Universiteit Gent
Faculteit Wetenschappen

Programmeren in Python

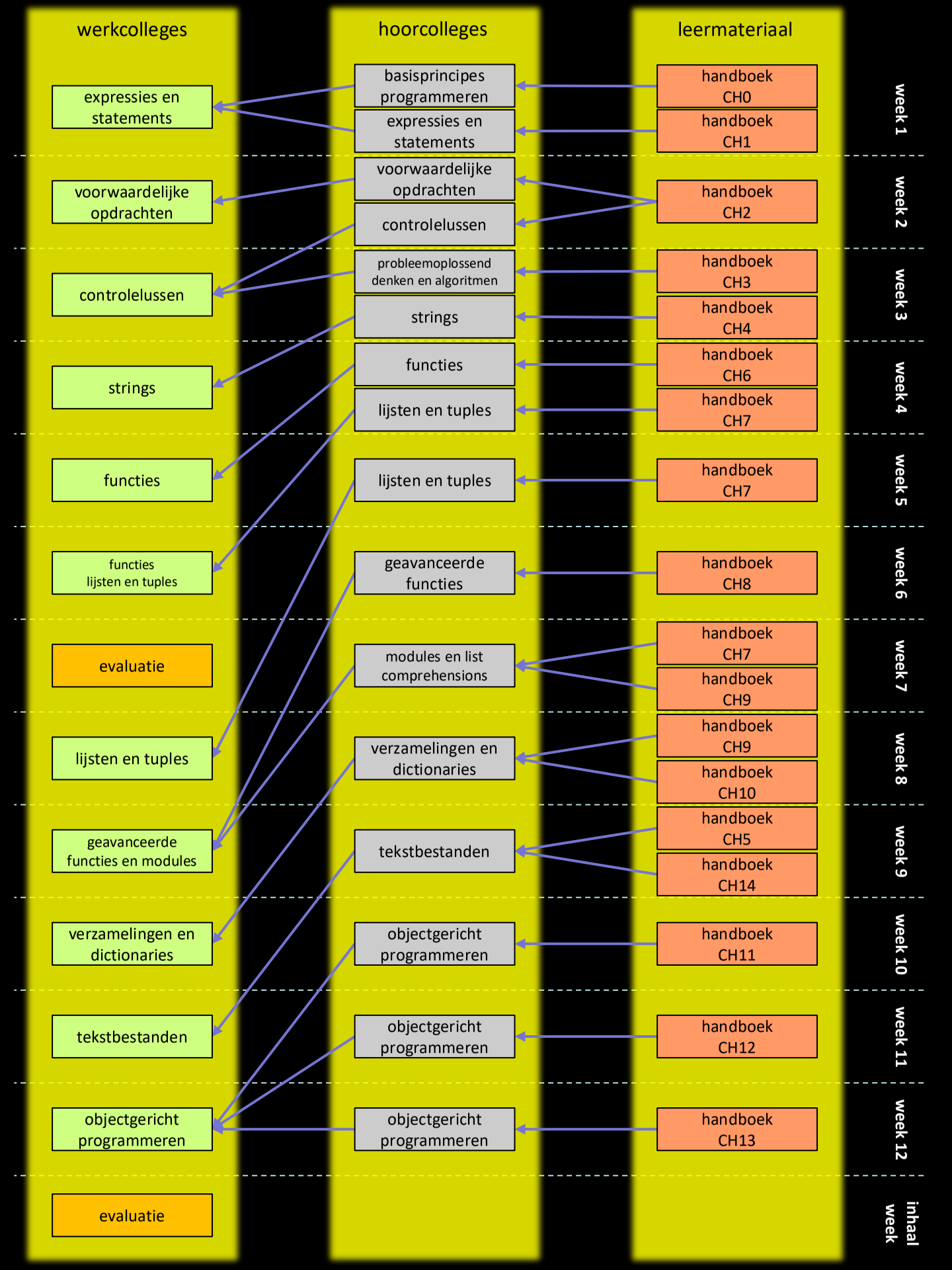
geneste lijsten

prof. dr. Peter Dawyndt

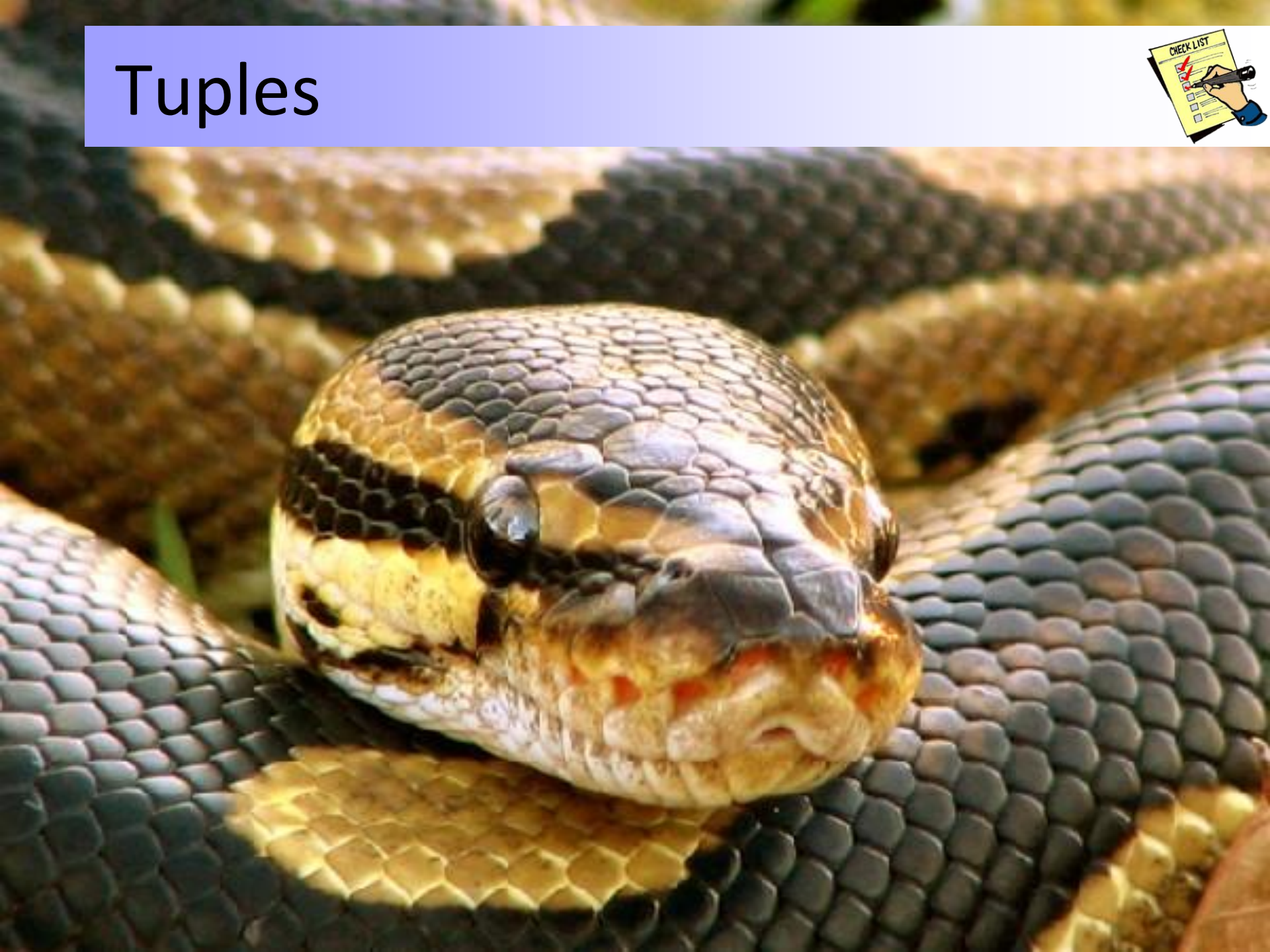
peter.dawyndt@ugent.be

@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

Tuples



Tuples

- **tuple**: *onveranderlijke* reeks van objecten
  - *immutable* variant van lijsten (geen wijzigingen na aanmaken)
  - geschreven met ronde ipv vierkante haakjes
    - `()` stelt lege tuple voor
    - tuple met één element → extra komma: `(55,)`

```
>>> gassen = ('He', 'Ne', 'Ar', 'Kr')
>>> print(gassen)
('He', 'Ne', 'Ar', 'Kr')
>>> print(gassen[0], gassen[-1])
He Kr
>>> print(gassen[3:2])
()
>>> print(gassen[2:3])
('Ar',)
```

UNIVERSITEIT GENT

Tuples uitpakken

- **tuple**: *onveranderlijke* reeks van objecten
  - ronde haakjes rond tuples meestal niet strikt noodzakelijk
  - toekenning aan tuple gebeurt per element (*tuple unpacking*)
    - aantal elementen in linkerlid van toekenningsopdracht moet gelijk zijn aan aantal elementen in rechterlid

```
>>> gassen = 'He', 'Ne', 'Ar'
>>> print(gassen)
('He', 'Ne', 'Ar')
>>> (gas1, gas2, gas3) = gassen
>>> print(gas1, gas2)
He Ne
>>> gas1, gas2, gas3 = gassen
>>> gas1, gas2 = gassen
ValueError: too many values to unpack
```

UNIVERSITEIT GENT

Tuples uitpakken

- **tuple**: *onveranderlijke* reeks van objecten
  - verschillende handige toepassingen van *tuple unpacking*
    - meerwaardige toekenning
    - waarden omwisselen
    - functies die meerdere waarden lijken terug te geven

```
>>> gas1, gas2 = 'He', 'Ne'
>>> gas1, gas2 = gas2, gas1
>>> def meerwaardig(n): return n**2, n**3
...
>>> x2, x3 = meerwaardig(10)
>>> print(x2, x3)
100 1000
```

UNIVERSITEIT GENT

Tuples uitpakken

- **tuple**: *onveranderlijke* reeks van objecten
  - verschillende handige toepassingen van *tuple unpacking*
    - meerwaardige toekenning
    - waarden omwisselen
    - functies die meerdere waarden lijken terug te geven
    - meerdere indexen in for-lussen

