

Python

Dag 3

Mees Meester



Vandaag

- Drukke dag!
- Businesscase
 - Toets

Introductie

- Vragen tot nu toe?



Name == main

- Zie je veel in code van andere
- Heeft als doel om te controleren of het om het hoofdprogramma gaat
- Vanuit ander programma: werkt niet

helper.py

```
print('Het bestand wordt alleen maar gebruikt')  
if __name__ == "__main__":  
    print('Het bestand zelf wordt aangeroepen')
```

```
[~] $ python helper.py  
Het bestand wordt alleen maar gebruikt  
Het bestand zelf wordt aangeroepen
```

main.py

```
import helper
```

```
[~] $ python main.py  
Het bestand wordt alleen maar gebruikt
```

Credits voorbeeld: <https://saturncloud.io/blog/what-does-if-name-main-do/>

Overzicht opleidingstraject

Lesdag 3

09:30-10:15	Blok 1	Datavisualisatie - Matplotlib
10:15-11:45	Blok 2	Classes
11:45-12:30		Pauze
12:30-13:15		Businesscase - Deel 1
13:15-14:15	Blok 3	Python & Data
14:15-15:30		Businesscase - Deel 2
15:30-16:00	Blok 4	Andere programmeertalen
16:00-16:30		Afsluiting



Blok 1 - Datavisualisatie - Matplotlib Theorie

09:30-10:15	Blok 1	Datavisualisatie - Matplotlib
10:15-11:45	Blok 2	Classes
11:45-12:30		Pauze
12:30-13:15		Businesscase - Deel 1
13:15-14:15	Blok 3	Python & Data
14:15-15:30		Businesscase - Deel 2
15:30-16:00	Blok 4	Andere programmeertalen
16:00-16:30		Afsluiting



Datavisualisatie & data storytelling

- Datavisualisatie
 - Data weergeven in beeld
- Data storytelling
 - Data weergeven in tekstje
 - Denk aan Spotify's jaarverslag

Soorten datavisualisatie

- Top 5

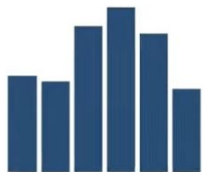
1. Staafdiagram
2. Histogram
3. ~~Taart diagram~~
4. Spreidingsdiagram
5. Lijndiagram

Bar Chart
Histogram
~~Pie Chart~~
Scatterplot
Line chart

1



2



3



4



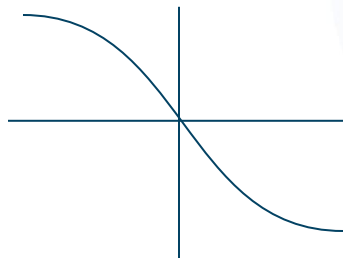
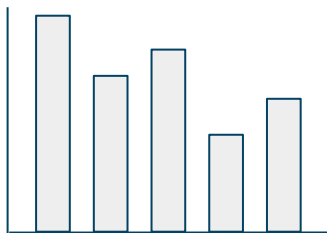
5



Credits: https://youtu.be/o7F-tbBl_hA

Discrete & Continuous

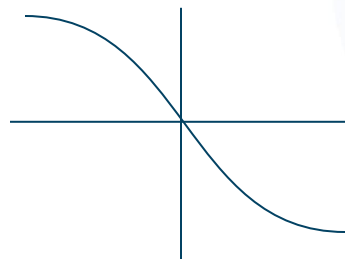
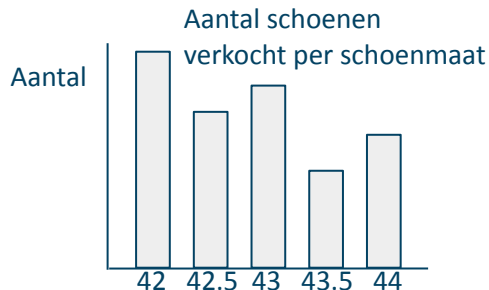
- Discrete
 - Finite - bepaalde waarden
 - Niets tussen vaste waardes
 - Dingen die je telt
 - Niet verbonden
- Voorbeelden
 - Aantal boeken
 - Aantal letters in berichtjes
 - Aantal sinaasappels
 - Dobbelsteen
- Continuous
 - Infinite - alle waarden
 - Elke waarde binnen een range
 - Dingen die je meet
 - Wel verbonden
- Voorbeelden
 - Temperatuur
 - Afstand
 - Lengte
 - Exacte tijd dat de bus te laat is



Schoenmaten? 42, 42.5, 43

Discrete & Continuous

- Discrete
 - Finite - bepaalde waarden
 - Niets tussen vaste waardes
 - Dingen die je telt
 - Niet verbonden
- Voorbeelden
 - Aantal boeken
 - Aantal letters in berichtjes
 - Aantal sinaasappels
 - Dobbelsteen
- Continuous
 - Infinite - alle waarden
 - Elke waarde binnen een range
 - Dingen die je meet
 - Wel verbonden
- Voorbeelden
 - Temperatuur
 - Afstand
 - Lengte
 - Exacte tijd dat de bus te laat is



Schoenmaten? 42, 42.5, 43
Discrete

Wanneer gebruik je welke?

