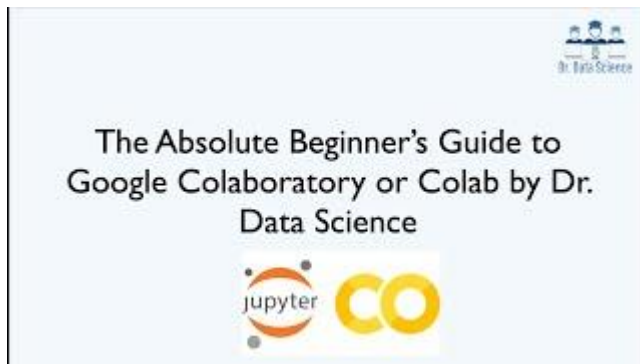




Vorbereiding

- Ga naar Google Drive
 - Maak een Google Account aan indien nodig
- Maak alvast een Google Colabratory bestand aan met de eerste drie minuten van <https://youtu.be/gLLTH2MJhmQ>
- Download de bestanden van readmees.github.io/python zet ze in Google Drive
- Kijk of je Dag1_Blok1.ipynb al kan openen in Google Colabratory





Python Basis

Dag 1

Mees Meester



Introductie

- Lang lesgeven
- Docent/trainer Opatel
- Python developer
- Hout- en metaalbewerking
- Kunstmatige Intelligentie
 - Universiteit van Amsterdam
 - Beeldverwerking



ChatGPT





Voorstellen

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



Opbouw

- 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

10:25-11:00	Blok 1	Inleiding
11:00-11:45	Blok 2	Programmeren in Python
11:45-12:15	Blok 3a	Data types - Individueel - Theorie
12:15-13:00		Pauze
13:00-13:35	Blok 3a	Data types - Individueel - Praktijk
13:40-15:25	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

10:25-11:00	Blok 1	Inleiding
11:00-11:45	Blok 2	Programmeren in Python
11:45-12:15	Blok 3a	Data types - Individueel - Theorie
12:15-13:00		Pauze
13:00-13:35	Blok 3a	Data types - Individueel - Praktijk
13:40-15:25	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Functies & Control flow 1
16:15-16:30		Afsluiting





Python

- Kunstmatige Intelligentie
 - Automatisering
 - Taalverwerking
 - Beeldverwerking
 - Reinforcement Learning (trial and error)
 - Deep learning (neural networks)
- (Big) data analyse
- Web development



Python

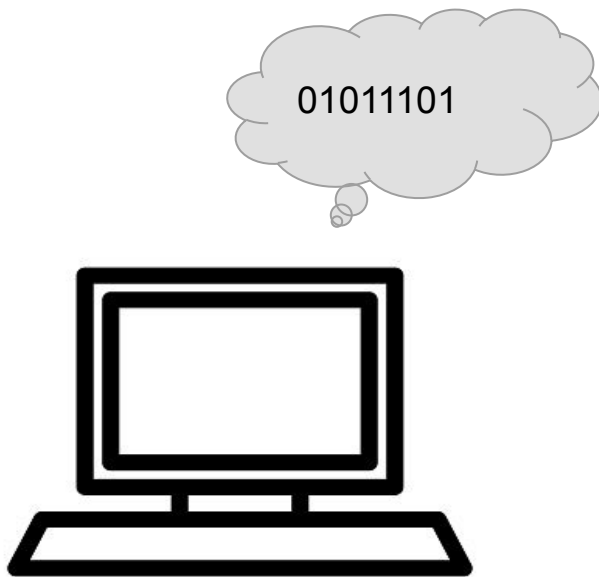
- Ontstaan in begin jaren '90
- Het Britse Monty Python
- Amsterdam
- Syntax

The image shows the title card for the Monty Python film 'Monty Python and the Holy Grail'. The text 'Monty Python' is written in a large, elegant, white cursive script. Below it, the words 'AND THE HOLY GRAIL' are written in a smaller, white, all-caps serif font. The entire text is centered on a solid red rectangular background.



