



VAJO LUKIC

# PYTHON DATA ENGINEERING RESOURCES

# **Python Data Engineering Resources**

Vajo Lukic

SAMPLE

## Copyright Notice

© 2024, Vajo Lukic. All rights reserved.

This book is published by Companion Code SL.

No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law. For permission requests, write to the publisher, addressed "Attention: Permissions Coordinator," at the address below.

Companion Code SL  
Calle Martinez Campos 16  
29001, Malaga,  
Spain  
[info@companioncode.com](mailto:info@companioncode.com)

SAMPLE

# Terms of Use

By accessing, reading, or using this eBook, you agree to adhere to the following terms of use. If you do not agree with these terms, you are prohibited from using or accessing this eBook.

1. **Licence:** Companion Code SL grants you a non-exclusive, non-transferable, revocable licence to use the eBook for personal, non-commercial purposes only. This means you may not use the eBook for commercial purposes without express written consent from the author/publisher.
2. **Copyright:** This eBook is protected by copyright and intellectual property laws. You may not reproduce, distribute, display, perform, publish, licence, create derivative works from, transfer, or sell any information, software, products, or services obtained from this eBook.
3. **Content Usage:** You may not use the eBook's content for commercial purposes, to infringe upon the eBook's intellectual property rights, or in any way that harms or can be reasonably expected to harm the author or publisher.
4. **Modifications:** Any unauthorised modification, alteration, or use of the eBook or its content is strictly prohibited.
5. **Limitation of Liability:** The eBook is provided "as is," and Companion Code SL shall not be liable for any direct, indirect, incidental, consequential, or punitive damages arising from or connected to your use or inability to use the eBook. All references and links are provided solely for informational purposes and do not guarantee the content's accuracy, reliability, or any other stated or implied objective. The author, company, and publisher make no guarantees regarding the performance, effectiveness, or relevance of any sites or references mentioned in this book.
6. **Governing Law:** These terms will be governed by and construed in accordance with the laws of Spain, without giving effect to any principles of conflicts of law.
7. **Changes to Terms of Use:** Companion Code SL reserves the right to modify these terms of use at any time. Your continued use of the eBook after any such changes indicates your acceptance of the new terms.
8. **Contact Information:** If you have any questions or concerns about these terms of use, please contact: [info@companioncode.com](mailto:info@companioncode.com)

## About

Hi, I'm Vajo Lukic! I've been exploring the worlds of data engineering and solution architecture for over 20 years, dabbling in everything from machine learning to business intelligence. I love getting my hands dirty with code and data and crafting smart systems. I'm always on a quest to learn something new and share that knowledge with anyone willing to listen. I truly hope this book will be a helpful resource for you and I'm grateful you've chosen to spend your time with it.

If you're into tech and innovation, feel free to catch up with me on [Substack](#), follow my musings on [Twitter](#), or connect with me on [LinkedIn](#). Let's learn and grow together!

SAMPLE

<b>Terms of Use</b>	<b>3</b>
<b>About</b>	<b>4</b>
<b>Who Is This Book For?</b>	<b>11</b>
<b>Introduction</b>	<b>12</b>
Data Engineering Done Right	12
What Is Inside This Book?	13
Each section includes:	13
Resources Included:	13
Giving Back to the Community	15
<b>ORMs for Python</b>	<b>16</b>
Introduction	16
List of ORMs	16
SQLAlchemy	16
Django ORM	16
Peewee	16
Pony ORM	17
SQLObject	17
Tortoise ORM	17
ORM	17
Gino	17
Most popular ORMs	18
How to choose ORM?	19
<b>Data/Schema Validation Tools for Python</b>	<b>20</b>
Introduction	20
List of Data/Schema validation tools	21
Pydantic:	21
Marshmallow:	21
Cerberus:	21
Voluptuous:	21
jsonschema:	21
Pandera:	21
Validr:	22
How to choose a Data/Schema validation tool?	23
<b>Database Migration Tools for Python</b>	<b>24</b>
Introduction	24
List of Database migration tools	25
Alembic:	25
Flyway:	25
South (for older Django projects):	25
Django Migrations:	25
yoyo-migrations:	25

