Murat Durmus

A Hands-On Introduction to

ESSENTIAL PYTHON LIBRARIES AND FRAMEWORKS

(WITH CODE SAMPLES)

Paperback available on Amazon:

https://www.amazon.com/dp/B0BW2MGYG4

Murat Durmus

A Hands-On Introduction to
Essential Python Libraries and Frameworks

(With Code Samples)

Cover design:

Murat Durmus

About the Author

Murat Durmus is CEO and founder of AISOMA (a Frankfurt am Main (Germany) based company specializing in AI-based technology development and consulting) and Author of the books "Mindful AI - Reflections on Artificial Intelligence".& "A Primer to the 42 Most commonly used Machine Learning Algorithms (With Code Samples)"

You can get in touch with the author via:

- LinkedIn: https://www.linkedin.com/in/ceosaisoma/

- E-Mail: murat.durmus@aisoma.de

Note:

The code examples and their description in this book were written with the support of ChatGPT (OpenAI).

"Python is not just a language,
it's a community
where developers can learn,
collaborate and create wonders."

- Guido van Rossum

(Creator of Python)

A BRIEF HISTORY OF PYTHON PROGRAMMING LANGUAGE

Python is a popular high-level programming language for various applications, including web development, scientific computing, data analysis, and machine learning. Its simplicity, readability, and versatility have made it a popular choice for programmers of all levels of expertise. Here is a brief history of Python programming language.

Python was created in the late 1980s by Guido van Rossum, who worked at the National Research Institute for Mathematics and Computer Science in the Netherlands. Van Rossum was looking for a programming language that was easy to read and write, and that could be used for various applications. He named the language after the British comedy group Monty Python, as he was a fan of their TV show.

The first version of Python, Python 0.9.0, was released in 1991. This version included many features still used in PythonPython today, such as modules, exceptions, and the core data types of lists, dictionaries, and tuples.

Python 1.0 was released in 1994 and included many new features, such as lambda, map, filter, and reduce. These features made it easier to write functional-style code in PythonPython.

Python 2.0 was released in 2000, introducing list comprehensions, a new garbage collector, and a cycle-detecting garbage collector. List comprehensions made

writing code that operated on lists and other iterable objects easier.

Python 3.0, a significant update to the language, was released in 2008. This version introduced many changes and improvements, including a redesigned print function, new string formatting syntax, and a new division operator. The latest version also removed some features that were considered outdated or redundant.

Since the release of Python 3.0, there have been several minor releases, each introducing new features and improvements while maintaining backward compatibility with existing code. These releases have included features such as async/await syntax for asynchronous programming, type annotations for improved code readability and maintainability, and improvements to the garbage collector and the standard library.

Python's popularity has grown steadily over the years, and it is now one of the most popular programming languages in the world. Web developers, data scientists, and machine learning engineers, among others, widely use it. Python's popularity has been driven by its simplicity, readability, and versatility, as well as its large and active community of developers who contribute to the language and its ecosystem of libraries and tools.

In conclusion, Python programming language has come a long way since its inception in the late 1980s. It has undergone many changes and improvements over the years, but its core values of simplicity, readability, and versatility have remained constant. Moreover, Python's

DATA SCIENCE

Data science is an interdisciplinary field that involves extracting, analyzing, and interpreting large, complex data sets. It combines elements of statistics, computer science, and domain expertise to extract insights and knowledge from data.

Data scientists use various tools and techniques to collect, process, and analyze data, including statistical analysis, machine learning, data mining, and data visualization. They work with large, complex data sets to uncover patterns, relationships, and insights that can inform decision-making and drive business value.

Data science has applications in various fields, including business, healthcare, finance, and social science. It informs different decisions, from product development to marketing to policy-making.

PANDAS

Python Pandas is an open-source data manipulation and analysis library for the Python programming language. It provides a set of data structures for efficiently storing and manipulating large data sets, as well as a variety of tools for data analysis, cleaning, and preprocessing.

