

# Introduktion til Pandas

AAMS - DLE

November 23, 2025

# Indholdsfortegnelse

- [\*\*1 Pandas Series\*\*](#)
- [\*\*2 Pandas DataFrames\*\*](#)
- [\*\*3 Pandas Read CSV\*\*](#)
- [\*\*4 Pandas Write CSV\*\*](#)
- [\*\*5 Pandas Read JSON\*\*](#)
- [\*\*6 Pandas Write JSON\*\*](#)
- [\*\*7 Pandas Statestik\*\*](#)

# Pandas Series

- En Pandas Series er en et-dimensionel datastruktur, som kan indeholde data af enhver type.
- Data i en Series er indekseret, hvilket betyder, at hvert element i en Series har en unik label.
- En Series kan oprettes fra en liste, et array eller en dictionary.

# Oprettelse af en Pandas Series

```
import pandas as pd
temperature = [20, 21, 22, 23, 24]
temp = pd.Series(temperature)
print(temp)
```

# Oprettelse af en Pandas Series

```
import pandas as pd
temperature = [20, 21, 22, 23, 24]
temp = pd.Series(temperature)
print(temp)
```

```
0      20
1      21
2      22
3      23
4      24
dtype: int64
```

# Indexing og slicing i Series

I Pandas Series er det muligt at tilgå elementer ved hjælp af indeksering og slicing.

```
import pandas as pd
temperature = pd.Series([20, 21, 22, 23, 24])
print(temperature[0])
print(temperature[1:3])
```

# Indexing og slicing i Series

I Pandas Series er det muligt at tilgå elementer ved hjælp af indeksering og slicing.

```
import pandas as pd
temperature = pd.Series([20, 21, 22, 23, 24])
print(temperature[0])
print(temperature[1:3])
```

```
20
[21, 22]
```

## Oprettelse af en Pandas Series fra en dictionary

```
import pandas as pd
temperature = {
    '2025:03:08:20:25': 20,
    '2025:03:08:20:30': 21,
    '2025:03:08:20:35': 22,
    '2025:03:08:20:40': 23,
    '2025:03:08:20:45': 24}
temp = pd.Series(temperature)
print(temp)
```

```
2025:03:08:20:25      20
2025:03:08:20:30      21
2025:03:08:20:35      22
2025:03:08:20:40      23
2025:03:08:20:45      24
dtype: int64
```

# Hvad er en Pandas DataFrame?

- En Pandas DataFrame er en todimensionel datastruktur, som kan indeholde data af enhver type.
- Data i en DataFrame er indekseret, hvilket betyder, at hvert element i en DataFrame har en unik label.
- En DataFrame kan oprettes fra en liste, et array, en dictionary eller en anden DataFrame.
- En DataFrame kan sammenlignes med en tabel i en database eller et regneark i Excel.

# Pandas Read CSV

- Pandas kan læse data fra en CSV-fil ved hjælp af `pd.read_csv()` metoden.
- Data fra en CSV-fil kan gemmes i en Pandas DataFrame.
- En CSV-fil er en tekstfil, hvor data er adskilt af kommaer.

# Læsning af en CSV-fil

```
import pandas as pd  
data = pd.read_csv('data.csv')  
print(data)
```

# Appending Series til en CSV-fil

```
import pandas as pd
data = pd.read_csv('data.csv')
humidity = pd.Series([50, 51, 52, 53, 54])
data['humidity'] = humidity
data.to_csv('data.csv', index=False)
```

Ellers

```
temperature = {
    'timestamp': ['2025:03:08:20:25', '2025:03:08:20:30',
                  '2025:03:08:20:35', '2025:03:08:20:40',
                  '2025:03:08:20:45'],
    'temperature': [21, 22, 23, 24, 25]}
df = pd.DataFrame(temperature)
df.to_csv('data.csv', index=False)
```

# filtering af data

Filtering af data i en Pandas DataFrame kan gøres ved hjælp af [] operator.

```
import pandas as pd  
data = pd.read_csv('data.csv')  
print(data.columns)
```

Output:

```
>>>Index(['timestamp', 'temperature'], dtype='object')
```

```
data = data[data['temperature'] > 22]  
data.to_csv('dataFiltreret.csv', index=False)
```

# Sletning af række og kolonne

- En kolonne kan slettes fra en Pandas DataFrame ved hjælp af `drop()` metoden.
- En række kan slettes fra en Pandas DataFrame ved hjælp af `drop()` metoden.
- `axis=1` angiver, at en kolonne skal slettes.
- `axis=0` angiver, at en række skal slettes.
- `inplace=True` angiver, at ændringerne skal gemmes i den oprindelige DataFrame.
- `inplace=False` angiver, at ændringerne skal gemmes i en ny DataFrame.
- `pd.drop(label, axis=0, inplace=True)`

| Parameter | Beskrivelse   | Eksempel                             |
|-----------|---|--------------------------------------|
| labels    | Navnet eller indekset på rækker/kolonner der skal fjernes   | drop(2), drop('column.name', axis=1) |
| axis      | 0 = fjerne rækker, 1 = fjerner kolonner   | axis=0 (rækker), axis=1 (kolonner)   |
| inplace   | True = ændringerne gemmes i den oprindelige DataFrame, False = ændringerne gemmes i en ny DataFrame | inplace=True, inplace=False          |
| errors    | Hvis labels ikke findes, så vil der komme en fejl, med mindre errors='ignore'                       | errors='ignore'                      |

# Eksempel

## Sletning af kolonne

```
import pandas as pd  
data = pd.read_csv('data.csv')  
data.drop('humidity', axis=1, inplace=True)  
print(data)
```

## Sletning af række

```
import pandas as pd  
data = pd.read_csv('data.csv')  
data.drop(2, axis=0, inplace=True)  
print(data)
```

# Pandas Read JSON

- Pandas kan læse data fra en JSON-fil ved hjælp af `pd.read_json()` metoden.
- Data fra en JSON-fil kan gemmes i en Pandas DataFrame.
- En JSON-fil er en tekstfil, hvor data er gemt i JSON-format.