Flask-Migrate:	25
SQLAlchemy-Migrate:	26
How to choose the right Database migration tool?	27
<b>Data Wrangling Tools for Python</b>	<b>28</b>
Introduction	28
List of data wrangling tools	29
Pandas:	29
Dask:	29
NumPy:	29
Beautiful Soup:	29
Scrapy:	29
OpenRefine (formerly Google Refine):	29
TextBlob:	30
How to choose the right data wrangling tool?	31
<b>ETL frameworks for Python</b>	<b>32</b>
Introduction	32
List of ETL frameworks	33
Pandas:	33
Petal:	33
PySpark:	33
Data Load Tool (DLT):	33
dbt (data build tool):	33
Bonobo:	34
Mage.AI:	34
How to choose the right ETL framework?	35
<b>Orchestration Tools for Python</b>	<b>36</b>
Introduction	36
List of orchestration tools	37
Apache Airflow:	37
Luigi:	37
Apache NiFi:	37
Prefect:	37
Dagster:	37
Argo Workflows:	37
How to choose the right orchestration tool?	38
<b>Data visualisation Libraries for Python</b>	<b>39</b>
Introduction	39
List of data visualisation tools	40
Matplotlib:	40
Seaborn:	40
Plotly:	40
Bokeh:	40

Altair:	40
How to choose the right data visualisation tool?	41
<b>Machine Learning Libraries for Python</b>	<b>42</b>
Introduction	42
List of ML libraries	43
Scikit-learn:	43
TensorFlow:	43
PyTorch:	43
Keras:	43
XGBoost:	43
LightGBM:	43
CatBoost:	44
How to choose the right ML library?	45
<b>Big Data Tools for Python</b>	<b>46</b>
Introduction	46
List of Big Data tools	47
Apache Spark:	47
Apache Hadoop:	47
Dask:	47
Apache Flink:	47
Apache Beam:	47
How to choose the right Big Data tool?	48
<b>Tools for Streaming Data Processing in Python</b>	<b>49</b>
Introduction	49
List of tools for stream data processing	50
Apache Kafka:	50
Apache Flink:	50
Apache Storm:	50
Faust:	50
Apache Spark Streaming:	50
Redpanda:	50
How to choose the right tool for stream data processing?	51
<b>Tools for Data Modeling in Python</b>	<b>52</b>
Introduction	52
List of tools for data modelling	53
dbdiagram.io:	53
ER/Studio:	53
MySQL Workbench:	53
ERAlchemy:	53
Dia:	53
Lucidchart:	54
How to choose the right tool for data modelling?	55



<b>API Development Frameworks for Python</b>	<b>56</b>
Introduction	56
List of API development frameworks	57
Flask:	57
Django REST framework:	57
FastAPI:	57
Tornado:	57
Falcon:	57
How to choose the right Python API development framework?	58
<b>Data Governance and Metadata Management Tools for Python</b>	<b>59</b>
Introduction	59
List of tools Data-Gov and Metadata management	59
Apache Atlas:	59
Amundsen:	59
CKAN:	59
Marquez:	60
DataHub:	60
Soda:	60
Collibra:	60
How to choose the right tool for this purpose?	61
<b>Cloud SDKs for Python</b>	<b>62</b>
Introduction	62
List of cloud SDKs	63
For AWS (boto3):	63
For GCP (google-cloud-python):	63
For Azure (azure-sdk-for-python):	63
IBM Cloud Python SDK:	63
Oracle Cloud Infrastructure SDK for Python:	63
How to choose the right cloud SDK?	64
<b>Cloud Services and Tools for Python</b>	<b>65</b>
Introduction	65
List of cloud services and tools	66
AWS	66
Amazon S3:	66
Amazon EC2:	66
Amazon Redshift:	66
Azure	66
Azure Blob Storage:	66
Azure Data Lake Storage:	66
Azure Synapse Analytics:	66
GCP	67
Google Cloud Storage:	67

