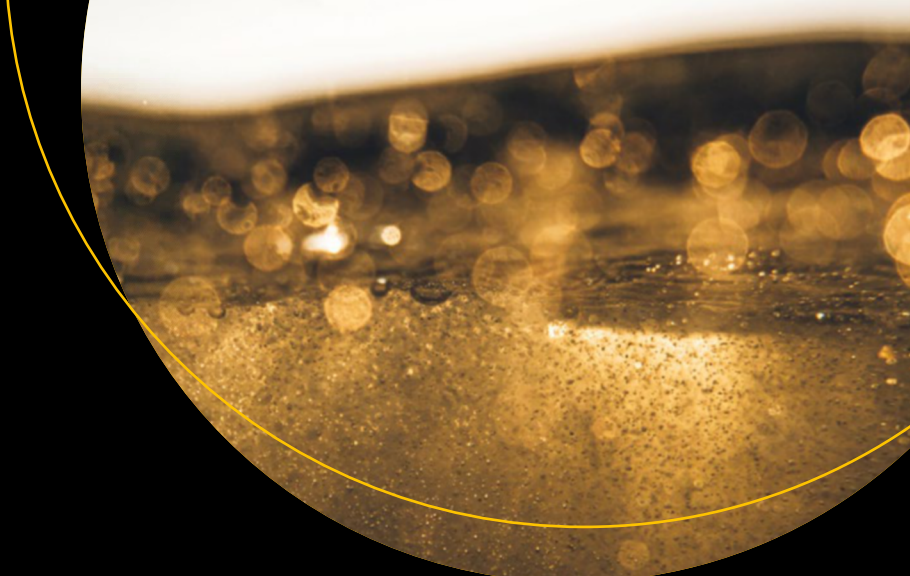




Data Engineering for Machine Learning Pipelines

From Python Libraries to ML Pipelines
and Cloud Platforms

Pavan Kumar Narayanan

Apress®

Data Engineering for Machine Learning Pipelines

**From Python Libraries to ML
Pipelines and Cloud Platforms**

Pavan Kumar Narayanan

Apress®

Data Engineering for Machine Learning Pipelines: From Python Libraries to ML Pipelines and Cloud Platforms

Pavan Kumar Narayanan
Sheridan, USA

ISBN-13 (pbk): 979-8-8688-0601-8
<https://doi.org/10.1007/979-8-8688-0602-5>

ISBN-13 (electronic): 979-8-8688-0602-5

Copyright © 2024 by Pavan Kumar Narayanan

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Celestin Suresh John
Development Editor: Laura Berendson
Editorial Project Manager: Gryffin Winkler

Cover designed by eStudioCalamar

Cover photo by Bradley Dunn on Unsplash

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, Suite 4600, New York, NY 10004-1562, USA. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at <http://www.apress.com/bulk-sales>.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub. For more detailed information, please visit <https://www.apress.com/gp/services/source-code>.

If disposing of this product, please recycle the paper

To my beloved father

Table of Contents

About the Author	xix
About the Technical Reviewer	xxi
Introduction	xxiii
Chapter 1: Core Technologies in Data Engineering	1
Introduction.....	1
Python Programming	2
F Strings	2
Python Functions	3
Advanced Function Arguments.....	4
Lambda Functions	5
Decorators in Python	6
Type Hinting.....	7
Typing Module	7
Generators in Python	8
Enumerate Functions.....	8
List Comprehension	11
Random Module	13
Git Source Code Management	16
Foundations of Git	16
GitHub.....	17
Core Concepts of Git.....	23
SQL Programming.....	31
Essential SQL Queries.....	31
Conditional Data Filtering	32
Joining SQL Tables	32

