Universiteit Gent
Faculteit Wetenschappen

Programmeren in Python

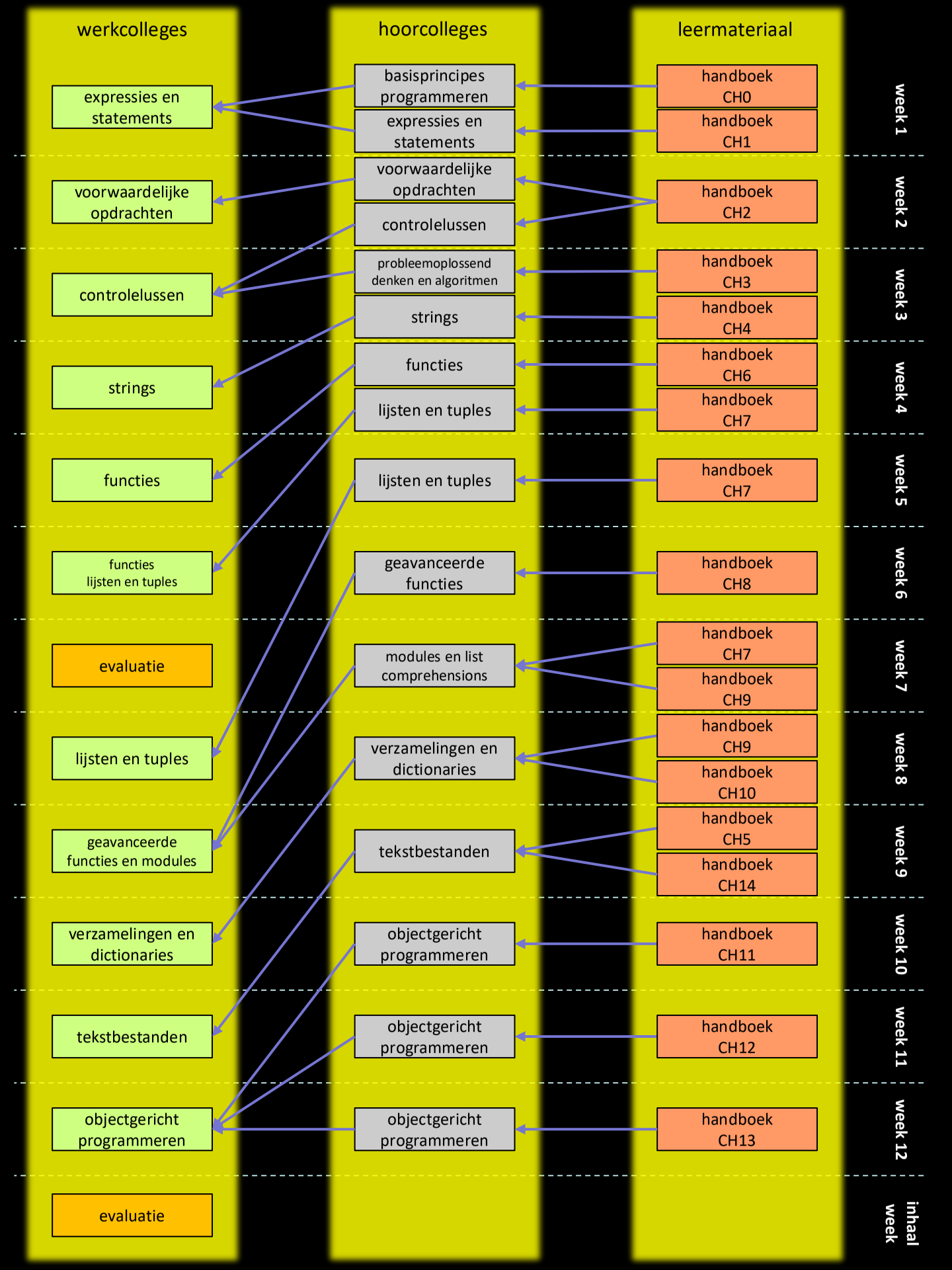
geavanceerde functies

prof. dr. Peter Dawyndt

peter.dawyndt@ugent.be

@dawyndt

UNIVERSITEIT GENT


werkcolleges
hoorcolleges
leermateriaal
expressies en statements
voorwaardelijke opdrachten
controlelussen
strings
functies
functies lijsten en tuples
evaluatie
lijsten en tuples
geavanceerde functies en modules
verzamelingen en dictionaries
tekstbestanden
objectgericht programmeren
evaluatie
basisprincipes programmeren
expressies en statements
voorwaardelijke opdrachten
controlelussen
probleemoplossend denken en algoritmen
strings
functies
lijsten en tuples
lijsten en tuples
geavanceerde functies
modules en list comprehensions
verzamelingen en dictionaries
tekstbestanden
objectgericht programmeren
objectgericht programmeren
objectgericht programmeren
handboek CH0
handboek CH1
handboek CH2
handboek CH3
handboek CH4
handboek CH6
handboek CH7
handboek CH7
handboek CH8
handboek CH7
handboek CH9
handboek CH9
handboek CH10
handboek CH5
handboek CH14
handboek CH11
handboek CH12
handboek CH13
week 1
week 2
week 3
week 4
week 5
week 6
week 7
week 8
week 9
week 10
week 11
week 12
inhaal week

Default params / Keyword args

Default parameters

- bij definiëren van functies kunnen standaardwaarden ingesteld worden voor parameters
  - gebruikt indien geen waarde doorgegeven voor deze parameters

```
>>> def wachter(naam, doel='Holy Grail', kleur='Blue'):
...     print('What is your name?')
...     print(f'My name is {naam}.')
...     print('What is your quest?')
...     print(f'To seek the {doel}.')
...     print('What is your favorite color?')
...     print(f'{kleur}.')

>>> wachter('Sir Launcelot of Camelot')
```

UNIVERSITEIT GENT

Default parameters

- bij definiëren van functies kunnen standaardwaarden ingesteld worden voor parameters
  - gebruikt indien geen waarde doorgegeven voor deze parameters

```
>>> def wachter(naam, doel='Holy Grail', kleur='Blue'):
...     print('What is your name?')
...     print(f'My name is {naam}.')
...     print('What is your quest?')
...     print(f'To seek the {doel}.'))
...     print('What is your favorite color?')
...     print(f'{kleur}.')

>>> wachter('Sir Launcelot of Camelot')
What is your name?
My name is Sir Launcelot of Camelot.
What is your quest?
To seek the Holy Grail.
What is your favorite color?
Blue.
```

UNIVERSITEIT GENT

Default parameters

- bij definiëren van functies kunnen standaardwaarden ingesteld worden voor parameters
  - gebruikt indien geen waarde doorgegeven voor deze parameters

```
>>> def wachter(naam, doel='Holy Grail', kleur='Blue'):
...     print('What is your name?')
...     print(f'My name is {naam}.')
...     print('What is your quest?')
...     print(f'To seek the {doel}.')
...     print('What is your favorite color?')
...     print(f'{kleur}.')

>>> wachter('Arthur, King of the Britons', 'Holy Hand Grenade')
What is your name?
My name is Arthur, King of the Britons.
What is your quest?
To seek the Holy Hand Grenade.
What is your favorite color?
Blue.
```

UNIVERSITEIT GENT

Default parameters

- bij definiëren van functies kunnen standaardwaarden ingesteld worden voor parameters
  - gebruikt indien geen waarde doorgegeven voor deze parameters

```
>>> def wachter(naam, doel='Holy Grail', kleur='Blue'):
...     print('What is your name?')
...     print(f'My name is {naam}.')
...     print('What is your quest?')
...     print(f'To seek the {doel}.')
...     print('What is your favorite color?')
...     print(f'{kleur}.')

>>> wachter('Sir Galahad', 'Holy Grail', 'Blue. No yellow')
What is your name?
My name is Sir Galahad.
What is your quest?
To seek the Holy Grail.
What is your favorite color?
Blue. No yellow.
```

UNIVERSITEIT GENT

Keyword arguments

- bij definiëren van functies kunnen standaardwaarden ingesteld worden voor parameters
  - gebruikt indien geen waarde doorgegeven voor deze parameters
  - doorgegeven argumenten krijgen voorrang op standaardwaarde
  - voor elke verplichte parameter moet argument aan functie doorgegeven worden
  - positionele argumenten
    - argumenten van links naar rechts gekoppeld aan parameters
    - parameters zonder standaardwaarde (*verplichte parameters*) moeten vóór default parameters (*optionele parameters*) staan bij definiëren van functies
  - benoemde argumenten (*keyword arguments*)
    - handig bij veel optionele parameters
    - makkelijker te lezen

UNIVERSITEIT GENT

Keyword arguments

- bij definiëren van functies kunnen standaardwaarden ingesteld worden voor parameters
  - benoemde argumenten (*keyword arguments*)

```
>>> def wachter(naam, doel='Holy Grail', kleur='Blue'):
...     print('What is your name?')
...     print(f'My name is {naam}.')
...     print('What is your quest?')
...     print(f'To seek the {doel}.')
...     print('What is your favorite color?')
...     print(f'{kleur}.')

>>> wachter('Sir Galahad', kleur='Blue. No yellow')
```

  - benoemde argumenten (*keyword arguments*)

