Pandas

Jannusch Bigge

15.01.2024

Pandas

Pandas

We now know how to get data into python.

Pandas

We now know how to get data into python.
This is nice but normaly we want to do stuff with it.

Pandas

We now know how to get data into python.
This is nice but normaly we want to do stuff with it.
Quite a lot of stuff is already implemented in numpy.

Pandas

We now know how to get data into python.
This is nice but normaly we want to do stuff with it.
Quite a lot of stuff is already implemented in numpy.
But loading data into numpy may be a bit tricky at some point.

Pandas

We now know how to get data into python.
This is nice but normaly we want to do stuff with it.
Quite a lot of stuff is already implemented in numpy.
But loading data into numpy may be a bit tricky at some point.
$\rightarrow$ Pandas - Python Data Analysis Library

Pandas

Pandas about itself:

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool [...].

Pandas

Pandas about itself:

> pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool [...].

What are the benefits of pandas?

Pandas

Pandas about itself:

> pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool [...].

What are the benefits of pandas?

- fast and efficient **Data Frames**

Pandas

Pandas about itself:

> pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool [...].

What are the benefits of pandas?

- fast and efficient **Data Frames**
- **labled** data

Pandas

Pandas about itself:

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool [...].

What are the benefits of pandas?

- fast and efficient **Data Frames**
- **labled** data
- group by and merging of data

Pandas

Pandas about itself:

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool [...].

What are the benefits of pandas?

- fast and efficient **Data Frames**
- **labled** data
- group by and merging of data
- Time Series functionality

Pandas - Data Frames

Data Frames are the core of pandas.

Pandas - Data Frames

Data Frames are the core of pandas.
They are basically a table with rows and columns.

Pandas - Data Frames

Data Frames are the core of pandas.
They are basically a table with rows and columns.

- columns are called **Series**

Pandas - Data Frames

Data Frames are the core of pandas.
They are basically a table with rows and columns.

- columns are called **Series**
- rows are called **Index**

Pandas - Data Frames

Data Frames are the core of pandas.
They are basically a table with rows and columns.

- columns are called **Series**
- rows are called **Index**
- you can access columns by name

Pandas - Data Frames

Data Frames are the core of pandas.
They are basically a table with rows and columns.

- columns are called **Series**
- rows are called **Index**
- you can access columns by name
- you can access rows by index

Pandas - Data Frames

Data Frames are the core of pandas.
They are basically a table with rows and columns.

- columns are called **Series**
- rows are called **Index**
- you can access columns by name
- you can access rows by index
- you can access cells by name and index

Pandas - Data Frames

First we want to crate a data frame:

```
import pandas as pd
df = pd.DataFrame(
    {'A': [1, 2, 3], 'B': [4, 5, 6]}
    )
print(df)
```

```
   A  B
0  1  4
1  2  5
2  3  6
```

Pandas - Data Frames

And now we want to access the data:

```
# access column
print( df['A'] )
# or
print( df.A )
```

```
0      1
1      2
2      3
Name: A, dtype: int64
```

Pandas - Data Frames

And now we want to access the data:

```
# access row
print( df.loc[0] )
# or
print( df.iloc[0] )
```

```
A      1
B      4
Name: 0, dtype: int64
```

Pandas - Data Frames

And now we want to access the data:

```
# access cell
print( df.loc[0, 'A'] )
# or
print( df.iloc[0, 0] )
# or
print( df.at[0, 'A'] )
# or
print( df.iat[0, 0] )
# or
print( df['A'][0] )
# or
print( df.A[0] )
```

1

Pandas - Data Frames

Beside accessing a whole column or row you can also access a subset of the data frame.

Pandas - Data Frames

Beside accessing a whole column or row you can also access a subset of the data frame.
$\rightarrow$ Slicing

Pandas - Data Frames

Beside accessing a whole column or row you can also access a subset of the data frame.
→ Slicing

```
df[1:3]
# or
df.loc[1:3]
```

```
     A  B
1    2  5
2    3  6
```

Pandas - Data Frames

Beside accessing a whole column or row you can also access a subset of the data frame.

→ Slicing

```
df[1:3]
# or
df.loc[1:3]
```

```
    A  B
1   2  5
2   3  6
```

```
df.loc[1:3, 'A']
```

```
1     2
2     3
Name: A, dtype: int64
```

Pandas - Data Frames

For the next things we need a more complex data frame.

Pandas - Data Frames

For the next things we need a more complex data frame.
First we create a data range:

```
dates = pd.date_range('1/1/2000', periods=8)
```

Pandas - Data Frames

For the next things we need a more complex data frame.
First we create a data range:

```
dates = pd.date_range('1/1/2000', periods=8)
```

And now we create a data frame with random values:

```
df = pd.DataFrame(
    np.random.randn(8, 4),
    index=dates,
    columns=['A', 'B', 'C', 'D']
    )
```

Pandas - Boolean Indexing

We do not know which rows we want to access.

Pandas - Boolean Indexing

We do not know which rows we want to access.
But we know the condition.

Pandas - Boolean Indexing

We do not know which rows we want to access.
But we know the condition.

```
df[df > 0]
```

Or we want to filter by a column:

```
df[df.A > 0]
```

But be carefull and inspect the returned data frame.
It may not be what you expect.

Pandas - Chaining

You can chain commands in pandas.

Pandas - Chaining

You can chain commands in pandas.

```
df[df > 0] = -df
```

Pandas - Chaining

You can chain commands in pandas.

```
df[df > 0] = -df
```

And we can also apply multiple conditions:

```
df[(df.A > 0) & (df.B < 0)]
```

Pandas - Filter with functions

We can also filter with functions.

Pandas - Filter with functions

We can also filter with functions.

```
df[df['A'].isin([1, 2])]
# in our case this will return a empty data frame
```

Pandas - Filter with functions

We can also filter with functions.

```
df[df['A'].isin([1, 2])]
# in our case this will return a empty data frame
```

Or we want to filter by a string:

```
df[df['E'].str.contains('foo')]
# complete mess, we do not even have a column E
```

Or we use a lambda function:

```
df[df['A'].apply(lambda x: x > 0)]
```

Pandas - More features

There are a lot more features in pandas.

Pandas - More features

There are a lot more features in pandas.

- group by

Pandas - More features

There are a lot more features in pandas.

- group by
- merging

Pandas - More features

There are a lot more features in pandas.

- group by
- merging
- time series

Pandas - More features

There are a lot more features in pandas.

- group by
- merging
- time series
- ploting

Pandas - More features

There are a lot more features in pandas.

- group by
- merging
- time series
- ploting
- calculating statistics

Pandas - More features

There are a lot more features in pandas.

- group by
- merging
- time series
- ploting
- calculating statistics
- ...

You can find a lot of examples in the documentation.

Pandas - More features

There are a lot more features in pandas.

- group by
- merging
- time series
- ploting
- calculating statistics
- ...

You can find a lot of examples in the documentation.

`https://pandas.pydata.org/pandas-docs/stable/`

Task

Load data of different rivers and store them in a data frame.

Load data of different rivers and store them in a data frame.

1. Plot the Data
2. Plot the mean for each week
3. Figure out what hight is normal and plot all not normal values