Some of the key data structures in Pandas include the Series, which is a one-dimensional array-like object that can hold any data type; and the DataFrame, which is a two-dimensional tabular data structure with rows and columns that can be thought of as a spreadsheet or a SQL table.

Pandas also provides a range of data manipulation functions and methods, such as filtering, sorting, merging, grouping, and aggregating data. It also supports data visualization tools that allow users to plot and visualize data in a variety of ways.

It is widely used in data analysis and data science, and is considered one of the essential tools for working with data in Python. It is also frequently used in conjunction with other popular data science libraries such as NumPy, Matplotlib, and SciPy.

An example of how you can use Pandas to read in a CSV file, manipulate the data, and then output it to a new file:

```
import pandas as pd

# Read in the CSV file
data = pd.read_csv('my_data.csv')

# Print the first few rows of the data
```

```
print(data.head())

# Filter the data to include only rows where
the 'score' column is greater than 90
filtered_data = data[data['score'] > 90]

# Create a new column that calculates the
average of the 'score' and 'time' columns
filtered_data['average'] =
(filtered_data['score'] +
filtered_data['time']) / 2

# Output the filtered data to a new CSV file
filtered_data.to_csv('my_filtered_data.csv',
index=False)
```

In this example, we first import the Pandas library using **import pandas as pd**. We then read in a CSV file called **my_data.csv** using the **pd.read_csv()** function, which creates a DataFrame object. We then use the **head()** method to print out the first few rows of the data.

Next, we filter the data to include only rows where the 'score' column is greater than 90 using boolean indexing. We then create a new column called 'average' that calculates the average of the 'score' and 'time' columns using basic arithmetic operations.

Finally, we use the **to_csv()** method to output the filtered data to a new CSV file called **my_filtered_data.csv**, with the **index=False** parameter indicating that we do not want to include the DataFrame index as a column in the output file.

Pros and Cons

Pros:

- Easy-to-use and highly versatile library for data manipulation and analysis.
- Provides powerful tools for handling large datasets, including fast indexing, filtering, grouping, and merging operations.
- Supports a wide range of input and output formats, including CSV, Excel, SQL databases, and JSON.
- Offers a rich set of data visualization tools, including line plots, scatter plots, histograms, and more.
- Has a large and active community of users and developers, which means that there is a wealth of online resources and support available.
- Can be used in conjunction with other popular data science libraries such as NumPy, SciPy, and Matplotlib.

Cons:

- Pandas can be memory-intensive when working with very large datasets, and may not be the best choice for real-time applications or very high-dimensional data.

- Some of the functions and methods can be complex and difficult to understand, especially for new users.
- Can be slow when performing certain operations, such as applying functions to large datasets or performing multiple merges or concatenations.
- May not always produce the desired results, especially when working with messy or unstructured data.
- Some users have reported issues with compatibility and portability between different versions of Pandas or between Pandas and other libraries.

MACHINE LEARNING

Machine learning is a subfield of artificial intelligence that develops algorithms that can automatically learn and improve from data.

In machine learning, a model is trained on a large dataset of input-output pairs, called a training set, and then used to make predictions on new, unseen data. The goal is to develop a model that can generalize well to new data by learning patterns and relationships in the training data that can be applied to new data.

There are several machine learning types, including **supervised**, **unsupervised**, and **reinforcement learning**. In supervised learning, the training set includes labeled examples of input-output pairs, and the goal is to learn a function that can accurately predict the output for new inputs. In unsupervised learning, the training set does not include labels; the goal is to discover patterns and relationships in the input data. Finally, in reinforcement learning, an agent learns to interact with an environment to achieve a goal by receiving rewards or penalties based on actions.

Machine learning has many applications, from image recognition and natural language processing to recommendation systems and predictive analytics. It is used in various industries, including healthcare, finance, and e-commerce, to automate decision-making, improve efficiency, and gain insights from data.

SCIKIT-LEARN

Python scikit-learn (also known as sklearn) is a popular machine learning library for the Python programming language. It provides a range of supervised and unsupervised learning algorithms for various types of data analysis tasks such as classification, regression, clustering, and dimensionality reduction.

