

Python

Dag 2
Mees Meester



Algemene vragen

- Tot nu toe?
- Vragen over de opdrachten kunnen tijdens het praktische deel van Blok 1

Maatwerk training

- Les eindigt om 16:15, daarna nog een kwartier om privé vragen te stellen
- Laten weten als je de opdrachten snel af hebt

Overzicht opleidingstraject

Lesdag 2

09:45-11:20	Blok 1	Herhaling
11:25-12:45	Blok 2	Loops
12:45-13:30		Pauze
13:30-14:45	Blok 3	Python geavanceerd
15:00-16:00	Blok 4	Libraries
16:00-16:15		Afsluiting



Blok 1 - Herhaling

Theorie

09:45-11:20	Blok 1	Herhaling
11:25-12:45	Blok 2	Loops
12:45-13:30		Pauze
13:30-14:45	Blok 3	Python geavanceerd
15:00-16:00	Blok 4	Libraries
16:00-16:15		Afsluiting



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 - Soorten

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

String

Integer, Float

Boolean

Nonetype

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 - 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 - 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

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 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-*

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'}

Functies

- 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

Blok 1 - Herhaling

Praktijk

09:45-11:20	Blok 1	Herhaling
11:25-12:45	Blok 2	Loops
12:45-13:30		Pauze
13:30-14:45	Blok 3	Python geavanceerd
15:00-16:00	Blok 4	Libraries
16:00-16:15		Afsluiting



Blok 2 - Loops

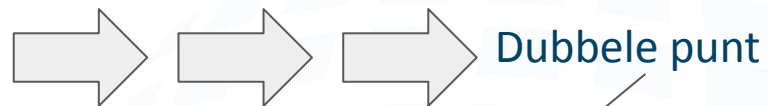
Theorie

09:45-11:20	Blok 1	Herhaling
11:25-12:45	Blok 2	Loops
12:45-13:30		Pauze
13:30-14:45	Blok 3	Python geavanceerd
15:00-16:00	Blok 4	Libraries
16:00-16:15		Afsluiting



Loops

- Doe iets meerdere keren

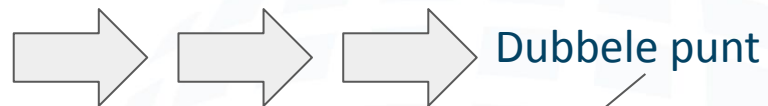


```
doe dit 3x:  
    print("vooruit")
```

Witruimte/tab

Loops

- Doe iets meerdere keren

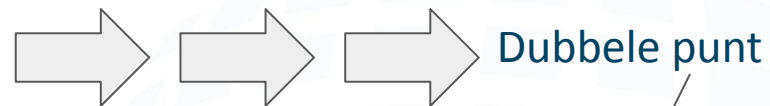


```
while not 3x:  
    print("vooruit")
```

Witruimte/tab

Loops

- Doe iets meerdere keren



```
stappen = 0
while stappen != 3:
    print("vooruit")
    stappen = stappen + 1
```

Witruimte/tab

```
vooruit
vooruit
vooruit
```

Range

- Vaak in combinatie met for loops
- Binnen bereik van 0-3, vallen 0, 1 en 2
- Syntax
 - `range()`

```
print(list(range(5)))
```

```
[0, 1, 2, 3, 4]
```


Loops - For

- Doe iets voor elk van iets
 - Doe voor elk getal in een range iets
 - Doe voor elk element van een lijst iets

```
for i in range(5):  
    print('Ik ga vooruit', i)
```

```
Ik ga vooruit 0  
Ik ga vooruit 1  
Ik ga vooruit 2  
Ik ga vooruit 3  
Ik ga vooruit 4
```

```
for i in [0,1,2,3,4]:  
    print('Ik ga vooruit', i)
```

```
Ik ga vooruit 0  
Ik ga vooruit 1  
Ik ga vooruit 2  
Ik ga vooruit 3  
Ik ga vooruit 4
```