TABLE OF CONTENTS

- Common Table Expressions..... 33
- Views in SQL..... 34
- Temporary Tables in SQL 36
- Window Functions in SQL..... 37
- Conclusion 39
- Chapter 2: Data Wrangling using Pandas 41**
 - Introduction..... 41
 - Data Structures 42
 - Series 42
 - Data Frame 43
 - Indexing 44
 - Essential Indexing Methods..... 45
 - Multi-indexing 49
 - Time Delta Index..... 52
 - Data Extraction and Loading 56
 - CSV 56
 - JSON..... 57
 - HDF5..... 57
 - Feather 58
 - Parquet..... 58
 - ORC..... 59
 - Avro 59
 - Pickle..... 60
 - Chunk Loading..... 60
 - Missing Values 61
 - Background 61
 - Missing Values in Data Pipelines 62
 - Handling Missing Values..... 63
 - Data Transformation..... 65
 - Data Exploration 66
 - Combining Multiple Pandas Objects..... 69

Data Reshaping.....	75
pivot().....	75
pivot_table().....	76
stack().....	78
unstack().....	80
melt().....	81
crosstab().....	82
factorize().....	85
compare().....	87
groupby().....	89
Conclusion	91
Chapter 3: Data Wrangling using Rust's Polars	93
Introduction.....	93
Introduction to Polars.....	94
Lazy vs. Eager Evaluation.....	95
Data Structures in Polars	96
Polars Series	97
Polars Data Frame.....	98
Polars Lazy Frame.....	100
Data Extraction and Loading.....	102
CSV.....	102
JSON.....	102
Parquet.....	103
Data Transformation in Polars.....	103
Polars Context	104
Basic Operations	109
String Operations.....	113
Aggregation and Group-By	113
Combining Multiple Polars Objects	114
Left Join.....	114
Outer Join	116

TABLE OF CONTENTS

Inner Join.....	117
Semi Join.....	118
Anti Join	119
Cross Join.....	119
Advanced Operations	120
Identifying Missing Values	121
Identifying Unique Values	122
Pivot Melt Examples	122
Polars/SQL Interaction	123
Polars CLI	128
Conclusion	130
Chapter 4: GPU Driven Data Wrangling Using CuDF.....	133
Introduction.....	133
CPU vs. GPU	134
Introduction to CUDA	135
Concepts of GPU Programming.....	135
Kernels	135
Memory Management	136
Introduction to CuDF	136
CuDF vs. Pandas.....	136
Setup	137
File IO Operations	143
Basic Operations.....	144
Column Filtering	145
Row Filtering	146
Sorting the Dataset.....	147
Combining Multiple CuDF Objects.....	148
Left Join.....	148
Outer Join	150
Inner Join.....	150

Left Semi Join.....	151
Left Anti Join.....	151
Advanced Operations	152
Group-By Function.....	152
Transform Function.....	153
apply()	154
Cross Tabulation	156
Feature Engineering Using cut().....	157
Factorize Function	158
Window Functions	159
CuDF Pandas	160
Conclusion	161
Chapter 5: Getting Started with Data Validation using Pydantic and Pandera	163
Introduction.....	163
Introduction to Data Validation.....	164
Need for Good Data	164
Definition	165
Principles of Data Validation.....	166
Introduction to Pydantic.....	168
Type Annotations Refresher.....	168
Setup and Installation.....	170
Pydantic Models	171
Fields	173
JSON Schemas	174
Constrained Types	176
Validators in Pydantic	176
Introduction to Pandera	178
Setup and Installation.....	179
DataFrame Schema in Pandera.....	180
Data Coercion in Pandera	182
Checks in Pandera.....	183

TABLE OF CONTENTS

Statistical Validation in Pandera	186
Lazy Validation.....	189
Pandera Decorators.....	192
Conclusion	196
Chapter 6: Data Validation using Great Expectations	197
Introduction.....	197
Introduction to Great Expectations.....	198
Components of Great Expectations	198
Setup and Installation	200
Getting Started with Writing Expectations	203
Data Validation Workflow in Great Expectations	207
Creating a Checkpoint.....	216
Data Documentation	220
Expectation Store.....	222
Conclusion	223
Chapter 7: Introduction to Concurrency Programming and Dask.....	225
Introduction.....	225
Introduction to Parallel and Concurrent Processing.....	226
History	226
Python and the Global Interpreter Lock	226
Concepts of Parallel Processing	227
Identifying CPU Cores	227
Concurrent Processing	228
Introduction to Dask.....	230
Setup and Installation.....	230
Features of Dask.....	231
Tasks and Graphs	231
Lazy Evaluation.....	232
Partitioning and Chunking	232

