SQL in production

An introduction to programming for data professionals

Why we need programming

Data-driven world

In today's data-driven world, organizations and industries rely heavily on data to make informed decisions and gain a competitive edge.

Volume, variety, velocity

Data are generated at an unprecedented scale, with diverse formats and arriving at high speeds. This is where the challenges lie.

Insights

The ultimate goal of working with data is to uncover valuable insights. Data-driven insights drive strategic decisions, enhance customer experiences, optimize operations, and enable innovation.

Programming languages act as the bridge between raw data and actionable insights, enabling professionals to handle massive datasets, build models, develop data pipelines, and deploy their solutions into the real world.

Applications of programming

Analysis and visualisation

Exploratory Data Analysis (EDA): Programming languages like Python and R empower data scientists to explore data, identify patterns, and gain initial insights.

Visualization: Through programming, professionals create informative data visualizations that facilitate better understanding and communication of insights.

Modelling and Machine Learning (ML)

Machine Learning: Programming is the backbone of machine learning, allowing the development and training of predictive models.

Algorithm implementation: Data engineers and scientists use programming to implement algorithms, fine-tune models, and optimize performance.

Data pipelines and automation

Data pipelines: Data engineers design and implement data pipelines that automate the movement, transformation, and storage of data. Programming is vital for orchestrating these pipelines.

Automation: Programming enables the automation of repetitive tasks, ensuring efficiency and consistency in data workflows.

And many others, such as **real-time data processing**, **customizing solutions**, **scalability and performance**, and **collaboration and reproducibility**.

But, beyond data, programming fosters **problem-solving skills** critical for tackling **complex**, **real-world challenges**.

Python for data professionals

Python traits

A dynamic, open-source, high-level programming language renowned for its simplicity, readability, and versatility.

Ease of use

It often serves as the go-to language for data professionals due to its vast ecosystem of libraries and easy integration with data tools.

Data science and engineering

Its ease of use and vast ecosystem of libraries and tools make it the preferred choice for data science tasks such as data cleaning, analysis, and machine learning model development.

It is also widely adopted for data engineering tasks such as building data pipelines, ETL (Extract, Transform, Load) processes, and big data workflows.

Python libraries

Python libraries provide **pre-built tools and functions** that simplify complex programming tasks such as doing numerical operations, visualizing data, and creating data models.

NumPy

A library for **numerical** and **scientific computing** that forms the foundation for many other data-related libraries.

Pandas

A **data manipulation** and **analysis** library that introduces **other data structures**, enabling us to work with **structured data** more efficiently.

Matplotlib

A fundamental **visualization library** that provides a flexible and customizable **plotting framework** for charts, graphs, and plots.

Scikit-Learn

A library that offers a wide range of **machine learning algorithms and tools** for model selection, evaluation, and deployment.

TensorFlow

A deep learning framework library for building and training complex **deep learning models**.

And many others, such as **pymysql** for a MySQL client, **OpenCV** for computer vision, and **pytest** for writing and running unit tests in Python.

Conflicting dependencies and configurations

alx

In order to leverage the full potential of a programming language such as Python, we will most likely need to **install many libraries and packages**. However, this often **introduces dependencies and configuration issues**.

Let's look at how this occurs:

We are working on **two different data projects** simultaneously. Project A requires version 1.0 of library X, but Project B requires a newer version of library X.

As we install version 2.0 of library X for Project B, version 1.0 is **overwritten**. Our code for Project A will most likely **no longer work as expected**. In other words, we have conflicting dependencies and configurations.

Operating system

Project A — Requires **library X (Version 1.0)**

Project B — Requires **library X (Version 2.0)**

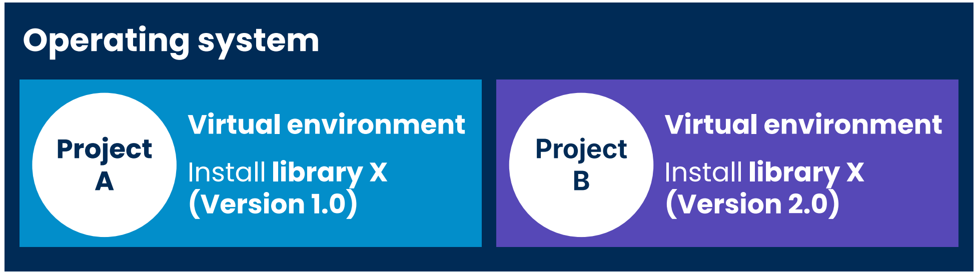
It becomes **challenging to switch** between our projects without uninstalling and reinstalling libraries. This process is **error-prone** and **time-consuming**.

Virtual environments

Virtual environments are **isolated**, **self-contained workspaces** within a computer's operating system where we can create, manage, and run projects with their own sets of **dependencies** and **configurations**.

What if we used virtual environments?