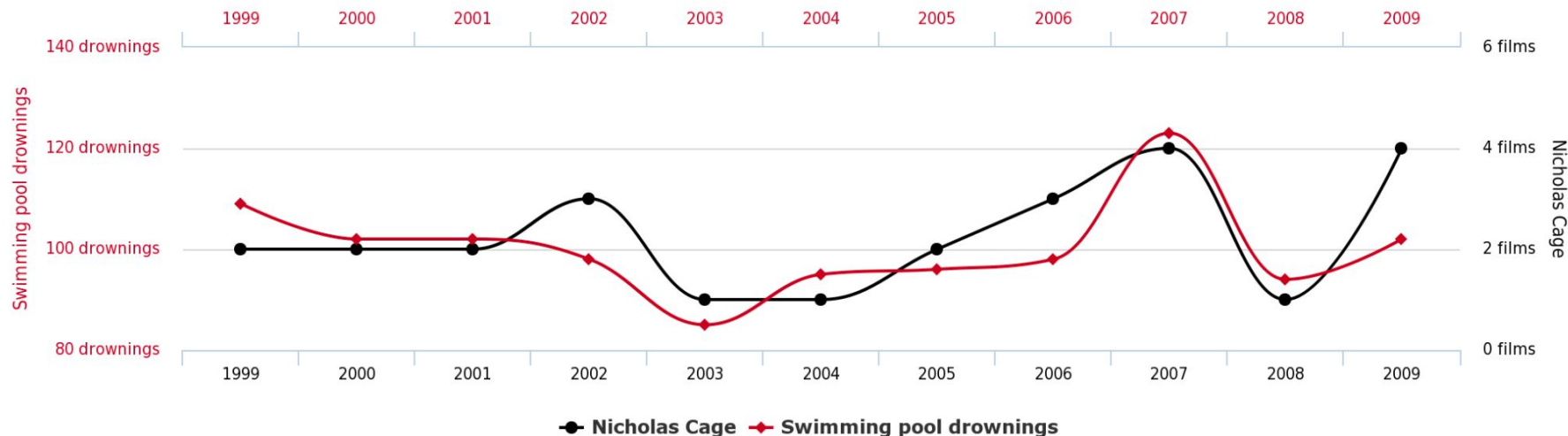


De kracht van visualisatie

Number of people who drowned by falling into a pool

correlates with

Films Nicolas Cage appeared in



Credits afbeelding: <https://www.tylervigen.com/>

Datavisualisatie in Python

- **Matplotlib**
- Plotly
- Seaborn

matplotlib

plotly

seaborn

Matplotlib

- Syntax
 - `import matplotlib.pyplot as plt`
 - `plt.plot(x, y, opties)`
 - `plt.show()`
- Extreem veel opties en mogelijkheden

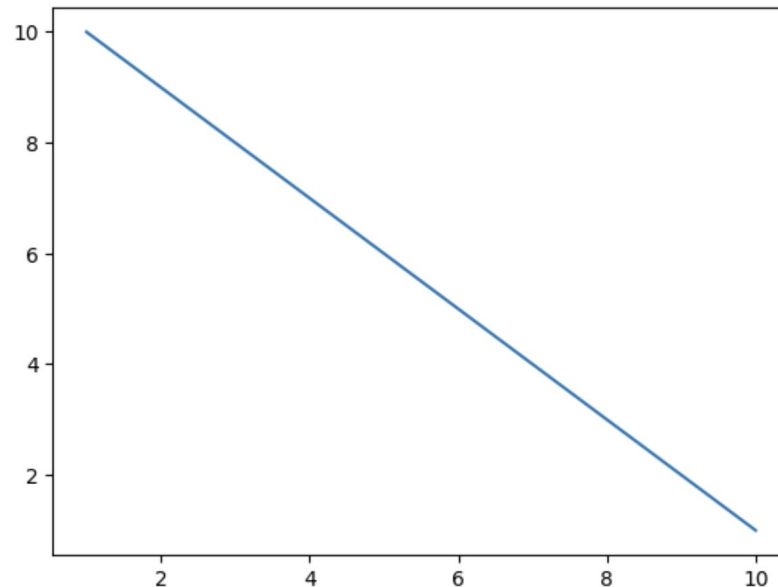
“bring in”
↓
import matplotlib.pyplot as plt

A **module** within the matplotlib library
↓

A **library** in python for data visualization
↑

An **alias**, so you don't have to write the library name on every call
↑

```
import matplotlib.pyplot as plt  
  
x = [1, 10]  
y = [10, 1]  
plt.plot(x, y)  
plt.savefig('plot.png')  
plt.show()
```



Naslagwerk datavisualisatie

- https://youtu.be/o7F-tbBI_hA
- <https://youtu.be/UO98IJQ3QGI>
- datavizcatalogue.com
- w3schools.com - matplotlib