Serialization and Pickling	232
Dask-CuDF	233
Dask Architecture	233
Core Library	234
Schedulers	234
Client	234
Workers	236
Task Graphs	236
Dask Data Structures and Concepts	236
Dask Arrays	236
Dask Bags	239
Dask DataFrames	241
Dask Delayed	244
Dask Futures	246
Optimizing Dask Computations	249
Data Locality	250
Prioritizing Work	251
Work Stealing	251
Conclusion	252
Chapter 8: Engineering Machine Learning Pipelines using DaskML	253
Introduction	253
Machine Learning Data Pipeline Workflow	254
Data Sourcing	255
Data Exploration	255
Data Cleaning	255
Data Wrangling	256
Data Integration	256
Feature Engineering	256
Feature Selection	257
Data Splitting	257
Model Selection	258

TABLE OF CONTENTS

Model Training	258
Model Evaluation	259
Hyperparameter Tuning	259
Final Testing	259
Model Deployment.....	260
Model Monitoring	260
Model Retraining	260
Dask-ML Integration with Other ML Libraries	260
scikit-learn	261
XGBoost.....	261
PyTorch.....	261
Other Libraries.....	262
Dask-ML Setup and Installation.....	262
Dask-ML Data Preprocessing.....	263
RobustScaler()	263
MinMaxScaler().....	264
One Hot Encoding.....	265
Cross Validation	266
Hyperparameter Tuning Using Dask-ML	269
Grid Search.....	269
Random Search	270
Incremental Search	270
Statistical Imputation with Dask-ML.....	272
Conclusion	276
Chapter 9: Engineering Real-time Data Pipelines using Apache Kafka	277
Introduction.....	277
Introduction to Distributed Computing.....	278
Introduction to Kafka.....	279
Kafka Architecture	281
Events.....	281
Topics	282

Partitions	282
Broker	282
Replication	282
Producers	283
Consumers	283
Schema Registry	284
Kafka Connect	284
Kafka Streams and ksqldb	284
Kafka Admin Client	286
Setup and Development	287
Kafka Application with the Schema Registry	296
Protobuf Serializer	301
Stream Processing	307
Stateful vs. Stateless Processing	308
Kafka Connect	320
Best Practices	321
Conclusion	322
Chapter 10: Engineering Machine Learning and Data REST APIs using FastAPI ...	323
Introduction	323
Introduction to Web Services and APIs	324
OpenWeather API	324
Types of APIs	324
Typical Process of APIs	326
Endpoints	326
API Development Process	327
REST API	328
HTTP Status Codes	329
FastAPI	330
Setup and Installation	331
Core Concepts	332
Path Parameters and Query Parameters	332