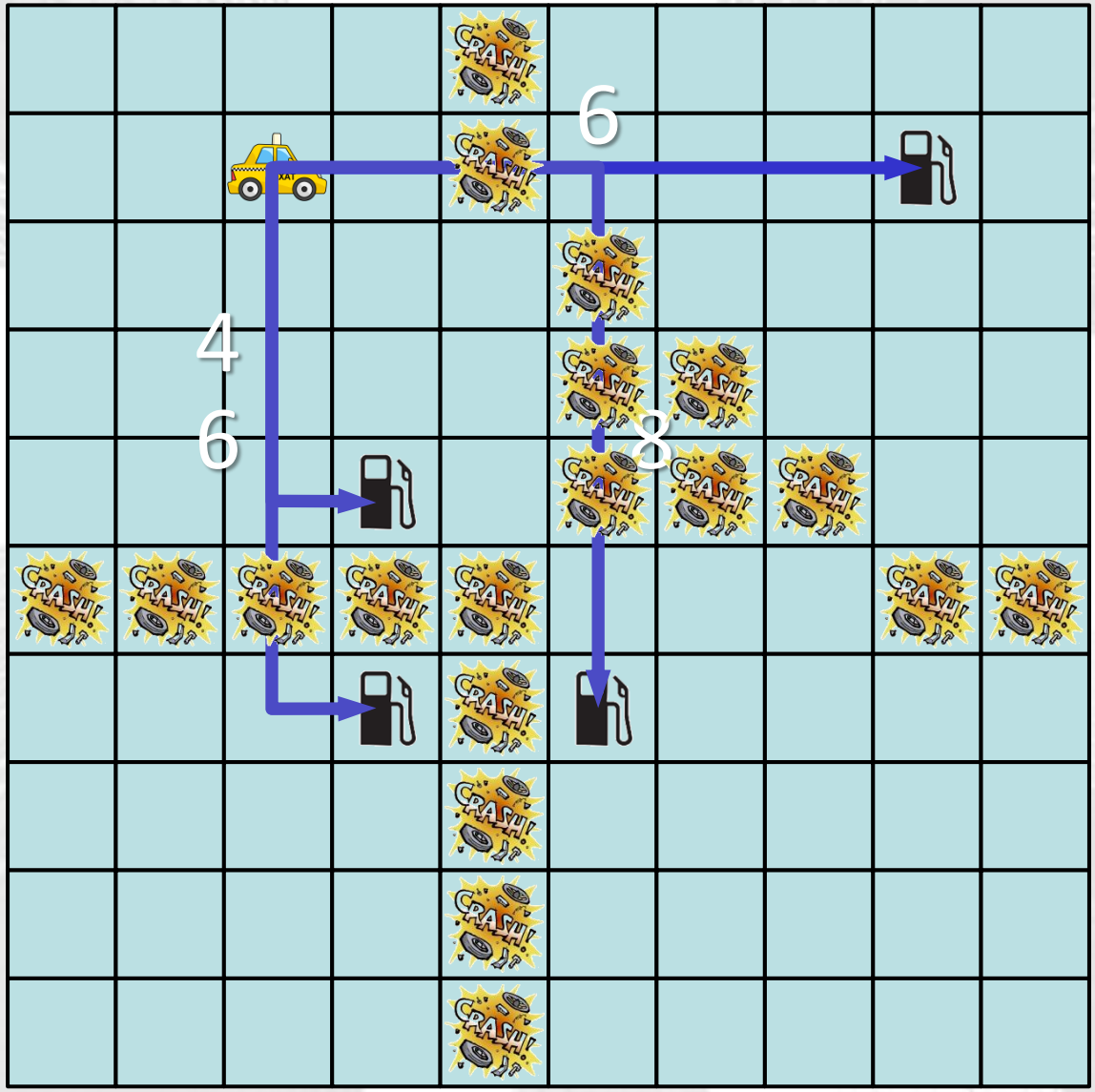
uitpakken.py

```
elementen = [
   ['H', 'waterstof', 1.008],
   ['He', 'helium', 4.003],
   ['Li', 'lithium', 6.941],
   ['Be', 'beryllium', 9.012]
]

for symbool, naam, massa in elementen:
    print(f'{naam} ({symbool}): {massa :.3f}')
```

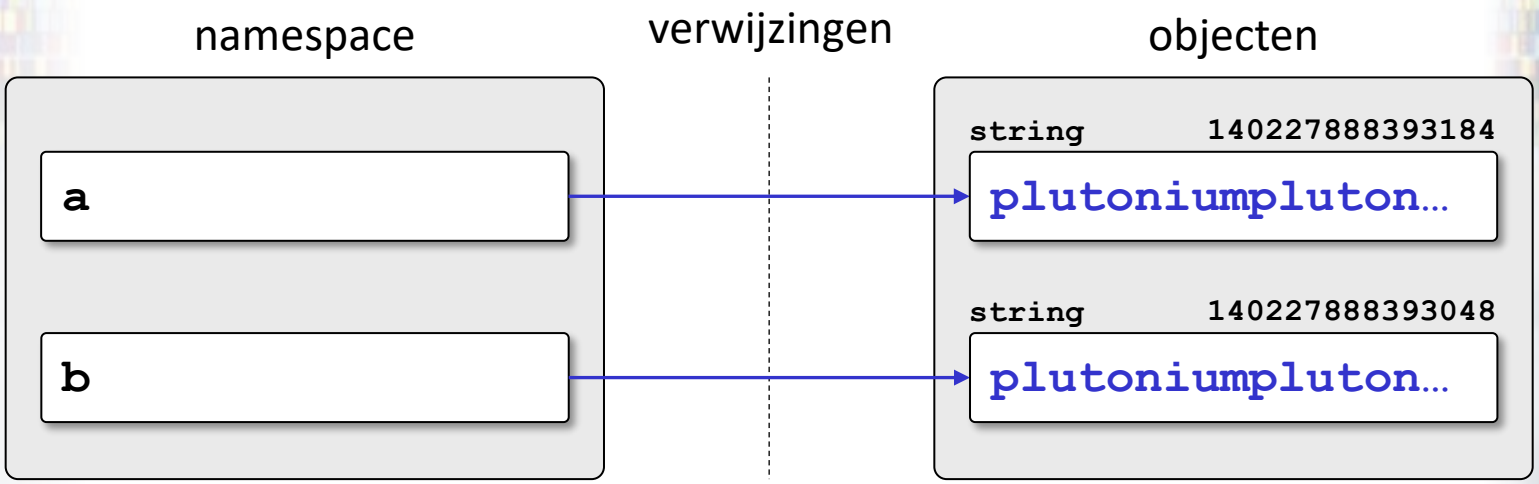
UNIVERSITEIT GENT

Manhattan



Aliassen

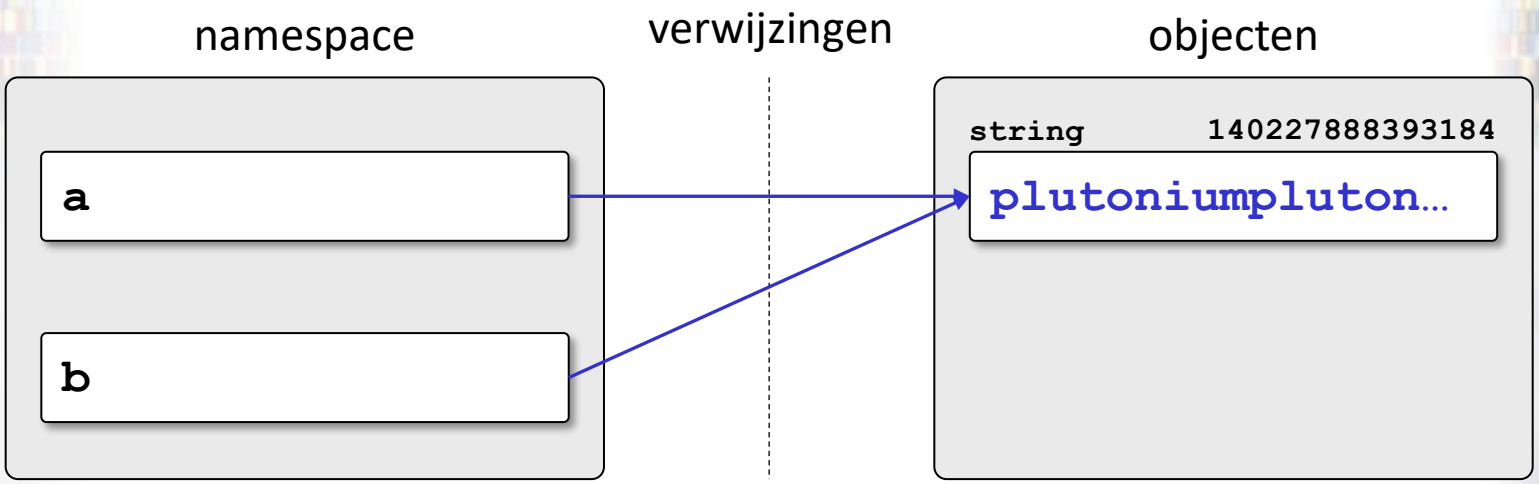
String aliassen




```
>>> a = 'plutonium' * 100
>>> b = 'plutonium' * 100
>>> a == b
True
>>> a is b
False
>>> id(a), id(b)
(140227888393184, 140227888393048)
```

UNIVERSITEIT GENT

String aliassen

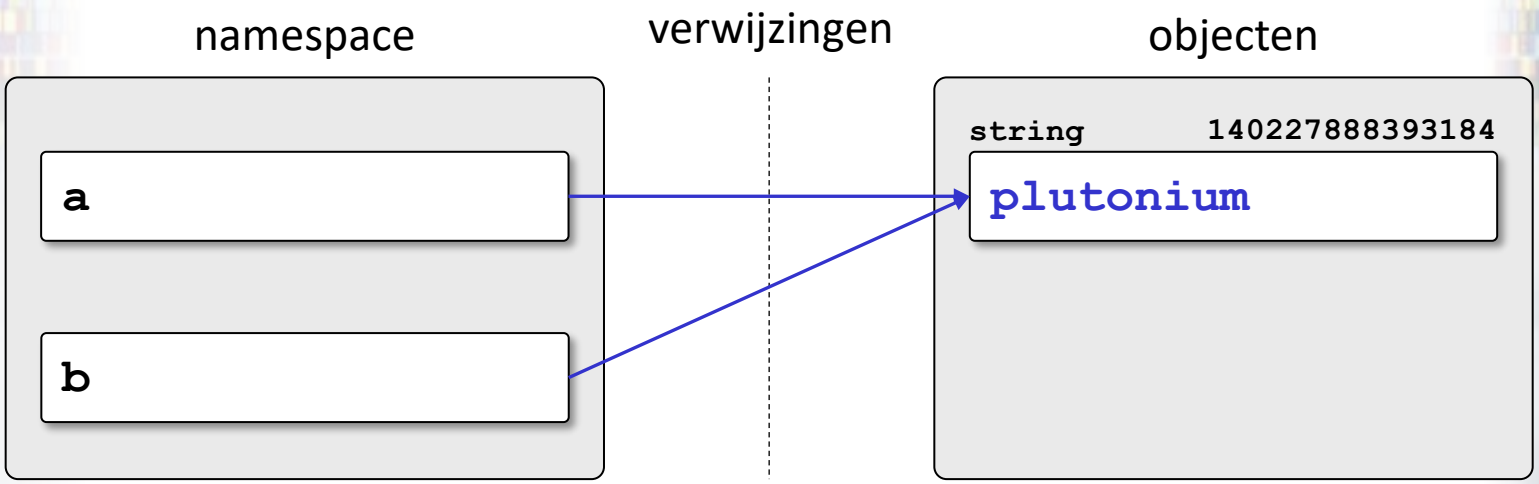



```
>>> a = 'plutonium' * 100
>>> b = a
>>> a == b
True
>>> a is b
True
>>> id(a), id(b)
(140227888393184, 140227888393184)
```

"a en b verwijzen naar hetzelfde object: het zijn aliassen."

