Universiteit Gent
Faculteit Wetenschappen

Programmeren in Python

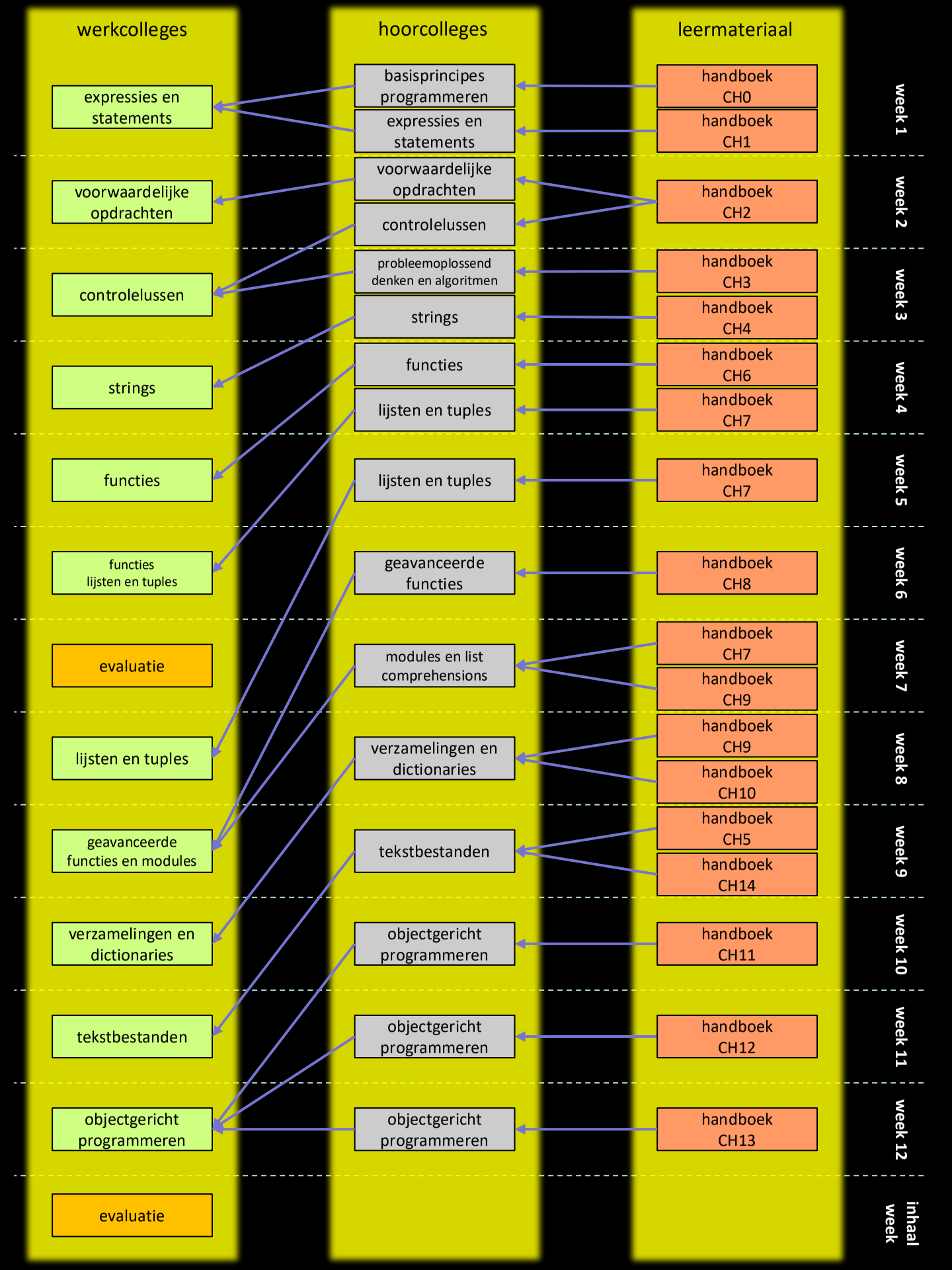
overerving

prof. dr. Peter Dawyndt

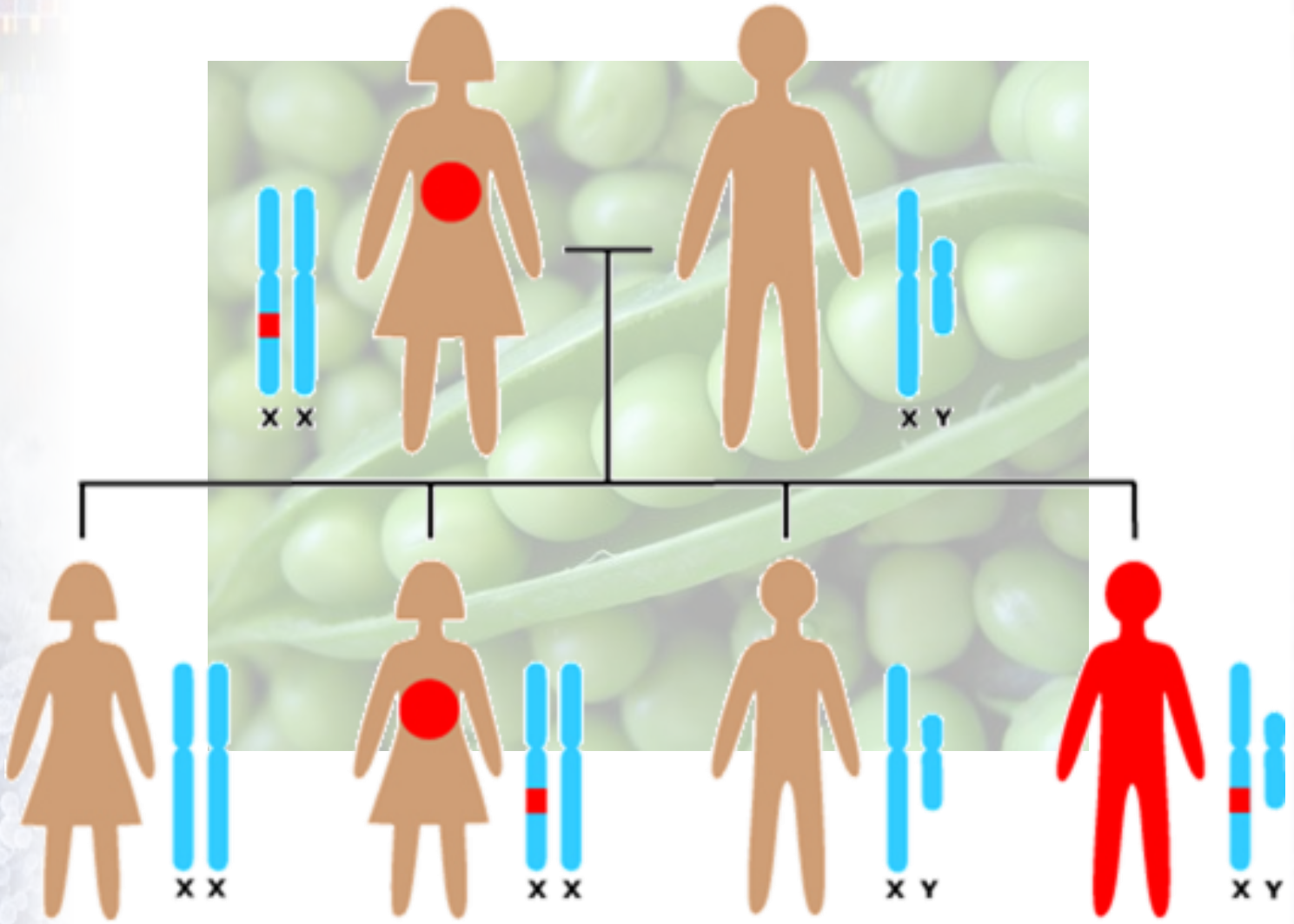
peter.dawyndt@ugent.be

@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

Overerving



UNIVERSITEIT GENT

OOP karakteristieken

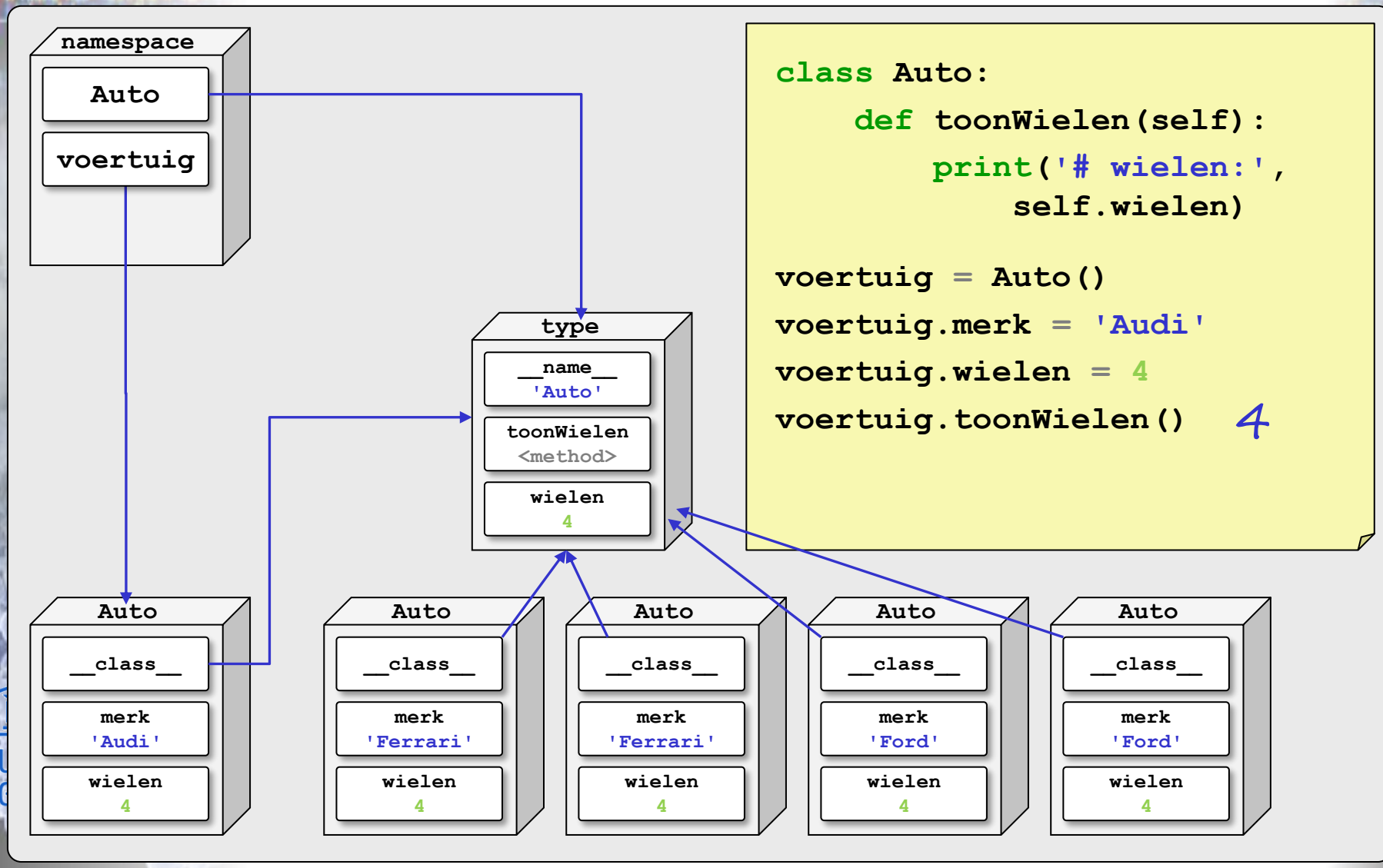
ENCAPSULATION

POLYMORPHISM

INHERITANCE