TABLE OF CONTENTS

Pydantic Integration 337

Dependency Injection in FastAPI 342

Database Integration with FastAPI 343

Object Relational Mapping 347

Building a REST Data API..... 350

Middleware in FastAPI..... 354

ML API Endpoint Using FastAPI..... 355

Conclusion 359

Chapter 11: Getting Started with Workflow Management and Orchestration 361

Introduction..... 361

Introduction to Workflow Orchestration 362

 Workflow 362

 ETL and ELT Data Pipeline Workflow 365

 Workflow Configuration..... 368

 Workflow Orchestration..... 369

Introduction to Cron Job Scheduler 371

 Concepts..... 371

 Crontab File 372

 Cron Logging 374

 Cron Job Usage 375

 Cron Scheduler Applications..... 380

 Cron Alternatives 381

Conclusion 381

Chapter 12: Orchestrating Data Engineering Pipelines using Apache Airflow 383

Introduction..... 383

Introduction to Apache Airflow 384

 Setup and Installation..... 384

Airflow Architecture 389

 Web Server 390

 Database 390

Executor	390
Scheduler	391
Configuration Files.....	391
A Simple Example.....	393
Airflow DAGs	395
Tasks	397
Operators.....	397
Sensors.....	399
Task Flow.....	400
Xcom.....	401
Hooks.....	402
Variables.....	403
Params	406
Templates	407
Macros.....	407
Controlling the DAG Workflow	408
Triggers.....	411
Conclusion	413
Chapter 13: Orchestrating Data Engineering Pipelines using Prefect	415
Introduction.....	415
Introduction to Prefect	416
Setup and Installation.....	417
Prefect Server	418
Prefect Development.....	421
Flows	421
Flow Runs.....	421
Interface	424
Tasks	424
Results.....	427
Persisting Results.....	428
Artifacts in Prefect.....	432

TABLE OF CONTENTS

States in Prefect 438

State Change Hooks 441

Blocks..... 442

Prefect Variables..... 444

Variables in .yaml Files..... 445

Task Runners..... 446

Conclusion 448

Chapter 14: Getting Started with Big Data and Cloud Computing 451

Introduction..... 451

Background of Cloud Computing 452

 Networking Concepts for Cloud Computing..... 453

Introduction to Big Data 454

 Hadoop 454

 Spark 455

Introduction to Cloud Computing 456

 Cloud Computing Deployment Models..... 456

 Cloud Architecture Concepts 458

 Cloud Computing Vendors 461

 Cloud Service Models 461

 Cloud Computing Services 463

Conclusion 471

Chapter 15: Engineering Data Pipelines Using Amazon Web Services 473

Introduction..... 473

AWS Console Overview 474

 Setting Up an AWS Account..... 474

 Installing the AWS CLI..... 488

AWS S3 489

 Uploading Files 492

AWS Data Systems..... 493

 Amazon RDS..... 493

Amazon Redshift	497
Amazon Athena.....	504
Amazon Glue.....	506
AWS Lake Formation	506
AWS SageMaker.....	519
Conclusion	530
Chapter 16: Engineering Data Pipelines Using Google Cloud Platform.....	531
Introduction.....	531
Google Cloud Platform	532
Set Up a GCP Account.....	533
Google Cloud Storage.....	534
Google Cloud CLI	535
Google Compute Engine.....	537
Cloud SQL.....	541
Google Bigtable.....	547
Google BigQuery	552
Google Dataproc.....	554
Google Vertex AI Workbench	559
Google Vertex AI	562
Conclusion	569
Chapter 17: Engineering Data Pipelines Using Microsoft Azure.....	571
Introduction.....	571
Introduction to Azure.....	572
Azure Blob Storage	574
Azure SQL.....	577
Azure Cosmos DB.....	584
Azure Synapse Analytics	591
Azure Data Factory.....	593
Azure Functions	602

TABLE OF CONTENTS

Azure Machine Learning 606

Azure ML Data Assets 609

Azure ML Job 611

Conclusion 616

Index..... 617

About the Author



Pavan Kumar Narayanan has an extensive and diverse career in the information technology industry, with a primary focus on the data engineering and machine learning domains. Throughout his professional journey, he has consistently delivered solutions in environments characterized by heterogeneity and complexity. His experience spans a broad spectrum, encompassing traditional data warehousing projects following waterfall methodologies and extending to contemporary integrations that involve application programming interfaces (APIs) and

message-based systems. Pavan has made substantial contributions to large-scale data integrations for applications in data science and machine learning. At the forefront of these endeavors, he has played a key role in delivering sophisticated data products and solutions, employing a versatile mix of both traditional and agile approaches. Pavan completed his Master of Science in Computational Mathematics at Buffalo State University, Operations Research at the University of Edinburgh, and Information Technology at Northern Arizona University. He can be contacted at pavan.narayanan@gmail.com.