UNIVERSITEIT GENT

String aliassen

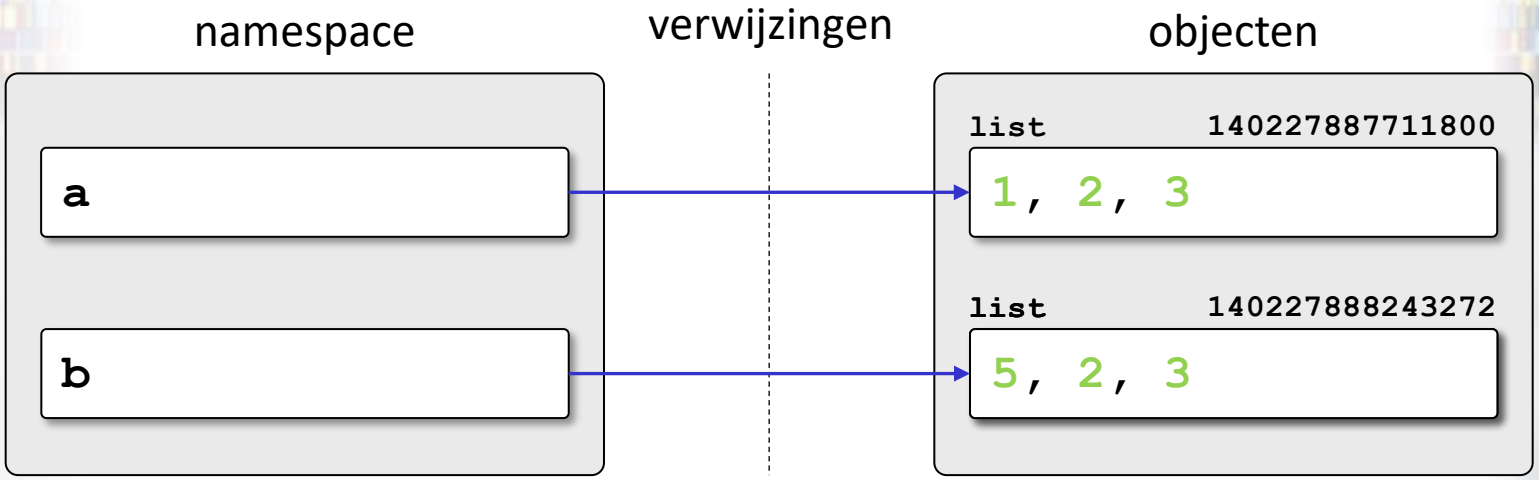



```
>>> a = 'plutonium'
>>> b = 'plutonium'
>>> a == b
True
>>> a is b
True
>>> id(a), id(b)
(140227888393184, 140227888393184)
```

"Python kan optimalisatie doorvoeren omdat strings onveranderlijk zijn."

UNIVERSITEIT GENT

Lijst aliassen

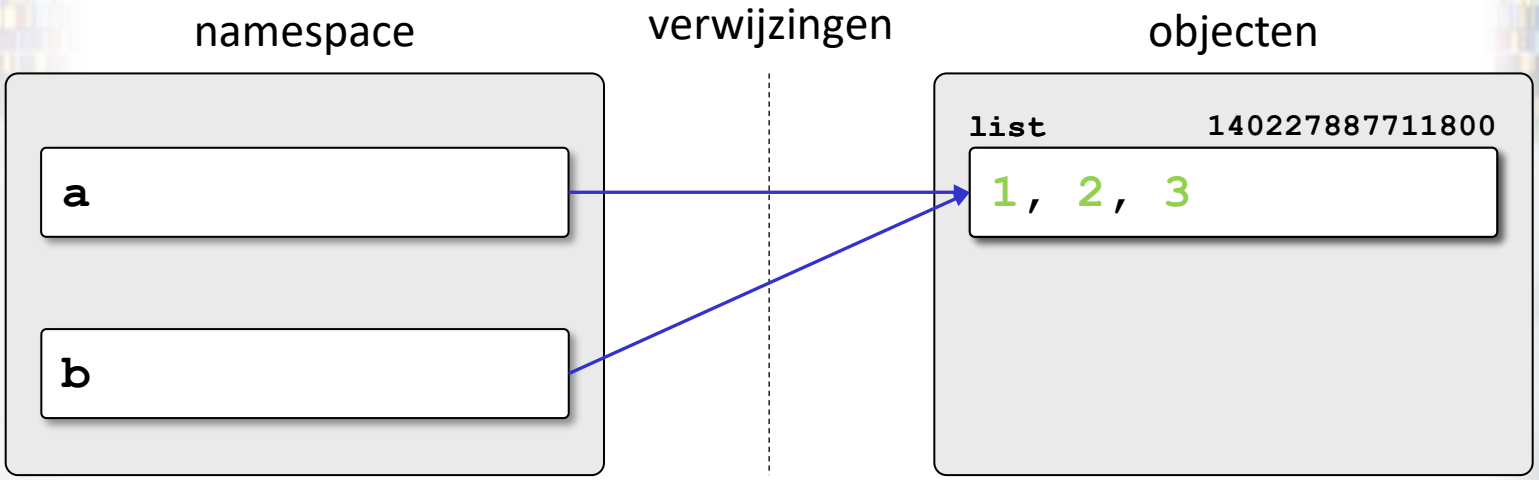



```
>>> a = [1, 2, 3]
>>> b = [1, 2, 3]
>>> a is b
False
>>> b[0] = 5
>>> a
[1, 2, 3]
```

"Python voert optimalisatie nooit door omdat lijsten veranderlijk zijn."

UNIVERSITEIT GENT

Lijst aliassen

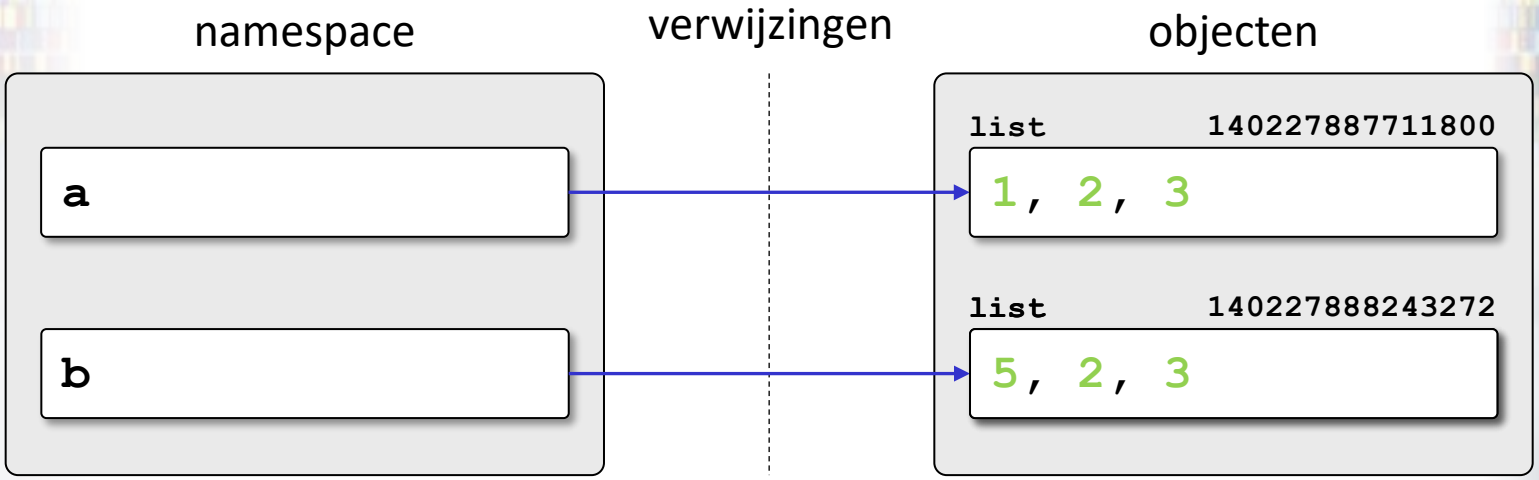



```
>>> a = [1, 2, 3]
>>> b = a
>>> a is b
True
>>> b[0] = 5
>>> a
[5, 2, 3]
```

"a en b verwijzen naar hetzelfde object: het zijn aliassen."

UNIVERSITEIT GENT

Lijsten kloneren

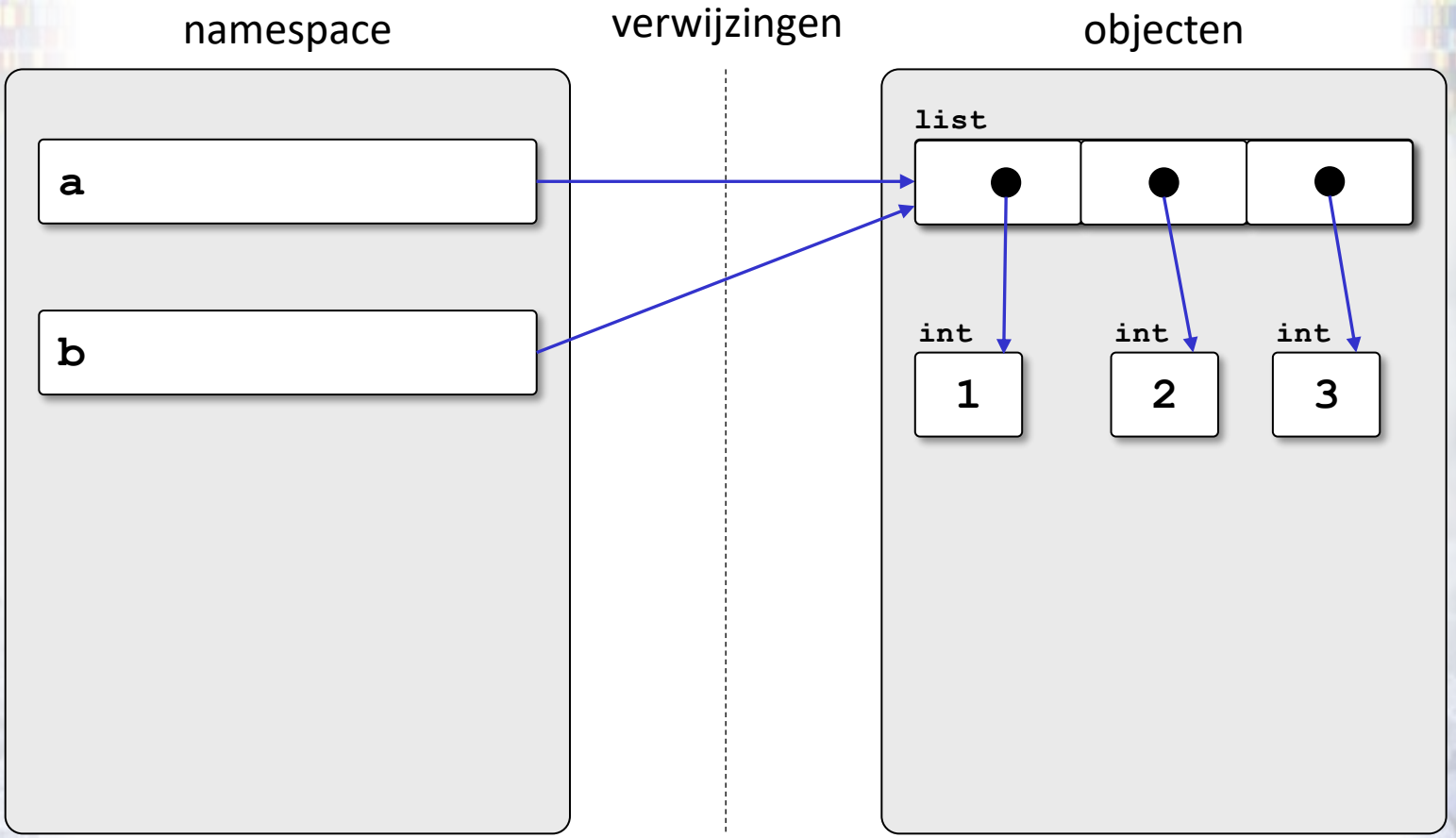



```
>>> a = [1, 2, 3]
>>> b = a[:]
>>> a is b
False
>>> b[0] = 5
>>> a
[1, 2, 3]
```

"Slicing maakt telkens een nieuwe lijst aan voor de subsequentie."