Google Compute Engine:	67
Google BigQuery:	67
How to choose cloud services and tools?	68
<b>Data Storage Solutions for Python</b>	<b>69</b>
Introduction	69
List of data storage solutions	70
Relational Databases (PostgreSQL):	70
NoSQL Databases (MongoDB):	70
Key-Value Store (Redis):	70
Wide-Column Store (Cassandra):	70
Graph Database (Neo4j):	70
Time Series Database (InfluxDB):	70
Search Engine (Elasticsearch):	71
How to choose the right data storage solution?	72
List of cloud-based Data warehousing tools	73
Cloudera:	74
Teradata:	74
Databricks:	74
Oracle Autonomous Database:	74
Snowflake:	74
<b>Data Quality Tools for Python</b>	<b>75</b>
Introduction	75
List of data quality tools	76
Ydata Profiling:	76
Great Expectations:	76
DataCleaner:	76
PyDeequ:	76
Dedupe:	76
Data Linter:	76
Soda Core:	77
How to choose the right data quality tool?	78
<b>Learning Resources for Python</b>	<b>79</b>
Introduction	79
List of Courses	79
DataCamp's Introduction to Data Engineering	79
Coursera's Data Engineering with Google Cloud Professional Certificate	79
edX's Professional Certificate in Data Engineering by IBM	79
Kaggle's Data Cleaning	79
LinkedIn Learning's Become a Data Engineer	80
List of Tutorials	81
Real Python's Data Engineering Course	81
Towards Data Science's Data Engineering Series on Medium	81

Apache Airflow in a Nutshell	81
DataQuest's Data Engineer Path	81
W3Schools Python Tutorial	81
YouTube: Corey Schafer's Python Tutorials	81
List of Books	83
"Python for Data Analysis"	83
"Designing Data-Intensive Applications"	83
"The Data Engineering Cookbook"	83
"Mining of Massive Datasets"	83
"Learning Spark: Lightning-Fast Big Data Analysis"	83
"Streaming Systems"	84
"Data Engineering with Python: Work with massive datasets to design data models and automate data pipelines using Python"	84
"Introducing Data Science: Big Data, Machine Learning, and more, using Python tools"	84
List of Blogs	85
Towards Data Science on Medium	85
Data Engineering Weekly	85
The Apache Software Foundation Blog	85
AWS Big Data Blog	85
Google Cloud Blog - AI & Machine Learning	85
DBT (Data Build Tool) Blog	86
<b>Communities and Forums for Data Engineering with Python</b>	<b>87</b>
Introduction	87
Why join communities?	87
List of communities and forums	88
Stack Overflow Data Engineering	88
Reddit - r/dataengineering	88
DBT (Data Build Tool) Slack Community	88
Kaggle	88
LinkedIn Groups (e.g., Data Engineering Social Club)	88
Towards Data Science on Medium	88
Operational Analytics Club	89
Data-Centric AI community	89
MLOps (@chipro)	89
DataTalks.club	89
MLOps Community	89
Locally Optimistic	89
Data Engineering Discord	89
<b>Free Datasets and APIs</b>	<b>90</b>
Free Datasets	90
Free APIs	99
<b>Final Thoughts and Further Exploration</b>	<b>105</b>

## Who Is This Book For?

This "book" started as a GitHub repository I created and shared [at this link](#), is a carefully assembled collection of resources designed to guide Python developers on their journey into data engineering, machine learning, and artificial intelligence. My aim in creating this repository is to share the knowledge and tools that I've found invaluable in my own career, hoping to ease the learning curve for newcomers and provide a reference point for seasoned professionals.

