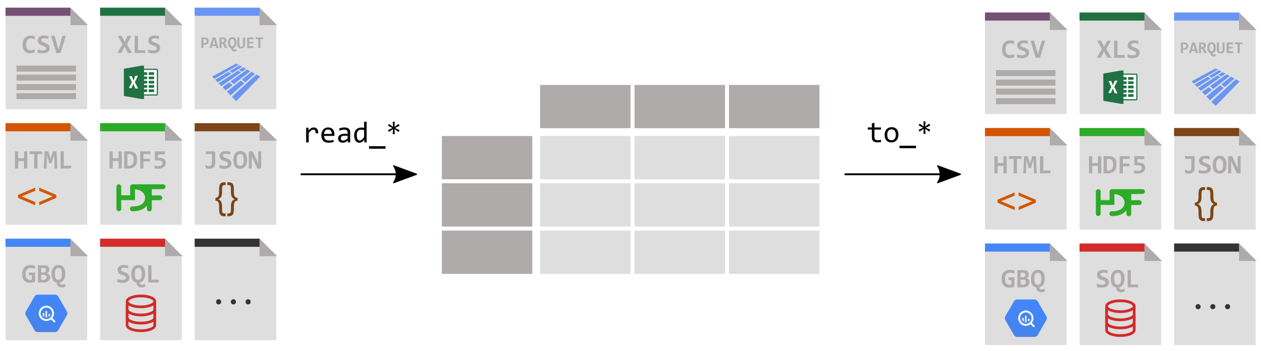
```
In [1]: import pandas as pd
```

Data used for this tutorial:

Titanic data

How do I read and write tabular data?



I want to analyze the Titanic passenger data, available as a CSV file.

```
In [2]: titanic = pd.read_csv("data/titanic.csv")
```

pandas provides the `read_csv()` function to read data stored as a csv file into a pandas `DataFrame`. pandas supports many different file formats or data sources out of the box (csv, excel, sql, json, parquet, …), each of them with the prefix `read_*`.

Make sure to always have a check on the data after reading in the data. When displaying a `DataFrame`, the first and last 5 rows will be shown by default:

```
In [3]: titanic
Out[3]:
     PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0              1         0       3  ...   7.2500   NaN         S
1              2         1       1  ...  71.2833   C85         C
2              3         1       3  ...   7.9250   NaN         S
3              4         1       1  ...  53.1000  C123         S
4              5         0       3  ...   8.0500   NaN         S
..           ...       ...     ...  ...      ...   ...       ...
886          887         0       2  ...  13.0000   NaN         S
887          888         1       1  ...  30.0000   B42         S
888          889         0       3  ...  23.4500   NaN         S
889          890         1       1  ...  30.0000  C148         C
890          891         0       3  ...   7.7500   NaN         Q

[891 rows x 12 columns]
```

I want to see the first 8 rows of a pandas DataFrame.

```
In [4]: titanic.head(8)
Out[4]:
   PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0            1         0       3  ...   7.2500   NaN         S
1            2         1       1  ...  71.2833   C85         C
2            3         1       3  ...   7.9250   NaN         S
3            4         1       1  ...  53.1000  C123         S
4            5         0       3  ...   8.0500   NaN         S
5            6         0       3  ...   8.4583   NaN         Q
6            7         0       1  ...  51.8625   E46         S
7            8         0       3  ...  21.0750   NaN         S

[8 rows x 12 columns]
```

To see the first N rows of a `DataFrame`, use the `head()` method with the required number of rows (in this case 8) as argument.

Note

Interested in the last N rows instead? pandas also provides a `tail()` method. For example, `titanic.tail(10)` will return the last 10 rows of the DataFrame.

A check on how pandas interpreted each of the column data types can be done by requesting the pandas `dtypes` attribute:

```
In [5]: titanic.dtypes
Out[5]:
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

For each of the columns, the used data type is enlisted. The data types in this `DataFrame` are integers (`int64`), floats (`float64`) and strings (`object`).

Note

When asking for the `dtypes`, no brackets are used! `dtypes` is an attribute of a `DataFrame` and `Series`. Attributes of a `DataFrame` or `Series` do not need brackets. Attributes represent a characteristic of a `DataFrame`/`Series`, whereas methods (which require brackets) *do* something with the `DataFrame`/`Series` as introduced in the first tutorial.

My colleague requested the Titanic data as a spreadsheet.

```
In [6]: titanic.to_excel("titanic.xlsx", sheet_name="passengers", index=False)
```

Whereas `read_*` functions are used to read data to pandas, the `to_*` methods are used to store data. The `to_excel()` method stores the data as an excel file. In the example here, the `sheet_name` is named *passengers* instead of the default *Sheet1*. By setting `index=False` the row index labels are not saved in the spreadsheet.

Skip to main content

```
In [7]: titanic = pd.read_excel("titanic.xlsx", sheet_name="passengers")
```

```
In [8]: titanic.head()
Out[8]:
   PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0            1         0       3  ...   7.2500   NaN         S
1            2         1       1  ...  71.2833   C85         C
2            3         1       3  ...   7.9250   NaN         S
3            4         1       1  ...  53.1000  C123         S
4            5         0       3  ...   8.0500   NaN         S

[5 rows x 12 columns]
```

I'm interested in a technical summary of a `DataFrame`

```
In [9]: titanic.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

The method `info()` provides technical information about a `DataFrame`, so let's explain the output in more detail:

- It is indeed a `DataFrame`.
- There are 891 entries, i.e. 891 rows.
- Each row has a row label (aka the `index`) with values ranging from 0 to 890.
- The table has 12 columns. Most columns have a value for each of the rows (all 891 values are `non-null`). Some columns do have missing values and less than 891 `non-null` values.
- The columns `Name`, `Sex`, `Cabin` and `Embarked` consists of textual data (strings, aka `object`). The other columns are numerical data with some of them whole numbers (aka `integer`) and others are real numbers (aka `float`).
- The kind of data (characters, integers,…) in the different columns are summarized by listing the `dtypes`.
- The approximate amount of RAM used to hold the DataFrame is provided as well.

REMEMBER

- Getting data in to pandas from many different file formats or data sources is supported by `read_*` functions.
- Exporting data out of pandas is provided by different `to_*` methods.
- The `head`/`tail`/`info` methods and the `dtypes` attribute are convenient for a first check.

Skip to main content

To user guide

For a complete overview of the input and output possibilities from and to pandas, see the user guide section about reader and writer functions.

Previous
What kind of data does pandas handle?

Next
How do I select a subset of a `DataFrame`?