Universiteit Gent
Faculteit Wetenschappen

Programmeren in Python

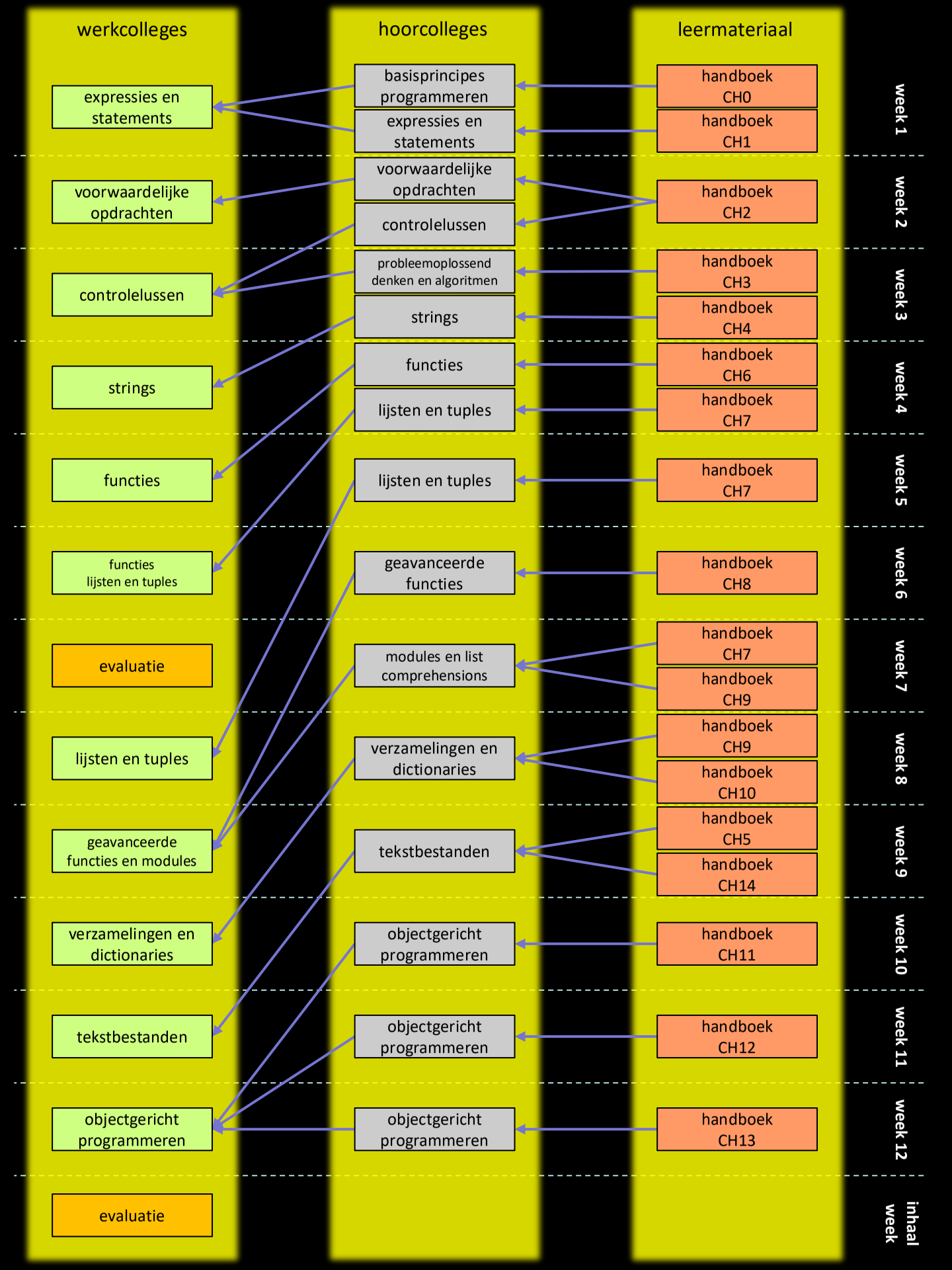
lijsten en tuples

prof. dr. Peter Dawyndt

peter.dawyndt@ugent.be

@dawyndt

UNIVERSITEIT GENT

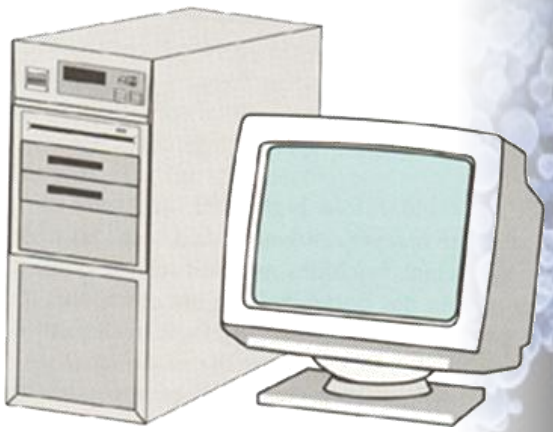

werkcolleges
hoorcolleges
leermateriaal
expressies en statements
voorwaardelijke opdrachten
controlelussen
strings
functies
functies lijsten en tuples
evaluatie
lijsten en tuples
geavanceerde functies en modules
verzamelingen en dictionaries
tekstbestanden
objectgericht programmeren
evaluatie
basisprincipes 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

Ontwerpcyclus

natuurlijke taal
(Nederlands, Engels, Frans, …)

programmeertaal
(Python, Java, C++, …)

machinetaal
(Pentium, Xeon, AMD, …)



UNIVERSITEIT GENT

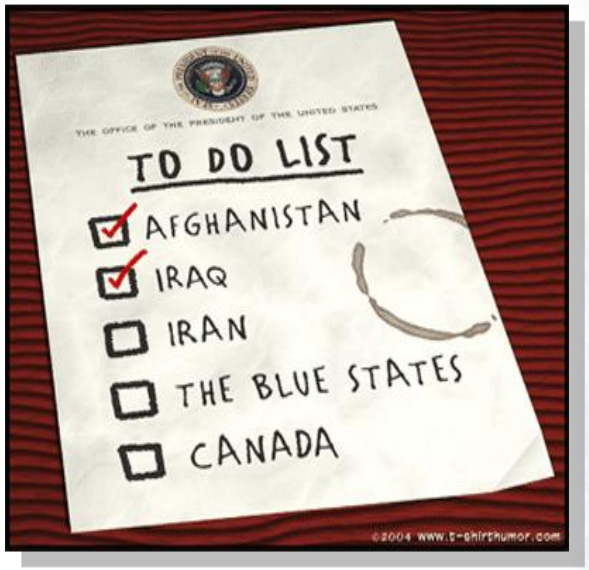
Ontwerpcyclus

1. beschrijf en analyseer het probleem
   - *wat moet het programma doen?*
   - *wat is de invoer, en wat de gewenste uitvoer?*
2. ontwerp een algoritme (*pseudocode*)
   - *hoe bereik je het vooropgestelde doel?*
3. zet algoritme om in broncode (*source code*)
   - volgens syntaxisregels van gekozen programmeertaal
4. compileer broncode tot machinetaal
5. voer het programma uit
6. spoor eventuele fouten op (*debuggen*)