Klassen en instanties

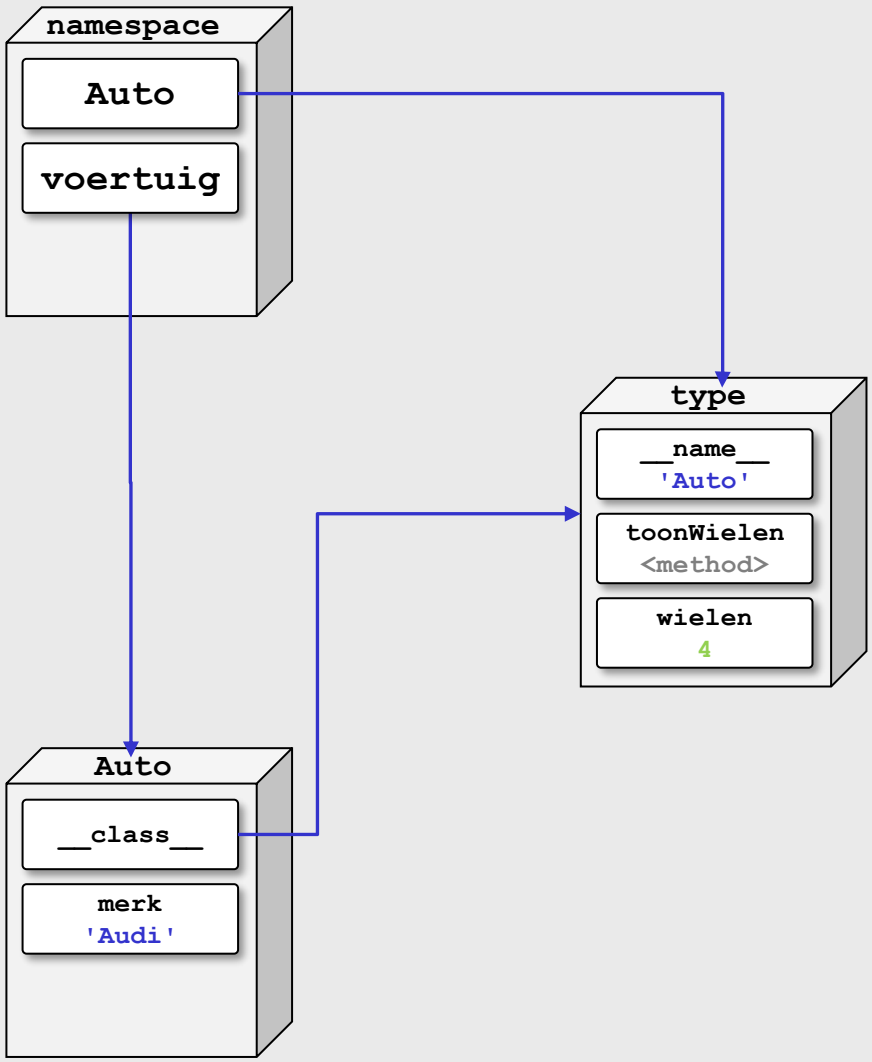
objecten (computergeheugen)




```
class Auto:
    def toonWielen(self):
        print('# wielen:',
              self.wielen)

voertuig = Auto()
voertuig.merk = 'Audi'
voertuig.wielen = 4
voertuig.toonWielen()   4
```

Klassen en instanties

objecten (computergeheugen)

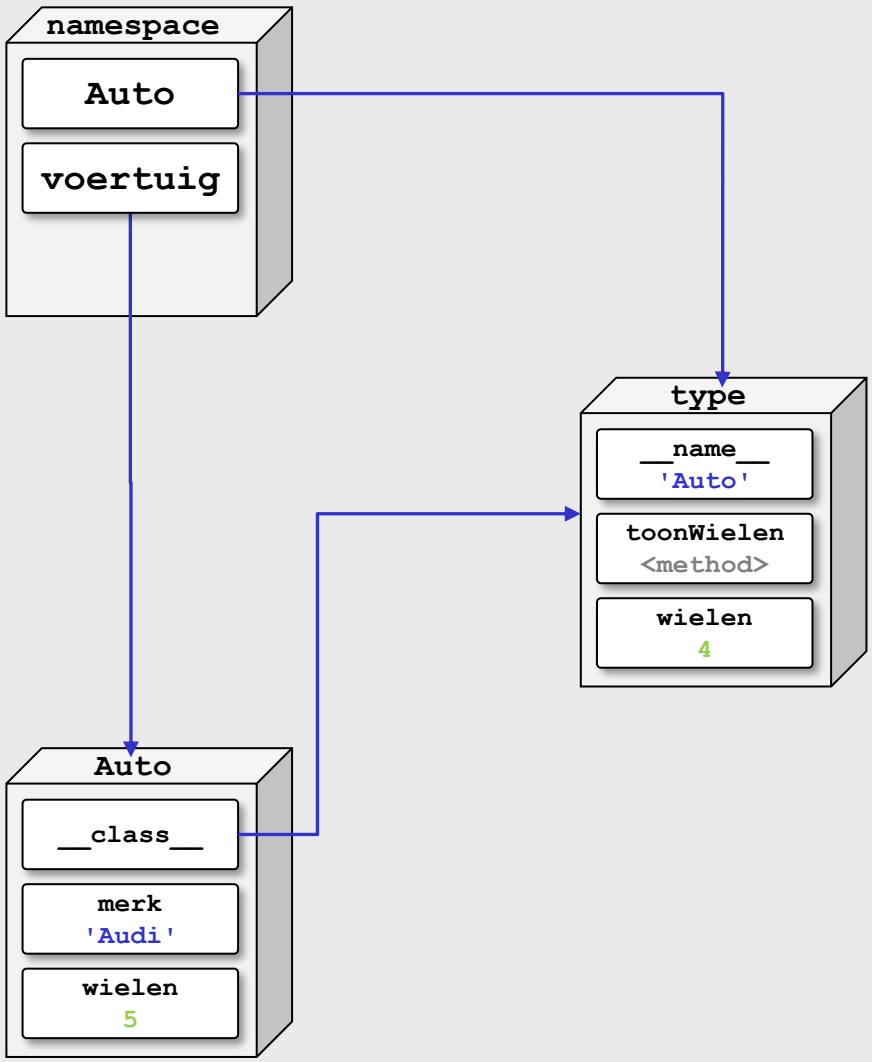



```
class Auto:
    wielen = 4
    def toonWielen(self):
        print('# wielen:',
              self.wielen)

voertuig = Auto()
voertuig.merk = 'Audi'
voertuig.toonWielen()
```

4

Klassen en instanties

objecten (computergeheugen)




```
class Auto:
    wielen = 4
    def toonWielen(self):
        print('# wielen:',
              self.wielen)

voertuig = Auto()
voertuig.merk = 'Audi'
voertuig.wielen = 5
voertuig.toonWielen()
```