Loops - For

- Doe iets voor elk van iets
- Binnen bereik van 0-3, vallen 0, 1 en 2
- Syntax
 - `range()`

```
for i in range(5):  
    print('Ik ga vooruit', i)  
print('Nu is i', i)
```

```
Ik ga vooruit 0  
Ik ga vooruit 1  
Ik ga vooruit 2  
Ik ga vooruit 3  
Ik ga vooruit 4  
Nu is i 4
```

```
i = 0  
while i != 5:  
    print('Ik ga vooruit', i)  
    i = i + 1  
print('Nu is i', i, ', dus i is niet 5 is onwaar')
```

```
Ik ga vooruit 0  
Ik ga vooruit 1  
Ik ga vooruit 2  
Ik ga vooruit 3  
Ik ga vooruit 4  
Nu is i 5 , dus i is niet 5 is onwaar
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
while True == True:  
    pass
```

```
i = 0  
while True == True:  
    i = i + 1  
    print(i)  
    break  
print("De loop is gestopt")
```

```
1  
De loop is gestopt
```

Blok 2 - Loops

Demonstratie

09:45-11:20

Blok 1 Herhaling

11:25-12:45

Blok 2 Loops

12:45-13:30

Pauze

13:30-14:45

Blok 3 Python geavanceerd

15:00-16:00

Blok 4 Libraries

16:00-16:15

Afsluiting



Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0 ←  
while True == True:  
    i = i + 1  
    print(i)  
  
    if i < 5:  
        continue  
    break  
  
print("De loop is gestopt")
```

```
1  
2  
3  
4  
5  
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True: ←
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```


Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)
    ←
    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5: ←
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

1
2
3
4
5
De loop is gestopt

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True: ←
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2 ←
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)
    ←
    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5: ←
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```


Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

1
2
3
4
5
De loop is gestopt

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True: ←
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

1

2

3

4

5

De loop is gestopt

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)
    ←
    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5: ←
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

1
2
3
4
5
De loop is gestopt

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True: ←
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```


Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)
    ←
    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5: ←
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

1
2
3
4
5
De loop is gestopt

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True: ←
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```


Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)
    ←
    if i < 5:
        continue
    break

print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```


```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break ← 
print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break
← print("De loop is gestopt")
```

```
1
2
3
4
5
De loop is gestopt
```

Loops - Overslaan en stoppen

- Pass
- Break
- Continue



```
i = 0
while True == True:
    i = i + 1
    print(i)

    if i < 5:
        continue
    break

print("De loop is gestopt")
```

1
2
3
4
5

De loop is gestopt

Blok 2 - Loops

Praktijk

09:45-11:20	Blok 1	Herhaling
11:25-12:45	Blok 2	Loops
12:45-13:30		Pauze
13:30-14:45	Blok 3	Python geavanceerd
15:00-16:00	Blok 4	Libraries
16:00-16:15		Afsluiting



Pauze
12:45-13:30



Blok 3 - Python geavanceerd

Theorie

09:45-11:20	Blok 1	Herhaling
11:25-12:45	Blok 2	Loops
12:45-13:30		Pauze
13:30-14:45	Blok 3	Python geavanceerd
15:00-16:00	Blok 4	Libraries
16:00-16:15		Afsluiting



List comprehension

- Manier van lijsten maken
 - Snel
 - Elegant

```
l = []  
l.append(3)  
print(l)
```

[3]

```
l = []  
for i in range(6):  
    l.append(i)  
print(l)
```

[0, 1, 2, 3, 4, 5]

```
l = [i for i in range(6)]  
print(l)
```

[0, 1, 2, 3, 4, 5]

List comprehension

- Manier van lijsten maken
 - Snel
 - Elegant
- Voorbij list comprehensions
 - Set comprehensions
 - Dict comprehensions

```
l = []  
l.append(3)  
print(l)
```