UNIVERSITEIT GENT

Kwantificatoren

- een *reeks* getallen
- een *aantal* jaren
- een *hoop* boeken
- een *lijst* namen
- een *stapel* pannenkoeken



UNIVERSITEIT GENT

Lijsten

Periodic Table of the Internet

I II III IV V VI VII VIII IX X XI XII XIII XIV XV XVI XVII XVIII

FaceBook.com
Fb
29

I Search Engines
II Internet Tools
III Site Ranking
IV Aggregators
V Webcomics
VI Productivity
VII Get Stuff
VIII Operating Systems
IX-XII Miscellaneous
XIII Blogs
XIV Social Networking
XV-XVI Podcasts
XVII Videos
XVIII News

WellingtonGrey.net
Wg
207,654
← Site URL
← Symbol
← Rank

Lijsten

- **lijst**: veranderlijke reeks van objecten
  - objecten uit lijst worden *elementen* genoemd
  - reeksen (*sequences*) zijn geordend
    - lijsten, tuples en strings zijn voorbeelden van *sequences*
  - element uit lijst kan individueel aangesproken worden via index
- eenvoudigste manier om lijsten aan te maken
  - elementen opsommen tussen vierkante haken
  - elementen uit lijst van elkaar scheiden door komma's

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

UNIVERSITEIT GENT

Lijsten

- **lijst**: veranderlijke reeks van objecten

"Denk aan een 1-dimensionale reeks die vanzelf groter of kleiner wordt naargelang dit nodig is."

