

Python

Dag 1
Mees Meester



Introductie

- Python developer
- Hout- en metaalbewerking
- Docent/trainer Global Training
- Meer dan decennium ervaring met lesgeven
- AI-Engineer
 - Computer Vision
- Kunstmatige Intelligentie
 - Universiteit van Amsterdam



Voorstellen

- Wat is je naam?
- Wat brengt je naar programmeren?
- Wat is je leerdoel?

Voorstellen

- Interactieve les
 - Veel zelf doen
 - Veel vragen -> grootste voordeel
- Blokken
 - Uitleg
 - Demonstratie
 - Zelf programmeren
- Heel veel nieuwe termen
 - Python of Frans?

Overzicht opleidingstraject

Lesdag 1

09:45-10:15	Blok 1	Inleiding
10:20-11:00	Blok 2	Programmeren in Python
11:00-12:45	Blok 3a	Data types - Individueel - Theorie
12:45-13:15		Pauze
13:15-15:15	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Functies & Control flow 1
16:15-16:30		Afsluiting



* Tijden dienen echter als richtlijn

Blok 1 - Inleiding

Theorie

09:45-10:15	Blok 1	Inleiding
10:20-11:00	Blok 2	Programmeren in Python
11:00-12:00	Blok 3a	Data types - Individueel - Theorie
12:00-12:45		Pauze
13:15-15:15	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Funcities & Control flow 1
16:15-16:30		Afsluiting



Python

- Kunstmatige Intelligentie
 - Automatisering
 - NLP
 - Computer Vision
 - Reinforcement Learning
 - Deep learning
- Big data analyse
- Web development

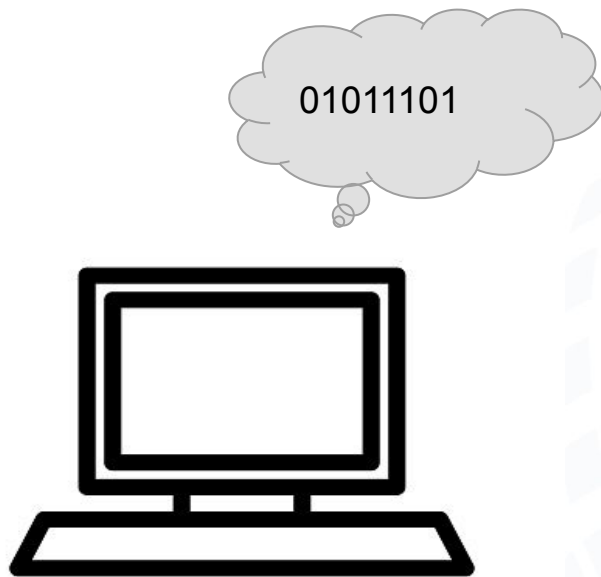
Python

- Ontstaan in begin jaren '90
- Het Britse Monty Python
- Amsterdam
- Syntax



The logo for the Monty Python film 'Monty Python and the Holy Grail'. It features the title in a white, stylized font on a dark blue rectangular background. 'Monty Python' is in a large, flowing script, while 'AND THE HOLY GRAIL' is in a smaller, blocky, all-caps font below it. The background of the slide has a faint, light blue grid pattern.