It was developed by David Cournapeau as a Google Summer of Code project in 2007 and is now maintained by a team of developers. It is open-source software and is available under a permissive BSD-style license.

Scikit-learn is built on top of other popular scientific computing libraries for Python, such as NumPy, SciPy, and matplotlib. It also integrates with other machine learning and data analysis libraries such as TensorFlow and Pandas.

Scikit-learn provides a wide range of machine learning algorithms, including:

- **Linear and logistic regression**
- **Support Vector Machines (SVM)**
- **Decision Trees and Random Forests**
- **K-Nearest Neighbors (KNN)**
- **Naive Bayes**
- **Clustering algorithms (e.g. K-Means)**

- **Dimensionality reduction techniques (e.g. Principal Component Analysis)**

It also provides utilities for model selection and evaluation, such as cross-validation, grid search, and performance metrics.

Scikit-learn is widely used in academia and industry for a variety of machine learning tasks, such as natural language processing, image recognition, and predictive analytics. It is considered one of the essential tools in the Python data science ecosystem.

An example code snippet that demonstrates how to use scikit-learn to train a simple logistic regression model:

```
from sklearn.linear_model import
LogisticRegression
from sklearn.datasets import load_iris

# Load the iris dataset
iris = load_iris()

# Split the dataset into features (X) and
labels (y)
X, y = iris.data, iris.target

# Create a LogisticRegression object
logreg = LogisticRegression()

# Fit the model using the iris dataset
logreg.fit(X, y)

# Predict the class labels for a new set of
features
new_X = [[5.1, 3.5, 1.4, 0.2], [6.2, 3.4, 5.4,
2.3]]
predicted_y = logreg.predict(new_X)
```

```
print(predicted_y)
```

In this example, we first import the necessary modules from scikit-learn (**LogisticRegression** for the model and **load_iris** for the iris dataset). We then load the iris dataset, which is a well-known dataset in machine learning consisting of 150 samples of iris flowers, with four features each.

We then split the dataset into features (the **X** variable) and labels (the **y** variable). We create a **LogisticRegression** object and fit the model to the dataset using the **fit** function.

Finally, we use the trained model to predict the class labels for two new sets of features (**new_X**). The predicted class labels are printed to the console.

This is just a simple example, and scikit-learn has many more advanced features and models for a wide range of machine learning tasks.

Pros and Cons

Pros:

- It's a powerful and comprehensive machine learning library that offers a wide range of algorithms for various tasks.
- Scikit-learn is easy to use and has a relatively simple API compared to other machine learning libraries.

- It does not include some newer or more advanced machine learning techniques that have been developed more recently, such as deep learning.
- Scikit-learn does not include built-in support for some popular machine learning frameworks such as TensorFlow or PyTorch, which may limit its flexibility in some use cases.

EXPLAINABLE AI

Explainable AI (XAI) is a set of techniques and practices that aim to make machine learning models and their decisions more transparent and understandable to humans.

XAI aims to provide insights into how a machine learning model works, how it makes decisions, and what factors influence its predictions. This is important because many modern machine learning models are complex and challenging to interpret, and their choices may significantly impact individuals and society.

XAI techniques include feature importance analysis, local and global model interpretability, counterfactual analysis, and model visualization. These techniques can help to identify the most critical factors that influence a model's predictions, provide explanations for specific predictions, and highlight potential biases or inaccuracies in the model.

Explainable AI is particularly important in applications where decisions made by machine learning models have significant consequences, such as healthcare, finance, and criminal justice. By making machine learning models more transparent and understandable, XAI can help build trust and confidence in these systems and ensure that they make fair and ethical decisions.

SHAP

SHAP (SHapley Additive exPlanations) is a popular open-source library for interpreting and explaining the predictions of machine learning models. SHAP is based on the concept of Shapley values, which are a method from cooperative game theory used to determine the contribution of each player to a cooperative game. In the context of machine learning, SHAP computes the contribution of each feature to a particular prediction, providing insight into how the model is making its predictions.