# Læsning af en JSON-fil

```
import pandas as pd  
data = pd.read_json('data.json')  
print(data)
```

# Appending Series til en JSON-fil

```
import pandas as pd
data = pd.read_json('data.json')
humidity = pd.Series([50, 51, 52, 53, 54])
data['humidity'] = humidity
data.to_json('data.json')
```

# filtering af data

Filtering af data i en Pandas DataFrame kan gøres ved hjælp af [] operator.

```
import pandas as pd  
data = pd.read_json('data.json')  
print(data.columns)
```

Output:

```
>>>Index(['timestamp', 'temperature'], dtype='object')
```

```
data = data[data['temperature'] > 22]  
data.to_json('dataFiltreret.json')
```

# Pandas Analyze Data

- Pandas kan bruges til at analysere data i en Pandas DataFrame.
- Data kan analyseres ved hjælp af forskellige metoder, såsom `mean()`, `median()`, `mode()`, `std()`, `min()`, `max()` og `describe()`.
- Disse metoder kan bruges til at finde gennemsnit, median, mode, standardafvigelse, minimum, maksimum og beskrivende statistikker for data.

# Eksempel

```
import pandas as pd
data = pd.read_csv('data.csv')
print(data['temperature'].mean())
print(data['temperature'].median())
print(data['temperature'].mode())
print(data['temperature'].std())
print(data['temperature'].min())
print(data['temperature'].max())
print(data['temperature'].describe())
```

# Statistik i Pandas

- `mean()` - Beregner gennemsnittet af data.
- `median()` - Beregner medianen af data.
- `mode()` - Beregner mode af data.
- `std()` - Beregner standardafvigelsen af data.
- `min()` - Finder minimumsværdien af data.
- `max()` - Finder maksimumsværdien af data.
- `describe()` - Giver en oversigt over data.
- `count()` - Tæller antallet af elementer i data.
- `sum()` - Beregner summen af data.
- `cumsum()` - Beregner den kumulative sum af data.

## mean

- `mean()` metoden beregner gennemsnittet af data.
- Gennemsnittet er summen af data divideret med antallet af elementer i data.
- Gennemsnittet er en måde at finde den centrale tendens i data.

# Eksempel

Formel for beregning af gennemsnit:

$$\text{mean} = \frac{\sum_{i=0}^n x_i}{n} \quad (1)$$

$$\text{mean} = \frac{20 + 21 + 22 + 23 + 24}{5} = \frac{110}{5} = 22.0$$

```
import pandas as pd
temperature = pd.read_csv('data.csv')
print(temperature['temperature'].mean())
```

22.0

# median

- `median()` metoden beregner medianen af data.
- Medianen er det midterste element i data, når data er sorteret i stigende rækkefølge.
- Medianen er en måde at finde den centrale tendens i data.

# Eksempel

Formel for beregning af median:

$$\text{median} = \begin{cases} x_{\left(\frac{n}{2}\right)} & \text{hvis } n \text{ er lige} \\ \frac{x_{\left(\frac{n}{2}\right)} + x_{\left(\frac{n}{2}+1\right)}}{2} & \text{hvis } n \text{ er ulige} \end{cases} \quad (2)$$

temperature = [20, 21, 22, 23]

$$\text{median} = \frac{21 + 22}{2} = \frac{43}{2} = 21.5$$

## Eksempel

```
import pandas as pd
temperature = pd.read_csv('data.csv')
print(temperature['temperature'].median())
```

22.0

# mode

- `mode()` metoden beregner mode af data.
- Mode er det element, der forekommer hyppigst i data.
- Mode er en måde at finde den centrale tendens i data.

## Eksempel

```
import pandas as pd
temperature = pd.DataFrame([20, 21, 22, 23, 24, 24,
                            24])
print(temperature.mode())
```

```
0    24
dtype: int64
```

- std() metoden beregner standardafvigelsen af data.
- Standardafvigelsen er en måling af spredningen af data.
- Standardafvigelsen er en måde at finde ud af, hvor meget data varierer fra gennemsnittet.

# Eksempel

Formel for beregning af varians

$$\sigma^2 = \text{variance} = \frac{\sum_{i=0}^n (x_i - \text{mean})^2}{n} \quad (3)$$

Formel for beregning af standardafvigelse

$$\text{std} = \sigma = \sqrt{\text{variance}} = \sqrt{\sigma^2} \quad (4)$$

$$\text{variance} = \frac{(20 - 22)^2 + (21 - 22)^2 + (22 - 22)^2 + (23 - 22)^2 + (24 - 22)^2}{5}$$

$$\text{std} = \sqrt{2.0} = 1.41$$

## cumsum

- `cumsum()` metoden beregner den kumulative sum af data.
- Den kumulative sum er summen af data op til det aktuelle element.
- Den kumulative sum er en måde at finde ud af, hvordan data ændrer sig over tid.

# Eksempel

Formel for beregning af kumulativ sum

$$\text{cumsum} = \sum_{i=1}^k (x_i - W) \quad (5)$$

Referencen er valgt til  $W = 0$

$$\text{cumsum} = [20, 20+21, 20+21+22, 20+21+22+23, 20+21+22+23+24] =$$

$$[20, 41, 63, 86, 110]$$

Tak for i dag!

# Spørgsmål?