About the Technical Reviewer



Suvoraj Biswas is a leading expert and emerging thought leader in enterprise generative AI, specializing in architecture and governance. He authored the award-winning book *Enterprise GENERATIVE AI Well-Architected Framework & Patterns*, acclaimed by industry veterans globally. Currently an enterprise solutions architect at Ameriprise Financial, a 130-year-old Fortune 500 organization, Biswas has over 19 years of IT experience across India, the United States, and Canada. He has held key

roles at Thomson Reuters and IBM, focusing on enterprise architecture, architectural governance, DevOps, and DevSecOps.

Biswas excels in designing secure, scalable enterprise systems, facilitating multi-cloud adoption, and driving digital transformation. His expertise spans Software as a Service (SaaS) platform engineering, Amazon Web Services (AWS) cloud migration, generative AI implementation, machine learning, and smart IoT platforms. He has led large-scale, mission-critical enterprise projects, showcasing his ability to integrate advanced cutting-edge technologies into practical business strategies.

Introduction

The role of data engineering has become increasingly crucial. Data engineering is the foundation on which organizations build their reporting, analytics, and machine learning capabilities. Data engineering as a discipline transforms raw data into reliable and insightful informational resources that provide valuable insights to organizations. However, data engineering is much more than that. It is a complex and iterative process.

If an enterprise project can be seen as constructing a multi-storied building, then data engineering is where the execution happens—from architecting and designing the structure of the data solution to laying the data storage foundation; installing the data processing frameworks; developing pipelines that source, wrangle, transform, and integrate data; and developing reports and dashboards while ensuring quality and maintenance of the systems, pipelines, reports, and dashboards, among several other things.

This book is designed to serve you as a desk reference. Whether you are a beginner or a professional, this book will help you with the fundamentals and provide you with job-ready skills to deliver high-value proposition to the business. This book is written with an aim to bring back the knowledge gained from traditional textbook learning, providing an organized exploration of key data engineering concepts and practices. I encourage you to own a paper copy. This book will serve you well for the next decade in terms of concepts, tools, and practices.

Though the content is exhaustive, the book covers around 60% of data engineering topics. Data engineering is a complex and evolving discipline; there are so many more topics that are not touched on. As a reader, I encourage you to contact me and share your remarks and things you witnessed in the data space that you wish to share. I am thankful to those who have already written to me, and I wish to see more of these efforts from the readers and enthusiasts.

INTRODUCTION

Having said that, this book covers a wide range of topics that are practiced in modern data engineering, with a focus on building data pipelines for machine learning development. This book is organized into the following topics:

- **Data wrangling:** Techniques for cleaning, transforming, and preparing data for analysis using Pandas, Polars, and GPU-based CuDF library
- **Data validation:** Ensuring data quality and integrity throughout the pipeline using Pydantic, Pandera, and Great Expectations
- **Concurrency computing:** Leveraging parallel processing for improved performance using Dask and DaskML
- **APIs:** Designing and working with APIs for data exchange using FastAPI
- **Streaming pipelines:** Designing and working with real-time data flows efficiently using Kafka
- **Workflow orchestration:** Managing complex data workflows and dependencies using cron, Airflow, and Prefect
- **Cloud computing:** Utilizing cloud services for scalable data engineering solutions using AWS, Azure, and Google Cloud Platform (GCP)

The book covers major cloud computing service providers. The reason is that organizations are increasingly moving toward a multi-cloud approach. And so, being equipped with knowledge on diverse cloud environments is crucial for your career success. This book delivers that.

If you are a data engineer looking to expand your skillset and move up the ladder, a data scientist who seeks to understand how things work under the hood, a software engineer who is already doing the work, or a student who is looking to gain a position, this book is essential for you. Even if you are in a functional domain or a founder of a start-up, interested in setting up a team and good data engineering practice, this book will help you from start to end. This book is a guide that will help with learning for people at every stage (from novice to expert).