It provides a range of tools for visualizing and interpreting model predictions, including summary plots, force plots, and dependence plots. It can be used with a wide range of machine learning models, including both black box and white box models.

Overall, Python SHAP is a powerful tool for understanding how machine learning models are making their predictions, and can be useful in a range of applications, including feature selection, model debugging, and model governance.

An example code usage of Python SHAP:

```
import shap
from sklearn.ensemble import
RandomForestClassifier
from sklearn.datasets import load_breast_cancer

# Load the Breast Cancer Wisconsin dataset
data = load_breast_cancer()
```

```
# Create a random forest classifier
clf = RandomForestClassifier(n_estimators=100,
random_state=0)

# Train the classifier on the breast cancer
dataset
clf.fit(data.data, data.target)

# Initialize the SHAP explainer
explainer = shap.Explainer(clf)

# Generate SHAP values for the first 5
instances in the dataset
shap_values = explainer(data.data[:5])

# Plot the SHAP values for the first instance
shap.plots.waterfall(shap_values[0])
```

In this example, we first load the Breast Cancer Wisconsin dataset and create a random forest classifier using the **RandomForestClassifier** class from scikit-learn. We then train the classifier on the dataset.

Next, we initialize a SHAP explainer using the **Explainer** class from the **shap** library. We then generate SHAP values for the first 5 instances in the dataset using the explainer.

Finally, we plot the SHAP values for the first instance using the **waterfall** function from the **shap.plots** module. This generates a waterfall plot showing the contribution of each feature to the model's prediction for the first instance.

This is just a simple example of how SHAP can be used to interpret the predictions of a machine learning model. In practice, SHAP can be used with a wide range of machine

learning models and datasets, and can provide valuable insights into how these models are making their predictions.

Pros and Cons

Pros:

- Provides a powerful tool for interpreting and explaining the predictions of machine learning models.
- Works with a wide range of machine learning models, including black box models.
- Can be used for a variety of tasks, including feature selection, model debugging, and model governance.
- Provides a range of visualizations for exploring and interpreting model predictions.
- Based on a well-established concept from cooperative game theory (Shapley values).
- Has an active community and is widely used in industry and academia.

Cons:

- Can be computationally intensive, especially for large datasets or complex models.

- Can be difficult to interpret and understand, especially for users who are not familiar with the underlying concepts and methods.
- Requires some knowledge of Python and machine learning concepts to use effectively.
- Can be sensitive to the choice of hyperparameters and other settings.
- May not always provide clear or definitive explanations for model predictions.

SHAP is a powerful and widely-used tool for interpreting and explaining machine learning models. However, as with any tool, it has its limitations and requires some expertise to use effectively. It is important to carefully consider the trade-offs and limitations of any model interpretability tool when choosing the right one for a particular application.

REGEX

Python Regex (Regular Expression) library is a powerful tool used for pattern matching and text processing. It provides a set of functions and meta-characters that allow us to search and manipulate strings using complex patterns. The regular expression is a sequence of characters that define a search pattern. Python's built-in **re** module provides support for regular expressions in Python. It is a widely used library for performing various text manipulation tasks such as string matching, searching, parsing, and replacing.

An example of using Python's regex library **re** to extract information from a complex string:

```
import re

# Example string to search through
text = "My phone number is (123) 456-7890 and
my email is example@example.com."

# Define regex patterns to search for
phone_pattern = re.compile(r'\(\d{3}\)\s\d{3}-
\d{4}')  # Matches phone numbers in (123) 456-
7890 format
email_pattern = re.compile(r'\b[\w.-
]+?@\w+?\.\w+?\b')  # Matches email addresses

# Search for matches in the text
phone_match = phone_pattern.search(text)
email_match = email_pattern.search(text)

# Print out the results
if phone_match:
    print("Phone number found:",
phone_match.group())
```

```
else:
    print("Phone number not found.")

if email_match:
    print("Email found:", email_match.group())
else:
    print("Email not found.")


Output:

Phone number found: (123) 456-7890
Email found: example@example.com
```

In this example, we use regular expressions to define patterns to search for a phone number and an email address in a complex string. The patterns are compiled using the **re.compile()** function, and then searched for using the **search()** function. The **group()** function is used to retrieve the actual matched text.