Programmeren

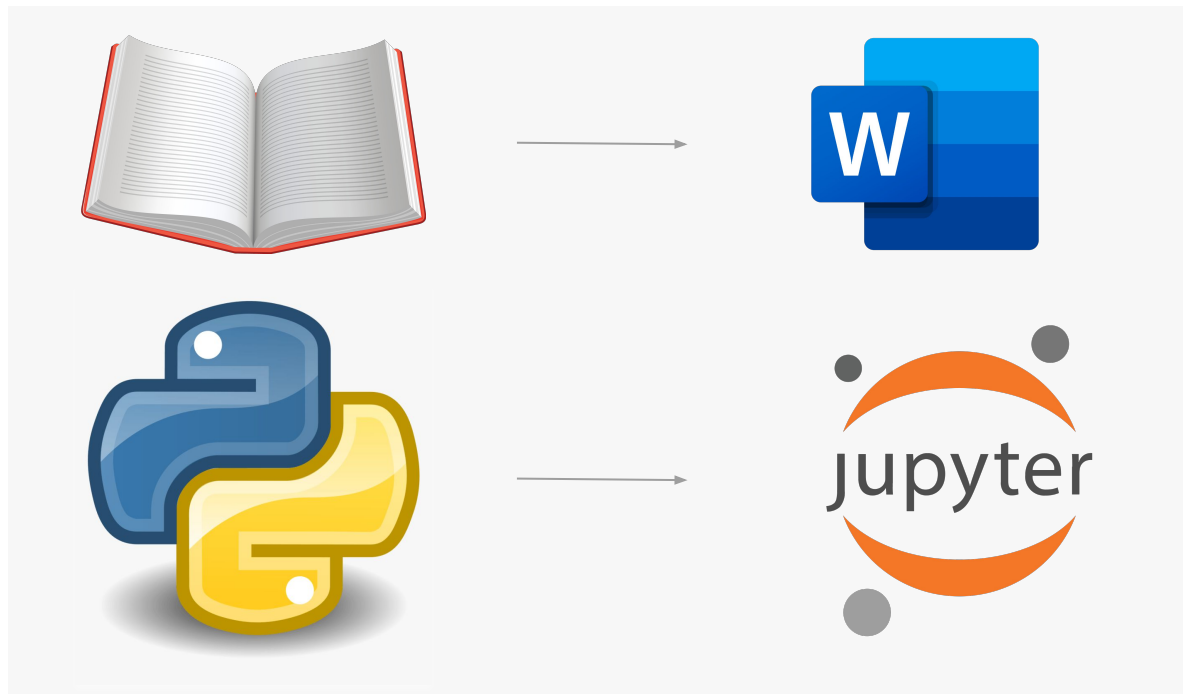
- Praten met een computer



Programmeren - Code schrijven?

- Programma's installeren
- Debuggen
- Google gebruiken
- Beter programmer = beter Googlen
 -  Hoe krijg ik Python zover om het antwoord 5+3 te laten uitrekenen?
 -  addition.py

Jupyter Lite



Blok 1 - Inleiding

Demonstratie - <https://shorturl.com/python>

09:45-10:15	Blok 1	Inleiding
10:20-11:00	Blok 2	Programmeren in Python
11:00-12:00	Blok 3a	Data types - Individueel - Theorie
12:00-12:45		Pauze
13:15-15:15	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Funcities & Control flow 1
16:15-16:30		Afsluiting



Blok 1 - Inleiding

Praktijk - <https://shorturl.com/python>

09:45-10:15	Blok 1	Inleiding
10:20-11:00	Blok 2	Programmeren in Python
11:00-12:00	Blok 3a	Data types - Individueel - Theorie
12:00-12:45		Pauze
13:15-15:15	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Funcities & Control flow 1
16:15-16:30		Afsluiting



Blok 2 - Programmeren in Python

Theorie

09:45-10:15	Blok 1	Inleiding
10:20-11:00	Blok 2	Programmeren in Python
11:00-12:00	Blok 3a	Data types - Individueel - Theorie
12:00-12:45		Pauze
13:15-15:15	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Funcities & Control flow 1
16:15-16:30		Afsluiting

Variabelen

- $x = 5$
- $y = x + 5$
- Wat is y ?