Blok 1 - Datavisualisatie - Matplotlib Demonstratie

09:30-10:15	Blok 1	Datavisualisatie - Matplotlib
10:15-11:45	Blok 2	Classes
11:45-12:30		Pauze
12:30-13:15		Businesscase - Deel 1
13:15-14:15	Blok 3	Python & Data
14:15-15:30		Businesscase - Deel 2
15:30-16:00	Blok 4	Andere programmeertalen
16:00-16:30		Afsluiting



Blok 1 - Datavisualisatie - Matplotlib Praktijk

09:30-10:15	Blok 1	Datavisualisatie - Matplotlib
10:15-11:45	Blok 2	Classes
11:45-12:30		Pauze
12:30-13:15		Businesscase - Deel 1
13:15-14:15	Blok 3	Python & Data
14:15-15:30		Businesscase - Deel 2
15:30-16:00	Blok 4	Andere programmeertalen
16:00-16:30		Afsluiting



Praktijk

- Laat nu zelf een grafiekje zien in Python

```
import matplotlib.pyplot as plt
```

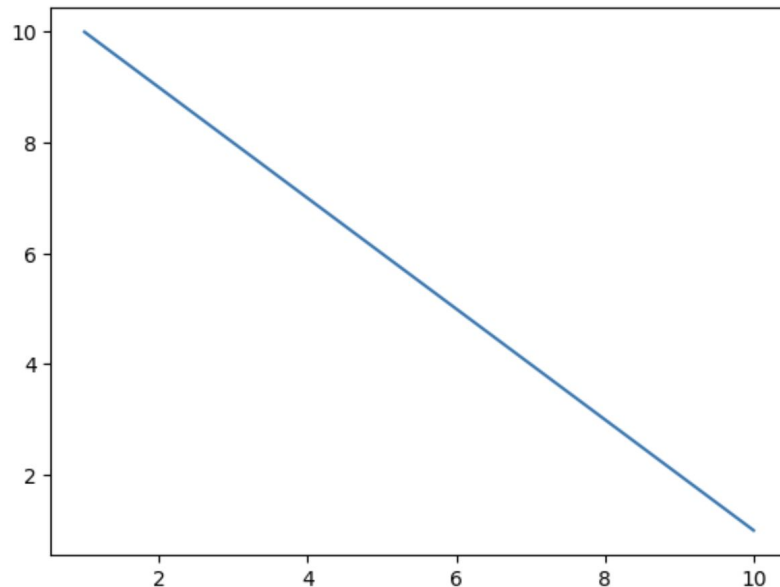
```
x = [1, 10]
```

```
y = [10, 1]
```

```
plt.plot(x, y)
```

```
plt.savefig('plot.png')
```

```
plt.show()
```



Blok 2 - Classes

Theorie & demonstratie

09:30-10:15	Blok 1	Datavisualisatie - Matplotlib
10:15-11:45	Blok 2	Classes
11:45-12:30		Pauze
12:30-13:15		Businesscase - Deel 1
13:15-14:15	Blok 3	Python & Data
14:15-15:30		Businesscase - Deel 2
15:30-16:00	Blok 4	Andere programmeertalen
16:00-16:30		Afsluiting



Class and object

- 'Python is an object oriented programming language. Almost everything in Python is an object, with its properties and methods.' ~ W3Schools
- OOP (Object Oriented Programming)
- Je kan zelf nieuwe objects maken met classes
- help() geeft je meer informatie
- dir() laat je de methods, properties en andere attributen van een object zien

```
help(str)
```

Help on class str in module builtins:

```
class str(object)
| str(object='') -> str
| str(bytes_or_buffer[, encoding[, errors]]) -> str
|
| Create a new string object from the given object. If encoding or
| errors is specified, then the object must expose a data buffer
| that will be decoded using the given encoding and error handler.
| Otherwise, returns the result of object.__str__() (if defined)
| or repr(object).
| encoding defaults to sys.getdefaultencoding().
| errors defaults to 'strict'.
```

```
dir(str)
```

```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
```

Class syntax

- Algemene syntax Class
class A():
 pass
- Methods
class A():
 def test(self):
 return 'test'
- Instances
 - a = A()

Class syntax

- Algemene syntax Class
- Methods
- Instances

```
class TestC:  
    def testM(self):  
        return 'Method'
```

```
a = TestC()
```

```
b = TestC()
```

```
print(a)
```

```
print(b)
```

```
print(TestC.testM(a))
```

```
print(b.testM())
```

```
<__main__.TestC object at 0x1097610>
```

```
<__main__.TestC object at 0x2cc17b0>
```

```
Method
```

```
Method
```

Self

- Je kunt het zien als een dict dat alle instance variables (properties/methods) van die Class bevat
- De meeste methoden gebruiken eigenschappen en dus self

```
class Person:
    def __init__(self):
        self.constant = 4
    def func(self):
        return 80
    def other_func(self, value):
        return self.func() + value + self.constant
```

Self

- Je kunt het zien als een dict dat alle instance variables (properties/methods) van die Class bevat
- De meeste methoden gebruiken eigenschappen en dus self
- Anders: @staticmethod

```
class Calculator:
    @staticmethod
    def add(a, b):
        return a + b

    @staticmethod
    def subtract(a, b):
        return a - b

print(Calculator.add(5, 3))
print(Calculator.subtract(5, 3))
```