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

UNIVERSITEIT GENT

Lijsten

- **lijst**: veranderlijke reeks van objecten
  - "veranderlijk" (*mutable*) betekent dat elementen *in place* kunnen vervangen worden
  - dit kan niet met strings
    - strings zijn onveranderlijk (*immutable*)

```
>>> gassen = ['He', 'Ne', 'Ar', 'Kr']
>>> gassen[0] = 'Xe'
>>> gassen[-1] = 'Rn'
>>> print(gassen)
['Xe', 'Ne', 'Ar', 'Rn']
```

UNIVERSITEIT GENT

Lijsten

- **lijst**: veranderlijke reeks van objecten
  - "van objecten" betekent dat lijst alle mogelijke gegevenstypes kan bevatten
    - zelfs elementen van verschillende types
    - zelfs lijsten van lijsten (*geneste lijsten*)

"Strings zijn homogeen, lijsten zijn heterogeen."

```
>>> gassen = ['He', 'Ne', 'Ar', 'Kr']
>>> atoomnummer = [2, 10, 18, 36]
>>> atoommassa = [4.00260, 20.179, 39.948, 83.80]
>>> helium = ['He', 'Helium', 2, 4.00260]
>>> gassen = [['He', 2], ['Ne', 10], ['Ar', 18]]
```

UNIVERSITEIT GENT

Lijsten

- **lijst**: veranderlijke reeks van objecten
  - `[]` stelt lege lijst voor
  - net zoals de numerieke waarde **0** en de lege string `''`, zijn ook lege lijsten **`False`** in Booleaanse expressies

```
>>> gassen = []
>>> if gassen:
...     print('Er zitten gassen in de lijst.')
... else:
...     print('Er zitten geen gassen in de lijst.')
...
Er zitten geen gassen in de lijst.
```

UNIVERSITEIT GENT

Lijsten

- **lijst**: veranderlijke reeks van objecten
  - lijsten bevatten objecten, maar zijn zelf ook objecten
    - kunnen toegekend worden aan variabelen
    - kunnen als resultaat teruggegeven worden door functie
    - kunnen als argument doorgegeven worden aan functie

```
>>> gassen = ['He', 'Ne', 'Ar', 'Kr']
>>> def omgekeerd(lijst):
...     return lijst[::-1]
...
>>> omgekeerd(gassen)
['Kr', 'Ar', 'Ne', 'He']
```

UNIVERSITEIT GENT

Lijstmethoden

- stringmethoden geven nieuwe string terug (*immutable*)
- lijstmethoden passen bestaande lijst aan (*mutable*)

```
>>> metalen = ['goud', 'ijzer', 'lood', 'goud']
```

methode	functionaliteit	voorbeeld	resultaat
`append`	toevoegen aan einde van lijst	`metalen.append('tin')`	`['goud', 'ijzer', 'lood', 'goud', 'tin']`
`count`	telt hoe vaak element voorkomt in lijst	`metalen.count('goud')`	`2`
`index`	positie opzoeken waar element eerste keer voorkomt in lijst (foutmelding indien geen voorkomens gevonden)	`metalen.index('ijzer')` `metalen.index('zwavel')`	`1` `ValueError`
`insert`	element toevoegen op aangegeven positie in lijst	`metalen.insert(2, 'zilver')`	`['goud', 'ijzer', 'zilver', 'lood', 'goud']`
`remove`	eerste voorkomen van element in lijst verwijderen	`metalen.remove('goud')`	`['ijzer', 'lood', 'goud']`
`reverse`	volgorde van elementen omkeren	`metalen.reverse()`	`['goud', 'lood', 'ijzer', 'goud']`
`sort`	elementen sorteren	`metalen.sort()`	`['goud', 'goud', 'ijzer', 'lood']`