Programmeren

- Praten met een computer



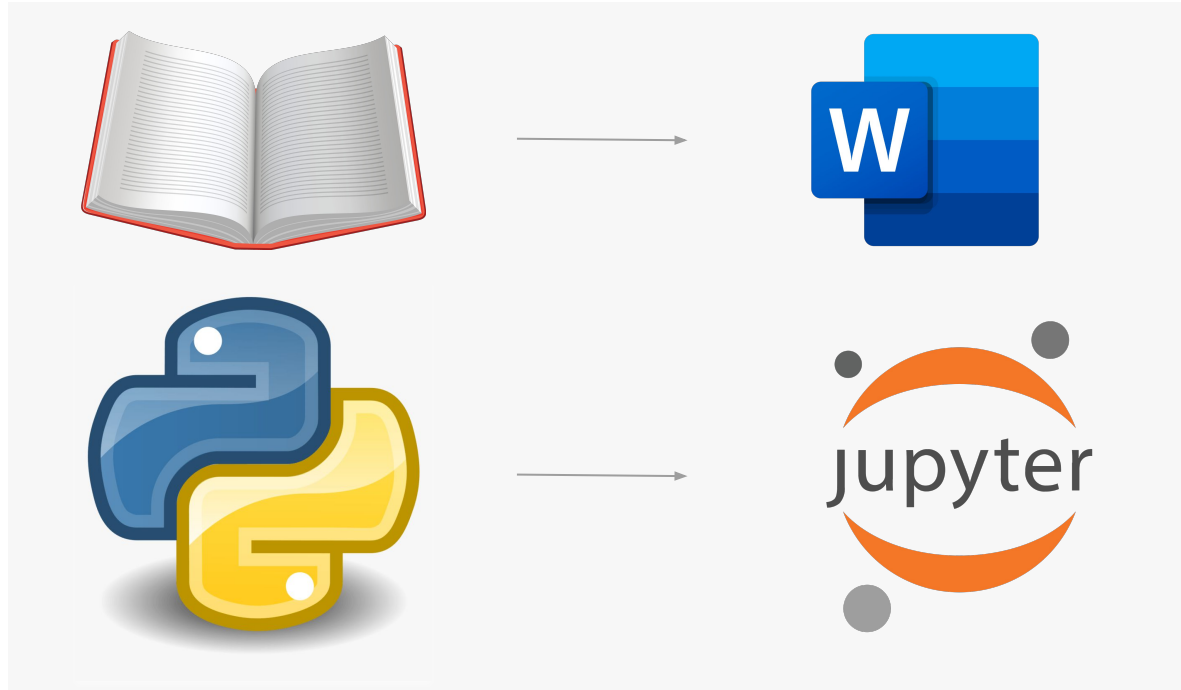


Programmeren - Code schrijven?

- Programma's installeren
- Debuggen
- Betere programmeur = beter Googlen
 -  Hoe krijg ik Python zover om het antwoord $5+3$ te laten uitrekenen?
 -  addition.py
- ChatGPT?
 - Ongelooflijk goed
 - PAS OP!
 - Gebruik als leraar, niet als antwoordmachine



Jupyter/Colab





Jupyter/Colab

- Cellen
- Voorbeelden

The screenshot displays a Jupyter Colab notebook titled "Voorbeeld_Jupyter.ipynb". The interface includes a top menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu is a toolbar with "Commands", "+ Code", "+ Text", and "Run all". On the left, a sidebar contains icons for a menu, search, code editor, keyboard shortcuts, and a file explorer. The main area shows two code cells under the heading "Voorbeeld Jupyter".

Cell [8]:

```
print('Welkom bij Python Basis')
```

Output: Welkom bij Python Basis

Cell [9]:

```
print('print() gebruiken we om output te genereren')
```

Output: ... print() gebruiken we om output te genereren



Businesscase

- Geautomatiseerde & gepersonaliseerde e-mails naar klanten sturen
- Een willekeurig aantal klanten
- Voorbeeld:

Beste Mees,

Als trouwe klant willen we jou, Mees Meester, de beste wensen toesturen.

Met vriendelijke groet,

Opatel

Blok 1 - Inleiding

Demonstratie - <https://readmees.github.io/python>

10:25-11:00	Blok 1	Inleiding
11:00-11:45	Blok 2	Programmeren in Python
11:45-12:15	Blok 3a	Data types - Individueel - Theorie
12:15-13:00		Pauze
13:00-13:35	Blok 3a	Data types - Individueel - Praktijk
13:40-15:25	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Functies & Control flow 1
16:15-16:30		Afsluiting