UNIVERSITEIT GENT

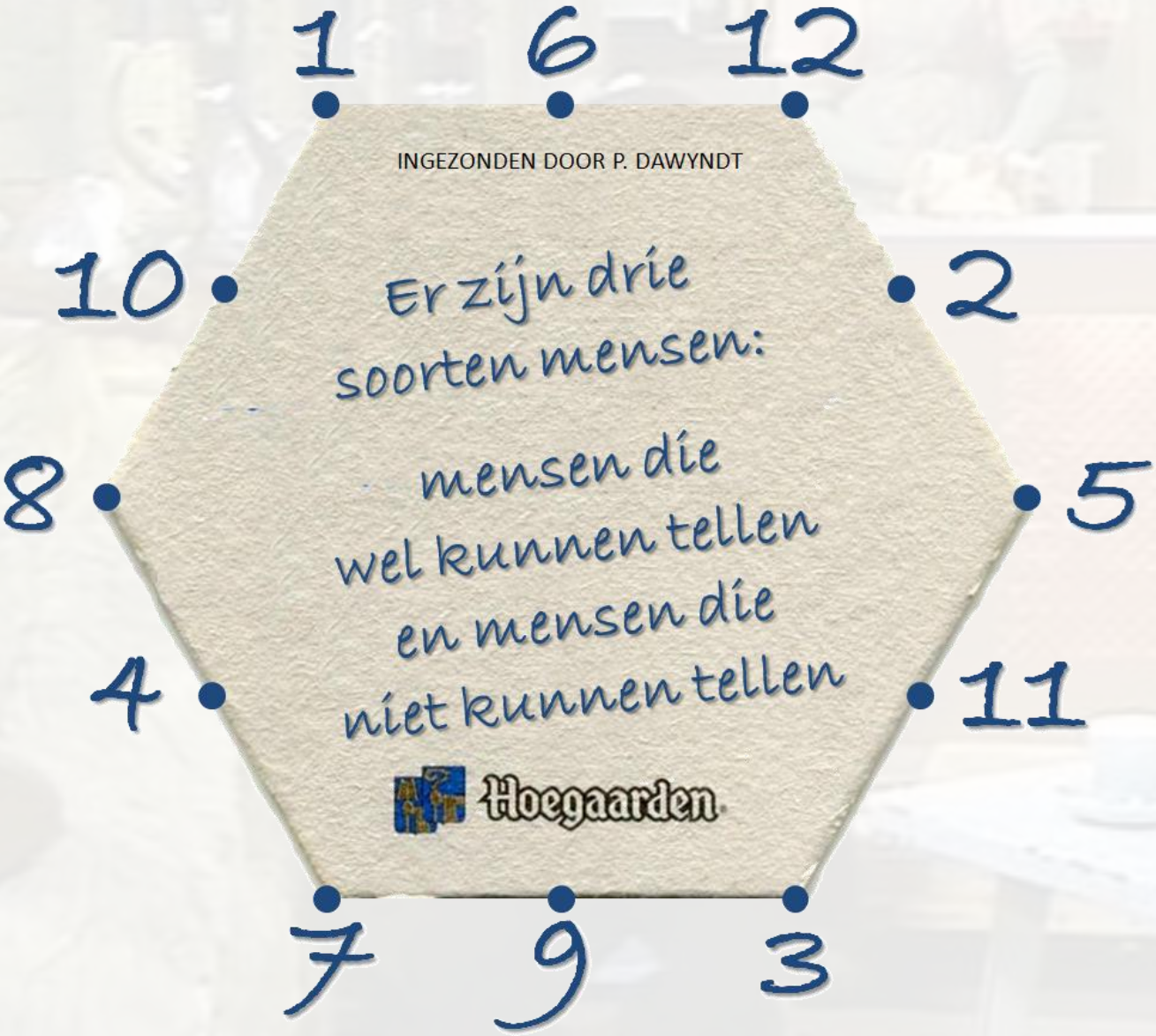
Keyword arguments

- bij definiëren van functies kunnen standaardwaarden ingesteld worden voor parameters
  - benoemde argumenten (*keyword arguments*)

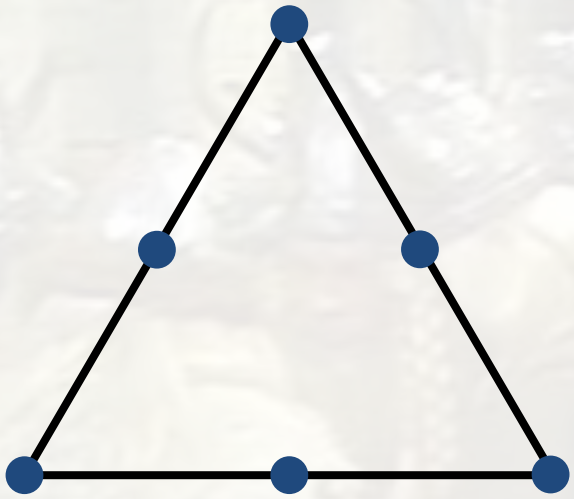
```
>>> def wachter(naam, doel='Holy Grail', kleur='Blue'):
...     print('What is your name?')
...     print(f'My name is {naam}.')
...     print('What is your quest?')
...     print(f'To seek the {doel}.')
...     print('What is your favorite color?')
...     print(f'{kleur}.')

>>> wachter('Sir Galahad', kleur='Blue. No yellow')
What is your name?
My name is Sir Galahad.
What is your quest?
To seek the Holy Grail.
What is your favorite color?
Blue. No yellow.
```