Lijstmethoden

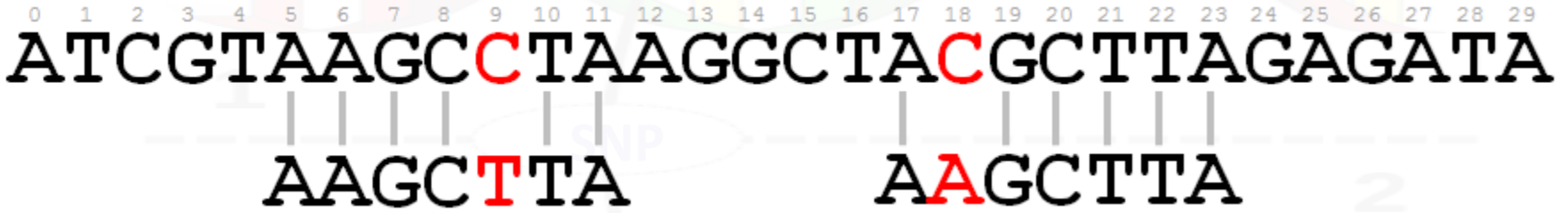
- methode **`index()`** geeft fout indien element niet gevonden
- alle methoden werken *in place*
  - veranderen originele lijst (behalve **`count()`** en **`index()`**)
  - geven waarde **`None`** terug als resultaat (behalve **`pop()`**)

```
>>> metalen = ['goud', 'ijzer', 'lood', 'goud']
>>> metalen.index('zwavel')
ValueError: list.index(x): x not in list
>>> metalen.sort()
>>> print(metalen)
['goud', 'goud', 'ijzer', 'lood']
>>> metalen = metalen.reverse()          # fout !!
>>> print(metalen)
None
```