5

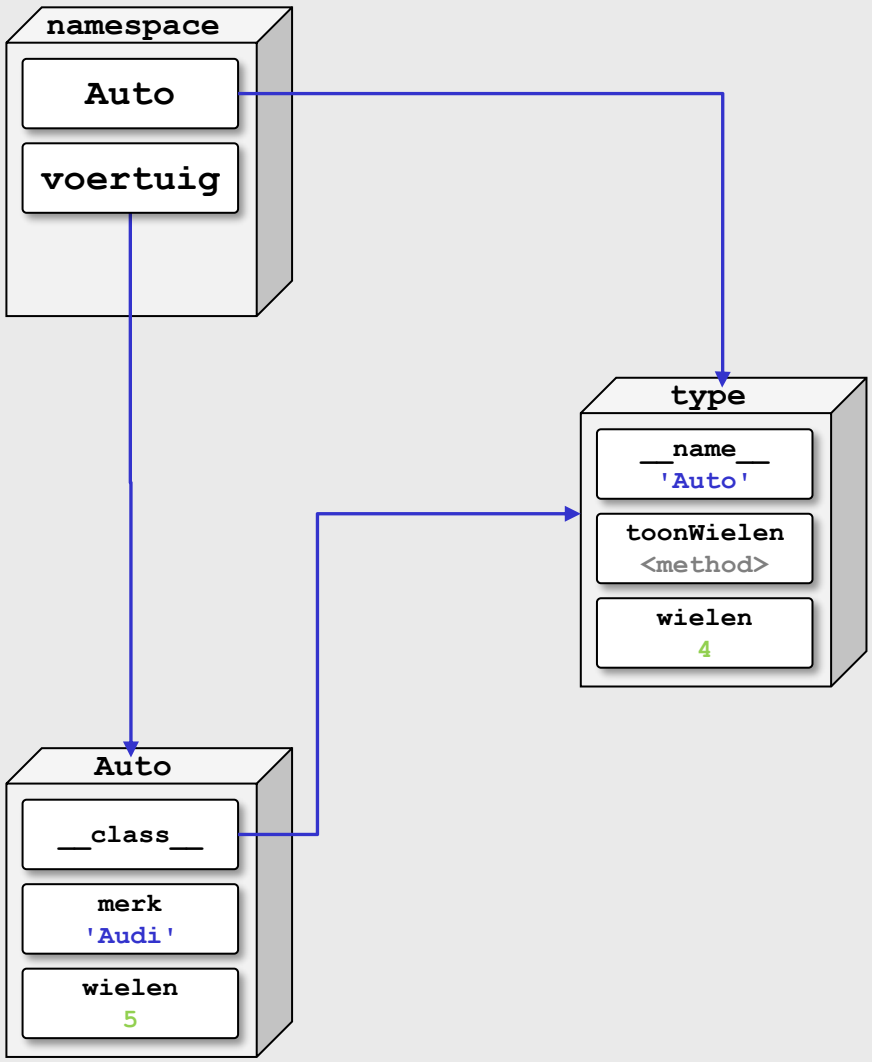
Romeins rekenen

I = 1
II = 2
III = 3
IV = 4
V = 5
VI = 6
VII = 7
VIII = 8
IX = 9
X = 10

XL = 40
L = 50
LX = 60
LXX = 70
LXXX = 80
XC = 90
C = 100
D = 500
M = 1000

Klassen en instanties

objecten (computergeheugen)

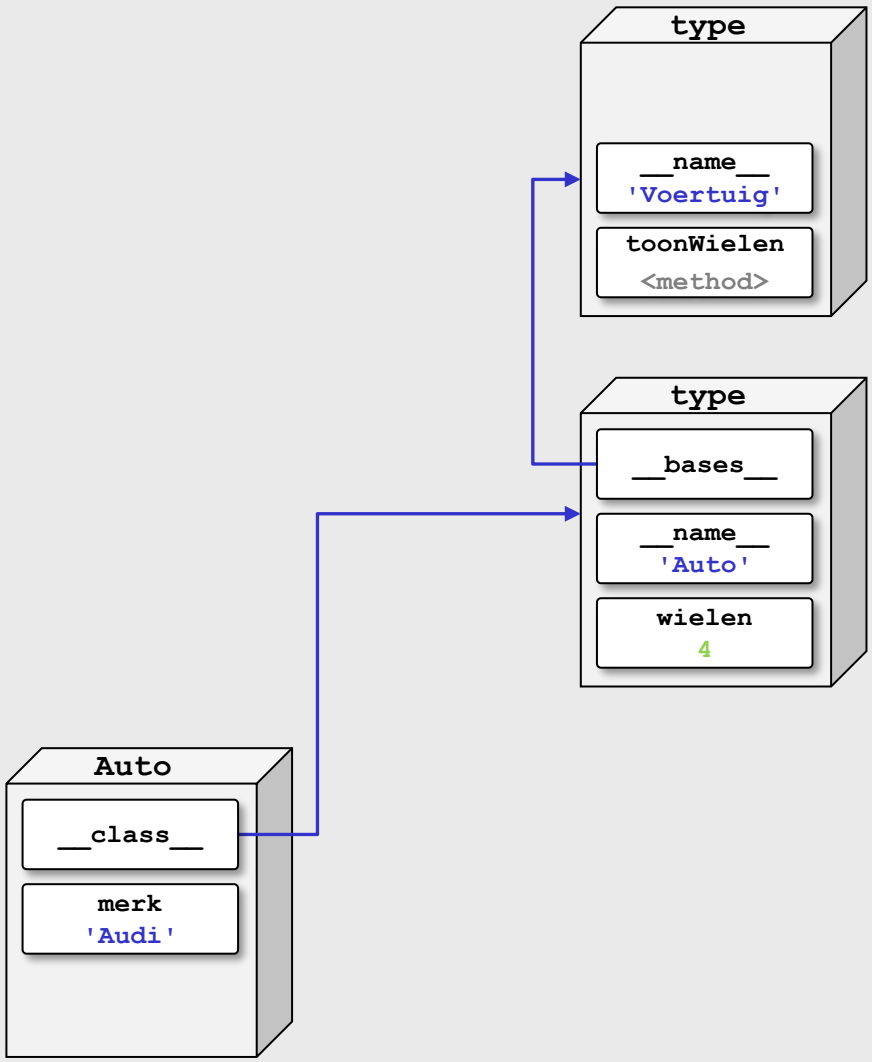



```
class Auto:
    wielen = 4
    def toonWielen(self):
        print('# wielen:',
              self.wielen)

voertuig = Auto()
voertuig.merk = 'Audi'
voertuig.wielen = 5
voertuig.toonWielen()
```

5

Overerving

objecten (computergeheugen)

