 3D, often used for dashboards. Supports real-time web apps.

E.g.: Time series visualisation, interactive bar charts.

Data visualisation

Visualisation libraries

2D/3D visualisation

Altair

Declarative library for data visualisation. Uses a JSON-based language to describe the visualisations, making it more intuitive.

E.g.: Interactive graphs, simple statistical visualisations.

Data visualisation

Visualisation libraries

Web visualisation (interactive and dashboards)

Dash (Plotly)

Allows the creation of interactive web applications. Very popular for dashboards and real-time data visualisation.

E.g.: Interactive dashboards, real-time visualisations.

Panel (HoloViz)

Library based on Bokeh for creating interactive dashboards. It offers great flexibility for integrating widgets and interactive controls.

E.g.: Interface for adjusting real-time visualisation parameters.

Data visualisation

Visualisation libraries

Statistical data visualisation

ggplot (ggplot2)

A port of the famous ggplot2 library from R. It follows a declarative approach to create visualisations based on the grammar of graphics.

E.g. : Elegant statistical graphics.

Statsmodels

Although primarily a library for statistical analysis, Statsmodels also offers visualisations for analysing regressions and time series.

E.g. : Residual plots, autocorrelation, time series analysis.

Data visualisation

Visualisation libraries

Mapping and geospatial visualisation

Geopandas

Pandas extension, Geopandas allows for working with geospatial data (geometric and geographic shapes) and easily visualising it.

Ex.: Maps, choropleth maps, visualisation of geographic areas.

Folium

Based on the JavaScript library Leaflet, Folium allows for the creation of interactive maps in Python. Ideal for visualising interactive geospatial data.

Ex.: Interactive maps, real-time geospatial data.

Data visualisation

Visualisation libraries

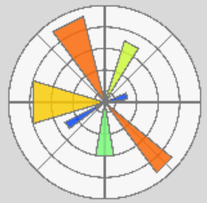
Mapping and geospatial visualisation

Cartopy

Library for geospatial mapping and the visualisation of climate and environmental data.

e.g. : Climate and geospatial data maps.

Basemap



Matplotlib extension for creating geographical maps and spatial data visualisations.

e.g. : Topographic maps, temperature maps, trajectory maps.

Data visualisation

Visualisation libraries

Big Data and visualisation of large datasets

Datashader (HoloViz)

A library designed for visualising very large amounts of data. It enables the creation of real-time graphics for massive datasets.

e.g. : Visualisation of massive data, such as spatial or temporal data in real time.

HoloViews (HoloViz)

Designed to visualise very large datasets (Big Data), HoloViews is based on high-level abstractions that allow for rapid visualisation.

E.g.: Interactive charts for massive data, visualisation of time series.

Data visualisation
Visualisation libraries

Big Data and visualisation of large datasets

Vaex

A library specialising in the visualisation and analysis of very large datasets. It is optimised to work on large datasets without requiring a lot of memory.

E.g.: Visualisation of large datasets in interactive charts.

Data visualisation

Visualisation libraries

Other libraries and useful tools for visualisation

Pyvis

A library that allows for the visualisation of interactive graphs (networks). It is used to represent graphs in an intuitive and interactive way.

E.g.: Visualisation of social networks, relational graphs.

WordCloud

Specifically designed to create word clouds, a simple yet powerful visual representation of word frequencies in a text.

E.g.: Text analysis, visualisation of keywords in corpora.

Data visualisation - Lab

Vizualising the location of French and American airports on a map

04

Time Series Analysis and Stochastic Models

Time Series Analysis and Stochastic Models

Time series analysis studies data indexed over time in order to understand patterns, dynamics, and uncertainty.

Key concepts:

- Time series components: trend, seasonality, noise
- Stationarity and transformations (differencing, scaling)
- Stochastic processes: white noise, random walk, AR, MA, ARMA
- Autocorrelation (ACF) and Partial Autocorrelation (PACF)

Time Series Analysis and Stochastic Models

Time series analysis studies data indexed over time in order to understand patterns, dynamics, and uncertainty.

Modeling & Forecasting

- Linear models: ARIMA / SARIMA
- Model selection and diagnostics (AIC, residual analysis)
- Short-term forecasting and uncertainty quantification

Python Ecosystem

- pandas → time series manipulation
- numpy → numerical computation
- statsmodels → ARIMA, ACF/PACF, statistical tests
- matplotlib / seaborn → visualization

Applications: finance, economics, signal processing, climate data, anomaly detection…

End of this section

Thank you!