Swiss UMEF
UNIVERSITY OF APPLIED SCIENCES INSTITUTE

Big Data Analytics

# 03: In-Memory Analytics with Pandas. Introduction to Pandas

Instructor: Oleh Tymchuk

#03: Agenda

- What is Pandas?
- Pandas Data Structures
- Loading Data into Pandas
- Practical cases
- Useful Links

What is Pandas?

What is Pandas?

- Fast, flexible, and open-source Python library
- Designed for data manipulation and analysis
- “Pandas” = Panel Data
- Optimized for in-memory data operations

What Types of Data Can Pandas Handle?

Tabular Data

- Similar to an SQL table or an Excel spreadsheet
- Supports heterogeneously-typed columns (e.g., numerical + categorical data)

Time Series Data

- Works with both ordered and unordered time series

Matrix Data

- Supports homogeneous or heterogeneous matrix-like structures

Observational & Statistical Data

- Can store any type of dataset, even unlabeled data

Key Features of Pandas

- Easy handling of missing data
- Label-based indexing and slicing
- Powerful data filtering and transformation
- Efficient group-by operations and aggregation
- Integration with other data analysis tools

Pandas Cohabitation

Jupyter Notebooks & Google Colab:

- Perfect for interactive data analysis and visualization

Plotting with:

- matplotlib: Basic plotting
- seaborn | plotly: Nicer statistical plots

Numerical analysis with:

- numpy: Efficient numerical computations

Modelling with:

- statsmodels: Statistical modeling
- scikit-learn: Machine learning

Installing & Setup

```
# Install via pip/conda:
!pip install pandas


# Import the library:
import pandas as pd


# Check version:
print(pd.__version__)
```

Pandas Data Structures

Data Structures

Dimensions	Name	Description
1	Series	1D labeled homogeneously-typed array
2	DataFrame	General 2D labeled, size-mutable tabular structure with potentially heterogeneously-typed column

Labeled Data

Labeled data refers to data that has identifiers (labels) for rows and columns.

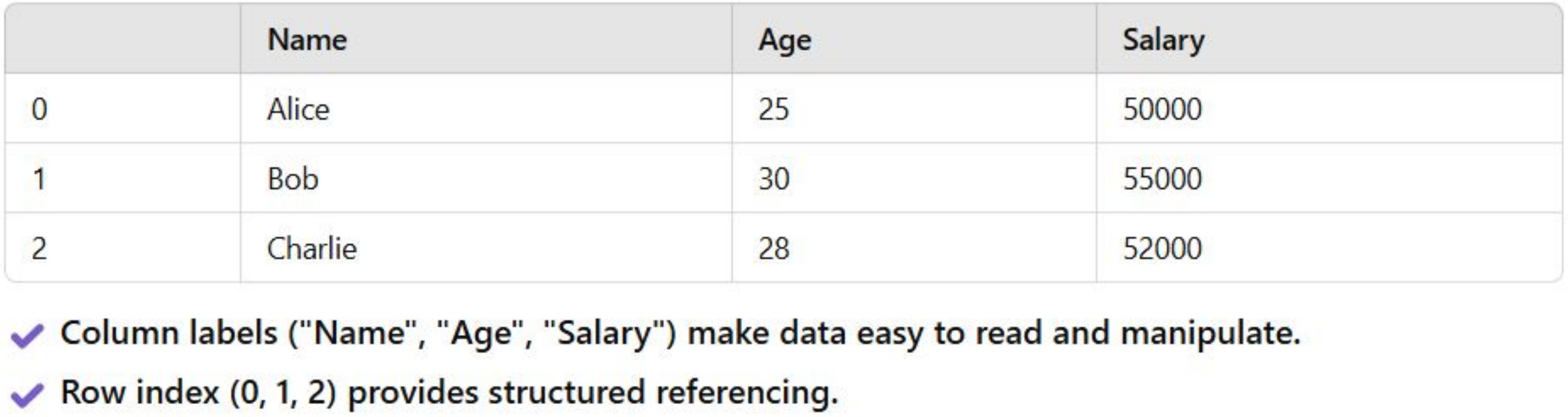
Key Components in Pandas:

- Index: labels for rows (e.g., 0, 1, 2, ... or custom labels like ['A', 'B', 'C']).
- Columns: labels for columns (e.g., ['Name', 'Age', 'City']).