I highly encourage you not to seek solutions from chat bots during learning. At the time of writing this (June 2024), many bot programs either provide legacy code or steer you into their own direction and approach. It may seem better because of the way information is presented, but it slows down your learning and professional development. Even to this date, I find search engines to be more effective, which take me to Q&A sites where one can witness how a solution has been arrived at and why other approaches didn't succeed. Furthermore, I find Google search to be effective in terms of locating a certain method or a function in an API documentation that may be better than the website itself.

At the end of this book, you will be equipped with skills, knowledge, and practices that can be derived from a decade of experience in data engineering. Many vendors and cloud computing service providers provide you with free credits if you register for the first time. I encourage you to make use of these free credits, explore various tools and services, and gain knowledge. And when you are done, please remember to clean up your workspace, shut down any virtual machines and databases, take a backup of your work, and delete your workspace in the cloud to prevent additional charges.

CHAPTER 1

Core Technologies in Data Engineering

Introduction

This chapter is designed to get you started. As we begin our journey into advanced data engineering practices, it is important to build a solid foundation in key concepts and tools. This chapter is designed to provide a refresher on Python programming and software version control using GitHub and a high-level review of structured query language (SQL). While these may seem not connected and different from each other, they play an important role in developing effective data engineering pipelines. Python's data ecosystem, software version controlling systems using Git, and working with relational database systems using SQL will certainly solidify your foundations and help you tackle data engineering challenges. These are challenging and complex topics, especially reviewing them together; I hope you persist, learn, and practice on your own.

By the end of the chapter, you will

- Understand some of the key Python programming concepts relevant to advanced software engineering practices.
- Gain an appreciation toward software version control systems and git; create git repositories; and perform essential version control operations like branching, forking, and submitting pull requests.
- Have enough knowledge to set up your own code repository in GitHub.

- Review fundamental SQL concepts for querying and manipulating data in database systems.
- Identify advanced SQL concepts and best practices for writing efficient and optimized SQL code.

Engineering data pipelines is not easy. There is a lot of complexity and it involves leveraging analytical thinking. The complexity includes but is not limited to varying data models, intricate business logic, tapping into various data sources, and so many other tasks and projects that make up the profession. Often, data engineers do build machine learning models to identify anomalies and build advanced imputations on the data. I have taken some of the topics in my experience that would serve value to the data engineer or software developer who is engineering pipelines.

Python Programming

There are two major versions of the Python programming language, Python 2 and Python 3. Python 2 has reached end of life for support. Python 3 is the de facto language version for Python programming. The Python programming language (version 3.*) is still the most preferred language for development, comes with extensive community-backed data processing libraries, and is widely adopted. In this section, let us take some time to look at some of the Python programming concepts that may help, in addition to your basic programming knowledge. These are discussed at a high level. I encourage you to perform further investigation into these classes, methods, and arguments that can be passed into these methods.

F Strings

Python f strings allow you to easily insert variables and expressions within a print string. When compared with traditional string formatting, f strings are a lot more readable and easy to understand. You can create an f string by placing the character F in front of the string, before quotes. The character is case insensitive, meaning you can either insert an F or an f. Here is an example:

```
Greetings = f"Hello, it is a beautiful day here, at the park today"
```

Now you can insert variables and expressions within the f string. We have shown the preceding f string without inserts for demonstration purposes. Here is how you can perform that:

```
Fruit = "berries"
Cost = 1.5
Number = 3

print(f"the {fruit} costs {cost} per piece, totals to $ {cost*number}
dollars")
```

will return

the oranges costs 1.5 per piece, totals to \$ 4.5 dollars

Python Functions

In programming, functions are a set of instructions that can be executed together to accomplish a specific task or operation. It may or may not accept inputs and may or may not return output. Let us look at an example of a function:

```
def add(a,b):
    c = a+b
    return c
```

When you call this function

```
print(add(2,3))
```

it will return

5

Let us look at a bit more substantial function. Here, we have a function that takes Celsius as input and calculates Fahrenheit as output:

```
def f_calc(celsius):
    return ((9/5 * celsius) + 32)
```

This is the same as the following function. Both are syntactically equal. We will discuss this way of defining functions in another section:

```
def f_calc(celsius: int) -> float:
    return ((9/5 * celsius) + 32)
```

In the preceding code, we have passed a variable named `celsius` of type integer. Inside the function we are calculating a value and returning that value in floating point type. This is how functions can accept input parameters, process them, and provide output.