This repository is meant for individuals who are passionate about Python and are either stepping into or advancing their careers in data engineering, machine learning, or AI. Whether you're just starting out and seeking a basic understanding of the necessary tools, or you're an experienced developer looking to upgrade your skill set, this collection is intended to be a helpful companion on your professional journey.

The tools, frameworks, libraries, and learning resources presented here will be valuable for individuals involved with Python in various roles, such as *software developers, data engineers, solution architects, data scientists*, and more.

Through this repository, I aim to foster a spirit of learning, sharing, and innovation, assisting you to achieve your goals in the quickly evolving world of data engineering.

# Introduction

During my time as a Data Engineer, I've gathered many bookmarks and resources that have really helped me learn and do my job. I organised these bookmarks and put them in this [repository](#) so they can help others too, whether you're new to Data Engineering with Python or looking to know more. I hope you find these resources as helpful as they were for me.

## Data Engineering Done Right

Every software we create must do two things well: first, it should solve a specific problem, providing the utility we need. Second, it should work reliably and predictably. To make sure software meets these standards, we focus on two kinds of requirements: functional and non-functional.

Functional requirements are about what the software should do, like its tasks and features. Non-functional requirements cover how the software performs these tasks, including its reliability, speed, and security.

This repository has tools organised to help address both kinds of requirements, reminding us that when we build data engineering systems, we need to consider all these aspects. Good software design isn't just about making it do its job; it's also about making sure it works well and can adapt to future needs. This approach helps us create software that's not only useful but also dependable and easy to maintain.

In a world where AI helpers can write code, just knowing how to code isn't going to be enough to keep our jobs as developers. You need to bring something extra to the table. Understanding how to design systems, build effective architectures, and manage data properly are key skills that will help you stand out and stay ahead in your career. It's all about adding that personal experience and going the extra mile in what you know and do. I hope that this book and the repository behind it are going to be a valuable support in that endeavour.

## What Is Inside This Book?

This book comes from a [repository](#) which was a handpicked collection of resources for Python developers in data engineering, machine learning, and AI. Inside, you'll discover a neatly arranged selection of frameworks, libraries, and tools crucial for machine learning, ETL, ORM, data/schema validation, database migration, and more, all focused on Python.

Resources are organised into sections and categories of tools to showcase different aspects of designing and developing robust Python applications. Different projects may have varying requirements and priorities concerning performance, data, legislation, and more. These categories are designed to cover all such aspects and beyond.

Each section includes:

- A concise description of the tools/resources within that category.
- A list of the most relevant tools found in that category.
- A guide on selecting the appropriate tool from each category.

Resources Included:

- **ORMs for Python:** Including popular ORMs like SQLAlchemy, Django ORM, Peewee, etc.
- **Data/Schema Validation:** Including libraries like Pydantic, Marshmallow, Cerberus, etc.
- **Database Migration Tools:** Tools like Alembic, Flyway, or Django's own migration system.
- **Data Wrangling Tools:** Libraries that help in cleaning, transforming, and preparing data, such as Pandas, Dask, etc.
- **ETL (Extract, Transform, Load) Frameworks:** Tools that help in the process of extracting data from various sources, transforming it, and loading it into a data store.
- **Orchestration Tools:** Tools such as Apache NiFi, Luigi, Airflow, and Prefect, are designed to automate and orchestrate ETL workflows, managing job scheduling and execution. However, the specific ETL tasks are typically defined with other dedicated libraries or frameworks.
- **Data Visualization Libraries:** Libraries that can help in visualising data, such as Matplotlib, Seaborn, Plotly, Bokeh, etc.
- **Machine Learning Libraries:** While not exclusively for data engineering, having resources related to machine learning is useful. This includes libraries like scikit-learn, TensorFlow, and PyTorch.
- **Big Data Processing Tools:** Includes links to resources for tools like Apache Spark, Apache Hadoop, etc.
- **Streaming Data Processing:** Tools and frameworks for processing streaming data, such as Apache Kafka, Apache Flink, and Apache Storm.
- **Data Modeling Tools:** Resources for data modelling tools that can help in designing database schemas, such as dbdiagram.io, ER/Studio, or MySQL Workbench.
- **API Development Frameworks:** Since data engineering often involves API development for data access, includes resources for frameworks like Flask, FastAPI, or Django REST Framework.
- **Data Governance and Metadata Management:** Tools and frameworks that help in managing data access, security, and compliance, such as Apache Atlas, Collibra, or Amundsen.