Variabelen

- $x = 5$
- $y = x + 5$
- Wat is y ?

```
x = 5  
y = x + 5
```


Variabelen

- Naam zelf verzinnen
- Syntax
 - Geen spaties en geen nummer als eerste letter
 - Geen naam van een functie
 - Omschrijvende & korte naam

```
In [1]: mijn_boek = 'Er was eens...'  
        mijn_nummer = 7
```

```
In [ ]: mijn_nummer + 3
```

```
In [3]: mijn_boek + 'meer tekst'
```

Rekenen met variabelen

- Min: -
- Plus: +
- Keer: *
- Delen door: /
- Vloerdeling: //
- Tot de macht: **
- Modulo: %

$25//7=3$

$3*7=21$, dus er blijft $25-21=4$ over

$25\%7= 4$

Input

- Syntax
 - `input()`

```
user_input = input('Wat is jouw naam?')
```

Wat is jouw naam?



```
input = input('Wat is jouw naam?')
```

Output

- Syntax
 - `print()`

```
mijn_variable = 10  
print(mijn_variable)
```

10

Comment

- Syntax
 - # notitie
 - """ tekst maar gebruikt als notitie """

```
# Notitie  
''' Notitie/tekst '''  
  
' Notitie/tekst '
```

```
# Notitie  
''' Notitie/tekst '''  
print(6)
```

6

Blok 2 - Programmeren in Python

Demonstratie

09:45-10:15	Blok 1	Inleiding
10:20-11:00	Blok 2	Programmeren in Python
11:00-12:00	Blok 3a	Data types - Individueel - Theorie
12:00-12:45		Pauze
13:15-15:15	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Funcities & Control flow 1
16:15-16:30		Afsluiting

Blok 2 - Programmeren in Python

Praktijk - <https://shorturl.com/python>

09:45-10:15	Blok 1	Inleiding
10:20-11:00	Blok 2	Programmeren in Python
11:00-12:00	Blok 3a	Data types - Individueel - Theorie
12:00-12:45		Pauze
13:15-15:15	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Funcities & Control flow 1
16:15-16:30		Afsluiting



Blok 3a - Data Types - Individueel

Theorie

09:45-10:15	Blok 1	Inleiding
10:20-11:00	Blok 2	Programmeren in Python
11:00-12:00	Blok 3a	Data types - Individueel - Theorie
12:00-12:45		Pauze
13:15-15:15	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Funcities & Control flow 1
16:15-16:30		Afsluiting

Data types - Individueel

- Tekst: String
- Nummers: Integer, Float
- Waar/onwaar: Boolean
- Niets: Nonetype



A diagram illustrating a change in a variable's value. It features a light gray background with a faint, stylized globe pattern. Two rectangular boxes are stacked vertically. The top box contains the text 'naam = None', where 'naam' is in black, '=' is in purple, and 'None' is in green. A blue arrow points downwards from the 'None' value in the top box to the 'Mees' value in the bottom box. The bottom box contains the text 'naam = 'Mees'', where 'naam' is in black, '=' is in purple, and 'Mees' is in red and enclosed in single quotes.

```
naam = None
```

```
naam = 'Mees'
```

Data types - Soorten

- `x = "Hello World"`
- `x = 20`
- `x = True`
- `x = None`
- `x = 20.5`

`str`
`int`
`bool`
`NoneType`
`float`

Data types - Type

- Syntax
 - `type()`

```
[1]: print(type('hello world'))  
  
      <class 'str'>
```

Data types - Type

- Syntax
 - `type()`

```
[1]: print(type('hello world'))
```

```
<class 'str'>
```

Data types - String

- String is tekst
- Syntax
 - `str()`



```
string_1 = 'Tekst'
string_2 = "Dit is hetzelfde, maar zorgt er bijvoorbeeld voor dat ik 'aanhalingstekens' ook kan opslaan in de variable"
string_3 = ''' Dit werkt op meerdere regels
Kijk maar
De enters worden ook opgeslagen
Enters kan ik ook opslaan door \n in een string te zetten'''
print(string_3)
```

```
Dit werkt op meerdere regels
Kijk maar
De enters worden ook opgeslagen
Enters kan ik ook opslaan door
  in een string te zetten
```