8

2

Property

- Geen haakjes
 - @property

```
class A:
    def no_prop(self):
        return 5

    @property
    def prop(self):
        return 5

a = A()
a.no_prop() + a.prop
```

10

Dunder/magic methods

- Alle speciale methods
 - Twee lage streepjes aan beide kanten
 - Dunder: Double Under (Underscores)
- `__init__`
 - Wordt altijd aangeroepen bij het initialiseren van een class/instance
- `__str__`
 - `str(object)`
 - `print(object)`
- `__len__`
 - `len(object)`

Dunder/magic methods

- Alle speciale methods
 - Twee lage streepjes aan beide kanten
 - Dunder: Double Under (Underscores)

```
class Eten():
    def __init__(self, naam):
        self.naam = naam
        self.houdbaarheid = 'wel'

    def over_datum(self):
        self.houdbaarheid = 'niet'

    def __str__(self):
        return f'Dit eten is {self.naam} en het product is {self.houdbaarheid} houdbaar'

kaas = Eten('Kaas')
print(kaas)
```

Dit eten is Kaas en het product is wel houdbaar

Dunder/magic methods

- Alle speciale methods
 - Twee lage streepjes aan beide kanten
 - Dunder: Double Under (Underscores)

```
class Eten():
    def __init__(self, naam):
        self.naam = naam
        self.houdbaarheid = 'wel'

    def over_datum(self):
        self.houdbaarheid = 'niet'

    def __str__(self):
        return f'Dit eten is {self.naam} en het product is {self.houdbaarheid} houdbaar'

kaas = Eten('Kaas')
kaas.over_datum()
print(kaas)
```

Dit eten is Kaas en het product is niet houdbaar

Python Inheritance

- Classes overnemen

```
class Eten():  
    def __init__(self, naam):  
        self.naam = naam  
        self.houdbaarheid = 'wel'  
  
    def over_datum(self):  
        self.houdbaarheid = 'niet'  
  
    def __str__(self):  
        return f'Dit eten is {self.naam} en het product'  
  
kaas = Eten('Kaas')  
kaas.over_datum()  
print(kaas)
```

Dit eten is Kaas en het product is niet houdbaar

```
class Brood(Eten):  
    pass
```

```
kaas = Brood('Brood')  
kaas.over_datum()  
print(kaas)
```

Dit eten is Brood en het product is niet houdbaar

Python Inheritance

- Classes overnemen
- Methods overschrijven

```
class Eten():
    def __init__(self, naam):
        self.naam = naam
        self.houdbaarheid = 'wel'

    def over_datum(self):
        self.houdbaarheid = 'niet'

    def __str__(self):
        return f'Dit eten is {self.naam} en het product'

kaas = Eten('Kaas')
kaas.over_datum()
print(kaas)
```

Dit eten is Kaas en het product is niet houdbaar

```
class Brood(Eten):
    def __init__(self):
        self.naam = 'Brood'
        self.houdbaarheid = 'niet'
```

```
kaas = Brood()
kaas.over_datum()
print(kaas)
```

Dit eten is Brood en het product is niet houdbaar

Python Inheritance

- Classes overnemen
- Methods overschrijven
- `super()`

```
class Eten():  
    def __init__(self, naam):  
        self.naam = naam  
        self.houdbaarheid = 'wel'  
  
    def over_datum(self):  
        self.houdbaarheid = 'niet'  
  
    def __str__(self):  
        return f'Dit eten is {self.naam} en het product'  
  
kaas = Eten('Kaas')  
kaas.over_datum()  
print(kaas)
```

Dit eten is Kaas en het product is niet houdbaar

```
class Brood(Eten):  
    def __init__(self):  
        super().__init__('brood')
```

```
kaas = Brood()  
kaas.over_datum()  
print(kaas)
```

Dit eten is brood en het product is niet houdbaar

Naslagwerk

- [W3schools class](#)



Blok 2 - Classes

Praktijk

09:30-10:15	Blok 1	Datavisualisatie - Matplotlib
10:15-11:45	Blok 2	Classes
11:45-12:30		Pauze
12:30-13:15		Businesscase - Deel 1
13:15-14:15	Blok 3	Python & Data
14:15-15:30		Businesscase - Deel 2
15:30-16:00	Blok 4	Andere programmeertalen
16:00-16:30		Afsluiting



Blok 3 - Data & Python

Praktijk

09:30-10:15	Blok 1	Datavisualisatie - Matplotlib
10:15-11:45	Blok 2	Classes
11:45-12:30		Pauze
12:30-13:15		Businesscase - Deel 1
13:15-14:15	Blok 3	Python & Data
14:15-15:30		Businesscase - Deel 2
15:30-16:00	Blok 4	Andere programmeertalen
16:00-16:30		Afsluiting



Pandas

- import pandas as pd
- Denk aan Excel qua weergave dataset
- Denk aan dictionary om er mee te werken
 - dataset['Fruits'] = column
 - dataset.fruits = column
 - dataset.get('Fruits') = ~~column~~

