Python Summer Course

Course 5: Pandas

Théophile Gentilhomme

October 24, 2025

Introduction

Pandas is a Python library for **data analysis and manipulation**.

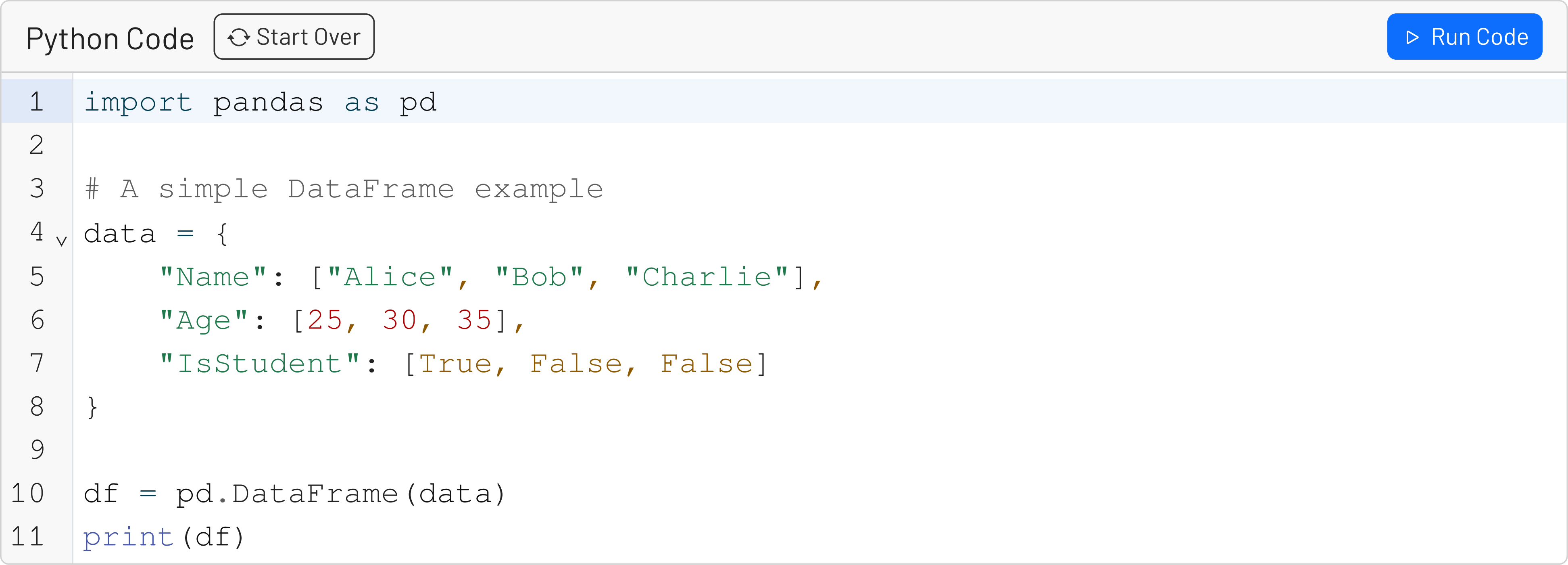
It provides two main data structures:

- **Series** → 1D labeled array
- **DataFrame** → 2D labeled table (like Excel or a database)

💡 Why Use Pandas?

- Built on top of NumPy
- Makes it easy to:
  - Load and clean data
  - Filter, transform, and group data
  - Perform statistics and summaries
- Works well with CSV, Excel, SQL, and more