UNIVERSITEIT GENT

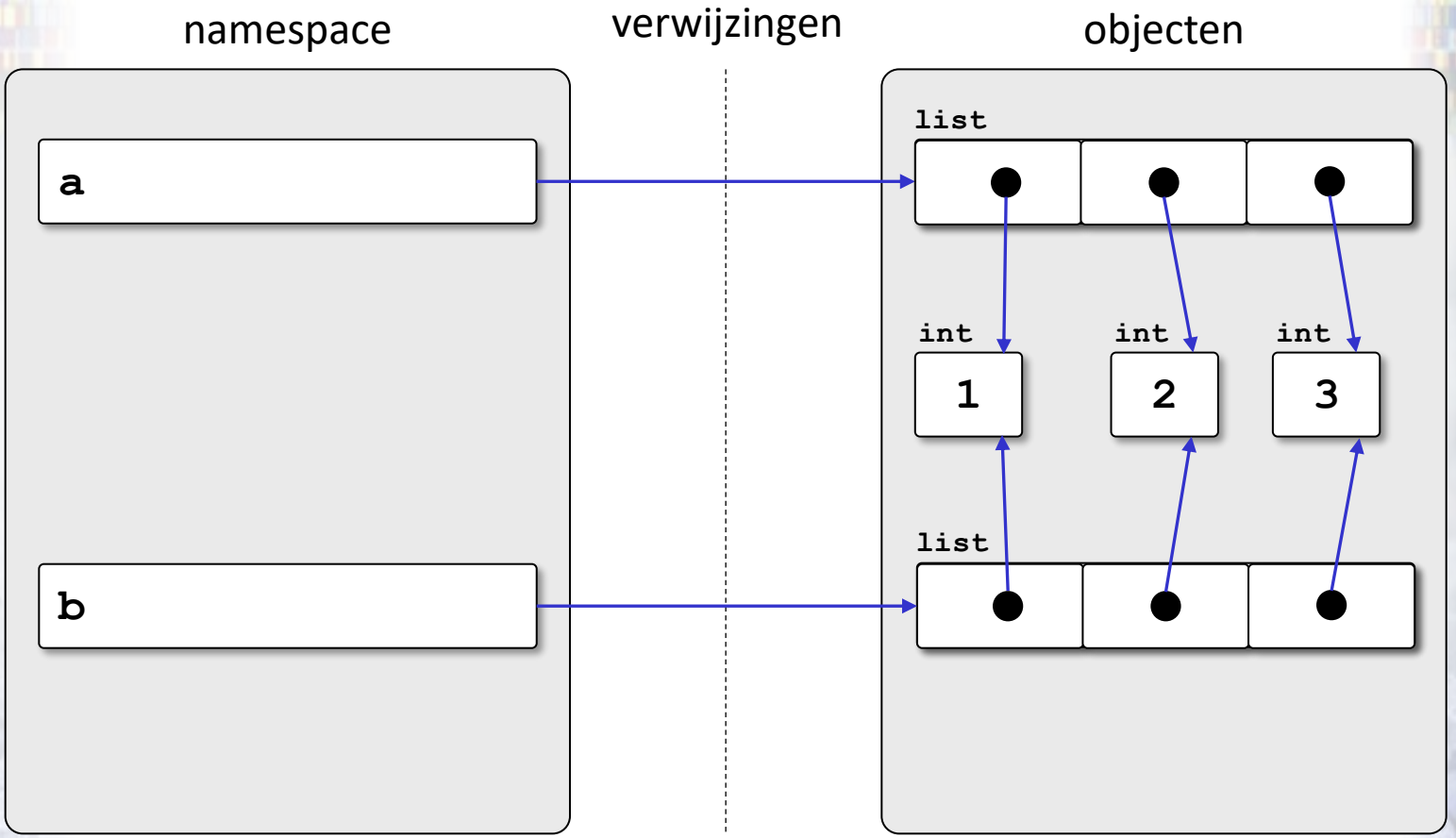
Alias vs copy vs deepcopy




```
>>> a = [1, 2, 3]
>>> b = a
```

UNIVERSITEIT GENT

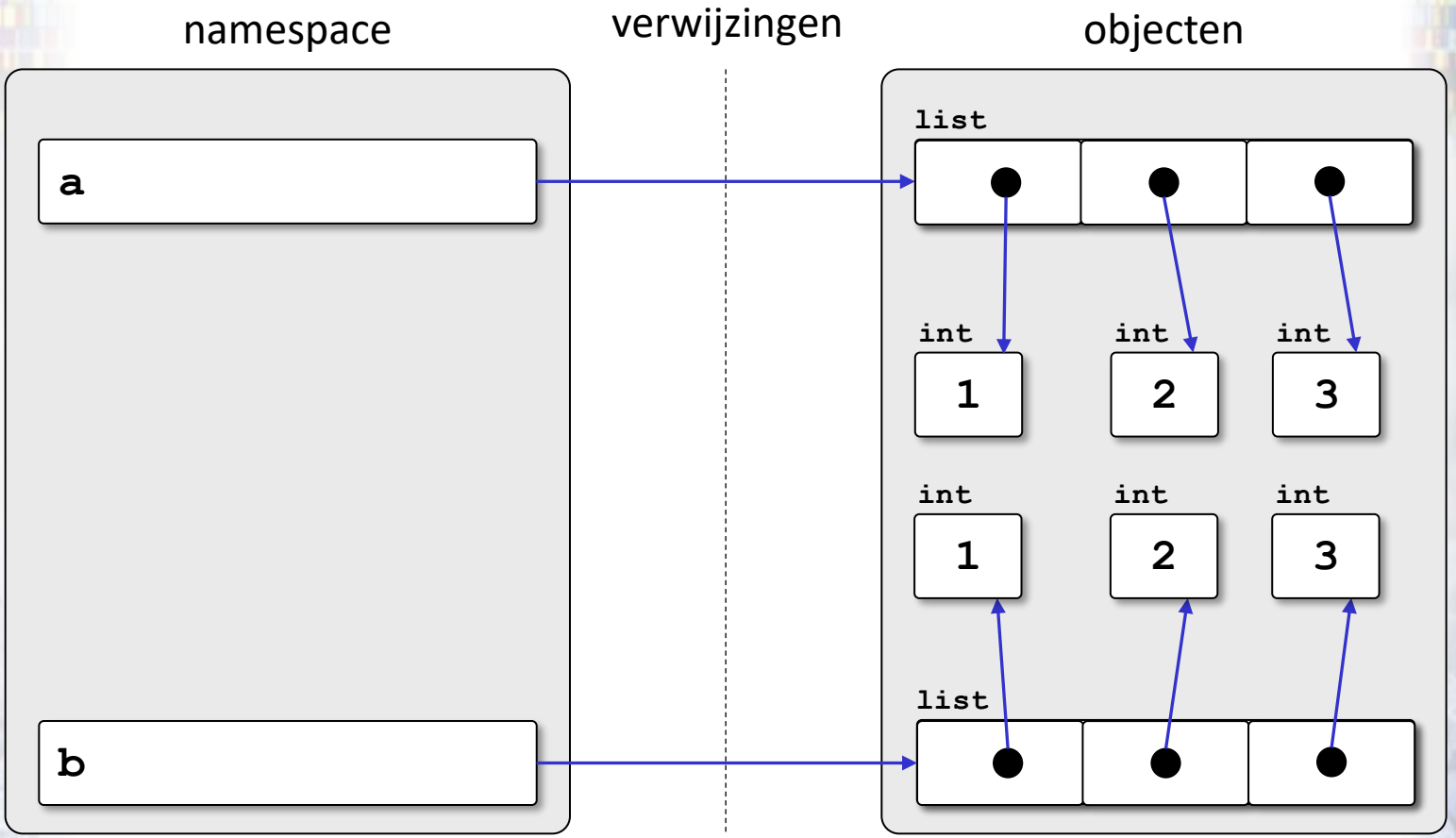
Alias vs **copy** vs deepcopy




```
>>> a = [1, 2, 3]
>>> b = a[:]
```

UNIVERSITEIT GENT

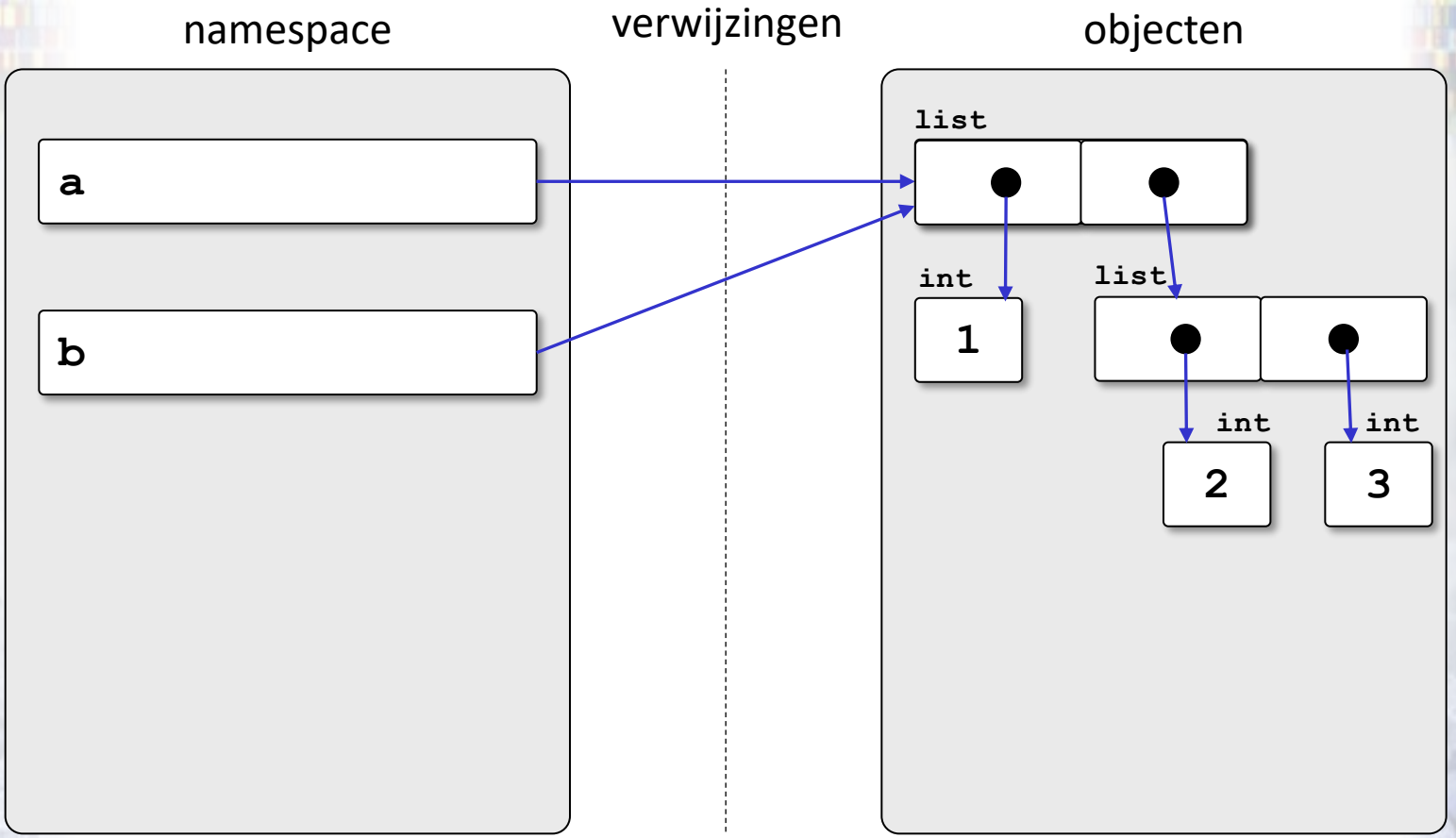
Alias vs copy vs **deepcopy**




```
>>> a = [1, 2, 3]
>>> b = copy.deepcopy(a)
```