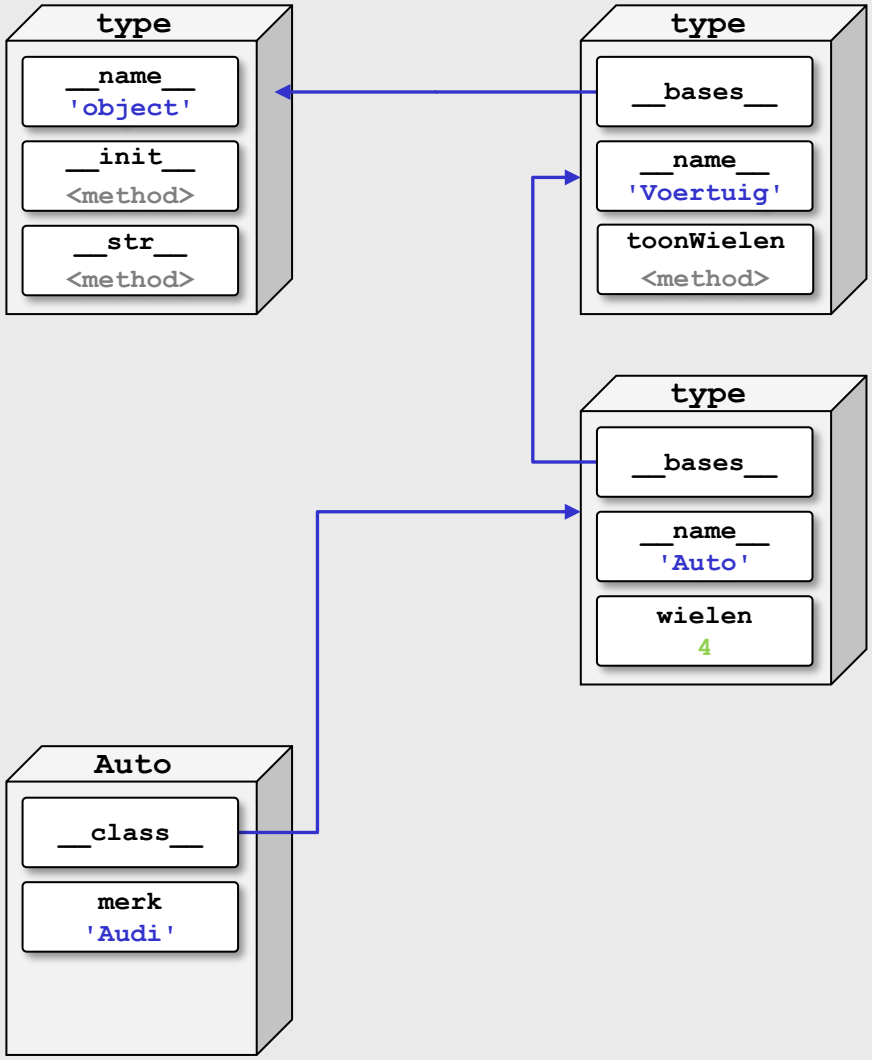



```
class Voertuig:
    def toonWielen(self):
        print('# wielen:',
              self.wielen)

class Auto(Voertuig):
    wielen = 4

voertuig = Auto()
voertuig.merk = 'Audi'
voertuig.toonWielen()
```

4

Overerving

objecten (computergeheugen)




```
class Voertuig(object):
    def toonWielen(self):
        print('# wielen:',
              self.wielen)

class Auto(Voertuig):
    wielen = 4

voertuig = Auto()
voertuig.merk = 'Audi'
voertuig.toonWielen()
```

4

Klassen: 'new style' vs 'old style'

- Python 2.x
  - onderscheid tussen 'new style' en 'old style' klassen
  - verschil zit enkel in geavanceerde (technische) details

```
class Punt:
    pass
```

old style

```
class Punt(object):
    pass
```

new style

- Python 3.x
  - geen onderscheid tussen 'new style' en 'old style' klassen

```
class Punt:
    pass
```