Why Labeled Data Matters:

- Enables intuitive data access using meaningful labels instead of numeric positions
- Supports alignment of data during operations
- Makes data more readable and interpretable

Labeled Data

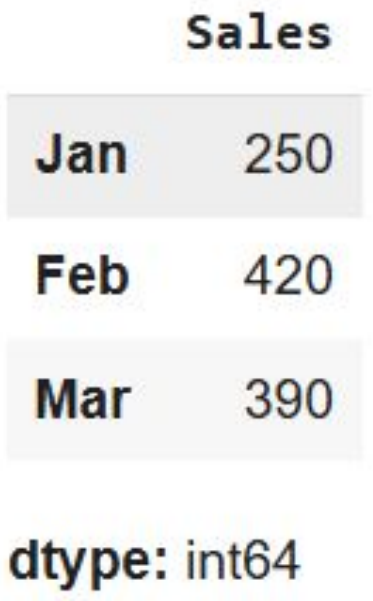


	Name	Age	Salary
0	Alice	25	50000
1	Bob	30	55000
2	Charlie	28	52000

- ✓ Column labels ("Name", "Age", "Salary") make data easy to read and manipulate.
- ✓ Row index (0, 1, 2) provides structured referencing.

Series

- A one-dimensional labeled array
- Can hold any data type: integers, strings, floats, etc.
- Each element has a value and a label (index)
- Can be created from lists, dictionaries, or NumPy arrays
- Similar to a single column in a spreadsheet or a list with labels

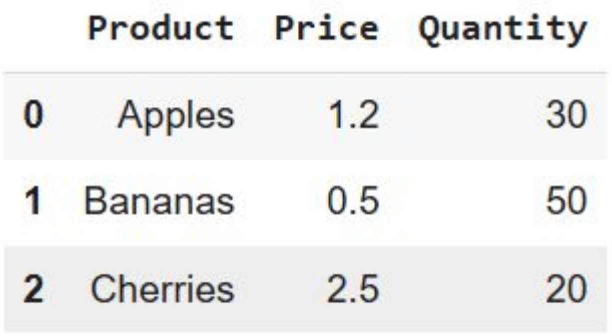


	Sales
Jan	250
Feb	420
Mar	390

dtype: int64

DataFrame

- A two-dimensional labeled data structure
- Similar to a spreadsheet or SQL table
- Consists of rows and columns
- Labeled axes: rows (index) and
  columns (column names)
- Each column is a Series
- Can be created from dictionaries, lists, NumPy arrays, or other DataFrames



	Product	Price	Quantity
0	Apples	1.2	30
1	Bananas	0.5	50
2	Cherries	2.5	20

Loading Data into Pandas

Common Data Types in Datasets

Text: Emails, customer reviews, chat logs, social media posts.

Numbers: Statistics, financial transactions, measurements, sensor data.

Categorical Data: Product categories, survey responses, customer segments.

Date/Time: Timestamps, event logs, transaction dates.

Boolean: Yes/No, True/False, binary indicators.

Common Data Formats. CSV

- CSV (Comma-Separated Values) is a simple file format for storing tabular data
- Each line represents a row, and columns are separated by commas
- Commonly used for data exchange between applications
- Example:

```
Name, Age, City

John, 28, Kyiv

Anna, 24, Lviv
```

Common Data Formats. JSON

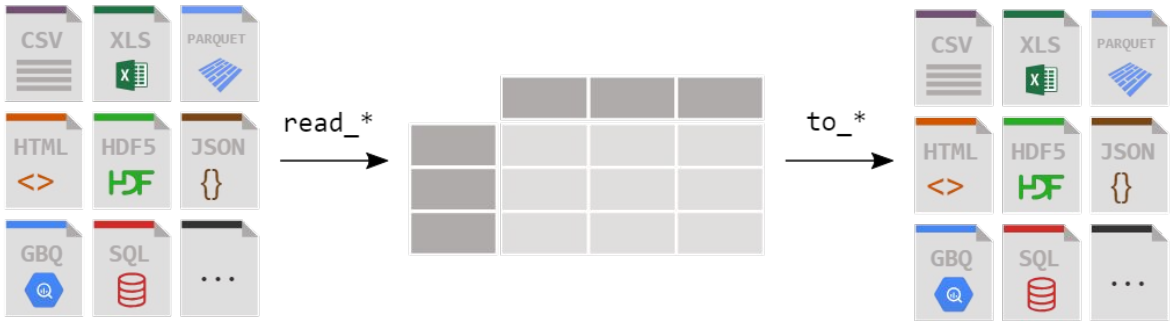
- JSON (JavaScript Object Notation) is a lightweight format for storing and exchanging data
- Uses key-value pairs and supports nested structures
- Commonly used in APIs and configuration files
- Example:

```
{
    "name": "John",
    "age": 28,
    "city": "Kyiv”
}
```

Loading Data Functions

Common data sources:

- CSV (.csv)
- Excel (.xlsx, .xls)
- SQL databases
- JSON (.json)



Functions to read data:

- pd.read_csv()
- pd.read_excel()
- pd.read_sql()
- pd.read_json()

Pandas Practice

Useful Links

Pandas. Getting started

Pandas. Intro to data structures

Pandas. How do I read and write tabular data?