UNIVERSITEIT GENT

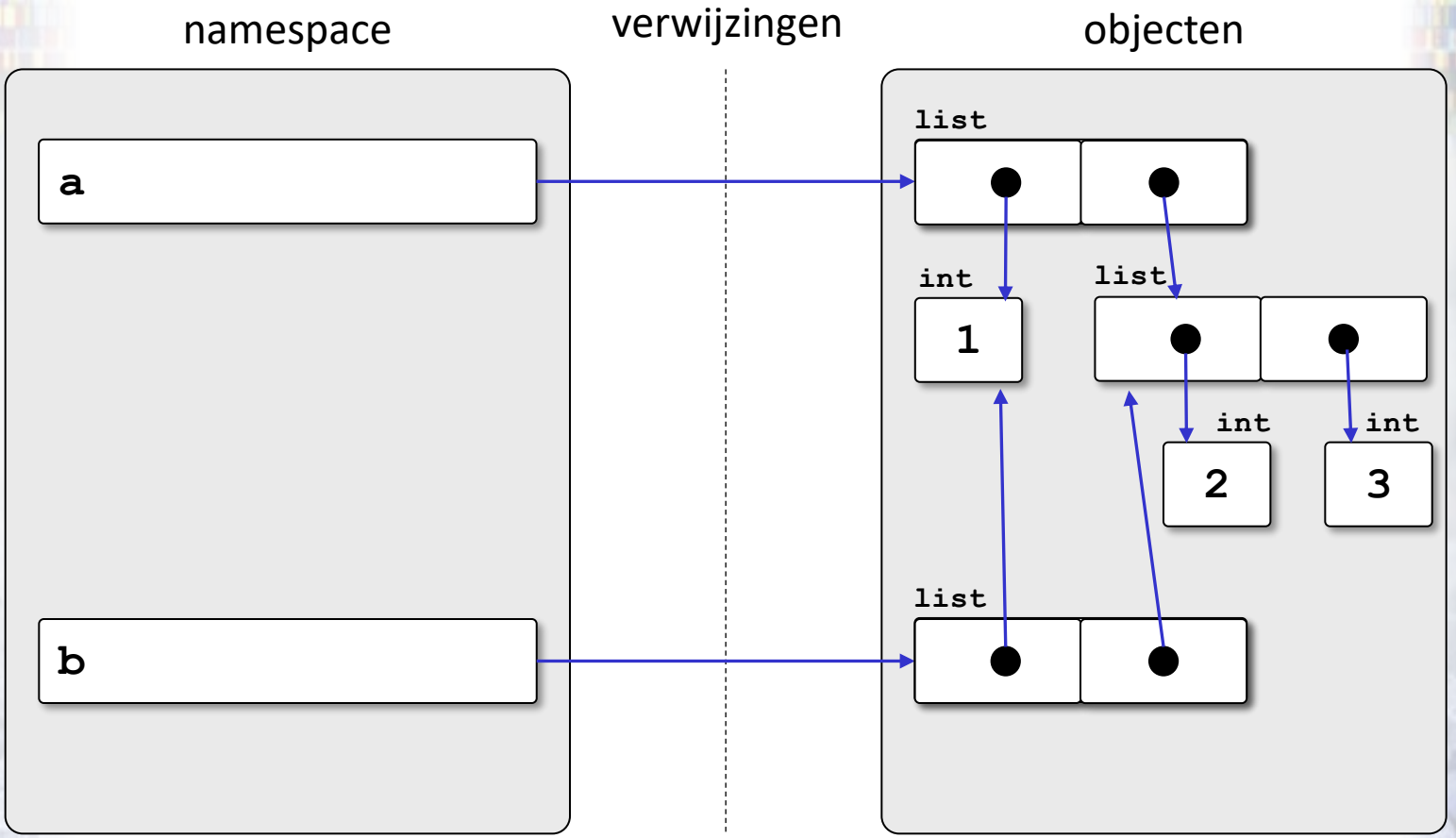
Alias vs copy vs deepcopy




```
>>> a = [1, [2, 3]]
>>> b = a
```

UNIVERSITEIT GENT

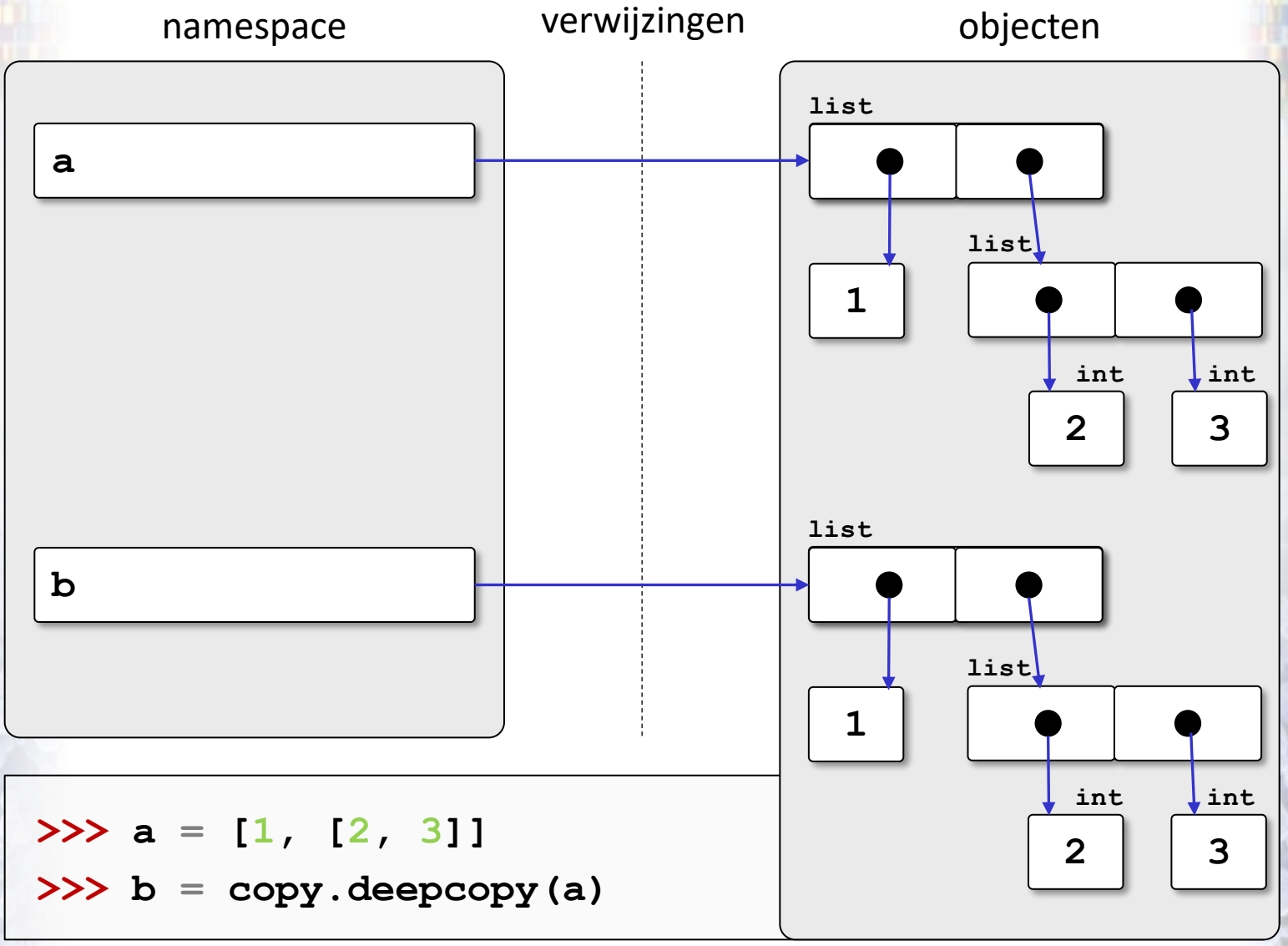
Alias vs **copy** vs deepcopy




```
>>> a = [1, [2, 3]]
>>> b = a[:]
```

UNIVERSITEIT GENT

Alias vs copy vs **deepcopy**




```
>>> a = [1, [2, 3]]
>>> b = copy.deepcopy(a)
```