new style

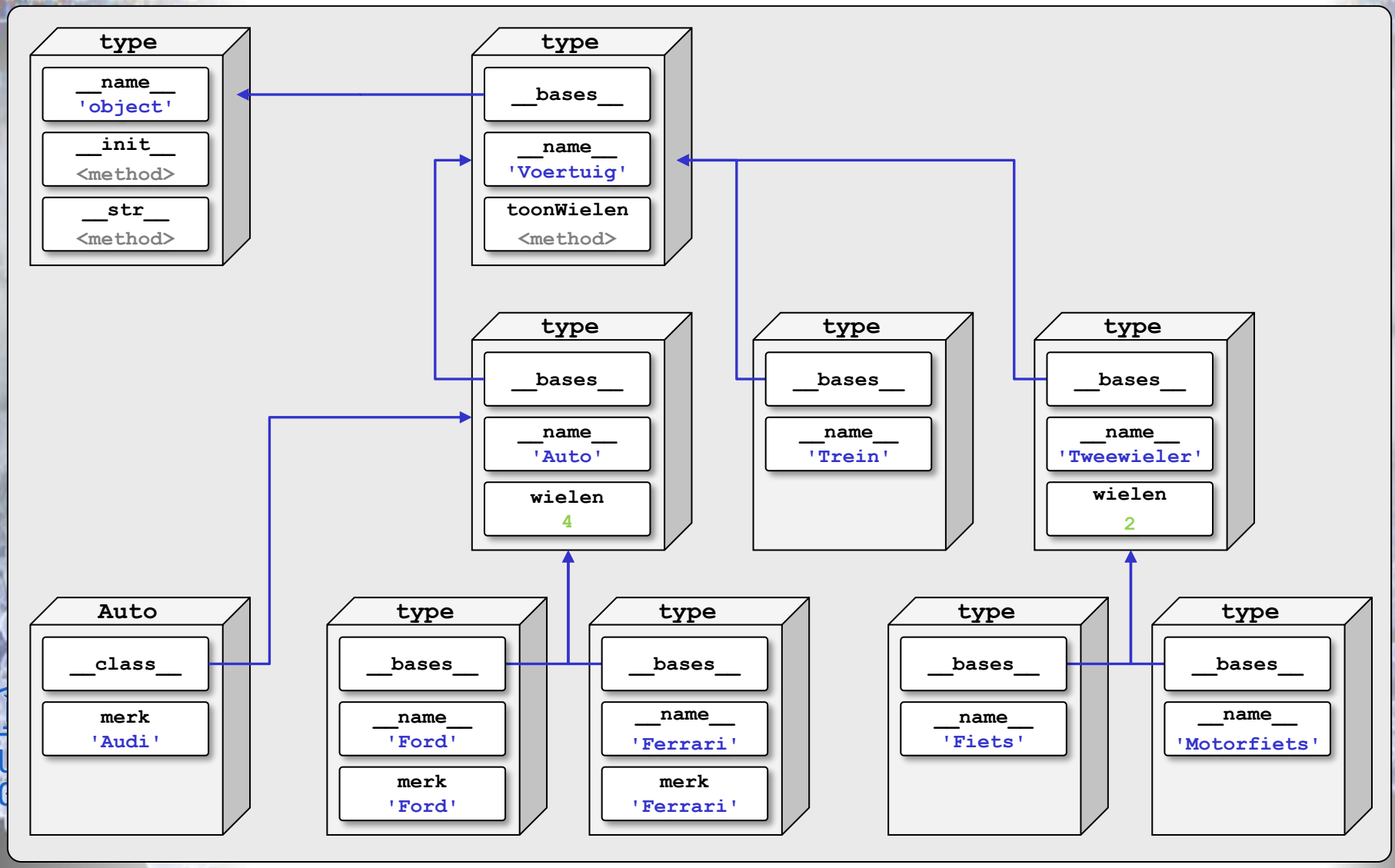
```
class Punt(object):
    pass
```

new style

UNIVERSITEIT GENT

Overerving

objecten (computergeheugen)



Overerving

```
class Voertuig:
  "basisklasse voor voertuigen"

  merk = 'onbekend'
  wielen = None

  def __init__(self, merk=None, wielen=None):
    if merk is not None: self.merk = merk
    if wielen is not None: self.wielen = wielen

  def __str__(self):
    return 'Een {} van merk {} met {} wielen'.format(
      self.__class__.__name__, self.merk, self.wielen)

  def toonWielen(self):
    if self.wielen is None:
      print('Onbekend aantal wielen')
    else:
      print(f'Aantal wielen: {self.wielen}')
```

```
class Auto(Voertuig):
    wielen = 4

class Ford(Auto):
    merk = 'Ford'

class Ferrari(Auto):
    merk = 'Ferrari'

class Trein(Voertuig):
    pass

class Tweewieler(Voertuig):
    wielen = 2

class Motorfiets(Tweewieler):
    pass

class Fiets(Tweewieler):
    pass
```

```
>>> fiets = Fiets(merk='Merckx')
>>> fiets.merk
'Merckx'
>>> fiets.toonWielen()
Aantal wielen: 2
>>> print(fiets)
Een Fiets van merk Merckx met 2 wielen
```

UNIVERSITEIT GENT

Overerving

```
class Voertuig:
  "basisklasse voor voertuigen"

  merk = 'onbekend'
  wielen = None

  def __init__(self, merk=None, wielen=None):
    if merk is not None: self.merk = merk
    if wielen is not None: self.wielen = wielen

  def __str__(self):
    return 'Een {} van merk {} met {} wielen'.format(
      self.__class__.__name__, self.merk, self.wielen)

  def toonWielen(self):
    if self.wielen is None:
      print('Onbekend aantal wielen')
    else:
      print(f'Aantal wielen: {self.wielen}')
```

```
class Auto(Voertuig):
    wielen = 4

class Ford(Auto):
    merk = 'Ford'

class Ferrari(Auto):
    merk = 'Ferrari'

class Trein(Voertuig):
    pass

class Tweewieler(Voertuig):
    wielen = 2

class Motorfiets(Tweewieler):
    pass

class Fiets(Tweewieler):
    pass
```

```
>>> auto = Ferrari()
>>> auto.merk
'Ferrari'
>>> auto.toonWielen()
Aantal wielen: 4
>>> print(auto)
Een Ferrari van merk Ferrari met 4 wielen
```

UNIVERSITEIT GENT

Overerving

```
class Voertuig:
  "basisklasse voor voertuigen"

  merk = 'onbekend'
  wielen = None

  def __init__(self, merk=None, wielen=None):
    if merk is not None: self.merk = merk
    if wielen is not None: self.wielen = wielen

  def __str__(self):
    return 'Een {} van merk {} met {} wielen'.format(
      self.__class__.__name__, self.merk, self.wielen)

  def toonWielen(self):
    if self.wielen is None:
      print('Onbekend aantal wielen')
    else:
      print(f'Aantal wielen: {self.wielen}')
```

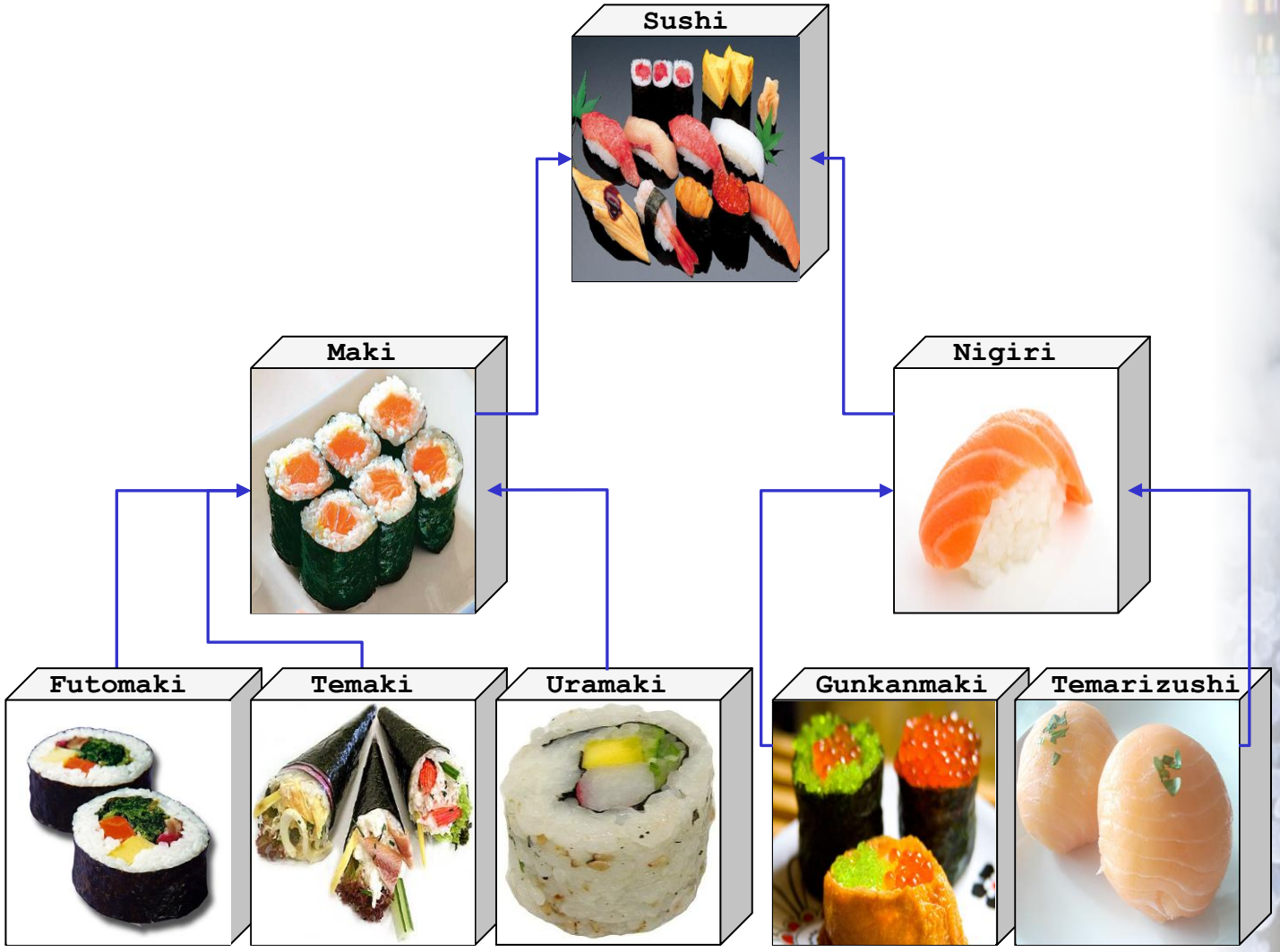
```
class Auto(Voertuig):
    wielen = 4

class Ford(Auto):
    merk = 'Ford'

class Ferrari(Auto):
    merk = 'Ferrari'

class Trein(Voertuig):
    pass

class Tweewieler(Voertuig):
    wielen = 2

class Motorfiets(Tweewieler):
    pass

class Fiets(Tweewieler):
    pass
```

```
>>> boot = Voertuig(merk='Ocean', wielen=0)
>>> boot.merk
'Ocean'
>>> boot.toonWielen()
Aantal wielen: 0
>>> print(boot)
Een Voertuig van merk Ocean met 0 wielen
```