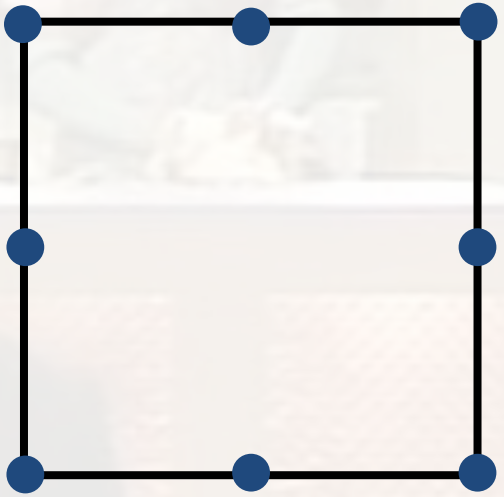
UNIVERSITEIT GENT

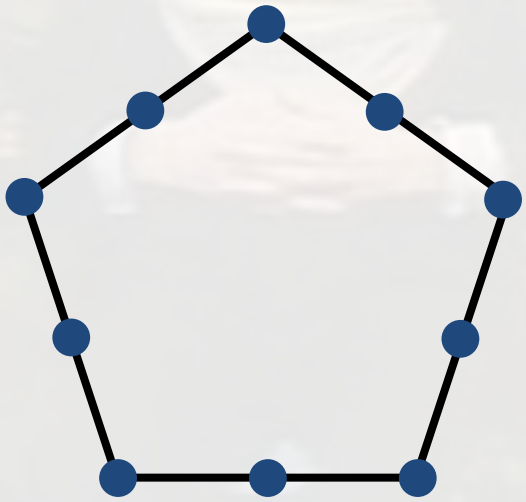
Caféspelletje



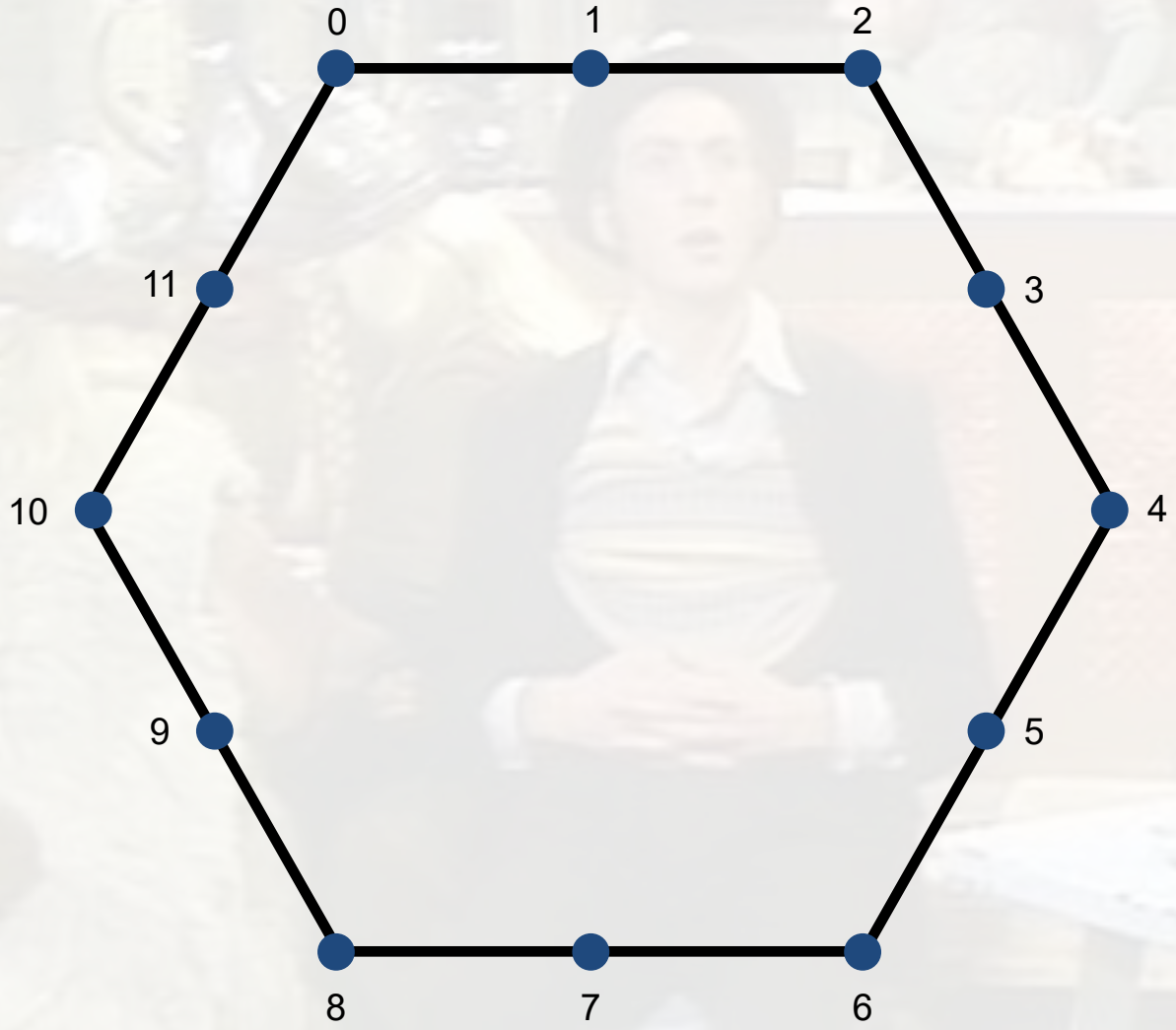
Caféspelletje







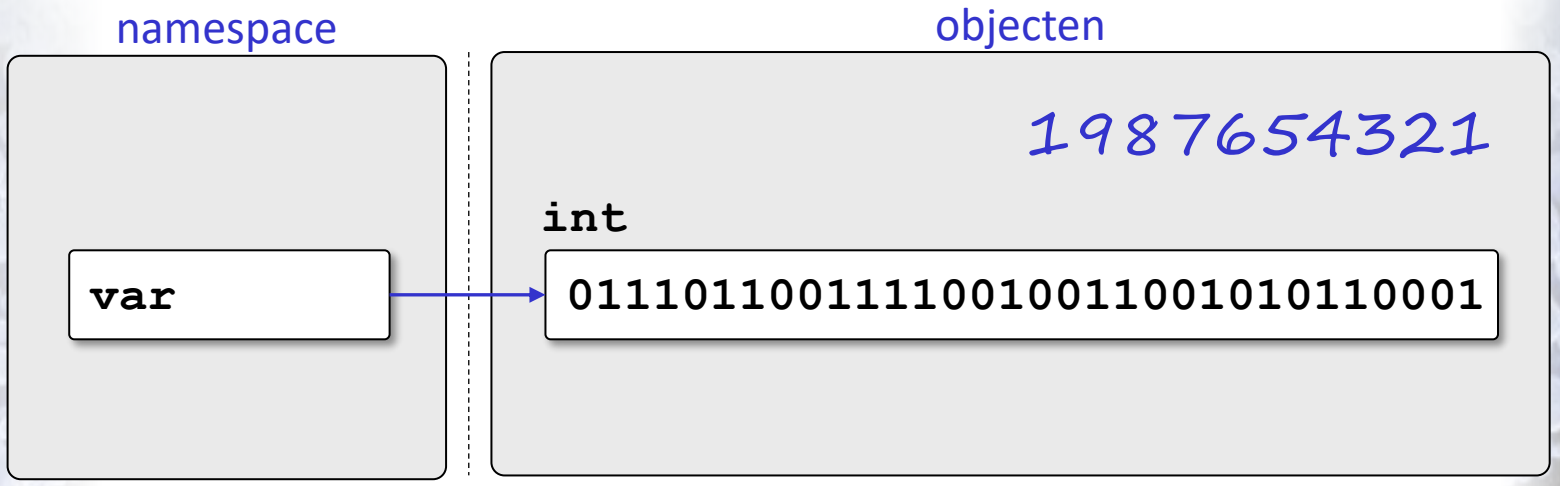
Caféspelletje



Functies zijn objecten

Functies zijn objecten

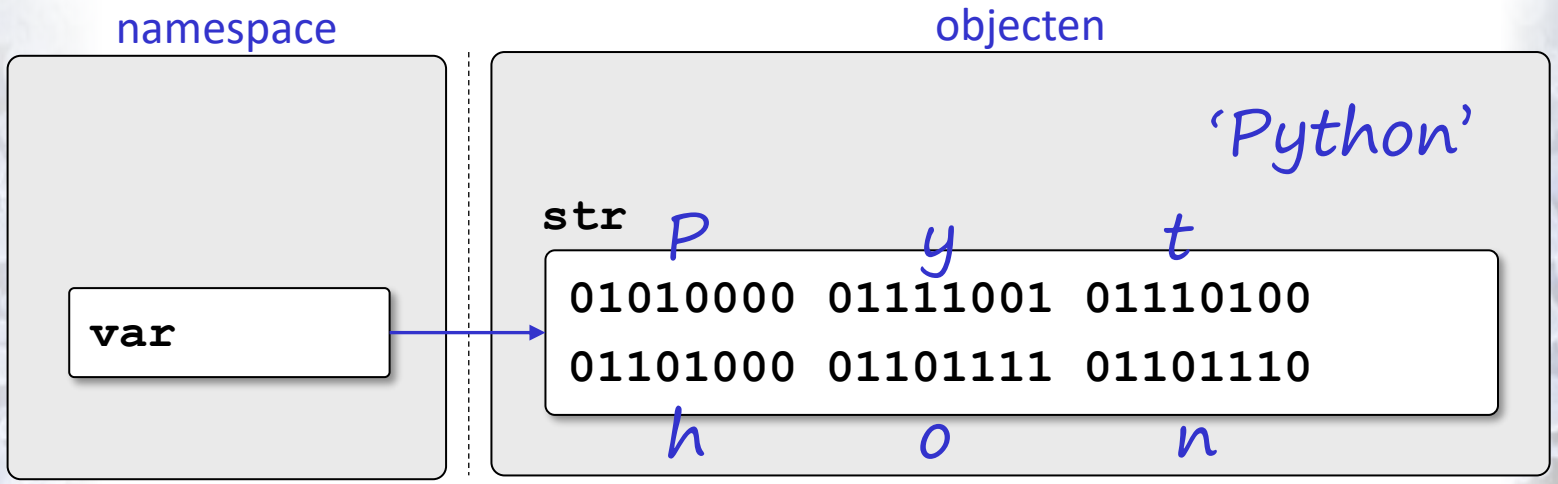
- integer is object dat 32 bits geheugen inpalmt …
  - … en waarnaar variabele kan verwijzen



UNIVERSITEIT GENT

Functies zijn objecten

- integer is object dat 32 bits geheugen inpalmt …
  - … en waarnaar variabele kan verwijzen
- string is reeks bytes die karakters voorstellen …
  - … en waarnaar variabele kan verwijzen
- functie is reeks bytes die instructies voorstellen …
  - … en natuurlijk kan ook daar een variabele naar verwijzen

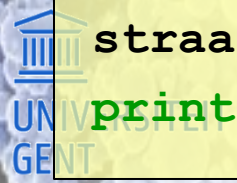


UNIVERSITEIT GENT

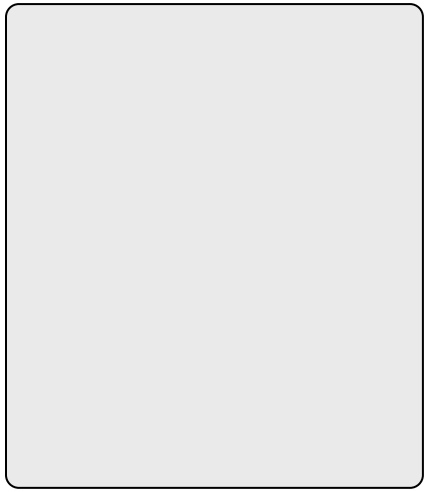
Functies zijn objecten

functies1.py

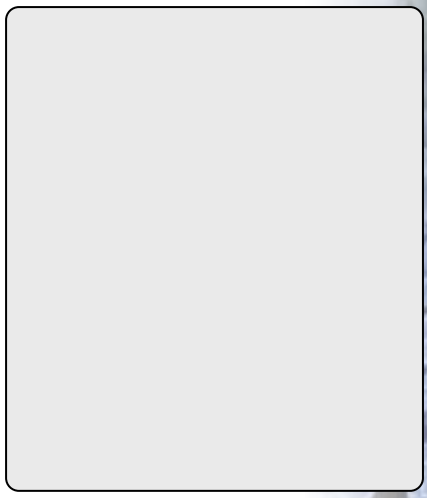
```
def omtrek(straal):
    import math
    return 2*math.pi*straal

straal = 2.87
print(omtrek(straal))
```



globals



namespace stack



objecten

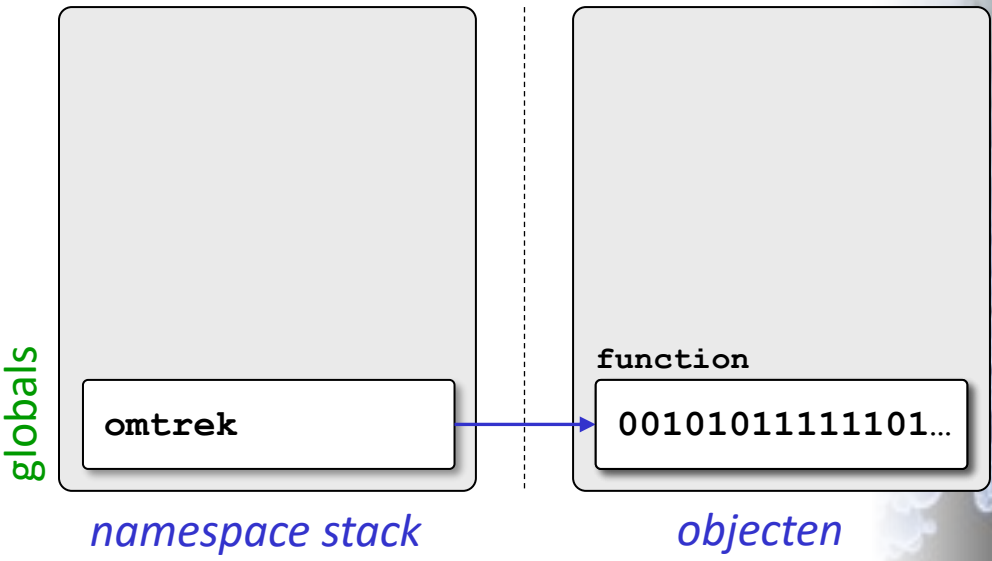
Functies zijn objecten

- functie is object dat kan aangeroepen worden
  - net zoals lijst of string object is dat kan geïndexeerd worden
  - **`def`** is dus korte notatie voor

