Pandas for data analysis

Call for pandas library

Call for pandas library

In
```
import pandas
import pandas as pd
```

The tail() method for outputting rows at the end of a table

In
```
df.tail() # last five rows
df.tail(15) # last fifteen rows
```

DataFrame() constructor for creating tables

In
```
pd.DataFrame(data = data, columns =
columns)
# data argument is a list of data,
# columns argument is a list of
# column names
```

The read_csv() method for reading CSV files

In
```
df = pd.read_csv('path to file')
```

The head() method for outputting starting rows of a table

In
```
df.head() # first five rows
df.head(10) # first ten rows
```

The columns attribute for outputting column names

In
```
df.columns
```

The shape attribute for outputting table size

In
```
df.shape
```

The dtypes attribute for getting information on the data types in a table

In
```
df.dtypes
```

The info() method to see summary information about the table

In
```
df.info()
```

The loc[row, column] attribute giving access to an element identified in DataFrame by row and column

In
```
df.loc[:, 'column']
```

Out

Type	Sample
One cell	`.loc[7, 'column']`
One column	`.loc[:, 'column']`
Multiple columns	`.loc[:, ['column_1', 'column_4']]`
Multiple consecutive columns (slice)	`.loc[:, 'column_5': 'column_8']`
One row	`.loc[1]`
All rows, starting with the given row	`.loc[1:]`
All rows, up to the given row	`.loc[:3]`
Multiple consecutive rows (slice)	`.loc[2:5]`

Boolean indexing to get the elements by a specific condition

Out

Type	Sample	Shortened notation
All rows meeting the condition	`'df.loc[df.loc[:,'column'] == 'X']'`	`'df[df['column'] == 'X']'`
One column of all rows meeting the condition	`'df.loc[df.loc[:,'column'] == 'X']['column']'`	`'df[df['column'] == 'X']['column']'`
Method application	`'df.loc[df.loc[:,'column'] == 'X']['column'].count()'`	`'df[df['column'] == 'X']['column'].count()'`

Indexing in Series

Out

Type	Sample	Shortened notation
One element	`` `df.loc[7]` ``	`` `df[7]` ``
Multiple elements	`` `df.loc[[5, 7, 10]]` ``	`` `df[[5, 7, 10]]` ``
Multiple consecutive elements (slice)	`` `df.loc[5:10]` `` including 10	`` `df[5:10]` `` not including 10
All, starting with the given element	`` `df.loc[1:]` ``	`` `df[1:]` ``
All, up to the given element	`` `df.loc[:3]` `` including 3	`` `df[:3]` `` not including 3

Glossary

Library
A set of ready-made methods for solving common problems.

CSV
A file format (Comma-Separated Values). Each row of a CSV file is a row in a table with the data separated by commas. The first row contains column headings (if there are any).

Tuple
One-dimensional unchangeable sequence. It's similar to a list; it can also be saved to a variable.

Series
A one-dimensional data structure in pandas where the elements can be accessed by their index. Each index is the number of a separate observation, which is why multiple different Series together make up a DataFrame.

- A Series stores one type of data.
- Every Series has a Name, information about the amount of data in the column (Length), and the data type stored in it (dtype).
- Indexing in a Series is analogous to indexing elements of a column in a DataFrame.

DataFrame
This is a two-dimensional pandas data structure where each element has two indices: row and column.

- Each row is one observation or record of the object. Columns are features of the object.
- DataFrame() is a pandas constructor, and it is used to create a **DataFrame**. Right before the constructor name there's a call to a variable storing a library: pd.DataFrame().
- DataFrame has inherent properties with values that can be requested. They are called attributes. For example, that could be df.shape, the table size, or df.columns, the number of columns.
- We can refer to a cell with data in the DataFrame by its index and the column name. That process is called indexing, and there are different ways to do it for a DataFrame.