UNIVERSITEIT GENT

Sushi

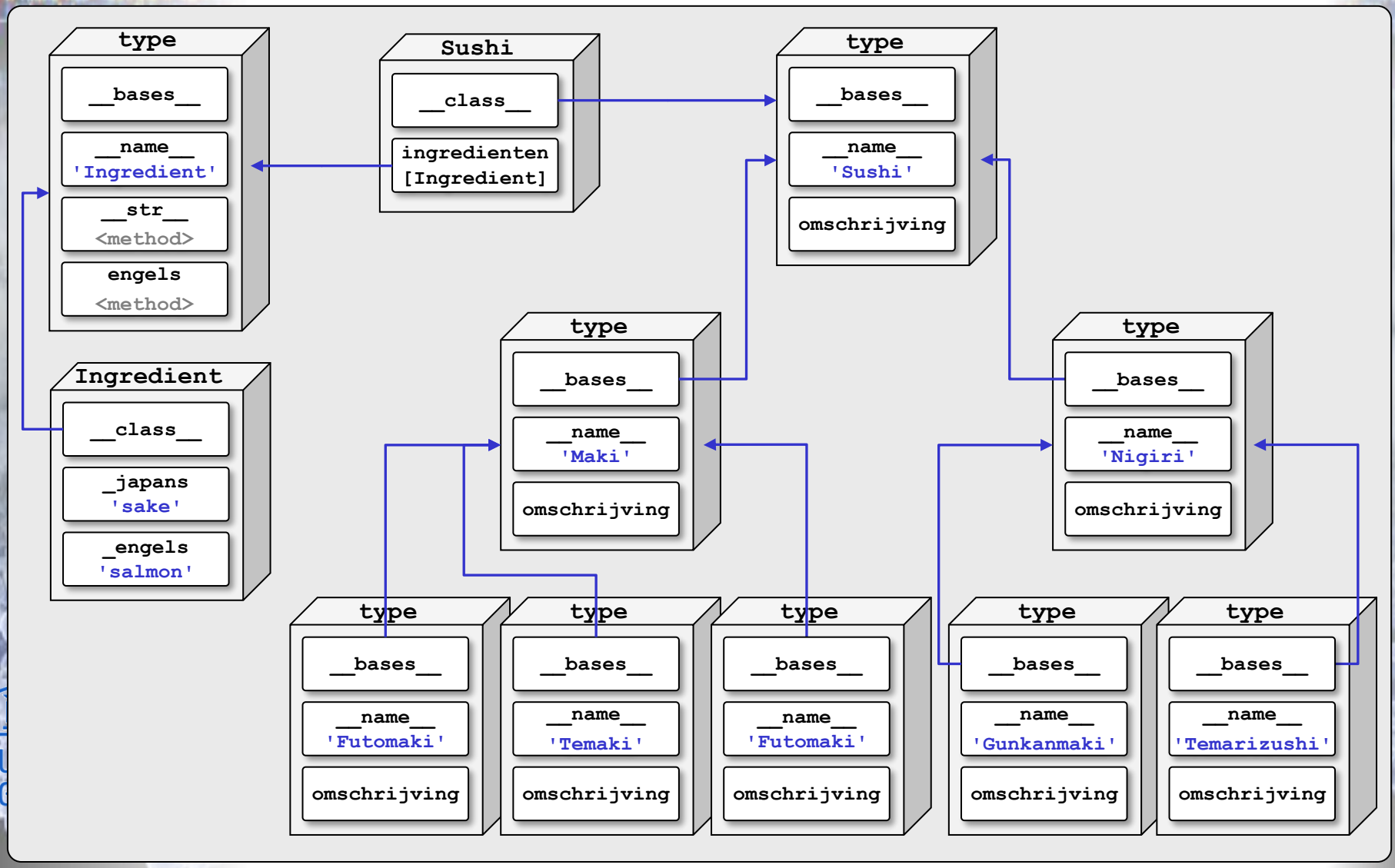
Sushi hiërarchie



UNIVERSITEIT GENT

Sushi hiërarchie

objecten (computergeheugen)