	Fruits	Supply
0	Apple	30
1	Mango	15
2	Grapes	10
3	Strawberry	25
4	Oranges	20

Dataframe maken

- DataFrame
 - `df = pd.DataFrame()`
 - 2D
 - Series
- `s = pd.Series()`
 - Column
 - 1D

```
### Creating a dataframe
import pandas as pd

dataset = {'Fruits': ["Apple", "Mango", "Grapes", "Strawberry",
"Oranges"], 'Supply': [30, 15, 10, 25, 20]}

# Create DataFrame
df = pd.DataFrame(dataset)

# Print the output.
df
```

	Fruits	Supply
0	Apple	30
1	Mango	15
2	Grapes	10
3	Strawberry	25
4	Oranges	20

Credits voorbeeld: <https://towardsdatascience.com/14-pandas-operations-that-every-data-scientist-must-know-cc326dc4e6ee>

Dataframe analyseren

- `df.shape`
 - Het aantal rijen en kolommen
 - *Vorm* van je data
- `df.describe()`
 - Samenvatting van beschrijvende statistieken
- `df.info()`
 - Informatie over de gegevenstypes en ontbrekende waarden
- `df.plot()`
 - Veel soorten visualisaties van je DF
 - lijngrafieken
 - histogrammen
 - taartdiagrammen

`pandas.DataFrame.plot`

`pandas.DataFrame.plot.area`

`pandas.DataFrame.plot.bar`

`pandas.DataFrame.plot.barh`

`pandas.DataFrame.plot.box`

`pandas.DataFrame.plot.density`

`pandas.DataFrame.plot.hexbin`

`pandas.DataFrame.plot.hist`

`pandas.DataFrame.plot.kde`

`pandas.DataFrame.plot.line`

`pandas.DataFrame.plot.pie`

`pandas.DataFrame.plot.scatter`

CSV & DataFrames

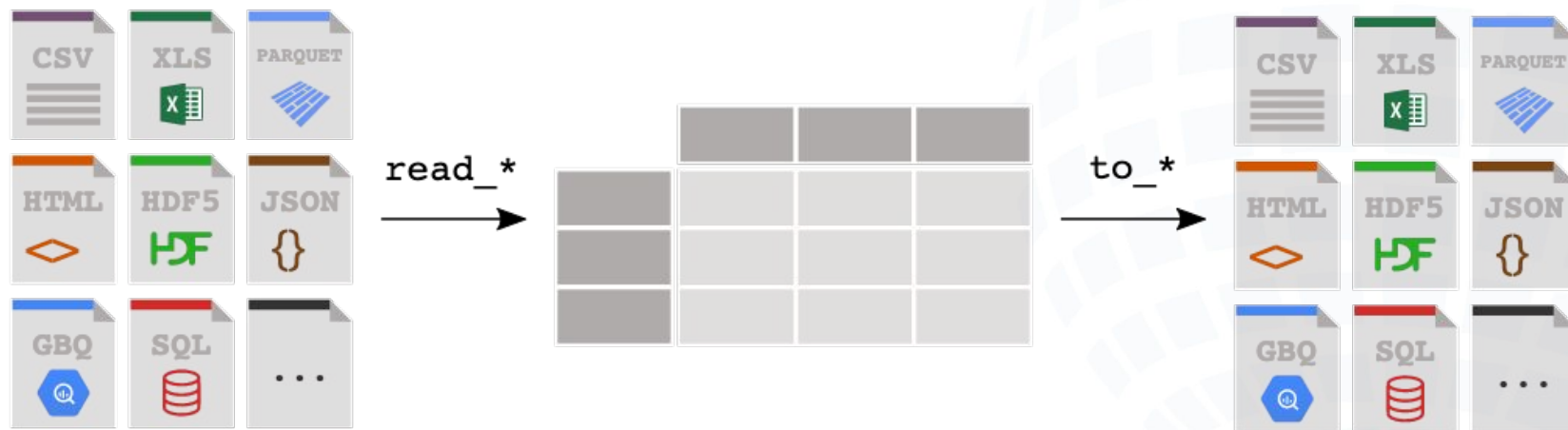
- `pd.read_csv()`
 - Dit is het primaire commando om gegevens in een Pandas DataFrame te lezen vanuit een CSV-bestand.
- `df.to_csv()`
 - Dit commando slaat je DataFrame op in een CSV-bestand.
- `code.py/code.ipynb`
 - Datasets/dataset1.csv
- `df = pd.read_csv('Datasets/dataset1.csv')`
- `df = pd.read_csv(URL NAAR DATASET)`

```
Data_Science
├── Datasets
│   ├── dataset1.csv
│   ├── dataset1.json
│   ├── dataset2.csv
│   ├── dataset3.csv
│   └── dataset4.csv
├── code.ipynb
└── code.py
```

Datasets & Dataframes

- `df = pd.read_json('Datasets/dataset1.json')`

```
Data_Science
├── Datasets
│   ├── dataset1.csv
│   ├── dataset1.json
│   ├── dataset2.csv
│   ├── dataset3.csv
│   └── dataset4.csv
├── code.ipynb
└── code.py
```