Blok 1 - Inleiding

Demonstratie - <https://readmees.github.io/python>

10:25-11:00	Blok 1	Inleiding
11:00-11:45	Blok 2	Programmeren in Python
11:45-12:15	Blok 3a	Data types - Individueel - Theorie
12:15-13:00		Pauze
13:00-13:35	Blok 3a	Data types - Individueel - Praktijk
13:40-15:25	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Functies & Control flow 1
16:15-16:30		Afsluiting



Blok 2 - Programmeren in Python

Theorie

10:25-11:00	Blok 1	Inleiding
11:00-11:45	Blok 2	Programmeren in Python
11:45-12:15	Blok 3a	Data types - Individueel - Theorie
12:15-13:00		Pauze
13:00-13:35	Blok 3a	Data types - Individueel - Praktijk
13:40-15:25	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Functies & 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'
```



Variabelen

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

```
sender = "pythonbasistest@gmail.com"  
receiver = "mm@opatel.nl"
```



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?')
```



Input

- Syntax
 - `input()`
- Businesscase

```
sender = "pythonbasistest@gmail.com"  
receiver = "mm@opatel.nl"  
password = input('Wat is je wachtwoord?')
```




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)
```



Comment

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

- Businesscase

```
sender = "pythonbasistest@gmail.com"  
receiver = "mm@opatel.nl"
```

```
# You password can be made at https://myaccount.google.com/apppasswords  
password = input('Wat is je wachtwoord?')
```

Blok 2 - Programmeren in Python

Demonstratie - <https://readmees.github.io/python>

10:25-11:00	Blok 1	Inleiding
11:00-11:45	Blok 2	Programmeren in Python
11:45-12:15	Blok 3a	Data types - Individueel - Theorie
12:15-13:00		Pauze
13:00-13:35	Blok 3a	Data types - Individueel - Praktijk
13:40-15:25	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Functies & Control flow 1
16:15-16:30		Afsluiting



Blok 2 - Programmeren in Python

Praktijk - <https://readmees.github.io/python>

10:25-11:00	Blok 1	Inleiding
11:00-11:45	Blok 2	Programmeren in Python
11:45-12:15	Blok 3a	Data types - Individueel - Theorie
12:15-13:00		Pauze
13:00-13:35	Blok 3a	Data types - Individueel - Praktijk
13:40-15:25	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Functies & Control flow 1
16:15-16:30		Afsluiting



Blok 3a - Data Types - Individueel

Theorie

10:15-10:35	Blok 1	Inleiding
10:40-11:30	Blok 2	Programmeren in Python
11:35-12:30	Blok 3a	Data types - Individueel - Theorie
12:45-13:30		Pauze
13:00-13:35	Blok 3a	Data types - Individueel - Praktijk
13:40-15:25	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Functies & Control flow 1
16:15-16:30		Afsluiting



Data types - Individueel

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

```
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 is tekst
- Syntax
 - `str()`
- f-string

```
naam = 'Tenzin'  
datum = '02-40-2013'  
  
mail = f'Beste {naam}, sinds {datum} hebben wij u met veel plezier als klant gehad!'  
  
print(mail)
```

```
Beste Tenzin, sinds 02-40-2013 hebben wij u met veel plezier als klant gehad!
```



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 - String

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

Omloopspaties verwijderen

```
print(name)  
print(name.strip().title())
```

MeEs MEeSTer
Mees Meester

Elk woord met een hoofdletter



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 - <https://readmees.github.io/python>

10:25-11:00	Blok 1	Inleiding
11:00-11:45	Blok 2	Programmeren in Python
11:45-12:15	Blok 3a	Data types - Individueel - Theorie
12:15-13:00		Pauze
13:00-13:35	Blok 3a	Data types - Individueel - Praktijk
13:40-15:25	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Functies & Control flow 1
16:15-16:30		Afsluiting



Pauze
12:45-13:30



Blok 3a - Data Types - Individueel

Praktijk - <https://readmees.github.io/python>

10:25-11:00	Blok 1	Inleiding
11:00-11:45	Blok 2	Programmeren in Python
11:45-12:15	Blok 3a	Data types - Individueel - Theorie
12:15-13:00		Pauze
13:00-13:35	Blok 3a	Data types - Individueel - Praktijk
13:40-15:25	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Functies & Control flow 1
16:15-16:30		Afsluiting