- **Cloud SDKs for Python:** These SDKs, like boto3 for AWS, provide Python developers with the tools necessary to interact with cloud services efficiently, allowing for the automation of resource management and the utilisation of cloud services within Python applications.
- **Cloud Services and Tools:** Include resources related to cloud services that are widely used in data engineering, like AWS, Azure, and GCP, particularly focusing on their data storage, processing, and analytics services.
- **Data Storage Solutions:** Resources on various data storage solutions like relational databases, NoSQL databases, data lakes, and data warehouses.
- **Data Quality Tools:** Tools that help in ensuring data quality, such as Great Expectations, Deequ, or Pandas Profiling.
- **Learning Resources:** Links to courses, tutorials, blogs, and books that offer in-depth knowledge about data engineering in Python.
- **Community and Forums:** Links to relevant forums and communities where developers can ask questions, share knowledge, and stay updated with the latest trends in data engineering.
- **Free datasets and APIs:** Great list of free datasets and APIs - a very useful collection of free data resources for people learning data engineering. These resources are great for getting hands-on experience.

But this repository is also more than just a set of links; it's a map to learning, offering a clear path for growth in data engineering with Python. Whether you're looking for free datasets and APIs for practical experience or detailed guides on various tools and frameworks, you'll find what you need here to improve your skills.

## Giving Back to the Community

All of us who work as Data engineers have used many free resources created by others, to learn, get new skills and grow as professionals. I created this free [repository](#) to give something back to the community that has helped me so much. I hope these resources will help others just like they helped me.

The reason this book is based on that same Git repository is to help people who want this info in a format that's easy to read on ebook readers. It's perfect if you like having your reading material handy and well-organised on your phone or tablet.

Let's assist each other out and learn together. By doing this, we're making the world a better, more welcoming place, and we're aiming to leave it better off for our kids than how we found it.

SAMPLE



# ORMs for Python

## Introduction

Object-Relational Mapping (ORM) in Python is a programming technique used to convert data between incompatible type systems in object-oriented programming languages and relational databases. ORM in Python allows developers to interact with a database using Pythonic objects instead of writing SQL queries. This abstraction facilitates data manipulation and management, making database operations more intuitive and integrated within the Python code.

ORMs serve as a bridge between the Python code and the database, enabling developers to perform CRUD (Create, Read, Update, Delete) operations on the database without having to write verbose SQL syntax. This not only speeds up the development process but also enhances code readability and maintainability. By using ORMs, developers can focus more on the business logic of the application rather than the details of database interaction.

## List of ORMs

Here's a list of some popular Object-Relational Mapping (ORM) libraries for Python, along with links to their main resources and a brief description of each:

### SQLAlchemy

Website: <https://www.sqlalchemy.org/>

Description: SQLAlchemy is one of the most widely used ORM libraries in the Python community. It provides a full suite of well-known enterprise-level persistence patterns and is designed for efficient and high-performing database access.

### Django ORM

Website: <https://docs.djangoproject.com/en/stable/topics/db/models/>

Description: Django ORM is part of the Django web framework. It allows you to define your data models entirely in Python and provides a powerful abstraction layer to translate your Python code to SQL.

### Peewee

Website: <http://docs.peewee-orm.com/en/latest/>

Description: Peewee is a small, expressive ORM that provides a simple and intuitive interface to interact with the database. It's lightweight and designed to be easy to use, making it a great choice for small to medium-sized applications.