Data types - String

- String slicing
- Syntax
 - tekst[int]
 - tekst[int:int]
 - tekst[:int]
 - tekst[int:]

```
tekst = 'abcd efg hij'  
print(tekst[0])  
print(tekst[:4])  
print(tekst[4:])  
print(tekst[-1])
```

```
a  
abcd  
  efg hij  
j
```

Data types - String

- Methods: ingebouwde instructies in
- Python die oa strings kunnen
- manipuleren
- Syntax
 - Voor lengte: len(tekst)
 - tekst.doeiets()

```
tekst = '    Python leren    '
```

Fout

```
tekst.upper()
print('upper() methode?:', tekst)
```

Goed

```
tekst = tekst.upper()
print('upper() methode:', tekst)
```

```
tekst = tekst.lower()
print('lower() methode:', tekst)
```

```
tekst = tekst.strip()
print('strip() methode:', tekst)
```

```
tekst = tekst.replace('python', 'programmeren')
print('replace() methode:', tekst)
```



```
upper() methode?:    Python leren
upper() methode:     PYTHON LEREN
lower() methode:     python leren
strip() methode:     python leren
replace() methode:   programmeren leren
```

Data types - Integers & Floats

- Nummers
- Integers zijn gehele nummers
- Floats zijn kommagetallen
- Python punt

```
cijfer1 = 6
cijfer2 = float(6)
print('Int:', cijfer1)
print('Float:', cijfer2)
```

Int: 6

Float: 6.0

Rekenen met variabelen

- Min: -
- Plus: +
- Keer: *
- Delen door: /
- Vloerdeling: //
- Tot de macht: **
- Modulo: %

$25//7=3$

$3*7=21$, dus er blijft $25-21=4$ over

$25\%7= 4$

Data types - Boolean

- Waar of onwaar
- Andere data types kunnen ook waar of onwaar zijn
- Syntax
 - `bool()`
 - `True`
 - `False`

Blok 3a - Data Types - Individueel

Demonstratie

09:45-10:15	Blok 1	Inleiding
10:20-11:00	Blok 2	Programmeren in Python
11:00-12:00	Blok 3a	Data types - Individueel - Theorie
12:00-12:45		Pauze
13:15-15:15	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Funcities & Control flow 1
16:15-16:30		Afsluiting

Blok 3a - Data Types - Individueel

Praktijk - https://tinyurl.com/gb_training

09:45-10:15	Blok 1	Inleiding
10:20-11:00	Blok 2	Programmeren in Python
11:00-12:00	Blok 3a	Data types - Individueel - Theorie
12:00-12:45		Pauze
13:15-15:15	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Funcities & Control flow 1
16:15-16:30		Afsluiting

Pauze

12:00-12:45



Blok 3b - Data Types - Verzamelingen

Theorie

09:45-10:15	Blok 1	Inleiding
10:20-11:00	Blok 2	Programmeren in Python
11:00-12:00	Blok 3a	Data types - Individueel - Theorie
12:00-12:45		Pauze
13:15-15:15	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Funcities & Control flow 1
16:15-16:30		Afsluiting

Data types - Soorten

- Individueel
 - Tekst:
 - Nummers:
 - Waar/onwaar:
 - Niets
- Verzamelingen van *elementen*
 - List
 - Tuple
 - Dict
 - Set

String

Integer, Float

Boolean

Nonetype

List

- Boodschappenlijstje
- Aanpasbaar
- Volgorde/index
- Syntax
 - `list()`
 - `lijst = [1,2,3]`
 - `lijst.append(1)`
 - `lijst[0] = 3` # aanpasbaar en volgorde

Tuple

- Niet aanpasbaar*
- Volgorde, dus index
- Ook niet met `.pop()`, `.append()`
- Syntax
 - `tuple()`
 - `tuple_1 = (1,2,'drie')**`