Blok 3b - Data Types - Verzamelingen

Theorie

10:15-10:35	Blok 1	Inleiding
10:40-11:30	Blok 2	Programmeren in Python
11:35-12:30	Blok 3a	Data types - Individueel - Theorie
12:45-13:30		Pauze
13:00-13:35	Blok 3a	Data types - Individueel - Praktijk
13:40-15:25	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Functies & Control flow 1
16:15-16:30		Afsluiting





Data types - Soorten

- Individueel
 - Tekst: String
 - Nummers: Integer, Float
 - Waar/onwaar: Boolean
 - Niets Nonetype
- Verzamelingen van *elementen*
 - List
 - Tuple
 - Dict
 - Set



List

- Boodschappenlijstje
- Aanpasbaar
- Volgorde/index
- Syntax
 - `list()`
 - `lijst = [1, 2, 3]`
 - `lijst.append(1)`
 - `print(lijst[0])`
 - 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 ~ 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-*

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
- Geen DUBBELE waarden
- 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 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-*

3/3

{‘hoofd’: ‘bovenste deel
menselijk lichaam’,

‘teen’: ‘vinger aan de voet’}

Blok 3b - Data Types - Verzamelingen

Demonstratie - <https://readmees.github.io/python>

10:25-11:00	Blok 1	Inleiding
11:00-11:45	Blok 2	Programmeren in Python
11:45-12:15	Blok 3a	Data types - Individueel - Theorie
12:15-13:00		Pauze
13:00-13:35	Blok 3a	Data types - Individueel - Praktijk
13:40-15:25	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Functies & Control flow 1
16:15-16:30		Afsluiting





Businesscase - Data Types

- Voorbeeld dictionary

```
klant1 = {'naam': 'Tenzin', 'achternaam': 'Pomare', 'datum': '02-40-2013'}
```

```
text = f'''Beste {klant1['naam']},  
Sinds {klant1['datum']} hebben wij u met veel plezier als klant gehad!'''
```

```
print(text)
```

```
Beste Tenzin,  
Sinds 02-40-2013 hebben wij u met veel plezier als klant gehad!
```

Blok 3b - Data Types - Verzamelingen

Praktijk - <https://readmees.github.io/python>

10:25-11:00	Blok 1	Inleiding
11:00-11:45	Blok 2	Programmeren in Python
11:45-12:15	Blok 3a	Data types - Individueel - Theorie
12:15-13:00		Pauze
13:00-13:35	Blok 3a	Data types - Individueel - Praktijk
13:40-15:25	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

Theorie

10:15-10:35	Blok 1	Inleiding
10:40-11:30	Blok 2	Programmeren in Python
11:35-12:30	Blok 3a	Data types - Individueel - Theorie
12:45-13:30		Pauze
13:00-13:35	Blok 3a	Data types - Individueel - Praktijk
13:40-15:25	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_naam(parameter):`
 `# 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 - <https://readmees.github.io/python>

10:25-11:00	Blok 1	Inleiding
11:00-11:45	Blok 2	Programmeren in Python
11:45-12:15	Blok 3a	Data types - Individueel - Theorie
12:15-13:00		Pauze
13:00-13:35	Blok 3a	Data types - Individueel - Praktijk
13:40-15:25	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 - <https://readmees.github.io/python>

10:25-11:00	Blok 1	Inleiding
11:00-11:45	Blok 2	Programmeren in Python
11:45-12:15	Blok 3a	Data types - Individueel - Theorie
12:15-13:00		Pauze
13:00-13:35	Blok 3a	Data types - Individueel - Praktijk
13:40-15:25	Blok 3b	Data types - Verzamelingen
15:30-16:15	Blok 4	Functies & Control flow 1
16:15-16:30		Afsluiting



Afsluiting

10:25-11:00

Blok 1 Inleiding

11:00-11:45

Blok 2 Programmeren in Python

11:45-12:15

Blok 3a Data types - Individueel - Theorie

12:15-13:00

Pauze

13:00-13:35

Blok 3a Data types - Individueel - Praktijk

13:40-15:25

Blok 3b Data types - Verzamelingen

15:30-16:15

Blok 4 Functies & Control flow 1

16:15-16:30

Afsluiting





Vragen

Zijn er nog vragen?

- Stel ze gerust!





Huiswerk

- Leer de begrippen uit deze presentatie
- Hulpmiddelen: W3schools, Codecademy, Stackoverflow en Youtube
- Antwoorden van ChatGPT heb je niets aan
- Leer de Logica & Algebra slide
- Maak de opdrachten af
 - <https://readmees.github.io/python>