"Maak een functie-object aan,
en ken dit toe aan een variabele."

functies1.py

```
def omtrek(straal):
    import math
    return 2*math.pi*straal

straal = 2.87
print(omtrek(straal))
```



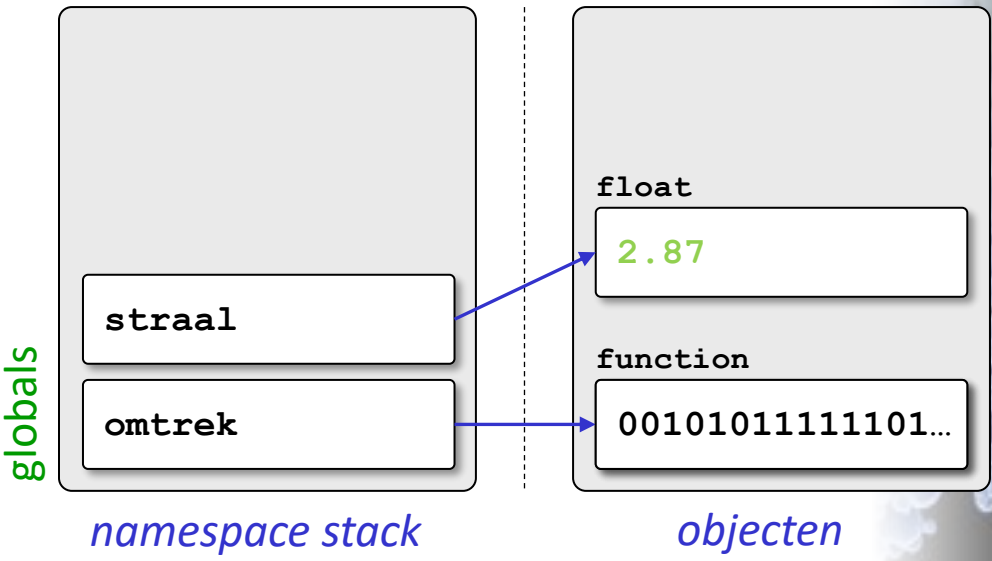
Functies zijn objecten

- functie is object dat kan aangeroepen worden
  - net zoals lijst of string object is dat kan geïndexeerd worden
  - **`def`** is dus korte notatie voor

"Maak een functie-object aan,
en ken dit toe aan een variabele."

functies1.py

```
def omtrek(straal):
    import math
    return 2*math.pi*straal

straal = 2.87
print(omtrek(straal))
```



Functies zijn objecten

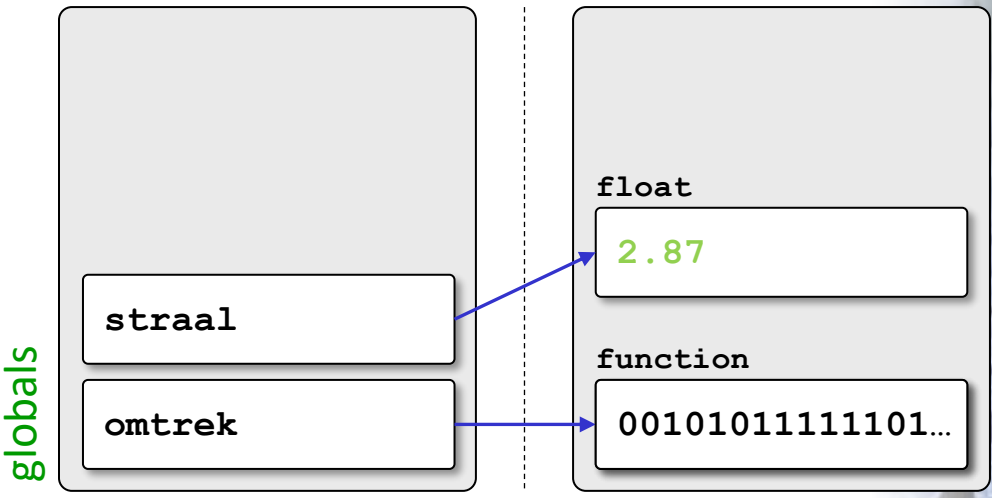
- functie is object dat kan aangeroepen worden
  - net zoals lijst of string object is dat kan geïndexeerd worden
  - **`def`** is dus korte notatie voor

"Maak een functie-object aan,
en ken dit toe aan een variabele."

functies1.py

```
def omtrek(straal):
    import math
    return 2*math.pi*straal

straal = 2.87
print(omtrek(straal))
```

18.0327418316

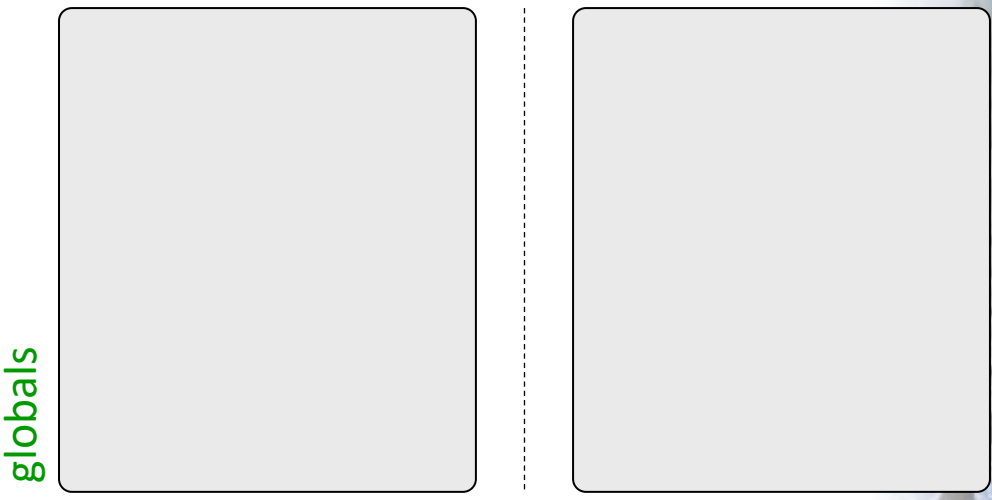


Functies zijn objecten

- dat betekent dat men
  - functies kan herdefiniëren

functies2.py

```
def omtrek(straal):
    return 2 * 3.14 * straal

def omtrek(straal):
    import math
    return 2*math.pi*straal

print(omtrek(2.87))
```

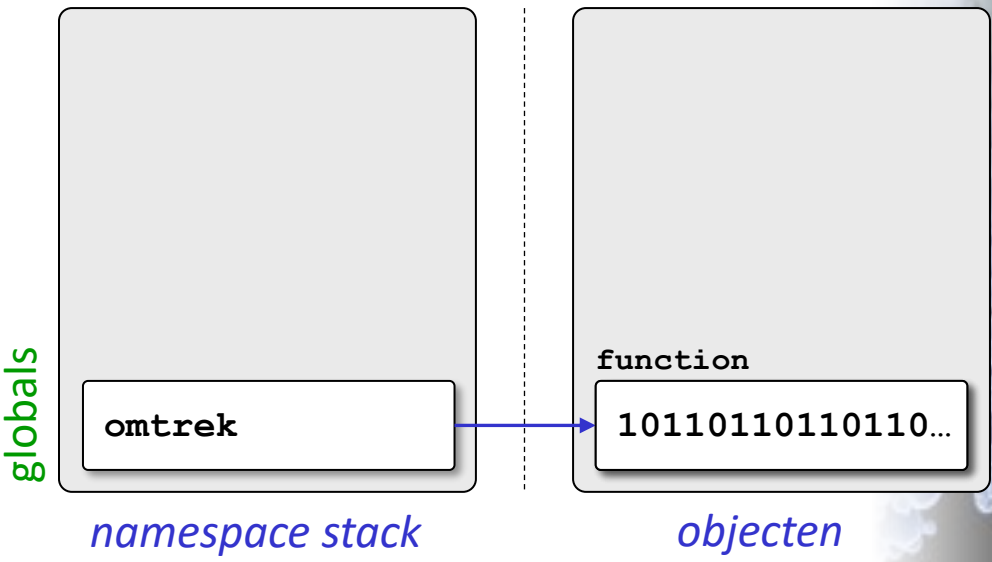


Functies zijn objecten

- dat betekent dat men
  - functies kan herdefiniëren

functies2.py

```
def omtrek(straal):
    return 2 * 3.14 * straal

def omtrek(straal):
    import math
    return 2*math.pi*straal

print(omtrek(2.87))
```