Pros and Cons

Pros:

- Powerful: Regular expressions are a powerful way to search and manipulate text.
- Efficient: The Python Regex library is optimized for performance and can handle large amounts of text quickly.
- Versatile: Regular expressions can be used for a wide range of tasks, from simple string matching to complex text parsing and manipulation.

- Flexible: The Python Regex library allows for a great deal of customization, allowing you to create complex patterns and match specific patterns in text.

Cons:

- Steep learning curve: Regular expressions can be difficult to learn, particularly for those new to programming.
- Easy to misuse: Because of their complexity, regular expressions can be prone to errors and can be difficult to debug.
- Limited functionality: While the Python Regex library is powerful, it has some limitations and may not be suitable for all text processing tasks.
- Less readable: Regular expressions can be less readable than other forms of text processing code, making it more difficult to maintain and update code.

WEB FRAMEWORK

A web framework is a software framework designed to simplify the development of web applications by providing a set of reusable components and tools for building and managing web-based projects. It provides a standardized way to build and deploy web applications by providing a structure, libraries, and pre-written code to handle everyday tasks such as request handling, routing, form processing, data validation, and database access.

Web frameworks typically include programming tools and libraries, such as templates, middleware, and routing mechanisms, that enable developers to write clean, maintainable, and scalable code for web-based projects. In addition, they abstract away much of the low-level details of web development, allowing developers to focus on the high-level functionality of their applications.

There are many web frameworks available in various programming languages, including Python (Django, Flask), Ruby on Rails, PHP (Laravel, Symfony), and JavaScript (React, Angular, Vue.js). These frameworks vary in features, performance, ease of use, and community support.

Web frameworks have become essential for web development because they provide a standardized way to build and maintain web applications, making it easier for developers to build complex web-based projects in less time and with fewer errors.

FLASK

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. There are extensions for object-relational mappers, form validation, upload handling, various open authentication technologies, and more.

An example code usage of Flask:

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run()
```

This creates a simple Flask web application that listens for requests on the root URL (**/**) and returns the string **'Hello, World!'** as a response. When you run this code and navigate to **http://localhost:5000/** in your web browser, you should see the message "Hello, World!" displayed on the page.

Pros and Cons

Pros:

- is a lightweight web framework that is easy to set up and use.
- has a simple and intuitive API that makes it easy to develop web applications.
- provides great flexibility when it comes to database integration, allowing developers to use any database they choose.
- The framework is highly customizable, with a large number of third-party extensions available to add functionality to your application.
- has good community support, with a large number of tutorials, resources, and examples available.

Cons:

- is not as powerful as some of the larger web frameworks, such as Django, which may make it less suitable for larger and more complex projects.
- requires developers to make more decisions about how to structure their application, which can make it more challenging for beginners.
- does not provide built-in support for tasks like form validation or user authentication, which can add additional development time for these features.

- As Flask is not an opinionated framework, it requires more configuration and setup, which can be daunting for developers who are not familiar with web development.
- It is not suitable for developing high-performance web applications that require a lot of concurrency, due to its single-threaded nature.

SELENIUM

Selenium is a library that enables web automation and testing by providing a way to interact with web pages programmatically. It allows developers to automate web browsers, simulate user interactions with websites, and scrape web data.

Selenium is widely used in testing and automation of web applications. It supports various programming languages including Python, Java, C#, Ruby, and JavaScript, and can work with different browsers such as Chrome, Firefox, Safari, and Internet Explorer.

With Selenium, you can create scripts to automate repetitive tasks such as form filling, clicking buttons, navigating through pages, and extracting data from web pages.

Overall, Selenium is a powerful tool for web automation and testing and can greatly simplify tasks that would otherwise be time-consuming and laborious.

An example code usage of Selenium for web scraping:

