

# Python Polars

Guide Complete - Comprehensive Guide

Chapter 1: Getting Started with Polars

What will you learn?	Working with DataFrames and Series
	Lazy evaluation and optimization
	Filtering and selecting data
	Method chaining
	Processing large files

# Table of Contents

1. Introduction to Polars	3
2. DataFrame - Basic Data Structure	4
3. Series - Single Column	6
4. LazyFrame - Lazy Evaluation	7
5. Selecting Columns and Filtering Data	9
6. Creating, Modifying and Deleting Columns	11
7. Method Chaining	13
8. Processing Large Files	14
9. Exercises	16
10. Summary and Resources	18

# 1. Introduction to Polars

Polars is a modern and extremely fast Python library for data processing. It serves as an excellent alternative to Pandas, with significantly improved performance.

## Advantages of Polars:

- Speed - Up to 10-100 times faster than Pandas
- Efficient memory utilization - Smart use of RAM
- Parallel processing - Utilizes all CPU cores
- Lazy Evaluation - Automatic query optimization
- Clean and intuitive syntax
- Type Safety - Strong type checking

## Installation:

```
pip install polars
```

## 2. DataFrame - Basic Data Structure

A DataFrame is a two-dimensional data structure (table) containing rows and columns. Each column contains data of the same type, and each row represents a single record.

### Creating a DataFrame:

```
import polars as pl

df = pl.DataFrame({
    'nums': [1, 2, 3, 4, 5],
    'letters': ['a', 'b', 'c', 'd', 'e']
})
```

### Reading from CSV file:

```
df = pl.read_csv('data.csv')
```

### Important DataFrame properties:

Property	Description	Example
schema	Table structure (columns + types)	df.schema
columns	List of column names	df.columns
dtypes	List of data types	df.dtypes
shape	Dimensions (rows, cols)	df.shape
height	Number of rows	df.height
width	Number of columns	df.width

### 3. Series - Single Column

A Series is a one-dimensional data structure representing a single column from a table. It contains values of the same type and is optimized for fast vectorized operations.

#### Creating a Series:

```
series = pl.Series('my_series', [1, 2, 3, 4, 5])
```

#### Accessing Series from DataFrame:

```
age_series = df['Age']
```

#### Common operations on Series:

- mean() - Average
- sum() - Sum
- min() / max() - Minimum / Maximum
- median() - Median
- std() - Standard deviation
- n\_unique() - Unique values
- null\_count() - Missing values

## 4. LazyFrame - Lazy Evaluation

LazyFrame is a 'lazy' version of DataFrame where operations are not executed immediately. Polars builds an optimal execution plan and only executes when calling .collect().

### Advantages of LazyFrame:

- Automatic optimization - Polars optimizes the query
- Better performance - Parallel and efficient execution
- Memory savings - Processes only what's needed
- Predicate Pushdown - Early filtering of data
- Projection Pushdown - Reads only required columns

### Example usage:

```
result = (
    df.lazy()
    .filter(pl.col('Age') > 30)
    .select(['Name', 'Age', 'Fare'])
    .sort('Fare', descending=True)
    .collect() # Execute here
)
```

## 5. Selecting Columns and Filtering Data

One of the most common uses when working with data is selecting specific columns and filtering rows.

### Selecting columns:

```
# Single column
df.select('Name')

# Multiple columns
df.select(['Name', 'Age', 'Fare'])

# Using pl.col() with transformations
df.select(pl.col('Name'), (pl.col('Age') * 12).alias('Age_in_months'))
```

### Filtering rows:

```
# Simple filter
df.filter(pl.col('Age') > 30)

# Multiple conditions with AND (&)
df.filter((pl.col('Age') > 30) & (pl.col('Sex') == 'female'))

# Conditions with OR (|)
df.filter((pl.col('Age') > 70) | (pl.col('Age') < 5))
```

# 10. Summary and Resources

## What did we learn in this chapter?

Topic	Key Concepts
DataFrame	Two-dimensional table, schema, columns, dtypes
Series	Single column, statistical operations
LazyFrame	Lazy evaluation, optimization, .collect()
Selecting	.select(), pl.col(), column transformations
Filtering	.filter(), conditions with & and
Modifying	.with_columns(), .drop(), .rename()
Chaining	Connecting multiple operations
Large Files	scan_csv(), streaming, batches

## Additional Resources:

- Official Polars Docs: <https://pola-rs.github.io/polars/>
- User Guide: <https://pola-rs.github.io/polars-book/>
- API Reference: <https://pola-rs.github.io/polars/py-polars/html/reference/>
- Discord Community: <https://discord.gg/4UfP5cfBE7>
- GitHub: <https://github.com/pola-rs/polars>

Good luck with Polars! Python Polars - Fast, efficient, and modern!