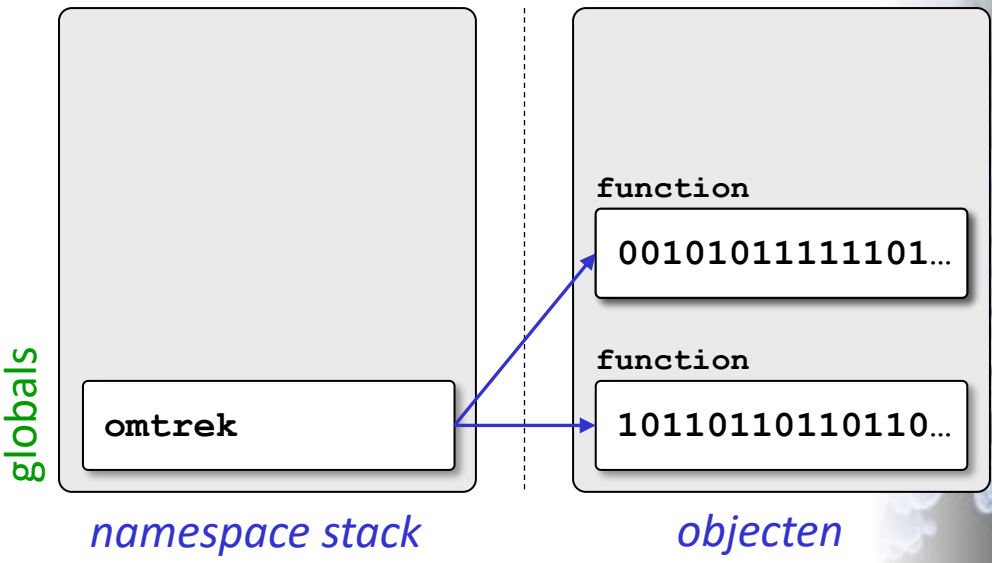


Functies zijn objecten

- dat betekent dat men
  - functies kan herdefiniëren

functies2.py

```
def omtrek(straal):
    return 2 * 3.14 * straal

def omtrek(straal):
    import math
    return 2*math.pi*straal

print(omtrek(2.87))
```

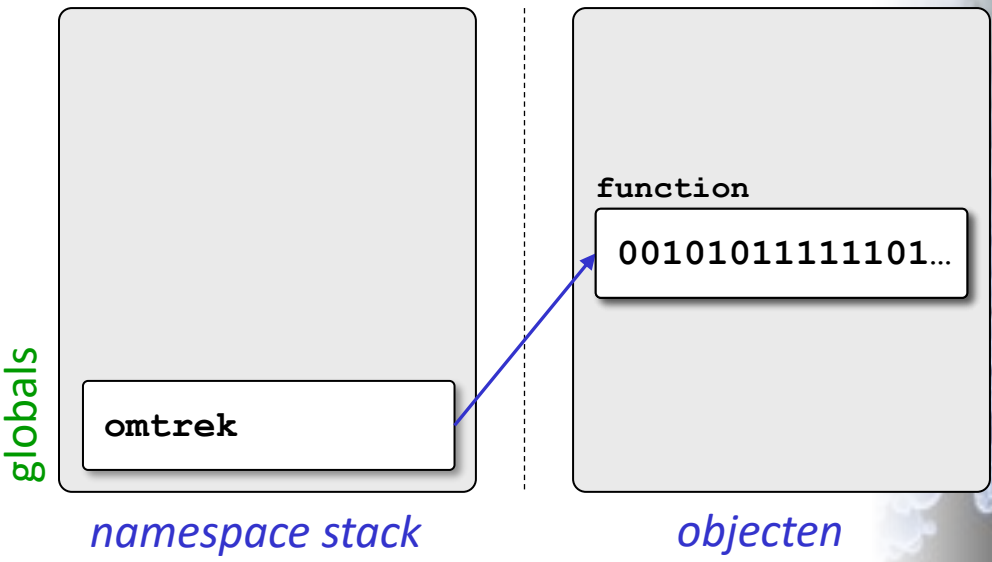


Functies zijn objecten

- dat betekent dat men
  - functies kan herdefiniëren

functies2.py

```
def omtrek(straal):
    return 2 * 3.14 * straal

def omtrek(straal):
    import math
    return 2*math.pi*straal

print(omtrek(2.87))
```

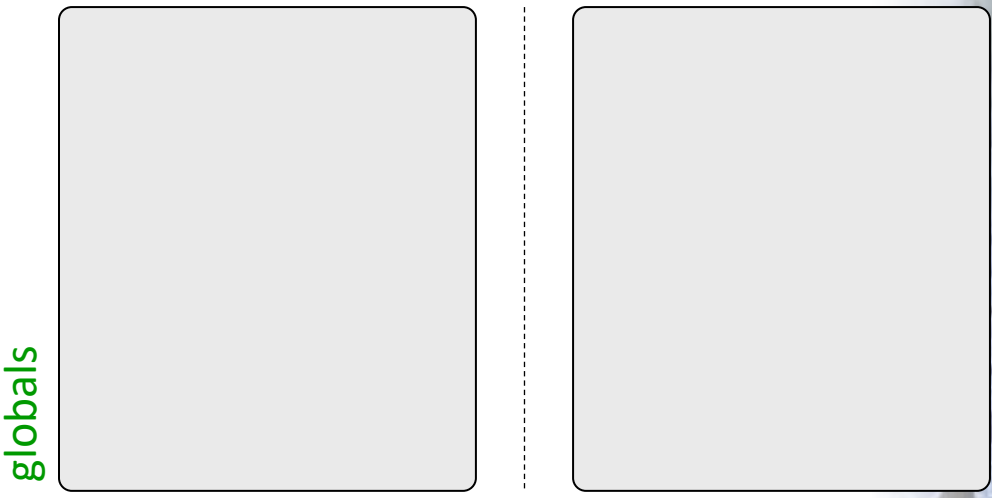


Functies zijn objecten

- dat betekent dat men
  - functies kan herdefiniëren
  - functies kan hernoemen (aliassen)

functies3.py

```
def omtrek(straal):
    import math
    return 2*math.pi*straal

contour = omtrek
print(contour(2.87))
```

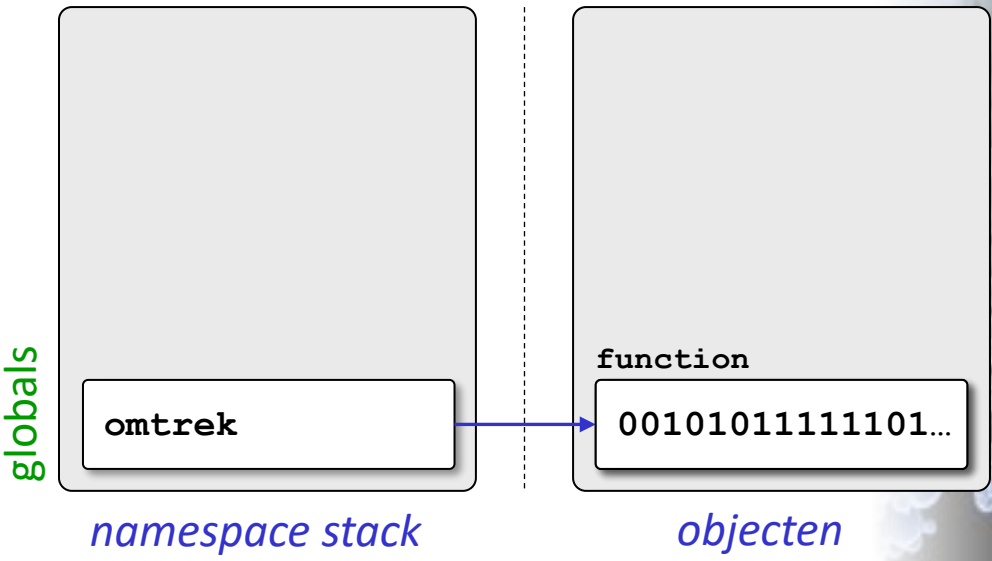



Functies zijn objecten

- dat betekent dat men
  - functies kan herdefiniëren
  - functies kan hernoemen (aliassen)

functies3.py

```
def omtrek(straal):
    import math
    return 2*math.pi*straal

contour = omtrek
print(contour(2.87))
```