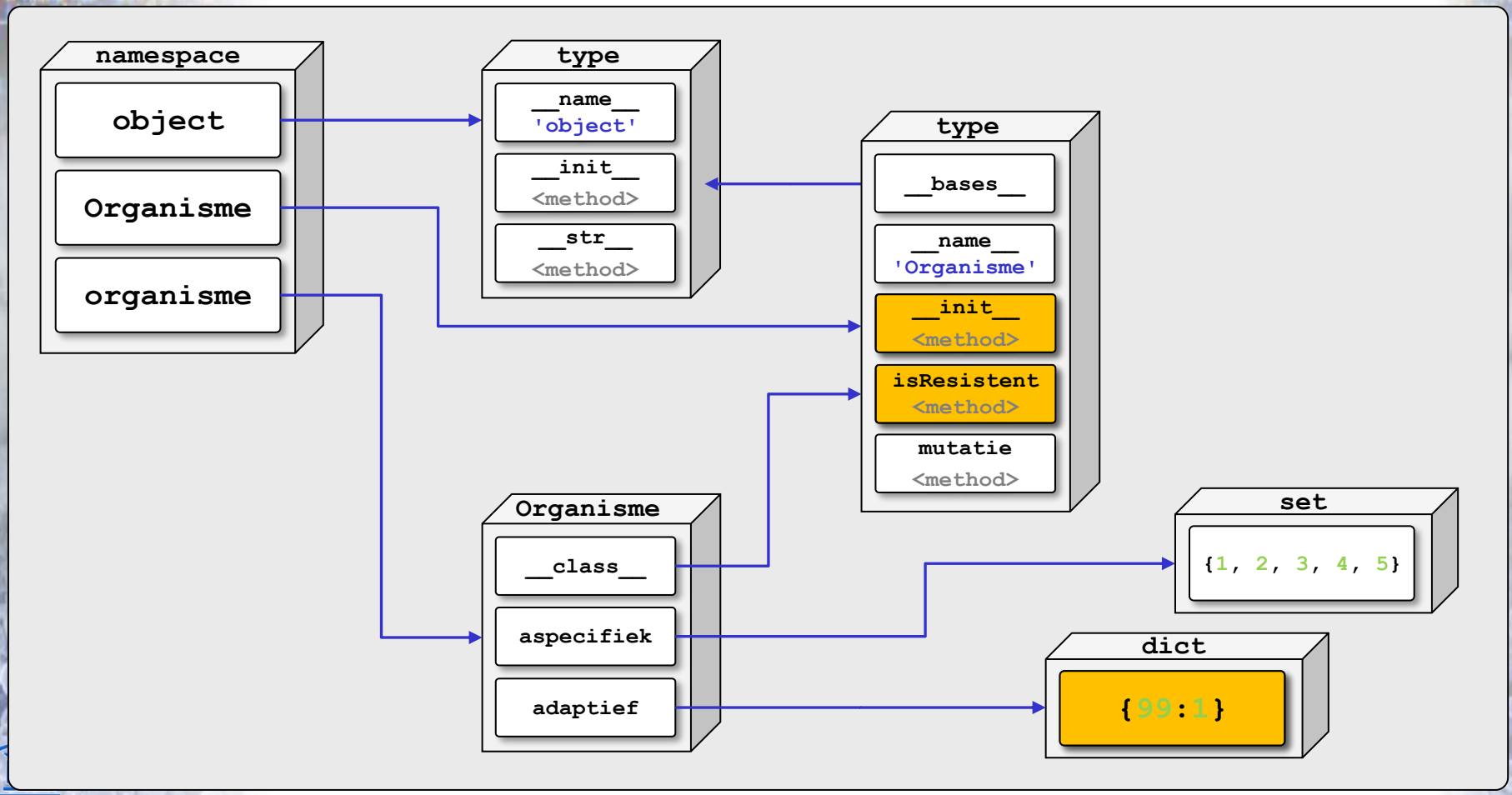


Immuunsysteem

aspecifiek

Immuunsysteem

objects (main memory)

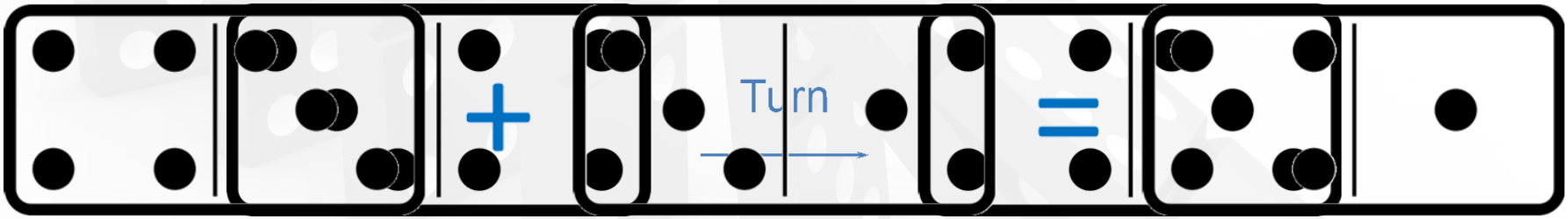


UNIVERSITEIT GENT

```
>>> organisme.isResistent(virus=99)
```

Dominostenen

```
+---+   +---+   +---+   +---+   +---+   +---+   +---+
|   |   |   |   |o  |   |o  |   |o o|   |o o|   |ooo|
|   |   | o |   |   |   | o |   |   |   | o |   |   |
|   |   |   |   |  o|   |  o|   |o o|   |o o|   |ooo|
+---+   +---+   +---+   +---+   +---+   +---+   +---+

  0       1       2       3       4       5       6
```



Quilten

`Quilt(2, 2, '//++')`

/	/
+	+

+

`Quilt(2, 2, '-\\||')`

-	\
|	|

=

`Quilt(2, 4, '//-\\++||')`

/	/	-	\
+	+	|	|

`Quilt(4, 2, '+\\+\\-|-/')`

+	\
+	\
-	|
-	/

draai

Huiswerk

- lees hoofdstuk 12 in Python handboek
  - bekijk sushi demo (oplossing op Ufora)
- lees hoofdstuk 13 in Python handboek
  - predator-prey demo
- afwerken opgelegde oefeningen reeks 10
  - deadline: dinsdag 17 december 2024
  - 2e evaluatie: 16 & 17 december 2024

UNIVI
GENT

Vragen of opmerkingen?

UNIVERSITEIT GENT

The sky is the limit …

"When a task cannot be partitioned because of sequential constraints, the application of more effort has no effect on the schedule.

The bearing of a child takes nine months, no matter how many women are assigned.

Many software tasks have this characteristic because of the sequential nature of debugging."

— Frederick P. Brooks, Jr.
The Mythical Man-Month

UNIVERSITEIT GENT