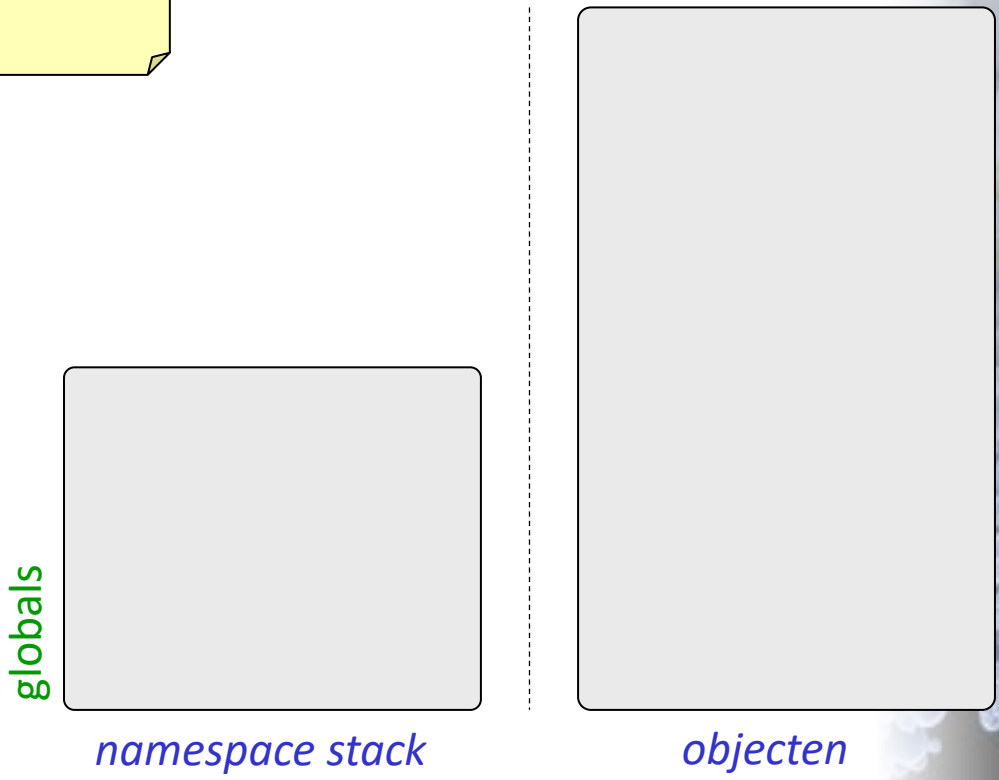
UNIVERSITEIT GENT

Lijsten doorgeven als argument

verdubbel.py

```
def verdubbel(lijst):
    for index, waarde in enumerate(lijst):
        lijst[index] = 2 * waarde

dingen = [2, 5, 'spam', 9.5]
verdubbel(dingen)
print(dingen)
```



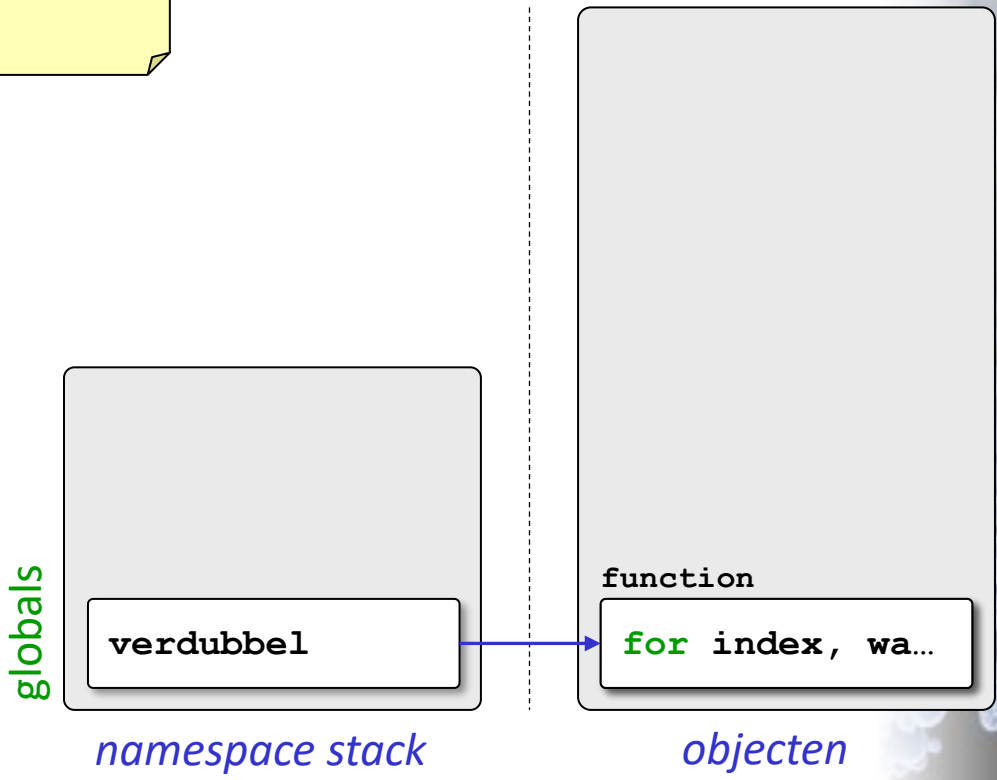
UNIVERSITEIT GENT

Lijsten doorgeven als argument

verdubbel.py

```
def verdubbel(lijst):
    for index, waarde in enumerate(lijst):
        lijst[index] = 2 * waarde

dingen = [2, 5, 'spam', 9.5]
verdubbel(dingen)
print(dingen)
```



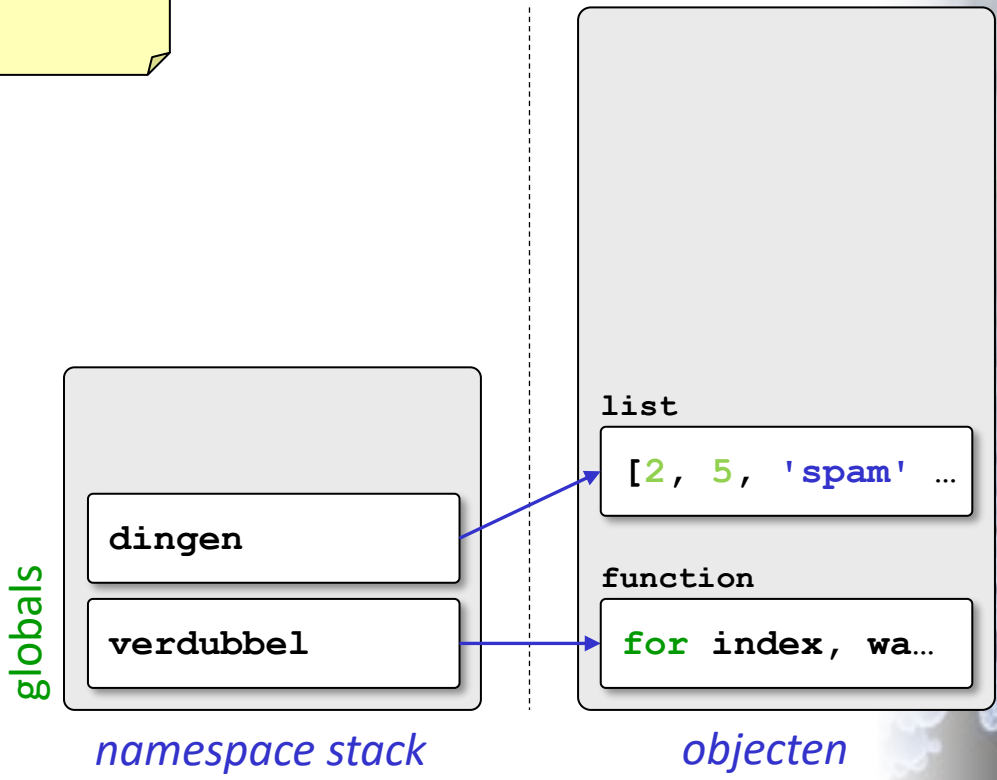
UNIVERSITEIT GENT

Lijsten doorgeven als argument

verdubbel.py

```
def verdubbel(lijst):
    for index, waarde in enumerate(lijst):
        lijst[index] = 2 * waarde

dingen = [2, 5, 'spam', 9.5]
verdubbel(dingen)
print(dingen)
```



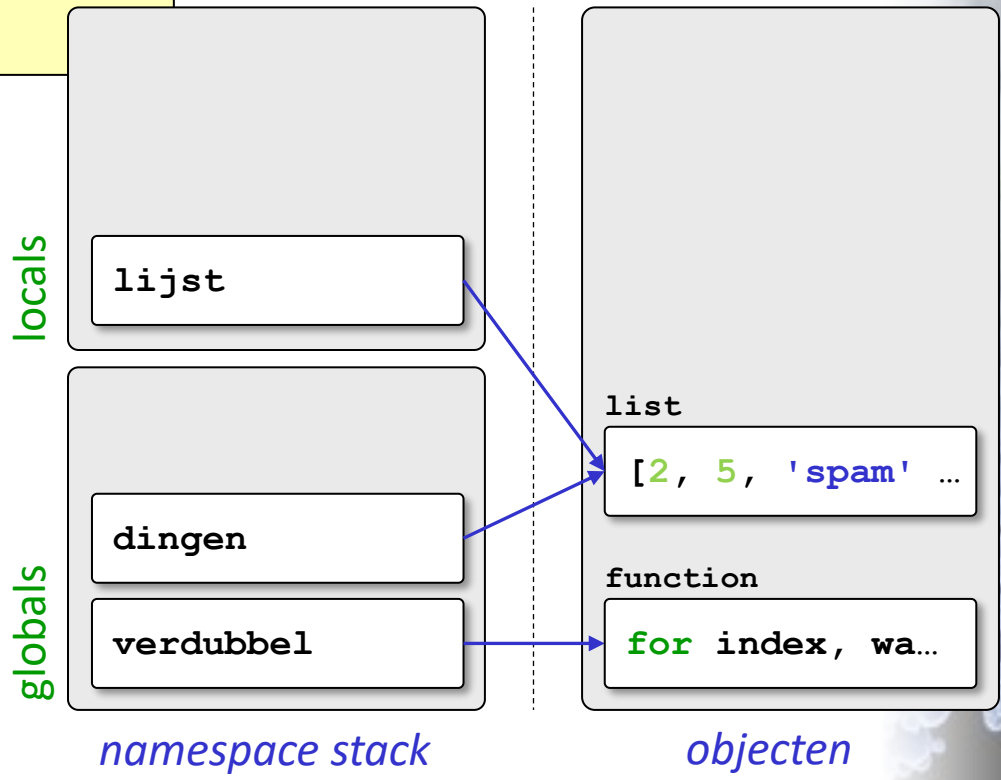
UNIVERSITEIT GENT

Lijsten doorgeven als argument

verdubbel.py

```
def verdubbel(lijst):
    for index, waarde in enumerate(lijst):
        lijst[index] = 2 * waarde

dingen = [2, 5, 'spam', 9.5]
verdubbel(dingen)
print(dingen)
```