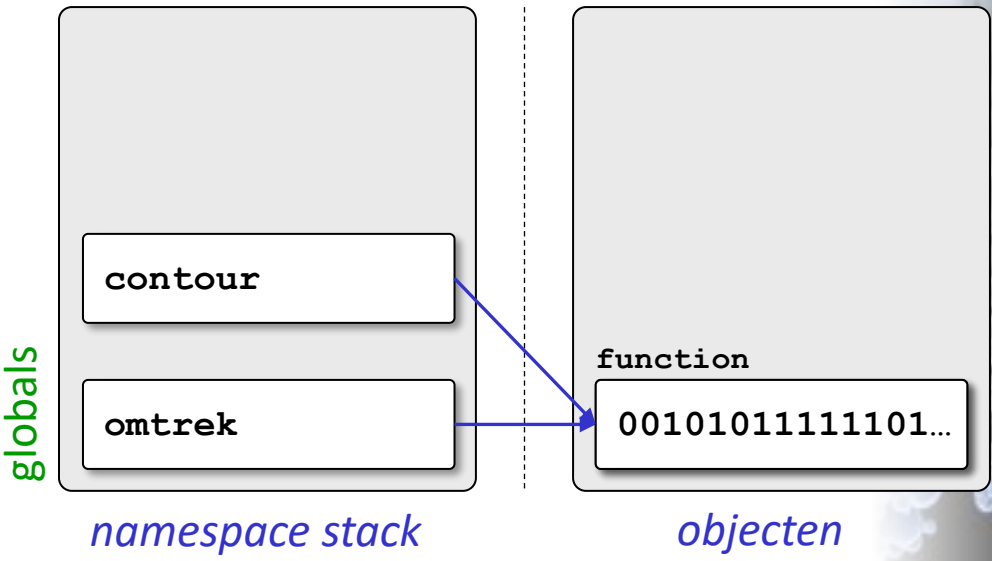


Functies zijn objecten

- dat betekent dat men
  - functies kan herdefiniëren
  - functies kan hernoemen (aliassen)

functies3.py

```
def omtrek(straal):
    import math
    return 2*math.pi*straal

contour = omtrek
print(contour(2.87))
```

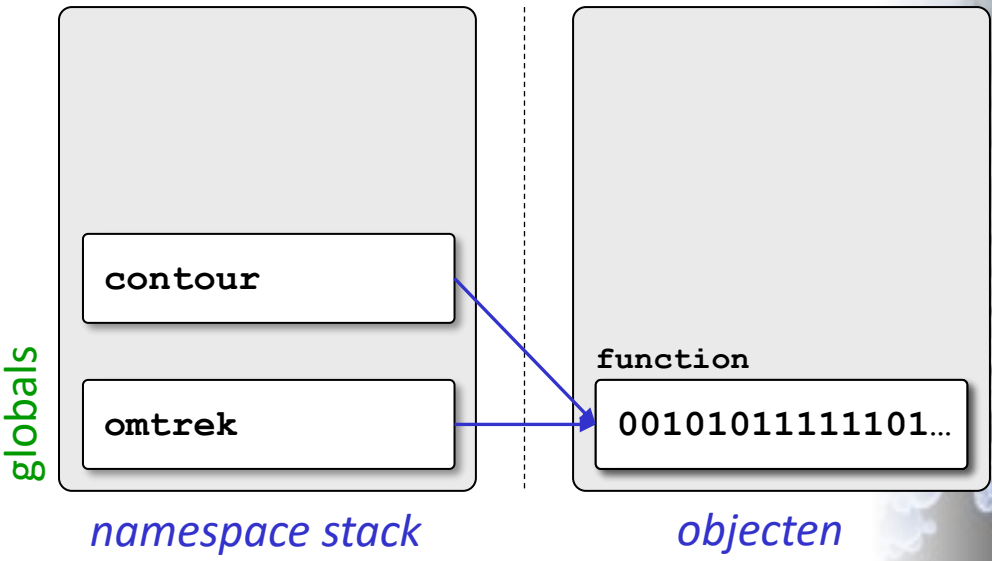


Functies zijn objecten

- dat betekent dat men
  - functies kan herdefiniëren
  - functies kan hernoemen (aliassen)

functies3.py

```
def omtrek(straal):
    import math
    return 2*math.pi*straal

contour = omtrek
print(contour(2.87))
```



Functies zijn objecten

- dat betekent dat men
  - functies kan herdefiniëren
  - functies kan hernoemen (aliassen)
  - functies als argument kan doorgeven aan functies

functies4.py

```
import math

def omtrek(straal):
    return 2 * math.pi * straal

def oppervlakte(straal):
    return math.pi * straal ** 2

def bereken(functie, waarde):
    return functie(waarde)

print(bereken(omtrek, 2.87))
print(bereken(oppervlakte, 2.87))
```

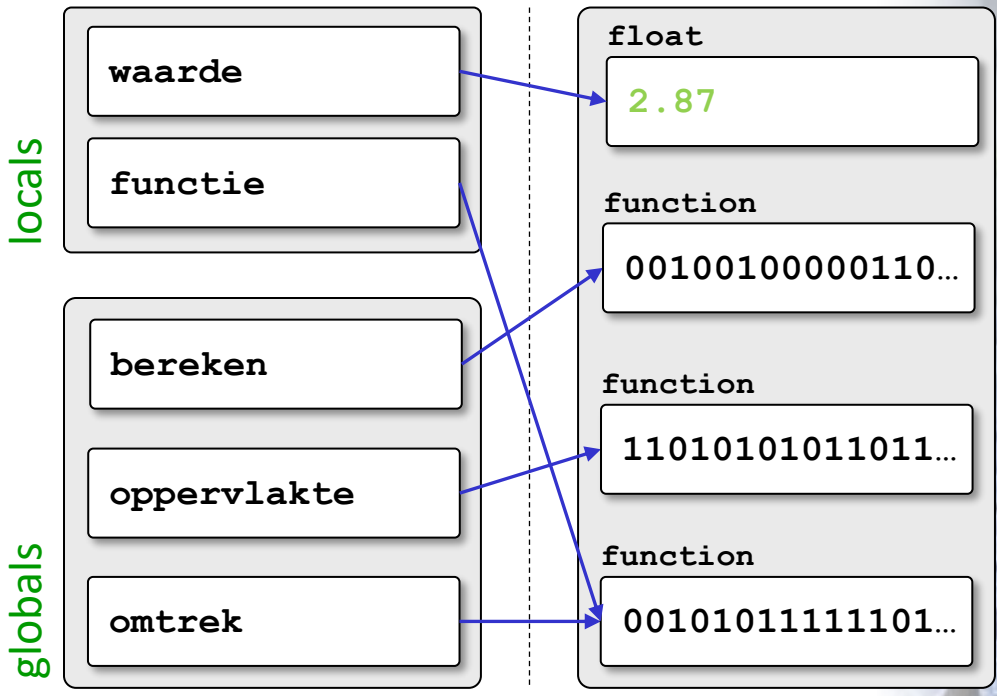
Functies zijn objecten

- dat betekent dat men
  - functies kan herdefiniëren
  - functies kan hernoemen (aliassen)
  - functies als argument kan doorgeven aan functies

functies4.py

```
import math

def omtrek(straal):
    return 2 * math.pi * straal

def oppervlakte(straal):
    return math.pi * straal ** 2

def bereken(functie, waarde):
    return functie(waarde)

print(bereken(omtrek, 2.87))
print(bereken(oppervlakte, 2.87))
```

18.0327418316



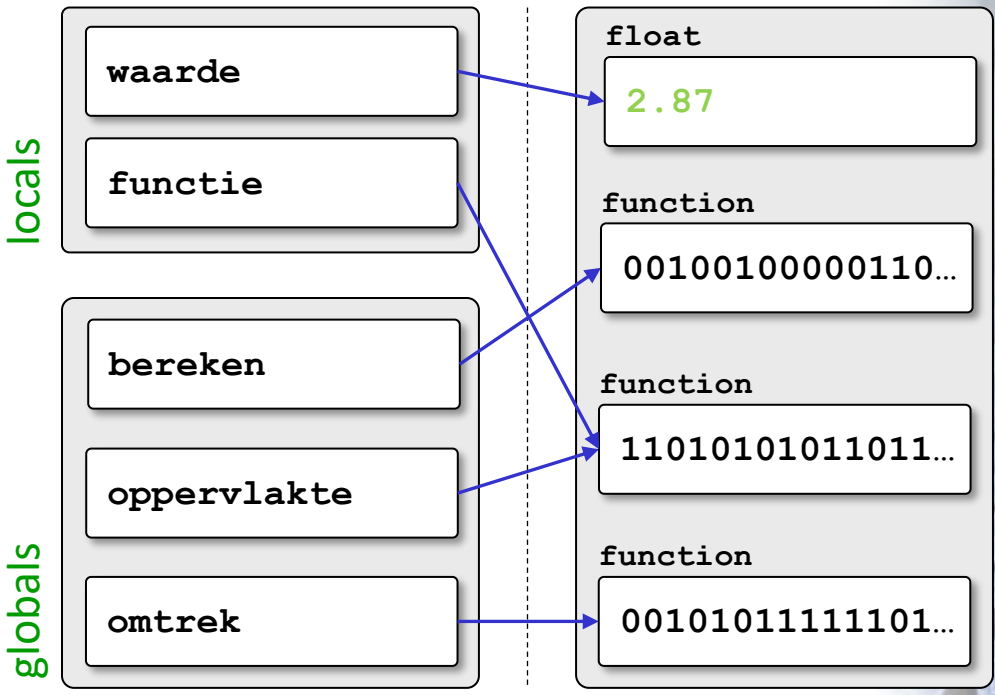
Functies zijn objecten

- dat betekent dat men
  - functies kan herdefiniëren
  - functies kan hernoemen (aliassen)
  - functies als argument kan doorgeven aan functies

functies4.py

```
import math

def omtrek(straal):
    return 2 * math.pi * straal

def oppervlakte(straal):
    return math.pi * straal ** 2

def bereken(functie, waarde):
    return functie(waarde)

print(bereken(omtrek, 2.87))
print(bereken(oppervlakte, 2.87))
```

```
18.0327418316
25.8769845284
```