Example



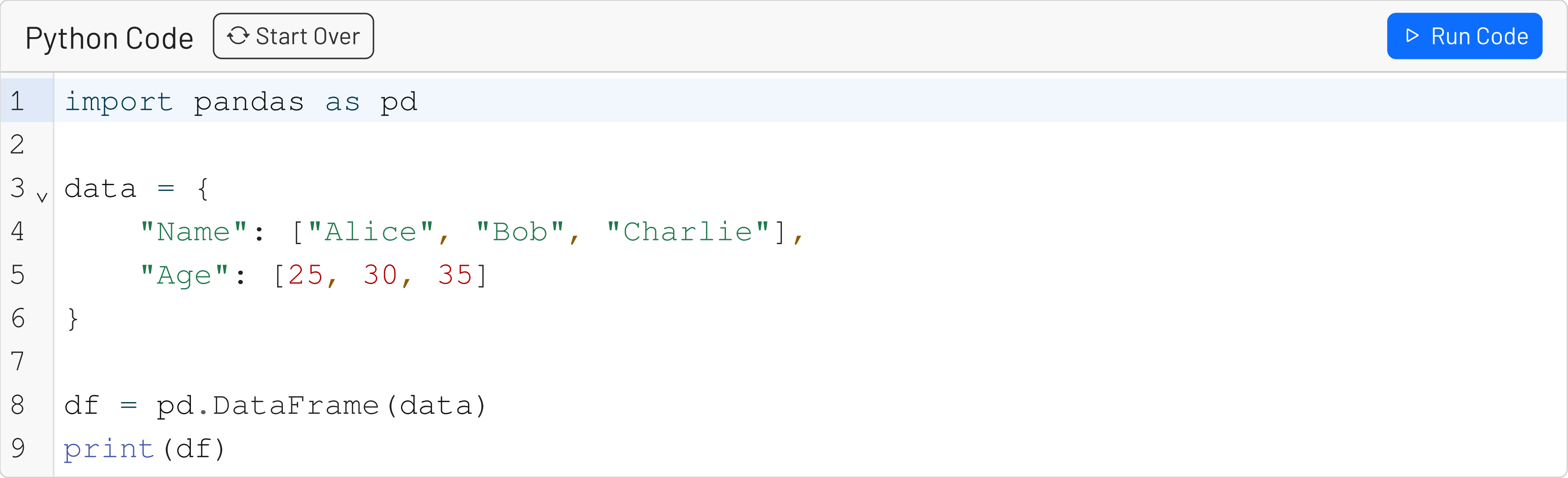
```python
import pandas as pd

# A simple DataFrame example
data = {
    "Name": ["Alice", "Bob", "Charlie"],
    "Age": [25, 30, 35],
    "IsStudent": [True, False, False]
}

df = pd.DataFrame(data)
print(df)
```

✅ Each **key** becomes a column name
✅ Each **value** must be a list or array of the **same length**

Pandas is the perfect tool for **data science**, **bioinformatics**, and any task involving **structured data**.

DataFrame = 2D Table

A DataFrame is like a table with **rows and columns**.



```python
import pandas as pd

data = {
    "Name": ["Alice", "Bob", "Charlie"],
    "Age": [25, 30, 35]
}

df = pd.DataFrame(data)
print(df)
```

- Rows are automatically indexed (0, 1, 2...)
- Columns are labeled (like a dict of Series)

Series = 1D Column or Row

A Series is like a **single column** with an index:

Python Code Start Over ▷ Run Code

```python
# Column access
ages = df["Age"]
print(ages)
```

- Has both values **and** an index
- Acts like a NumPy array + labels

Access by Label `.loc[row_label, column_label]`

You can access specific element of the table using row/column label with `.loc[]`.

Python Code ⟳ Start Over ▷ Run Code

```python
# Access one element
print(df.loc[0, "Age"])     # Age of Alice
```

Modify an element

Python Code ⟳ Start Over ▷ Run Code

```
df.loc[2, "Age"] = 31
print(df)
```

Row/column access

Python Code Start Over Run Code

```python
print(df.loc[1])          # Entire row (Series)
print(df.loc[:, "Name"])  # Entire column (Series)
```

Access by Index

`.iloc[row_index, column_index]`

Python Code ⟳ Start Over ▷ Run Code

```python
print(df.iloc[0, 1])          # Same as above (row 0, column 1)
```

Same as `.loc[]` but with indexes

Python Code ⟳ Start Over ▷ Run Code

```python
# Slicing
print(df.iloc[0:2, 0:2])   # First 2 rows and 2 columns

# Can be done with .loc[] too
```

Add a Column / Filter Rows

You can add columns just like a dictionnary

Python Code ⟳ Start Over ▷ Run Code

```python
df["HES"] = [True, False, False]
```

Filter with conditions (Boolean mask)

You can filter the row using boolean mask with the same lenght as the columns.

Usually, you filter using value in one or several columns

Python Code ⟳ Start Over ▷ Run Code

```python
print(df.loc[df["Age"] > 30])  # Rows where Age > 30

# Combine conditions with `&` (and) `|` (or)
```

Loading CSV and Excel Files

Pandas makes it easy to import real-world data from common formats.

This is usually what your are going to do.

Examples for CSV and Excel files

Loading files

```python
import pandas as pd
# Load a CSV file from a URL or local path
df = pd.read_csv("my_data.csv")  # or from a URL
print(df.head())  # Show first 5 rows
```

- Use `sep=";"` if using semicolon separator
- Use `encoding="utf-8"` if needed
- Can use a column as row indexes `index_col="thisCol"`

```python
# Load an Excel File
df = pd.read_excel("my_data.xlsx", sheet_name="Sheet1")
```

Compute Stats and Explore Data

Pandas offers many built-in tools to summarize and analyze your data quickly.