Sort

- Syntax
 - `df.sort_value(by = [column])`

```
import pandas as pd

dataset = {'Fruits': ["Apple", "Mango", "Grapes", "Strawberry",  
"Oranges"], 'Supply': [30, 15, 10, 25, 20]}
df = pd.DataFrame(dataset)

df.sort_values(by = ["Supply"])
```

	Fruits	Supply
2	Grapes	10
1	Mango	15
4	Oranges	20
3	Strawberry	25
0	Apple	30

Credits voorbeeld: <https://towardsdatascience.com/14-pandas-operations-that-every-data-scientist-must-know-cc326dc4e6ee>

Group by

- Syntax
 - `df.groupby([column name])`
 - mean, first, max, min

```
df = pd.DataFrame({'Fruit': ['Appel', 'Appel',  
                             'Wortel', 'Wortel', 'Wortel'],  
                  'Prijs': [1.21, .60, .20, .30, .25]})  
df
```

	Fruit	Prijs
0	Appel	1.21
1	Appel	0.60
2	Wortel	0.20
3	Wortel	0.30
4	Wortel	0.25

```
group = df.groupby(['Fruit'])  
group.first()
```

	Prijs
Fruit	
Appel	1.21
Wortel	0.20

Group by

- Syntax
 - `df.groupby([column name])`
 - mean, first, max, min
 - `group.get_group(group name)`

```
group = df.groupby(['Fruit'])  
group.first()
```

Prijzen	
Fruit	
Appel	1.21
Wortel	0.20

```
group = df.groupby(['Fruit'])  
group.get_group('Appel')
```

	Fruit	Prijzen
0	Appel	1.21
1	Appel	0.60

Loc

- Syntax
 - `df.loc[row, columns]`

```
df = pd.DataFrame({'Fruit': ['Appel', 'Appel',  
                             'Wortel', 'Wortel', 'Wortel'],  
                  'Prijs': [1.21, .60, .20, .30, .25]})  
df
```

	Fruit	Prijs
0	Appel	1.21
1	Appel	0.60
2	Wortel	0.20
3	Wortel	0.30
4	Wortel	0.25

```
result = df.loc[1:3, 'Prijs']  
print(type(result))  
print(list(result))  
  
<class 'pandas.core.series.Series'>  
[0.6, 0.2, 0.3]
```

Fillna/Dropna

- Syntax Fillna
 - `df.fillna(value=None, method=None, axis=None, inplace=False, limit=None, ...)`
 - `df.gewicht = df.gewicht.fillna(df.gewicht.mean())`
- Syntax Dropna
 - `df.dropna()`
 - `df.dropna(axis='columns')`

Pandas

- `import pandas as pd`
- Denk aan dictionary om er mee te werken
 - Goed om vele soorten data op te slaan
- Krijg direct aantal columns en rijen met `shape`

	Fruits	Supply
0	Apple	30
1	Mango	15
2	Grapes	10
3	Strawberry	25
4	Oranges	20

Numpy analytics

- import numpy as np
- Statistieken
 - np.mean()
 - np.std()
 - np.sum()

```
[1]: import numpy as np
```

```
[2]: lijst = [1,2,3,4]  
print(np.mean(lijst))  
  
2.5
```

```
[3]: array = np.array(lijst)  
print(array.mean())  
  
2.5
```

Overeenkomsten met Pandas

- Krijg direct aantal kolommen en rijen met shape
 - `array.shape()`
 - `df.shape()`
- Transpose gemakkelijk
 - Rijen en kolommen omdraaien
 - `array.T`
 - `df.T`
- Statistieken
 - `array.mean()`
 - `df.mean(numeric_only=True)`

```
df.mean()
```

```
column1    0.036381  
column2   -0.150868  
dtype: float64
```

```
array.mean()
```

```
-0.04421953494140423
```

Data naslagwerk

- Pandas
 - [pandas.pydata.org - Pandas Cheat Sheet](https://pandas.pydata.org/docs/10min/cheat_sheet.html)
 - [pandas.pydata.org - pandas 2.0.2 documentation](https://pandas.pydata.org/docs/10min/cheat_sheet.html)
 - [w3schools.com - pandas/](https://www.w3schools.com/pandas/)
- Numpy
 - [analyticsvidhya.com - Numpy tutorial](https://analyticsvidhya.com/numpy-tutorial/)
 - [numpy.org - the absolute basics for beginners](https://numpy.org/doc/stable/user/absolute_beginners.html)
- Alles
 - <https://learn.microsoft.com/en-us/training/modules/explore-analyze-data-with-python/1-introduction>

Blok 3 - Data & Python

Praktijk

09:30-10:15	Blok 1	Datavisualisatie - Matplotlib
10:15-11:45	Blok 2	Classes
11:45-12:30		Pauze
12:30-13:15		Businesscase - Deel 1
13:15-14:15	Blok 3	Python & Data
14:15-15:30		Businesscase - Deel 2
15:30-16:00	Blok 4	Andere programmeertalen
16:00-16:30		Afsluiting