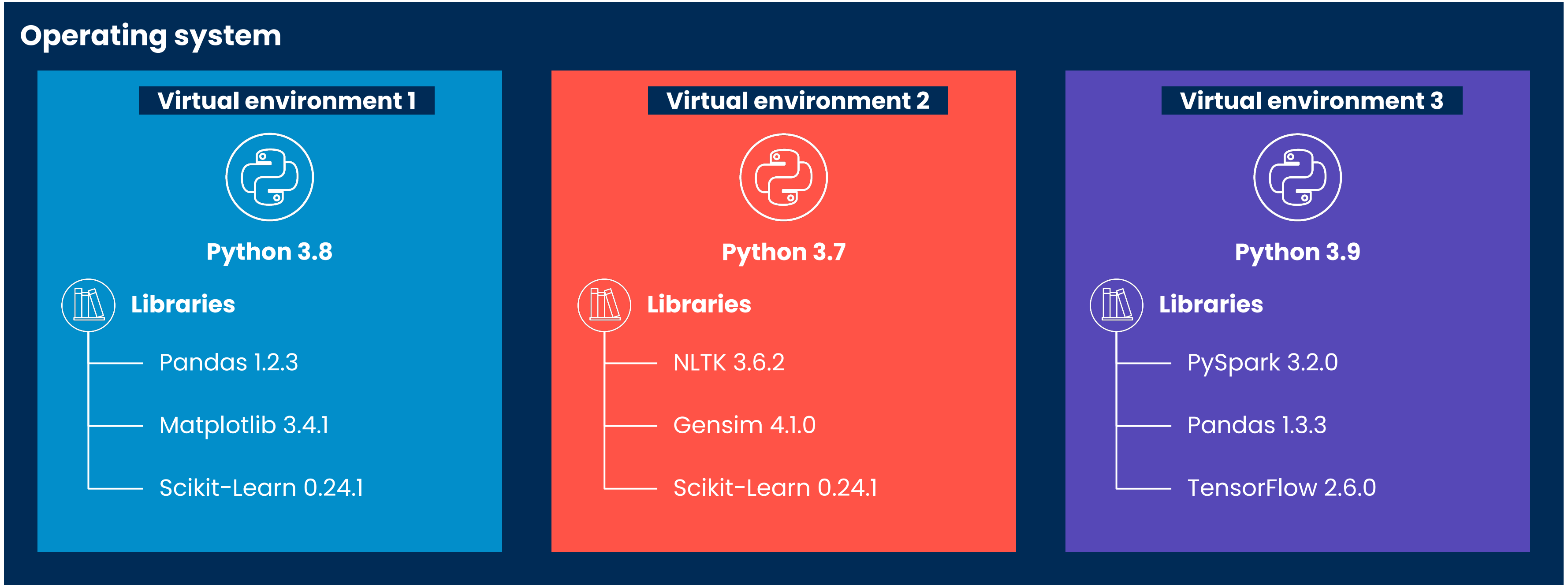
We would **create a virtual environment for each project**. This means we can install version 2.0 of library X for Project B in our second virtual environment **without overwriting** the version 1.0 installation we need for Project A.



We can imagine that each of the virtual environments acts as a **separate** operating system in the sense that each has **distinct installations** and **configurations**.

Python virtual environments

As good practice, we should always aim to use **virtual environments** when programming in Python. *An example of how this would look:*



Other programming languages

While Python is often preferred by data professionals, other programming languages like **R**, **Julia**, and **Scala** can also be used for most data-related tasks.

Selecting the appropriate programming language **depends on the specific requirements** of a data project. While **Python offers versatility** and a comprehensive set of libraries, alternative languages shine in their **respective domains**.

R

A favored language for **statistical analysis**, specifically for tasks such as hypothesis testing, linear and nonlinear modeling, and visualization.

Julia

A high-level, **high-performance** programming language designed for **computationally intensive tasks** such as complex numerical and scientific computing.

Scala

A favored programming language for **distributed data processing** and **big data analytics** that offers the scalability needed for big data workloads.

Leveraging programming to solve problems

Let's look at an example of how we can **leverage Python to solve a problem** related to the United Nations Sustainable Development Goal (SDG) "zero hunger," SDG 2.

Problem statement: Efficiently distribute surplus food to organizations serving vulnerable populations while minimizing food waste.

Food inventory management and data collection: Using Python, develop a comprehensive food inventory management system that collects data on surplus food items from various sources, including restaurants, supermarkets, and farms.

Data validation and cleaning: Use the Python Pandas library for data validation, cleaning, and normalization. This ensures the accuracy and consistency of food inventory data.

Data integration: Integrate data from different sources into a unified database using Python. This could include data on food expiration dates, storage conditions, donation histories, and geographic information.

Leveraging programming to solve problems

Geospatial analysis: Implement geospatial analysis using Python libraries like GeoPandas to optimize food distribution routes based on the proximity of food sources to distribution centers and beneficiaries.

Time series analysis: Apply time series analysis techniques in Python to predict food availability patterns, taking into account seasonal variations and historical data. This aids in the efficient management of the food inventory.

Machine learning for demand forecast: Enhance demand forecasting with advanced machine learning models implemented in Python, which can consider factors like demographic data, socioeconomic indicators, and historical consumption patterns.

Big data analytics: Utilizing big data processing frameworks like Apache Spark with Python to analyze large datasets related to food donations, distribution trends, and beneficiary needs. This can uncover insights for optimizing food allocation strategies.

Real-time data updates: Create mechanisms for real-time data updates using Python, allowing for dynamic adjustments in food distribution plans as new information becomes available.