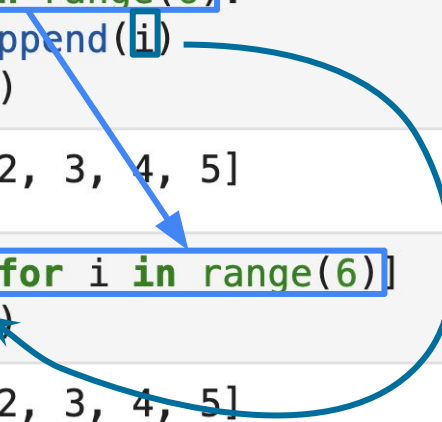
[3]

```
l = []  
for i in range(6):  
    l.append(i)  
print(l)
```

[0, 1, 2, 3, 4, 5]

```
l = [i for i in range(6)]  
print(l)
```

[0, 1, 2, 3, 4, 5]



Iterable

- Wanneer is iets iterable?
 - Je kan er doorheen lopen/lopen
 - Voorbeelden: lists, tuples, sets, dictionaries, strings

Iterator vs Iterable

- Iterables zijn containers/pakketjes die een iterator bevatten
 - Je kan een iterable maken van een verzameling
 - Syntax
 - `iter()`
- Waarom een iterator?
 - Je genereert maar een waarde tegelijk (als je die nodig hebt)
 - Kan efficiënter zijn

```
i = iter([1,2,3])
print(next(i))
print(next(i))
print(next(i))
print(next(i))
```

```
1
2
3
```

```
-----
StopIteration
Cell In[1], line 5
      3 print(next(i))
      4 print(next(i))
----> 5 print(next(i))
```

```
StopIteration:
```

Try/except

- “Ask forgiveness, not permission”
- In het voorbeeld is get netter

```
namen = {'Voornaam': 'Yvo'}  
try:  
    print(namen['Achternaam'])  
except KeyError:  
    print('Geen naam')
```

Geen naam

```
print(namen.get('Achternaam', 'Geen naam'))
```

Geen naam

Variabelen updaten

- Als je een variabele wilt update is er een elegantere manier

```
x = 0  
x = x + 5  
print(x)
```

5

```
x += 5  
print(x)
```

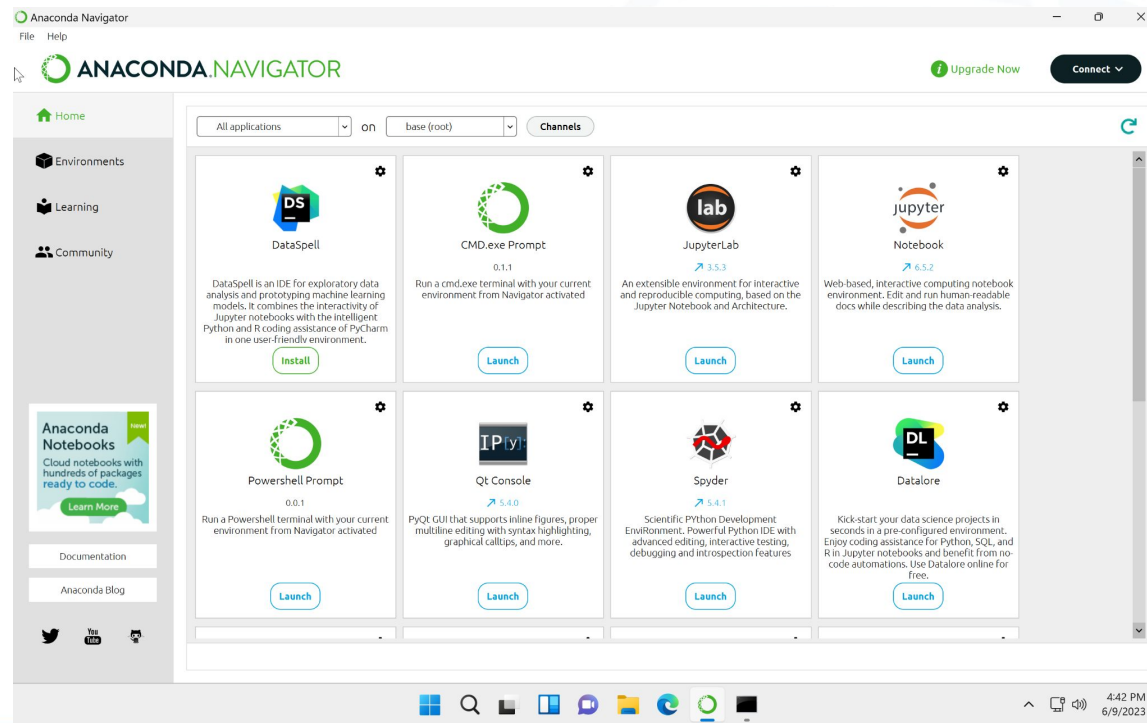
10

Wanneer geen Python?