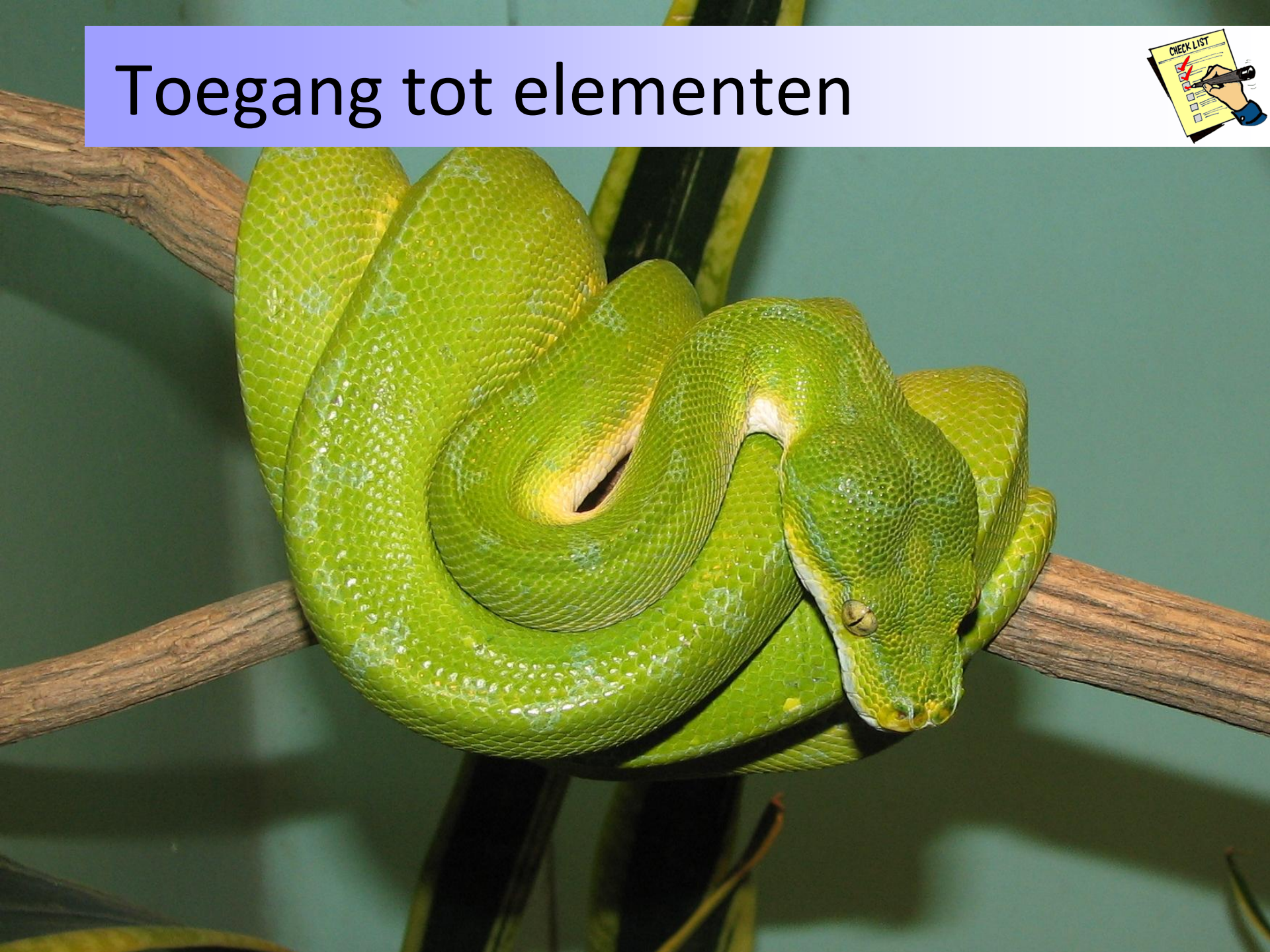
UNIVERSITEIT GENT

SNPs


CHECK LIST


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
ATCGTAAGCCTAAGGCTACGCTTAGAGATA
AAGCTTA
AAGCTTA

Toegang tot elementen



Toegang tot elementen

- aanloog als bij strings (*bracket operator*)
  - indexeren: toegang tot individuele elementen
    - integerexpressie tussen vierkante haakjes

```
>>> gassen = ['He', 'Ne', 'Ar', 'Kr']
>>> print(gassen)
['He', 'Ne', 'Ar', 'Kr']
>>> print(gassen[0])
He
>>> print(gassen[-1])
Kr
>>> print(gassen[4])
IndexError: list index out of range
>>> gassen[-5]
IndexError: list index out of range
```

UNIVERSITEIT GENT

Toegang tot elementen

- aanloog als bij strings (*bracket operator*)
  - indexeren: toegang tot individuele elementen
    - integerexpressie tussen vierkante haakjes

```
>>> gassen = ['He', 'Ne', 'Ar', 'Kr']
>>> gassen[9 - 8]
'Ne'
>>> gassen[1.0]
TypeError: list indices must be integers, not float
>>>
>>> gassen = [['He', 2], ['Ne', 10], ['Ar', 18]]
>>> gassen[1]
['Ne', 10]
>>> gassen[1][0]
'Ne'
```

UNIVERSITEIT GENT

Toegang tot elementen

- aanloog als bij strings (*bracket operator*)
  - indexeren: toegang tot individuele elementen
    - integerexpressie tussen vierkante haakjes
    - **`lijst[i] = obj`**
      - kent object **`obj`** toe aan positie **`i`** in **`lijst`**
      - lijsten zijn veranderlijk !!

```
>>> gassen = ['He', 'Ne', 'Ar', 'Kr']
>>> print(gassen)
['He', 'Ne', 'Ar', 'Kr']
>>> gassen[0] = 'H'
>>> gassen[-1] = 'Xe'
>>> print(gassen)
['H', 'Ne', 'Ar', 'Xe']
```

UNIVERSITEIT GENT

Toegang tot elementen

- aanloog als bij strings (*bracket operator*)
  - indexeren: toegang tot individuele elementen
    - integerexpressie tussen vierkante haakjes
    - **`lijst[i] = obj`**
    - methode **`append()`**: element toevoegen achteraan lijst

```
>>> gassen = ['H', 'He', 'Ne', 'Ar']
>>> gassen.append('Kr')
>>> gassen.append('Xe')
>>> gassen.append('Rn')
>>> gassen.append('Uuo')
>>> print(gassen)
['H', 'He', 'Ne', 'Ar', 'Kr', 'Xe', 'Rn', 'Uuo']
```

UNIVERSITEIT GENT

Toegang tot elementen

- aanloog als bij strings (*bracket operator*)
  - indexeren: toegang tot individuele elementen
  - slicing: subsequentie van elementen uit lijst genereren

```
>>> gassen = ['H','He','Ne','Ar','Kr','Xe','Rn']
>>> gassen[1:3]
['He', 'Ne']
>>> gassen[:4]
['H', 'He', 'Ne', 'Ar']
>>> gassen[-3:]
['Kr', 'Xe', 'Rn']
>>> gassen[:]
['H', 'He', 'Ne', 'Ar', 'Kr', 'Xe', 'Rn']
>>> gassen[::2]
['H', 'Ne', 'Kr', 'Rn']
```