Python Code ⟳ Start Over ▷ Run Code

```python
import pandas as pd

df = pd.DataFrame({
    "Age": [25, 30, 35, 28],
    "Score": [80, 90, 88, 75],
    "Something": ["A", "B", "C", "D"]
})
```

Quick Exploration

Python Code ⟳ Start Over ▷ Run Code

```python
print(df.head())         # First 5 rows
print(df.tail(2))        # Last 2 rows
print(df.columns)        # List of column names
print(df.index)          # Row index
```

Basic Statistics

Python Code ⟳ Start Over ▷ Run Code

```python
print(df.mean())       # Column-wise average
print(df["Age"].max())   # Max value in a column
```

Data Overview

Python Code Start Over ▷ Run Code

```
print(df.info())      # Structure and types
print(df.describe()) # Summary stats for numeric columns
```

Handling Missing Values in Pandas

Missing values are common in real-world data. Pandas uses `NaN` (Not a Number) to represent them.

Python Code ⟳ Start Over ▷ Run Code

```python
import pandas as pd
import numpy as np

df = pd.DataFrame({
    "Name": ["Alice", "Bob", "Charlie"],
    "Age": [25, np.nan, 35]
})

### Detect Missing Data
print(df.isna())        # True where values are missing
print(df.isna().sum()) # Count missing per column
```

Drop or Fill

Python Code Start Over Run Code

```
df.dropna()                    # Remove rows with missing values
df.fillna(0)                   # Replace missing with 0
df["Age"].fillna(df["Age"].mean(), inplace=True)  # Replace with column mean
```

Simple Plots with Pandas

Pandas makes it easy to plot data directly from a DataFrame.

It uses Matplotlib, which is a Python library commuly used for plots.

You can customize the plots using Matplolib functionalities.

Line Plot

Python Code ⟳ Start Over ▷ Run Code

```python
# You do not need this code: just for the presentation
import micropip
await micropip.install("matplotlib")  # One-time install
```

Python Code ⟳ Start Over ▷ Run Code

```python
import pandas as pd
import matplotlib.pyplot as plt

df = pd.DataFrame({
    "Year": [2020, 2021, 2022, 2023],
    "Revenue": [100, 150, 200, 250]
})

df.plot(x="Year", y="Revenue", kind="line", title="Revenue Over Time")

# Call pyplot show
plt.show() # You normally do need this (only for the presentation)
```

Other Plot Types

Python Code ⟳ Start Over ▷ Run Code

```
df.plot(kind="bar")          # Bar chart
plt.show()
```

Python Code ⟳ Start Over ▷ Run Code

```
df.plot(kind="hist")         # Histogram
plt.show()
```

Python Code ⟳ Start Over ▷ Run Code

```
df.plot(kind="box")          # Boxplot
plt.show()
```

Python Code ⟳ Start Over ▷ Run Code

```
df.plot(kind="scatter", x="Year", y="Revenue")  # Scatter plot
plt.show()
```

And many more to cover on Pandas!

- `group_by()`
- Time series
- interpolation
- ...

Your turn!

You are given a dataset `gene_expression_data.cvs`. There is no descriptions, but you need to explore and make some analyses.

1. Load the file using Pandas
2. Explore it using the functions seen in this course
3. Check how many treated vs healthy samples there are using `value_counts()` on the column or using boolean condition

4. Compute the mean expression for `Tissue_B` (optional: all tissues)
5. Calculate the difference between Tissue A and B and store it in a new column
6. Filter the genes where Tissue C expression is greater than 9 and Biomarker is True (i.e. two boolean expression combined with `&`)

7. Plot a scatter plot of Tissue A vs Tissue B, colored with biomarker. Tip: use `scatter` with

```
colors = ["red" if b else "blue" for b in df["Biomarker"]]
```

🔙 main

Solution

1. Load the CSV

Show Solution Enter code

2. Explore the Data

Show Solution Enter code

🔙 main

3. Count Conditions

Show Solution Enter code

4. Mean Expression per Tissue

Show Solution Enter code

5. Add Difference Column (Tissue A – B)

Show Solution Enter code

6. Filter: Tissue C > 9 and Biomarker = True

Show Solution Enter code

7. Scatter Plot (Tissue A vs B, colored by Biomarker)

Show Solution Enter code

More references

Python course for data analysis

PANDAS-TUTORIAL

Pandas