* Er zijn uitzonderingen

** Werkt gek met één element

Dictionary

- Aanpasbaar
- Geen volgorde, dus geen index
- Waardes kunnen gevonden worden aan de hand van een 'key'

het **hoofd** (o; -en) **1** bovenste deel van het menselijk lichaam: *aan iets het ~ **bieden** zich ertegen verzetten; iemands ~ **eisen** zijn aftreden eisen; een **hard** ~ in iets hebben een zaak somber inzien; heel wat **aan** zijn ~ hebben de zorg voor veel dingen hebben; er hangt ons iets **boven** het ~ er dreigt gevaar; iem., iets **over** het ~ zien (per ongeluk) niet zien; **uit** het ~ leren van buiten; iem. voor het ~ **stoten** kwetsend behandelen; zich het ~ **breken** over iets erover tobben; het **groeit** me bo-*

1/3

hoofd: bovenste deel
menselijk lichaam

teen: vinger aan de voet

Dictionary

- Aanpasbaar
- Geen volgorde, dus geen index
- Waardes kunnen gevonden worden aan de hand van een 'key'

het **hoofd** (o; -en) **1** bovenste deel van het menselijk lichaam: *aan iets het ~ **bieden** zich ertegen verzetten; iemands ~ **eisen** zijn af-treden eisen; een **hard** ~ in iets hebben een zaak somber inzien; heel wat **aan** zijn ~ hebben de zorg voor veel dingen hebben; er hangt ons iets **boven** het ~ er dreigt gevaar; iem., iets **over** het ~ zien (per ongeluk) niet zien; **uit** het ~ leren van buiten; iem. voor het ~ **stoten** kwetsend behandelen; zich het ~ **breken** over iets erover tobben; het **groeit** me bo-*

2/3

{hoofd: bovenste deel
menselijk lichaam,

teen: vinger aan de voet}

Dictionary

- Aanpasbaar
- Geen volgorde, dus geen index
- Waardes kunnen gevonden worden aan de hand van een 'key'
- Syntax
 - dict()
 - dict_1 = {'hoofd': 'bovenste deel menselijk lichaam',
 'teen': 'vinger aan de voet'}
 - dict_1['hond'] = 'beste vriend van een mens'

het **hoofd** (o; -en) **1** bovenste deel van het menselijk lichaam: *aan iets het ~ **bieden** zich ertegen verzetten; iemands ~ **eisen** zijn af-treden eisen; een **hard** ~ in iets hebben een zaak somber inzien; heel wat **aan** zijn ~ heb-ben de zorg voor veel dingen hebben; er hangt ons iets **boven** het ~ er dreigt gevaar; iem., iets **over** het ~ zien (per ongeluk) niet zien; **uit** het ~ leren van buiten; iem. voor het ~ **stoten** kwetsend behandelen; zich het ~ **bre-ken** over iets erover tobben; het **groeit** me bo-*

3/3

{'hoofd': 'bovenste deel
menselijk lichaam',

'teen': 'vinger aan de voet'}

Set

- Aanpasbaar
- Geen volgorde, dus geen index
- Ook geen key
- Syntax
 - dict()
 - `set_1 = {'Mees', '180'}`
 - ~~×~~ `set_1['achternaam'] = 'Meester'`
 - ~~×~~ `set_1[0] = 'Meester'`

{'bovenste deel menselijk
lichaam',

'vinger aan de voet'}

het **hoofd** (o; -en) **1** bovenste deel van het menselijk lichaam: *aan iets het ~ **bieden** zich ertegen verzetten; iemands ~ **eisen** zijn af-treden eisen; een **hard** ~ in iets hebben een zaak somber inzien; heel wat **aan** zijn ~ hebben de zorg voor veel dingen hebben; er hangt ons iets **boven** het ~ er dreigt gevaar; iem., iets **over** het ~ zien (per ongeluk) niet zien; **uit** het ~ leren van buiten; iem. voor het ~ **stoten** kwetsend behandelen; zich het ~ **bre-ken** over iets erover tobben; het **groeit** me bo-*