Blok 4 - Andere programmeertalen

Theorie

09:30-10:15	Blok 1	Datavisualisatie - Matplotlib
10:15-11:45	Blok 2	Classes
11:45-12:30		Pauze
12:30-13:15		Businesscase - Deel 1
13:15-14:15	Blok 3	Python & Data
14:15-15:30		Businesscase - Deel 2
15:30-16:00	Blok 4	Andere programmeertalen
16:00-16:30		Afsluiting



Python vs andere talen

- C#/C++
- JavaScript
- PHP

Python vs C#

- Beide zijn object-oriented (class-based)
- C# is sneller
- Python is makkelijker te leren
- Beide zijn open source

```
print('hello world')
```

```
Console.WriteLine("Hello World!");
```

Python vs Javascript

- Javascript
 - Is niet pure object-based
 - Is beter voor real-time applicaties
 - Vaak in combinatie met HTML
 - Is weakly-typed
 - 3 + '3' wordt '33' (string)

```
print('hello world')
```

```
console.log('Hello World');
```

Python vs PHP

- PHP
 - Vooral voor web
 - Is weakly-typed

```
print('hello world')
```

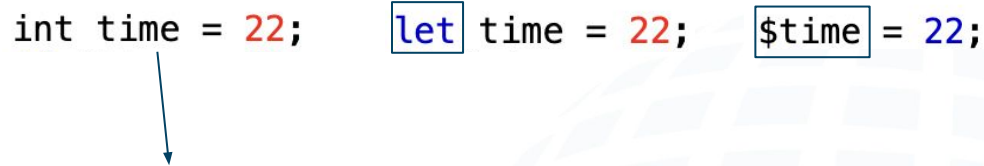
```
echo "Hello world";
```

If statements

- C#

```
time = 22
if time < 10:
    print("Good morning.")
elif time < 20:
    print("Good day.")
else:
    print("Good evening.")
```

```
int time = 22;    let time = 22;    $time = 22;
```



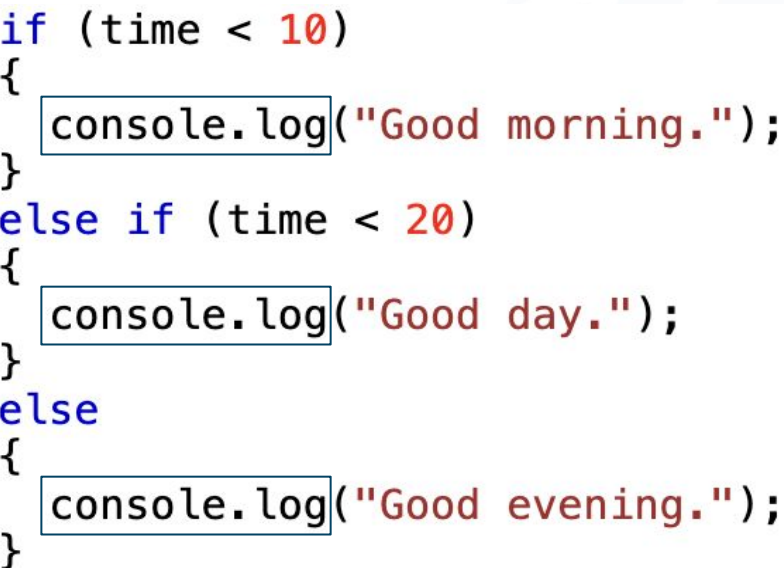
```
if (time < 10)
{
    Console.WriteLine("Good morning.");
}
else if (time < 20)
{
    Console.WriteLine("Good day.");
}
else
{
    Console.WriteLine("Good evening.");
}
```

If statements

- Javascript

```
time = 22
if time < 10:
    print("Good morning.")
elif time < 20:
    print("Good day.")
else:
    print("Good evening.")
```

```
int time = 22;    let time = 22;    $time = 22;
```



The diagram illustrates the syntax for variable declaration and conditional logic in three programming languages. At the top, three lines show variable declarations: `int time = 22;` (C/C++), `let time = 22;` (JavaScript), and `$time = 22;` (PHP). Below these, an if statement is shown for each language. A blue arrow points from the `let` declaration to the `if` statement in the JavaScript example. In the JavaScript example, the `if` statement is written as `if (time < 10) { console.log("Good morning."); } else if (time < 20) { console.log("Good day."); } else { console.log("Good evening."); }`. The `if`, `else if`, and `else` keywords are highlighted in blue, while the `console.log` function calls are highlighted in a blue box. The `if` statement in the C/C++ example is written as `if (time < 10) { console.log("Good morning."); } else if (time < 20) { console.log("Good day."); } else { console.log("Good evening."); }`. The `if`, `else if`, and `else` keywords are highlighted in blue, while the `console.log` function calls are highlighted in a blue box. The `if` statement in the PHP example is written as `if ($time < 10) { echo "Good morning."; } elseif ($time < 20) { echo "Good day."; } else { echo "Good evening."; }`. The `if`, `elseif`, and `else` keywords are highlighted in blue, while the `echo` function calls are highlighted in a blue box.

```
if (time < 10)
{
    console.log("Good morning.");
}
else if (time < 20)
{
    console.log("Good day.");
}
else
{
    console.log("Good evening.");
}
```