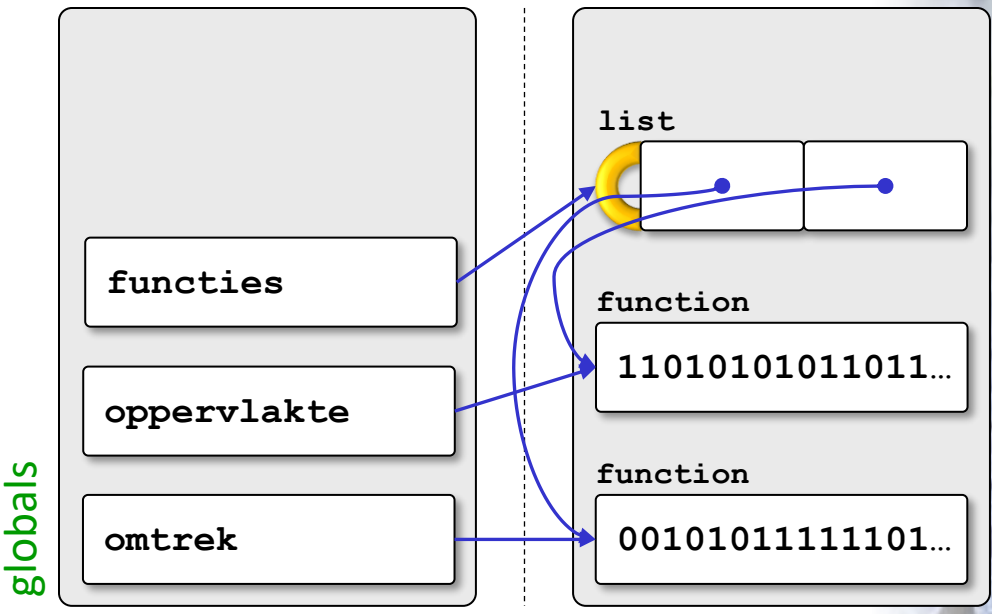


Functies zijn objecten

- dat betekent dat men
  - functies kan herdefiniëren
  - functies kan hernoemen (aliassen)
  - functies als argument kan doorgeven aan functies
  - functies in lijsten kan opslaan

functies5.py

```
import math

def omtrek(straal):
    return 2 * math.pi * straal

def oppervlakte(straal):
    return math.pi * straal ** 2

functies = [omtrek, oppervlakte]
for functie in functies:
    print(functie(2.87))
```



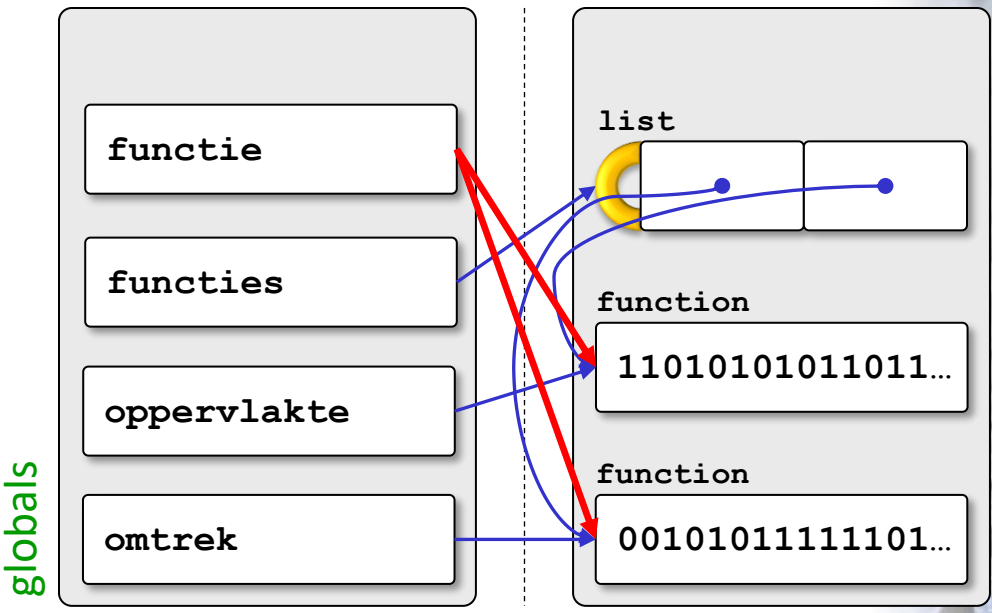
Functies zijn objecten

- dat betekent dat men
  - functies kan herdefiniëren
  - functies kan hernoemen (aliassen)
  - functies als argument kan doorgeven aan functies
  - functies in lijsten kan opslaan

functies5.py

```
import math

def omtrek(straal):
    return 2 * math.pi * straal

def oppervlakte(straal):
    return math.pi * straal ** 2

functies = [omtrek, oppervlakte]
for functie in functies:
    print(functie(2.87))
```

```
18.0327418316
25.8769845284
```



Functies zijn objecten

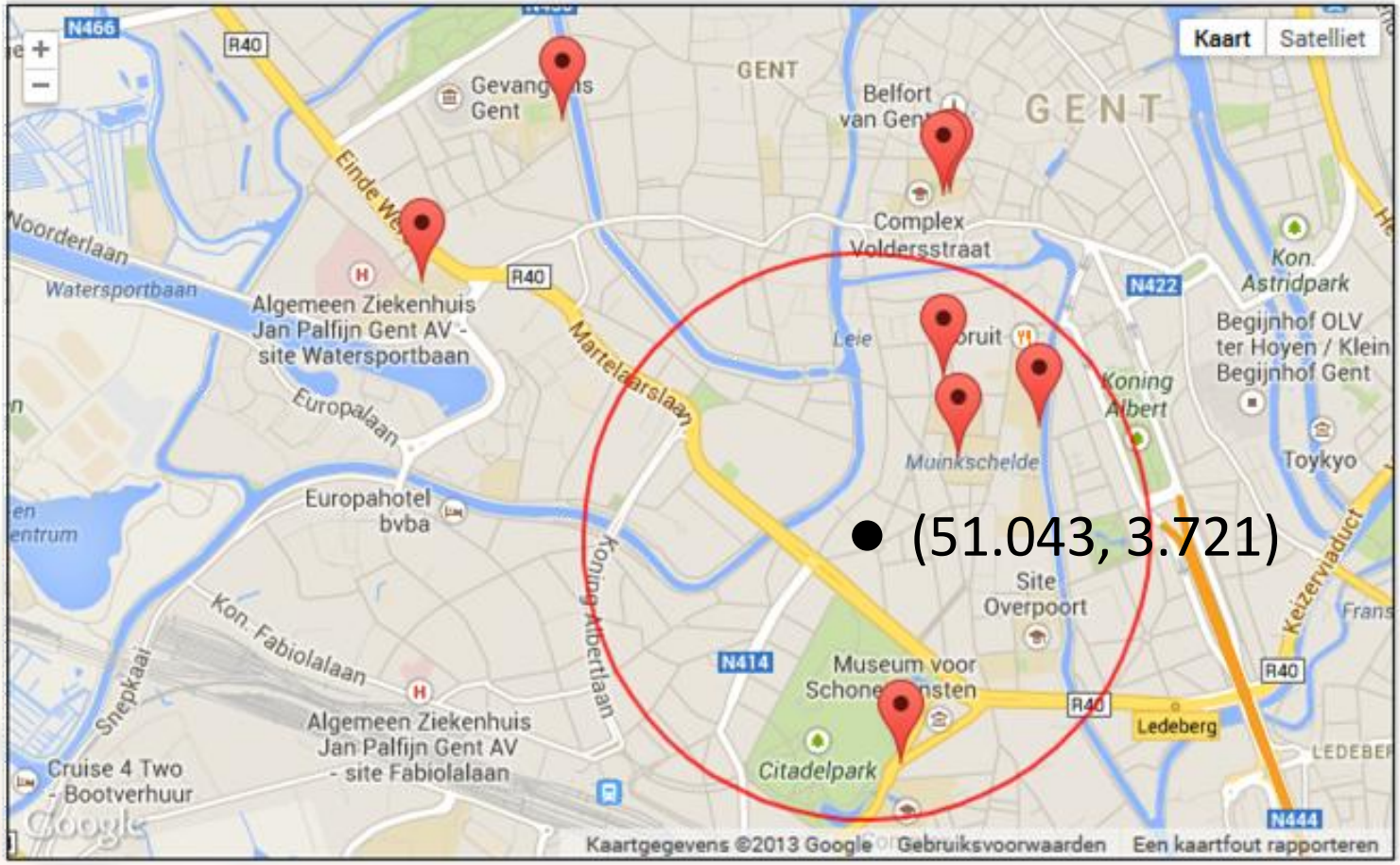
- functie-object heeft ook eigenschappen en methoden

```
>>> def omtrek(straal):
...     """Berekent omtrek van cirkel met gegeven straal."""
...     return 2 * straal * math.pi
...
>>> dir(omtrek)
['__annotations__', '__call__', '__class__', '__closure__', '__code__', '__defaults__',
'__delattr__', '__dict__', '__doc__', '__eq__', '__format__', '__ge__', '__get__',
'__getattribute__', '__globals__', '__gt__', '__hash__', '__init__', '__kwdefaults__',
'__le__', '__lt__', '__module__', '__name__', '__ne__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__']
>>> omtrek.__name__
'omtrek'
>>> omtrek.__doc__
'Berekent omtrek van cirkel met gegeven straal.'
>>> help(omtrek)
Help on function omtrek in module __main__:

omtrek(straal)
    Bereken omtrek van cirkel met gegeven straal.
```