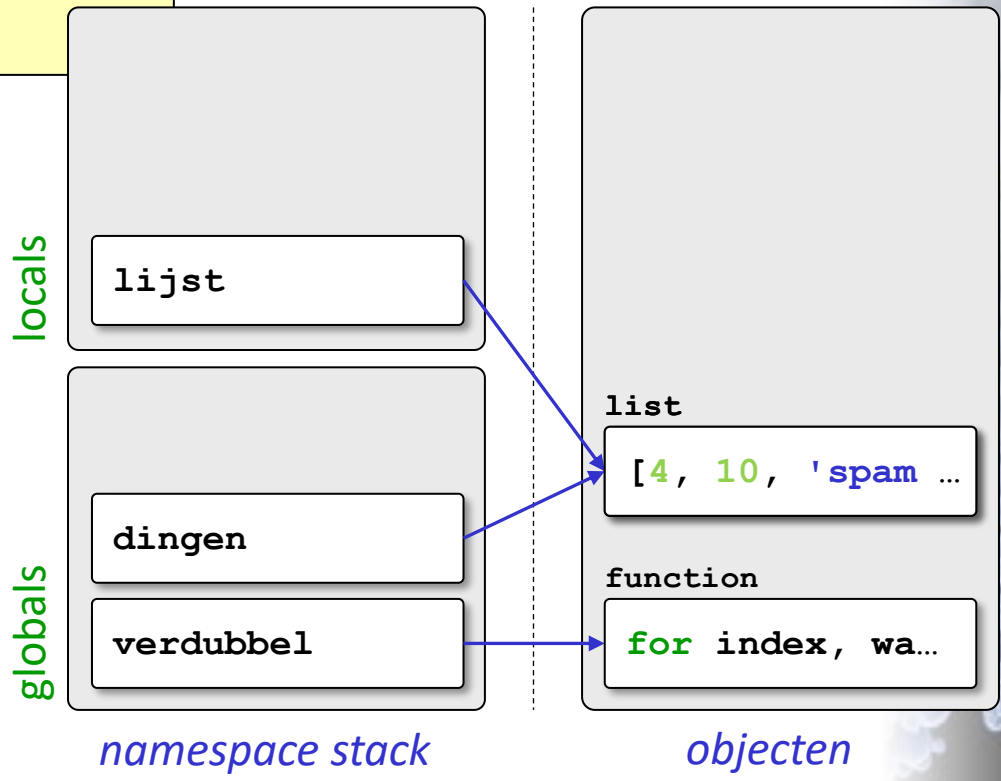
UNIVERSITEIT GENT

Lijsten doorgeven als argument

verdubbel.py

```
def verdubbel(lijst):
    for index, waarde in enumerate(lijst):
        lijst[index] = 2 * waarde

dingen = [2, 5, 'spam', 9.5]
verdubbel(dingen)
print(dingen)
```



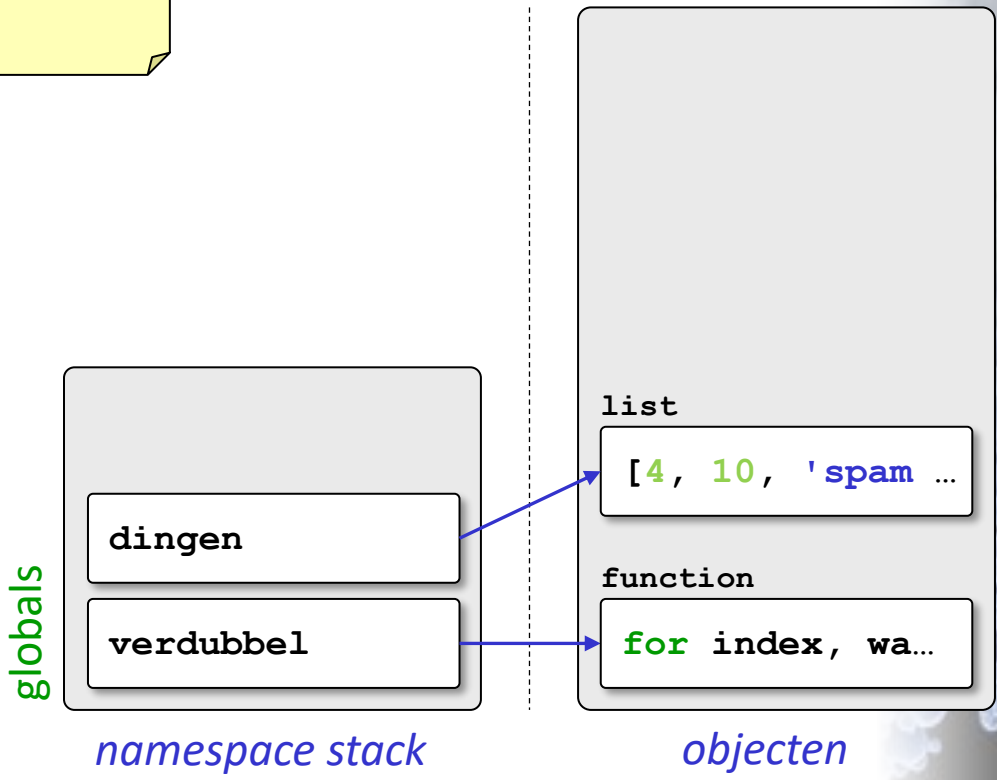
UNIVERSITEIT GENT

Lijsten doorgeven als argument

verdubbel.py

```
def verdubbel(lijst):
    for index, waarde in enumerate(lijst):
        lijst[index] = 2 * waarde

dingen = [2, 5, 'spam', 9.5]
verdubbel(dingen)
print(dingen)
```



UNIVERSITEIT GENT

Heremietkreeften

Geneste lijsten

Geneste lijsten

- **geneste lijst**: lijst die als element in andere lijst voorkomt
  - *bracket operators* evalueren van links naar rechts

```
>>> gassen = [['He', 2], ['Ne', 10], ['Ar', 18]]
>>> gas = gassen[2]
>>> gas
['Ar', 18]
>>> gas[0]
'Ar'
>>> gassen[2][0]
'Ar'
```

UNIVERSITEIT GENT

Matrices

- **geneste lijst**: lijst die als element in andere lijst voorkomt
  - *bracket operators* evalueren van links naar rechts
  - wordt vaak gebruikt om roosters/matrices voor te stellen

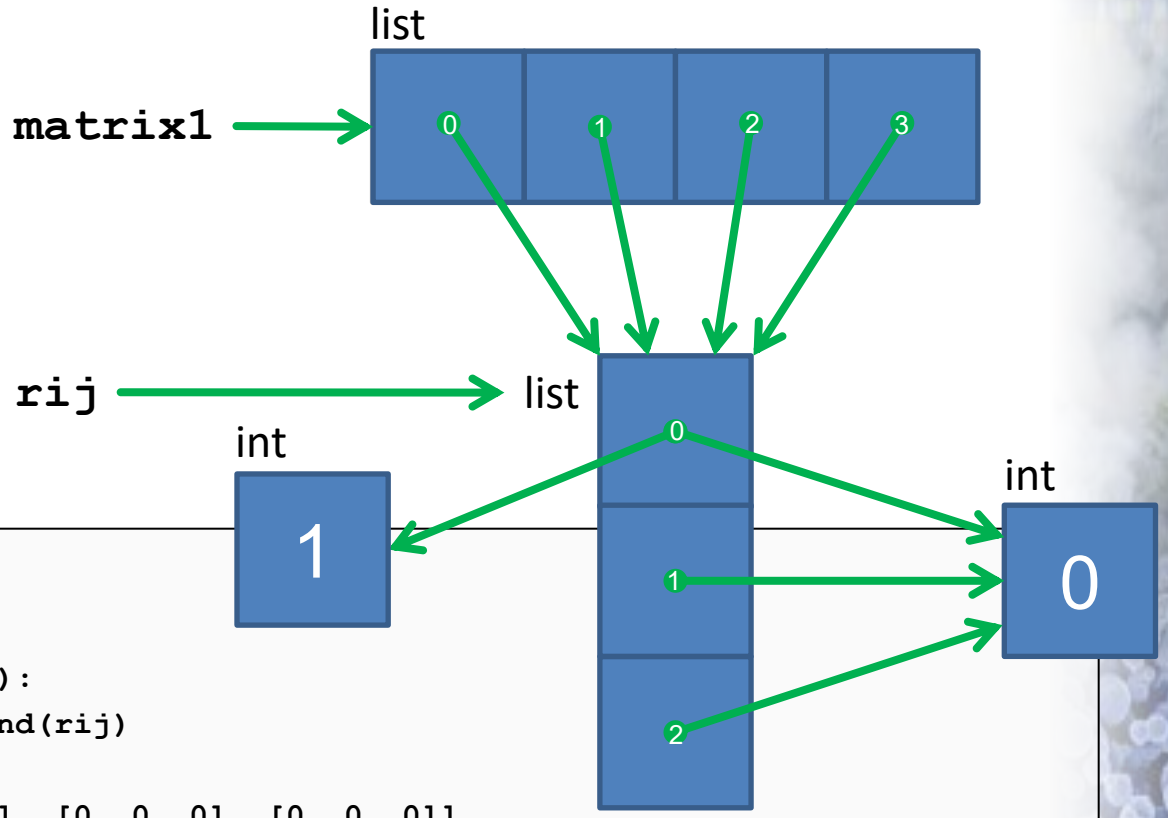
$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

```
>>> matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> rij = matrix[1]
>>> rij
[4, 5, 6]
>>> rij[2]
6
>>> matrix[1][2]
6
```

UNIVERSITEIT GENT

Matrices

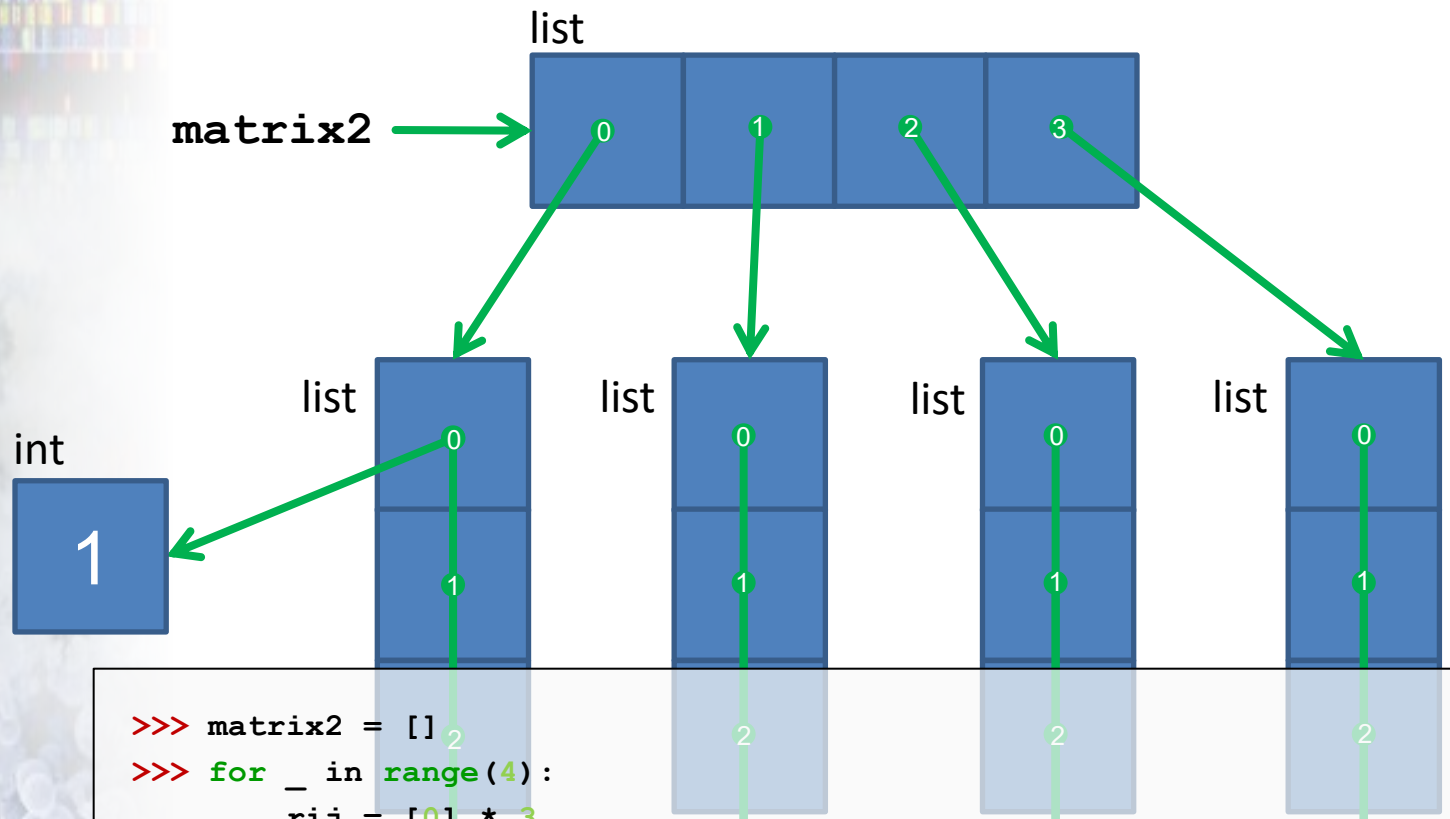
- **opgepast**: lijsten zijn veranderlijk




```
>>> rij = [0] * 3
>>> matrix1 = []
>>> for _ in range(4):
...     matrix1.append(rij)
>>> matrix1
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>> matrix1[0][0] = 1
>>> matrix1
[[1, 0, 0], [1, 0, 0], [1, 0, 0], [1, 0, 0]]
```