If statements

- PHP

```
time = 22
if time < 10:
    print("Good morning.")
elif time < 20:
    print("Good day.")
else:
    print("Good evening.")
```

```
int time = 22;    let time = 22;    $time = 22;
```

```
if ($time < 10)
{
    echo "Good morning.";
}
else if ($time < 20)
{
    echo "Good day.";
}
else
{
    echo "Good evening.";
}
```



For loops


- For loop
 1. Maak een nieuwe variabele
 2. Conditie voor wanneer de loop door blijft gaan
 3. Maak de variabele steeds eentje hoger
- C#

```
for i in range(5):  
    print(i)
```

`int time = 22;` `let time = 22;` `$time = 22;`

1 2 3

```
for (int i = 0; i < 5; i++)  
{  
    Console.WriteLine(i);  
}
```



For loops

- For loop
 1. Maak een nieuwe variabele
 2. Conditie voor wanneer de loop door blijft gaan
 3. Maak de variabele steeds eentje hoger
- Javascript

```
for i in range(5):  
    print(i)
```

`int time = 22;` `let time = 22;` `$time = 22;`

1 2 3

```
for (let i = 0; i < 5; i++)  
{  
    console.log(i);  
}
```

For loops

- For loop
 1. Maak een nieuwe variabele
 2. Conditie voor wanneer de loop door blijft gaan
 3. Maak de variabele steeds eentje hoger
- PHP

```
for i in range(5):  
    print(i)
```

```
int time = 22;      let time = 22;      $time = 22;
```

```
1                      2              3  
for ($i = 0; $i < 5; $i++)  
{  
    echo "$i <br>";  
}
```

Functies

- Python vs Javascript & PHP

```
def som(a, b):  
    return a + b
```

PHP

```
function som($a, $b) {  
    return $a + $b;  
}
```

Javascript

```
function som(a, b) {  
    return a + b;  
}
```

Afsluiting

16:00-16:30

09:30-10:15	Blok 1	Datavisualisatie - Matplotlib
10:15-11:45	Blok 2	Classes
11:45-12:30		Pauze
12:30-13:15		Businesscase - Deel 1
13:15-14:15	Blok 3	Python & Data
14:15-15:30		Businesscase - Deel 2
15:30-16:00	Blok 4	Andere programmeertalen
16:00-16:30		Afsluiting



Vragen

Zijn er nog vragen?

- Stel ze gerust!



Tips

- Docstrings gebruiken
 - Comments van een functie
- Logging gebruiken in plaats van print
- Handige tools
 - Black
 - Formateert je code naar PEP8
 - Pylint
 - Laat je waarschuwingen en errors zien
 - Flake8
 - Stijltips
 - Pydocstyle
 - Stijl van docstrings



Huiswerk

- Lekker aan de slag
- Mag altijd mailen als je nog vastloopt



Administratief

- Enquête
- Presentielijst
- Foto

Class bonus

16:30-17:00

09:30-10:15	Blok 1	Datavisualisatie - Matplotlib
10:15-11:45	Blok 2	Classes
11:45-12:30		Pauze
12:30-13:15		Businesscase - Deel 1
13:15-14:15	Blok 3	Python & Data
14:15-15:30		Businesscase - Deel 2
15:30-16:00	Blok 4	Andere programmeertalen
16:00-16:30		Afsluiting



Iterable Class

- Objecten waar je overheen kan lopen/lopen
- Hebben allemaal een **iter()** method waarmee een **iterator** gereturned kan worden

```
class A:  
    def __iter__(self):  
        return iter([1,2,3])
```

```
for i in A():  
    print(i)
```

1
2
3

Iterable Class

- Objecten waar je overheen kan lopen/lopen
- Hebben allemaal een **iter()** method waarmee een **iterator** gereturned kan worden

```
class ReeksGetallen:
    def __init__(self, eind_nummer):
        print('Init functie wordt aangeroepen')
        self.eind_nummer = eind_nummer
        self.nummer = 0

    def __iter__(self):
        print('Iter functie wordt aangeroepen')
        return self

    def __next__(self):
        print('Next functie wordt aangeroepen')
        if self.nummer >= self.eind_nummer:
            raise StopIteration
        self.nummer += 1
        return self.nummer - 1
```

Iterable Class

- Objecten waar je overheen kan lopen/lopen
- Hebben allemaal een **iter()** method waarmee een **iterator** gereturned kan worden

```
class ReeksGetallen:
    def __init__(self, eind_nummer):
        print('Init functie wordt aangeroepen')
        self.eind_nummer = eind_nummer
        self.nummer = 0

    def __iter__(self):
        print('Iter functie wordt aangeroepen')
        return self

    def __next__(self):
        print('Next functie wordt aangeroepen')
        if self.nummer >= self.eind_nummer:
            raise StopIteration
        self.nummer += 1
        return self.nummer - 1
```

```
a = ReeksGetallen(3)
```

```
b = iter(a)
```

```
next(b)
```

```
b.__next__()
```

```
Init functie wordt aangeroepen
```

```
Iter functie wordt aangeroepen
```

```
Next functie wordt aangeroepen
```

```
Next functie wordt aangeroepen
```