3/3

{'hoofd': 'bovenste deel
menselijk lichaam',

'teen': 'vinger aan de voet'}

Blok 3b - Data Types - Verzamelingen

Demonstratie

09:45-10:15	Blok 1	Inleiding
10:20-11:00	Blok 2	Programmeren in Python
11:00-12:00	Blok 3a	Data types - Individueel - Theorie
12:00-12:45		Pauze
13:15-15:15	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Funcities & Control flow 1
16:15-16:30		Afsluiting

Blok 3b - Data Types - Verzamelingen

Praktijk - https://tinyurl.com/gb_training

09:45-10:15	Blok 1	Inleiding
10:20-11:00	Blok 2	Programmeren in Python
11:00-12:00	Blok 3a	Data types - Individueel - Theorie
12:00-12:45		Pauze
13:15-15:15	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Funcities & Control flow 1
16:15-16:30		Afsluiting



Blok 4 - Functies & Control flow 1

Theorie

09:45-10:15	Blok 1	Inleiding
10:20-11:00	Blok 2	Programmeren in Python
11:00-12:00	Blok 3a	Data types - Individueel - Theorie
12:00-12:45		Pauze
13:15-15:15	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Functies & Control flow 1
16:15-16:30		Afsluiting

Funcities

- Mini-programma
- Handig als je operaties wil herhalen
- Syntax
 - `def functie(input):`
 - `# doe iets (mini Python programma)`
 - `return output (output voor programmeur)`

If, else if, else

- Syntax
 - if conditie:
 # doe iets
 - elif conditie:
 # doe iets anders
 - else:
 # doe iets

Logica & Algebra

- and en or
- Booleans
- Syntax
 - $5 < 10$ kleiner dan
 - $10 > 5$ groter dan
 - $5 == 5$ gelijk aan
 - $5 != 10$ niet gelijk aan
 - $5 <= 10$ kleiner dan of gelijk aan
 - $5 <= 5$ kleiner dan of gelijk aan
 - $5 >= 3$ groter dan of gelijk aan
 - $5 >= 5$ groter dan of gelijk aan

Logica & Algebra

- and en or
- Booleans
- Booleans
- Syntax
 - $x \text{ and } y$ # waar als x en y waar zijn
 - $x \text{ or } y$ # waar als x of y waar is

Blok 4 - Functies & Control flow 1

Demonstratie

09:45-10:15	Blok 1	Inleiding
10:20-11:00	Blok 2	Programmeren in Python
11:00-12:00	Blok 3a	Data types - Individueel - Theorie
12:00-12:45		Pauze
13:15-15:15	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Functies & Control flow 1
16:15-16:30		Afsluiting

Blok 4 - Functies & Control flow 1

Praktijk

09:45-10:15	Blok 1	Inleiding
10:20-11:00	Blok 2	Programmeren in Python
11:00-12:00	Blok 3a	Data types - Individueel - Theorie
12:00-12:45		Pauze
13:15-15:15	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Functies & Control flow 1
16:15-16:30		Afsluiting

Afsluiting

16:15-16:30

09:45-10:15

10:20-11:00

11:00-12:00

12:00-12:45

13:15-15:15

15:30-16:15

16:15-16:30

Blok 1

Inleiding

Blok 2

Programmeren in Python

Blok 3a

Data types - Individueel - Theorie

Pauze

Blok 3b

Data types - Verzamelingen

Blok 4

Functies & Control flow 1

Afsluiting



Vragen

Zijn er nog vragen?

- Stel ze gerust!



Huiswerk

- Leer de begrippen uit deze presentatie
- Hulpmiddelen: W3schools, Codecademy, Stackoverflow en Youtube
- Leer de Logica & Algebra slide
- Maak de opdrachten af
 - https://tinyurl.com/gb_training

