Theory: Introduction to pandas

9 minutes 0 / 4 problems solved

In practice, data is often stored in the form of a table, for example, an Excel spreadsheet, a CSV file, or an SQL database. Imagine that you need to analyze this tabular data and get some useful insights from it. Let's think of the task you will need to perform.

First, you will need to load data from different formats preserving its tabular structure and probably join several tables together. Then, to perform the actual data analysis, you will definitely want to access different columns, rows, and cells of the tables, compute some overall statistics, create pivot tables and maybe even make basic plots. Is there a tool in Python that combines all these functionalities? And the answer is yes!

This topic will introduce you to the `pandas`, a powerful open-source library for data manipulation and analysis. You will install the library and get an idea of its main functionality.

§1. Installing pandas

`pandas` is not included in the standard Python library, so you might need to install it separately, for example, using `pip`. Type the following in your command line:

```
pip install pandas
```

Note that pandas is also dependent on other libraries (for example, `numpy` for vector computations) that you need to be installed as well. Besides, there are many optional dependencies. For instance, if you want to use pandas data visualization functionality, you need to install `matplotlib`, a plotting library in Python. You install those libraries using `pip`, too.

Once the installation is complete, you will be able to import it in your code. Since `pandas` is quite a long name, it is commonly abbreviated and imported as `pd`:

```
import pandas as pd
```

Note that new versions of pandas are released once in a while, with new functionality and fixed bugs. So it would be a good idea to keep an eye on the updates. You can easily upgrade the version of pandas on your machine with the command: `pip install --upgrade pandas`.

§2. Data structures in pandas

The two data structures of `pandas` are `Series` (1D) and `DataFrame` (2D). You will get more familiar with them in the dedicated topics, but an overview is given below.

`Series` is a one-dimensional array that stores elements of the same data type.

Each element stored in a `Series` is associated with a label called **index**. By default, this index is just the sequence 0, 1, 2, ... However, any custom values can be used. For example, when analyzing time series, timestamps are typically set as indexes.

Current topic:

Introduction to pandas

Topic depends on:

- Typical ML pipeline — Stage 1
- Pip

Topic is required for:

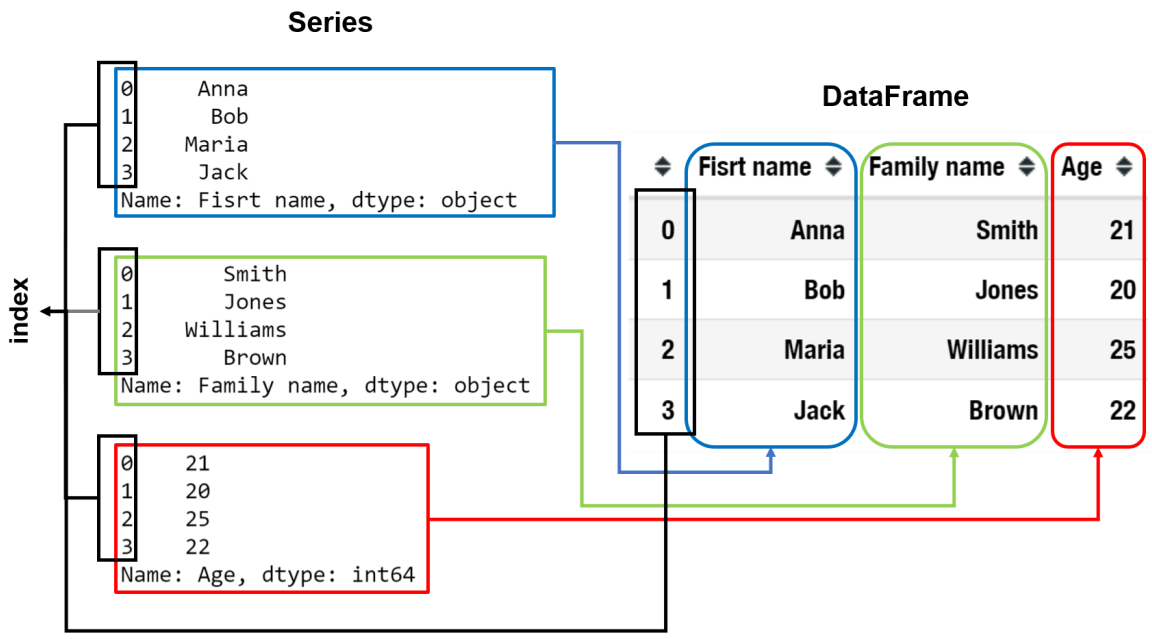
- Series

Table of contents:

Indexes, as well as automatic and explicit data alignment based on them, are the core of `pandas`.

`DataFrame`, in turn, is a two-dimensional data structure that is used to represent tabular data with columns of potentially different data types. You can see `DataFrame` as a table each column of which is a `Series` object. In other words, the `DataFrame` is a container for `Series`, while `Series` is a container for scalars.

This is illustrated with an example below. Three `Series` objects store names, surnames, and ages of students respectively, while the `DataFrame` combines this information in a single table.



Note that each row in the `DataFrame` is also associated with an index. It is worth mentioning that even though `DataFrame` is a 2D data structure, one can still use it to represent higher-dimensional data by using so-called **multi-index**, that is, assigning several indexes, or labels, to each row. However, this is rarely used in practice.

Tables represented as `DataFrame` slightly differ from spreadsheets to optimize computations. `Series` (= columns of a `DataFrame`) can store data of one type only to perform operations on them faster. It is also not possible to add or remove elements from a `Series`, which helps pandas efficiently store `Series` objects in memory.

§3. What is inside pandas?

As stated in its guidelines, `pandas` aims to become 'the most powerful and flexible open-source data analysis/manipulation tool available in any language'.

The name `pandas` is derived from **'panel data'**, a term that is used in statistics and econometrics to refer to the data sets containing observations over multiple time periods for the same individuals. This library will be helpful if you are working with tabular data, such as data stored in spreadsheets or databases. Apart from that, `pandas` offers great support for time series and provides extensive functionality for working with dates, times, and time-indexed data.

With the help of `pandas`, one can easily perform the most typical data processing steps.

In particular, the package makes it convenient to load and save data, as it supports out-of-the-box integration with many commonly-used tabular formats such as `.csv`, `.xlsx`, as well as SQL databases.

Let's look at what we can get from pandas:

- Intuitive merging and joining data sets allows one to easily combine data from different sources, while flexible reshaping tools help construct pivot tables.
- Missing values in the data are represented as NaN and can be easily handled, for example, replaced by some value, using built-in functionality.
- Besides, if you have `matplotlib`, a Python plotting library, installed, then you can use pandas built-in plotting functionality to make a basic plot

from your data to better understand it.
- Get basic statistical information about our data literally with one line of code.
- Integrate it with other libraries for machine learning like `sklearn`.

Finally, `pandas` is open-source software, which makes it very popular in both academic and commercial domains.

§4. Conclusions

- `pandas` is a flexible tool for data analysis.
- `pandas` is a perfect tool to work with heterogeneous data.
- Two data structures in `pandas` are `Series` (1D) and `DataFrame` (2D).
- `Series` stores values of the same data type, while columns in a `DataFrame` can be of different types.

Report a typo

23 users liked this theory. **0** didn't like it. What about you?

Start practicing

Comments (0) Hints (0) Useful links (0) Show discussion