Advanced Function Arguments

***args**

Let us look at using the “*args” parameter in a function. The arguments parameter in functions can be used to pass a variable number of arguments to a function. If you have the syntax “*args” once, you can pass several numbers of variables, without having to define each one of them. Let us look at two examples in contrast to understand this concept better:

```
def add(a,b):
    c = a+b
    return c
```

When you run this

```
add(2,3)
```

it will return

5

What if we do not know how many input parameters we need? We can use arguments here:

```
def add_args(*args):
    total_sum = 0
    for i in args:
        total_sum += i
    return total_sum
```

So when you run this

```
add_args(3,4,5,6,7)
```

it will return

25

Note You can also pass a list to the function and still obtain the functionality. But with arguments, you can call an empty function and return none.

****kwargs**

This is quite similar to arguments, and the difference is that you can have a key associated with each item in the argument:

```
def add_kwargs(**kwargs):  
    total_sum = 0  
    for i in kwargs.values():  
        total_sum += i  
    return total_sum
```

Use of args and kwargs can be highly beneficial in enhancing the flexibility of a function, complementing the decorator implementation and passing a varying number of parameters as inputs.

Lambda Functions

In Python, lambdas are short for lambda expressions. They are inline functions that evaluate one single expression. That single expression is evaluated when the expression is called. Let us look at this with an illustration:

```
calculate_fahrenheit = lambda celsius: 9/5*celsius + 32
```

So when you call this function

```
calculate_fahrenheit(27)
```

it will return

80.6

Decorators in Python

Decorators in Python are a way to add additional functionality to your existing function. The decorator itself is a function. In the parameter column, it would take another function, as an input parameter. In calculus terms, it is equal to the function of a function. If math is not your cup of tea, then let's look at a simple example.

Here is a test decorator:

```
def testdec(func):  
    def wrapper():  
        print("+++Beginning of Decorator Function+++")  
        func()  
        print("+++End of Decorator Function+++")  
    return wrapper
```

Let us try to use this decorator on a sample function:

```
@testdec  
def hello():  
    print("Hello!")
```

When we run this function

```
hello()
```

it will provide us

```
+++Beginning of Decorator Function+++  
Hello!  
+++End of Decorator Function+++
```

Type Hinting

Type hinting is a Python feature that lets you specify the data types of the variable declarations, arguments in functions, and the variables that these functions return. Type hints make it easier to understand what data types to expect and which function returns what type of data. The idea behind type hinting is better readability, code maintainability, and being able to debug and troubleshoot errors faster. Type hints are not mandatory; you can still pass an argument without specifying the data type (Python will still check them during runtime).

Here is a simple example demonstrating the type hinting functionality in Python:

```
def some_function(a: float, b: float) -> float:
    c = a+b
    return c
```

Typing Module

The typing module supports type hints for advanced data structures with various features. Here is an example for a dictionary:

```
from typing import Dict

var2: Dict[int, str] = {234: "JohnDoe"}
```

Notice the usage of “Dict.” It serves the same purpose as “dict,” a dictionary type offered by core Python. “Dict” does not provide additional functionality except you have to specify the data type for both key and value aspects of a dictionary. If you are using Python 3.9 or later, you can use either “Dict” or “dict.”

Let us look at another example called the Union, where a variable could be one of few types:

```
from typing import Union

var4: list[Union[int, float]] = [3, 5, 5.9, 6.2]
```