## Pony ORM

Website: <https://ponyorm.org/>

Description: Pony ORM is a unique ORM that uses generator expressions to define queries. It's designed to be intuitive and user-friendly, allowing for complex queries to be written in pure Python, closely mirroring human language.

## SQLObject

Website: <http://www.sqlobject.org/>

Description: SQLObject is a popular ORM that provides an object-oriented interface to your database, with tables as classes and rows as instances. It supports a variety of database backends and is designed for simplicity and ease of use.

## Tortoise ORM

Website: <https://tortoise.github.io/>

Description: Tortoise ORM is an easy-to-use async ORM (Object Relational Mapper) inspired by Django. It's designed to be used with async and await syntax of Python, making it a great choice for asynchronous applications.

## ORM

Website: <https://github.com/encode/orm>

Description: The ORM library is a lightweight and async-ready ORM that is designed to work with FastAPI and Starlette. It's particularly suited for applications that require asynchronous database operations.

## Gino

Website: <https://python-gino.org/>

Description: Gino is an async ORM built on SQLAlchemy core. It's designed for async programming with asyncio and provides a simple and intuitive API for interacting with the database asynchronously.

These ORMs cover a wide range of use cases, from simple and lightweight libraries to more robust and feature-rich solutions, providing various options depending on the requirements of your Python project.

## Most popular ORMs

As of early 2024, several Python ORMs remain popular and actively used within the developer community. Here's a brief overview of some of the most notable ones:

- **SQLAlchemy:** Widely regarded for its "just right" level of abstraction, making complex database queries easier compared to other ORMs source
- **Django ORM:** Integrated with the Django web framework, it works well for a variety of database operations but can be complex for more advanced queries.
- **Peewee:** Known for being simpler and more hackable than SQLAlchemy, aiming for ease of use and readability.

These ORMs vary in their design philosophies, ease of use, and specific features, catering to different types of projects and developer preferences. Whether you need synchronous or asynchronous operations, a simple interface, or deep customization, there's likely a Python ORM that fits the bill.

## How to choose ORM?

When deciding among the three most popular Python ORMs—SQLAlchemy, Django ORM, and Peewee—your choice should depend on your project's specific needs and context.

Here's a general guide on when to choose each:

- **SQLAlchemy:** Choose SQLAlchemy when you need a powerful, flexible ORM that can handle complex queries and supports a wide variety of database backends. It's a great choice if you want the flexibility to drop down to raw SQL when necessary and prefer a data-mapper pattern over active record. It's ideal for applications where the database is a central component and requires sophisticated interactions and transformations. SQLAlchemy is well-suited for both small-scale and enterprise-level applications due to its scalability and comprehensive feature set.
- **Django ORM:** Opt for Django ORM if you're already using the Django framework for your web application. Since it's tightly integrated with Django, it offers a seamless development experience within this ecosystem. Django ORM simplifies the process of performing common web application tasks, making it a good choice for developers who prefer convention over configuration. It's particularly well-suited for rapid development and applications where the database interactions are relatively straightforward or of moderate complexity.
- **Peewee:** Peewee is an excellent choice for smaller applications or when you prefer a lightweight, easy-to-learn ORM. Its simple and intuitive interface is great for quick development cycles and smaller codebases. If you're working on a project where you don't need the extensive features of larger ORMs like SQLAlchemy and prefer a more straightforward, less verbose ORM, Peewee is a strong candidate. It's particularly appealing for developers who prioritise readability and simplicity in their code and don't require advanced features like those offered by SQLAlchemy.

In summary, if you need a powerful and flexible ORM for complex applications, go with SQLAlchemy. If you're working within the Django ecosystem and want tight integration with your web framework, choose Django ORM. And if you prefer simplicity and are working on a smaller project, Peewee might be the best fit. Your specific project requirements, including the complexity of database interactions, the size of your application, and your development environment, will ultimately guide your choice among these ORMs.