```
from selenium import webdriver
from selenium.webdriver.common.by import By

# Set up the driver
driver =
webdriver.Chrome('path/to/chromedriver')

# Navigate to the website you want to scrape
driver.get('https://www.example.com')
```

```
# Find the element you want to interact with
and perform actions
element = driver.find_element(By.XPATH,
'//button[@id="button-id"]')
element.click()

# Extract the data you want from the website
data_element = driver.find_element(By.XPATH,
'//div[@class="data-class"]')
data = data_element.text

# Clean up and close the driver
driver.quit()
```

In this example, we're using the Chrome driver and navigating to a website. We then find a button element and click it, which causes some data to load on the page. We then find the element that contains the data we want to scrape and extract its text. Finally, we clean up and close the driver.

Note that web scraping can be a legally and ethically gray area, and some websites may have terms of service that prohibit it. Be sure to check the website's policies and be respectful in your scraping activities.

Pros and Cons

Pros:

- Can interact with web pages as if you were using a web browser, allowing for more complex scraping tasks

- Supports a wide range of browsers including Chrome, Firefox, Safari, and Internet Explorer
- Can handle dynamic content loaded by JavaScript, AJAX, and other technologies
- Supports headless browsing, which allows you to run the scraping tasks without a graphical user interface
- Supports various programming languages including Python, Java, Ruby, and C#

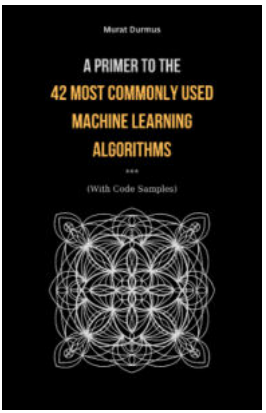
Cons:

- Can be slower than other web scraping libraries due to its reliance on browser automation
- Requires more setup and configuration compared to other libraries
- Can be more resource-intensive, as it requires a browser instance to run
- May not be suitable for all web scraping tasks, particularly those that require high speed and scalability

Also available from the Author

A PRIMER TO THE 42 MOST COMMONLY USED MACHINE LEARNING ALGORITHMS

(WITH CODE SAMPLES)



Whether you're a data scientist, software engineer, or simply interested in learning about machine learning, "A Primer to the 42 Most commonly used Machine Learning Algorithms (With Code Samples)" is an excellent resource for gaining a comprehensive understanding of this exciting field.

Available on Amazon:
https://www.amazon.com/dp/B0BT911HDM

Kindle: **(B0BT8LP2YW)**
Paperback: **(ISBN-13: 979-8375226071)**

MINDFUL AI

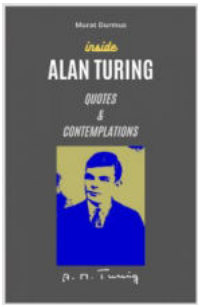
Reflections on Artificial Intelligence

Inspirational Thoughts & Quotes on Artificial Intelligence
(Including 13 illustrations, articles & essays for the fundamental understanding of AI)

Available on Amazon:
https://www.amazon.com/dp/B0BKMK6HLJ

Kindle: **(ASIN: B0BKLCKM22)**
Paperback: **(ISBN-13: 979-8360396796)–**

INSIDE ALAN TURING: QUOTES & CONTEMPLATIONS



Alan Turing is generally considered the father of computer science and artificial intelligence. He was also a theoretical biologist who developed algorithms to explain complex patterns using simple inputs and random fluctuation as a side hobby. Unfortunately, his life tragically ended in suicide in 1954, after he was chemically castrated as punishment (instead of prison) for 'criminal' gay acts.

"We can only see a short distance ahead, but we can see plenty there that needs to be done." ~ Alan Turing

Available on Amazon:
https://www.amazon.com/dp/B09K25RTQ6

Kindle: **(ASIN: B09K3669BX)**
Paperback: **(ISBN- 979-8751495848)**