docstring

UNIVERSITEIT GENT

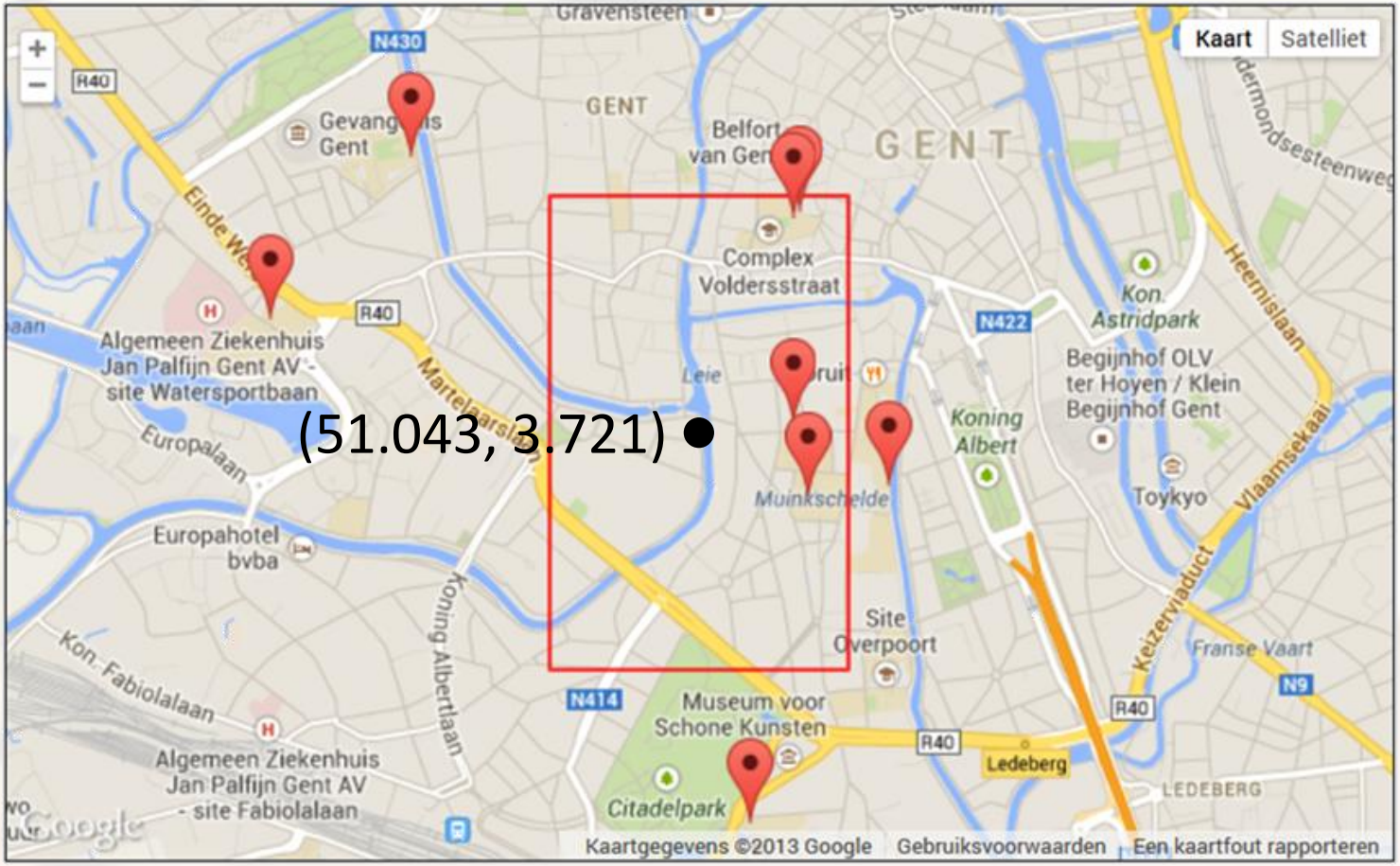
Locatiefilter



- (51.043, 3.721)

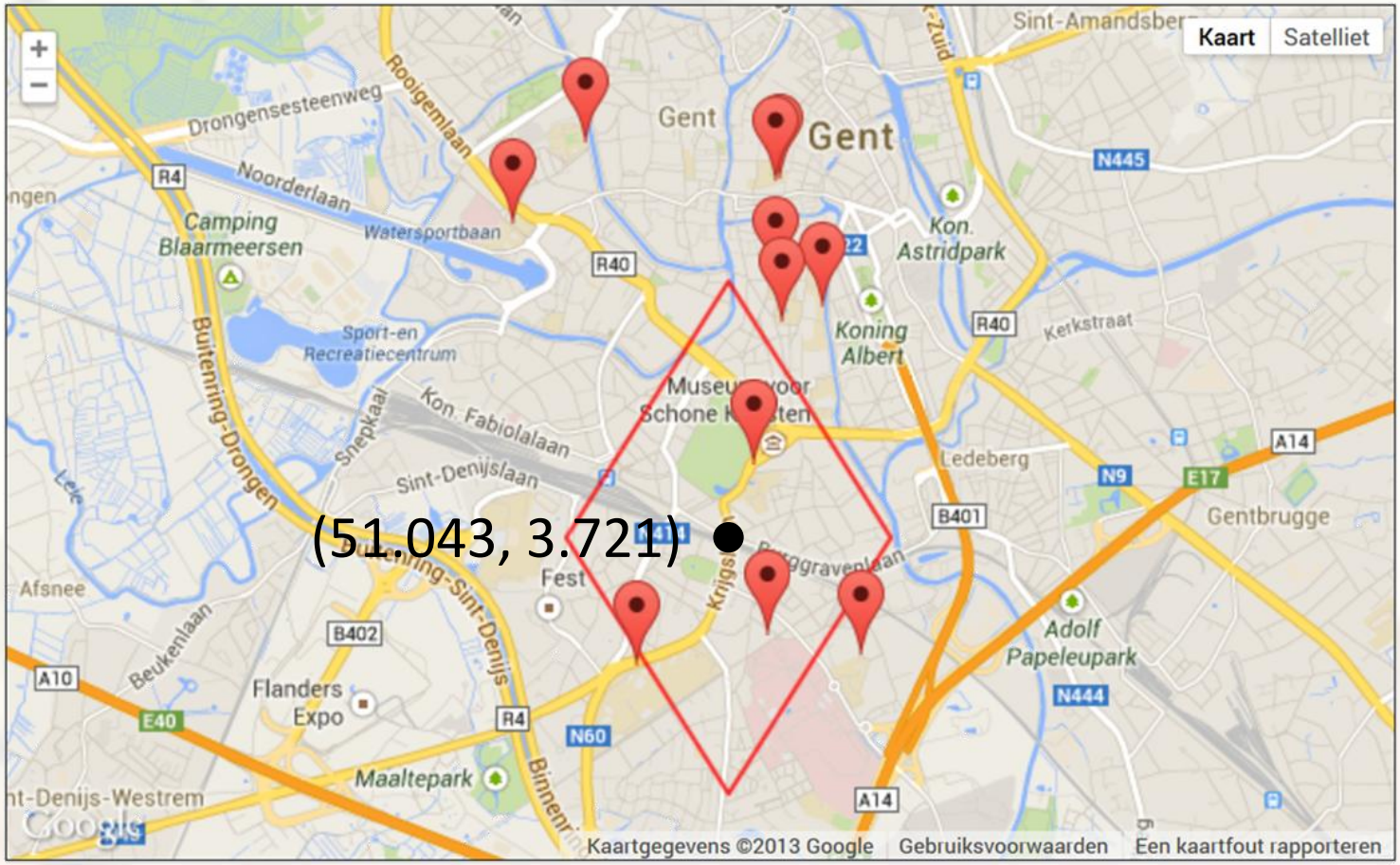
Euclidische afstand $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

Locatiefilter



schaakbord-afstand $d = max(|x_1 - x_2|, |y_1 - y_2|)$

Locatiefilter



Manhattan-afstand $$d = |x_1 - x_2| + |y_1 - y_2|$$

Huiswerk (volgende les)

- voorbeeldopgaven volgende les (reeks 7)
  - Maanfasen

Huiswerk (werkcolleges)

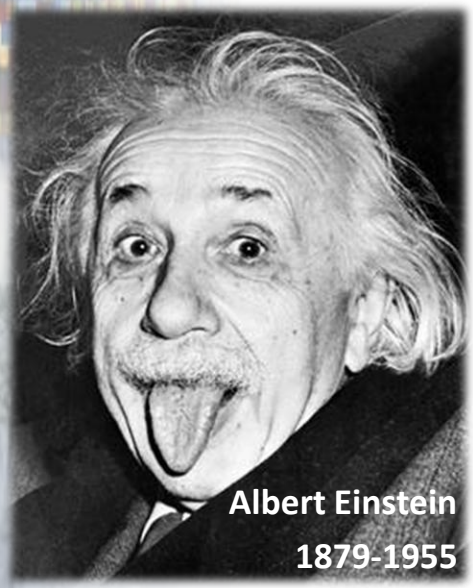
- maak opgavereeks 7 (geavanceerde functies)
  - gebruik doctests
  - vanaf volgende week terug wekelijks nieuwe opgelegde oefeningen waarin tijdens werkcolleges kan gewerkt worden
  - maak eerste opgelegde oefeningen bij voorkeur voorafgaand aan werkcollege
  - plagiaatdetectie !!!

UNIVI
GENT

Vragen of opmerkingen?

UNIVERSITEIT GENT

The sky is the limit…



"Everything should be simple:
as simple as possible,
but no simpler."

— A. Einstein

UNIVERSITEIT GENT