- Als het safety-critical applicatie is
- Als snelheid of low-memory-usage belangrijk is
- Als je per regel code betaalt krijgt ;)

Anaconda Navigator

- Run Python lokaal
 - Veiliger/onveiliger?
- Ook Jupyter Notebooks mogelijk
- VS code



Blok 3 - Python geavanceerd

Praktijk

09:45-11:20	Blok 1	Herhaling
11:25-12:45	Blok 2	Loops
12:45-13:30		Pauze
13:30-14:45	Blok 3	Python geavanceerd
15:00-16:00	Blok 4	Libraries
16:00-16:15		Afsluiting



Blok 4 - Libraries

Theorie

09:45-11:20	Blok 1	Herhaling
11:25-12:45	Blok 2	Loops
12:45-13:30		Pauze
13:30-14:45	Blok 3	Python geavanceerd
15:00-16:00	Blok 4	Libraries
16:00-16:15		Afsluiting



Wat zijn libraries?

- Hulpmiddelen die je kan gebruiken
- Importen -> in je programma binnen halen
- Geen wiskunde? Import wiskunde!



Wat zijn libraries?

- Hulpmiddelen die je kan gebruiken
- Importen -> in je programma binnen halen
- Geen wiskunde? Import wiskunde!



```
import math  
wortel9 = math.sqrt(9)  
print(wortel9)
```

3.0

Wat zijn libraries?

- Hulpmiddelen die je kan gebruiken
- Importen -> in je programma binnen halen
- Geen wiskunde? Import wiskunde!



```
import math  
wortel9 = math.sqrt(9)  
print(wortel9)
```

3.0

Wat zijn libraries?

- Hulpmiddelen die je kan gebruiken
- Importen -> in je programma binnen halen
- Geen wiskunde? Import wiskunde!





```
import Google
wortel9 = Google
print(wortel9)
```

3.0

Libraries downloaden

- Syntax:
 - `pip install`
 - `pip install math`
- Syntax Google Colab
 - `!pip install math`
- Math is standaard al geïnstalleerd

Libraries downloaden

- Syntax:
 - `pip install`
 - `pip install` 
- Syntax Google Colab
 - `!pip install` 
- Math is standaard al geïnstalleerd

Blok 4 - Libraries

Demonstratie

09:45-11:20

Blok 1 Herhaling

11:25-12:45

Blok 2 Loops

12:45-13:30

Pauze

13:30-14:45

Blok 3 Python geavanceerd

15:00-16:00

Blok 4 Libraries

16:00-16:15

Afsluiting



Blok 4 - Libraries

Praktijk

09:45-11:20	Blok 1	Herhaling
11:25-12:45	Blok 2	Loops
12:45-13:30		Pauze
13:30-14:45	Blok 3	Python geavanceerd
15:00-16:00	Blok 4	Libraries
16:00-16:15		Afsluiting



Afsluiting

16:00-16:15

09:45-11:20	Blok 1	Herhaling
11:25-12:45	Blok 2	Loops
12:45-13:30		Pauze
13:30-14:45	Blok 3	Python geavanceerd
15:00-16:00	Blok 4	Libraries
16:00-16:15		Afsluiting



Vragen

Zijn er nog vragen?

- Stel ze gerust!



Afsluitingen

- Leer de begrippen uit deze presentatie
- Hulpmiddelen: W3schools, Codecademy, Stackoverflow en Youtube
 - ChatGPT: maar niet te vertrouwen
- Maak de opdrachten af
 - https://readmees.github.io/python_maatwerk
- Mag altijd vragen mailen!