UNIVERSITEIT GENT

Matrices



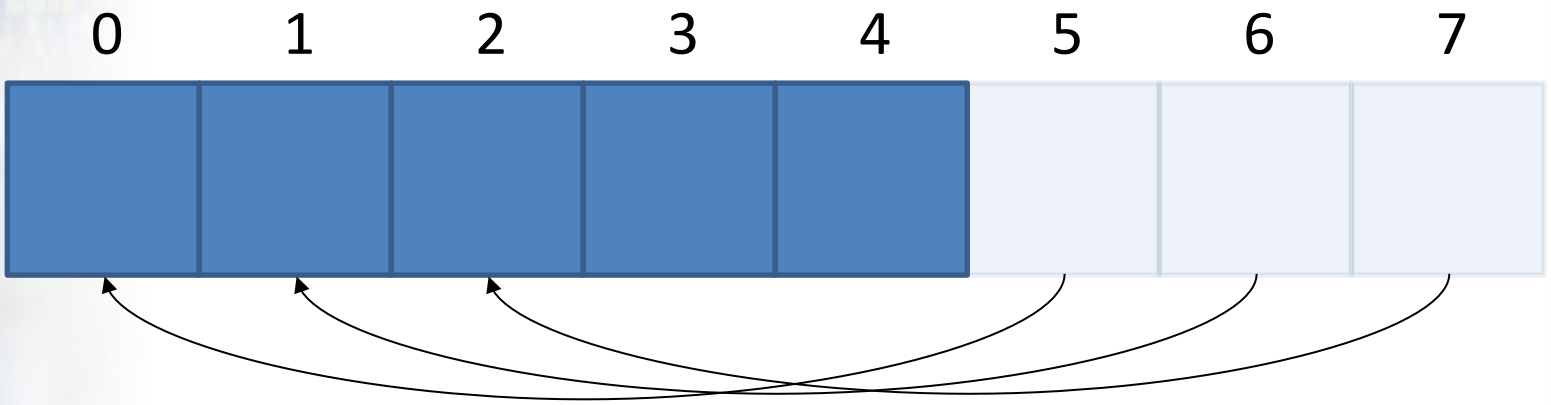

```
>>> matrix2 = []
>>> for _ in range(4):
...     rij = [0] * 3
...     matrix2.append(rij)
>>> matrix2
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

```
>>> matrix2[0][0] = 1
>>> matrix2 = [[0] * 3 for _ in range(4)]
>>> matrix2
>>> matrix2[0][0] = 1
[[1, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

UNIVERSITEIT GENT

Circulaire lijsten

Circulaire lijsten




```
>>> lijst = ['spam', 'eggs', 'bacon', 'shrubbery', 'ni']
>>> lijst[6]
Traceback (most recent call last):
IndexError: list index out of range
>>> lijst[6 % len(lijst)]
'eggs'
>>> lijst[-18 % len(lijst)]
'bacon'
```

UNIVERSITEIT GENT

Strings en lijsten

- ingebouwde functie **`list()`**
  - neemt *iterable object* als argument
  - geeft lijst terug met de elementen van *iterable object*
- ingebouwde functie **`str()`**
  - zet Python-object (*argument*) om in string (*return value*)

```
>>> list('stikstof')
['s', 't', 'i', 'k', 's', 't', 'o', 'f']
>>> str(5)
'5'
>>> str(None)
'None'
>>> str(list('stikstof'))
"['s', 't', 'i', 'k', 's', 't', 'o', 'f']"
```

UNIVERSITEIT GENT

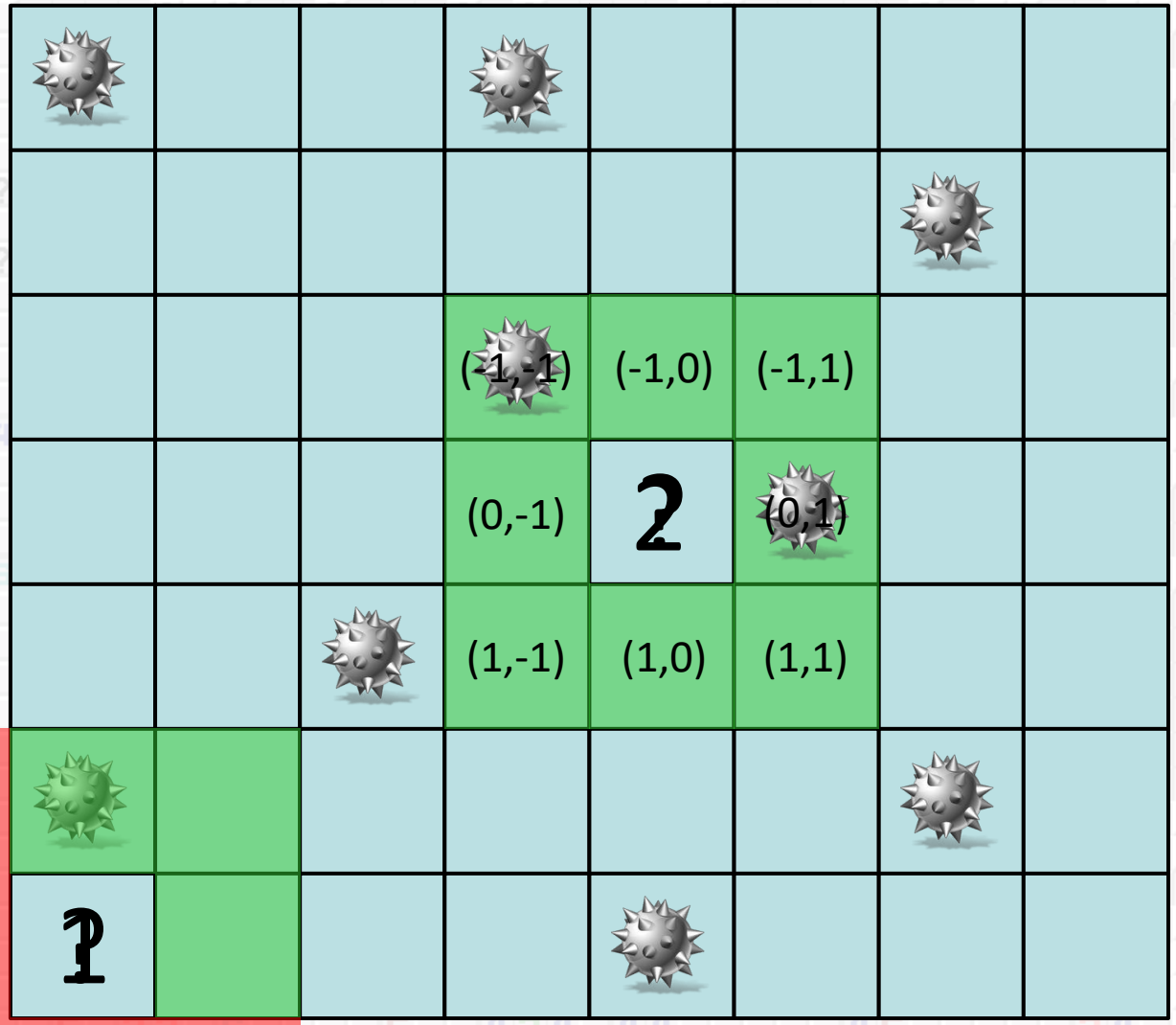
Strings en lijsten

- stringmethode **`split()`**
  - string → lijst van strings (opsplitsen met scheidingsteken)
  - standaard wordt witruimte als scheidingsteken gebruikt
- stringmethode **`join()`**
  - lijst van strings → string (samenvoegen met scheidingsteken)

```
>>> zin = 'de koetsier poetst de postkoets'
>>> zin.split()
['de', 'koetsier', 'poetst', 'de', 'postkoets']
>>> zin.split('oe')
['de k', 'tsier p', 'tst de postk', 'ts']
>>> ' '.join(['de','koetsier','poetst','de','postkoets'])
'de koetsier poetst de postkoets'
>>> 'oe'.join(['de k', 'tsier p', 'tst de postk', 'ts'])
'de koetsier poetst de postkoets'
```

UNIVERSITEIT GENT

Mijnenveger



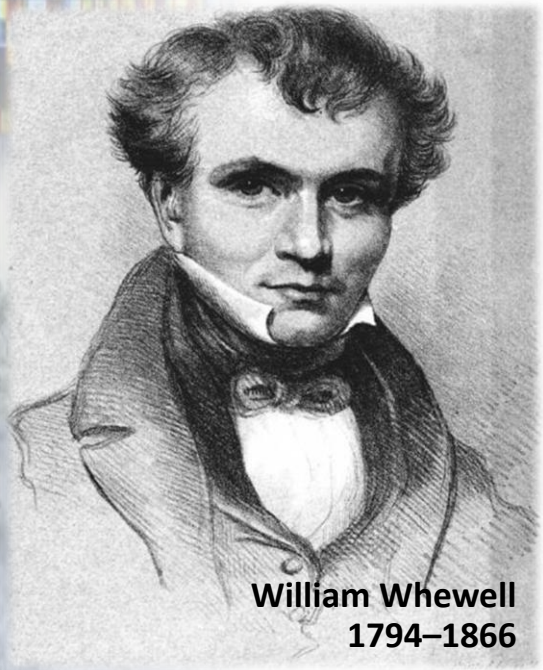
Huiswerk (volgende les)

- handboek: lees hoofdstuk 8 (modules)
- voorbeeldopgaven volgende les (reeks 7)
  - Caféspelletje
  - Locatiefilter

Vragen of opmerkingen?

UNIVERSITEIT GENT

The sky is the limit …



William Whewell
1794–1866

UNIVERSITEIT GENT

"Every failure is a step towards success."

— William Whewell