UNIVERSITEIT GENT

Elementen overlopen

Elementen overlopen

- **list traversal**: verwerkingspatroon waarbij elementen van een lijst één voor één doorlopen worden

traversal1.py

```
gassen = ['H','He','Ne','Ar','Kr','Xe','Rn','Uuo']

i = 0
while i < 8:
    print(gassen[i])
    i += 1
```

UNIVERSITEIT GENT

Elementen overlopen

- **list traversal**: verwerkingspatroon waarbij elementen van een lijst één voor één doorlopen worden
  - gebruik ingebouwde functie **`len()`** om aantal elementen in een lijst te bepalen

```
>>> len(['spam!', 1, [1, 2, 3],
...      ['Brie', 'Roquefort', 'Pol le Veq']])
4
```

traversal2.py

```
gassen = ['H','He','Ne','Ar','Kr','Xe','Rn','Uuo']

i = 0
while i < len(gassen):
    print(gassen[i])
    i += 1
```

UNIVERSITEIT GENT

Elementen overlopen

- **list traversal**: verwerkingspatroon waarbij elementen van een lijst één voor één doorlopen worden
  - gebruik ingebouwde functie **`len()`** om aantal elementen in een lijst te bepalen

traversal3.py

```
gassen = ['H','He','Ne','Ar','Kr','Xe','Rn','Uuo']

for i in range(len(gassen)):
    print(gassen[i])
```

UNIVERSITEIT GENT

Elementen overlopen

- **list traversal**: verwerkingspatroon waarbij elementen van een lijst één voor één doorlopen worden
  - gebruik ingebouwde functie **`len()`** om aantal elementen in een lijst te bepalen
  - lijsten zijn iteratorobjecten (*iterable objects*)
    - elementen kunnen rechtstreeks met for-lus doorlopen worden

traversal4.py

```
gassen = ['H','He','Ne','Ar','Kr','Xe','Rn','Uuo']

for gas in gassen:
    print(gas)
```

UNIVERSITEIT GENT

Elementen overlopen

- **list traversal**: verwerkingspatroon waarbij elementen van een lijst één voor één doorlopen worden
  - gebruik ingebouwde functie **`len()`** om aantal elementen in een lijst te bepalen
  - lijsten zijn iteratorobjecten (*iterable objects*)

traversal5.py

```
getallen = [1, 2, 3, 4, 5]

for index in range(len(getallen)):
    getallen[index] = getallen[index] ** 2

print(getallen)
```

UNIVERSITEIT GENT

Elementen overlopen

- **list traversal**: verwerkingspatroon waarbij elementen van een lijst één voor één doorlopen worden
  - gebruik ingebouwde functie **`len()`** om aantal elementen in een lijst te bepalen
  - lijsten zijn iteratorobjecten (*iterable objects*)
    - ingebouwde functie **`enumerate()`** geeft iterator terug die zowel index als waarde van een element teruggeeft

traversal6.py

```
getallen = [1, 2, 3, 4, 5]

for index, waarde in enumerate(getallen):
    getallen[index] = waarde ** 2

print(getallen)
```

UNIVERSITEIT GENT

Lidmaatschap

- ***element* in *lijst***
  - Booleaanse expressie is **`True`** als het element voorkomt in de lijst

```
>>> from klasse import klasse
>>> klasse('Po')
'halfgeleider'
>>> klasse('Cr')
'metaal'
```

klasse.py

```
def klasse(element):
    if element in ['He','Ne','Ar','Kr','Xe','Rn','Uuo']:
        return 'edelgas'
    elif element in ['B','Si','Ge','As','Sb','Te','Po','At','Uus']:
        return 'halfgeleider'
    elif element in ['H','C','N','O','F','P','S','Cl','Se','Br','I']:
        return 'niet-metaal'
    else:
        return 'metaal'
```

UNIVERSITEIT GENT

Lidmaatschap

- ***element* in *lijst***
  - Booleaanse expressie is **`True`** als het element voorkomt in de lijst

```
>>> from klasse2 import klasse
>>> klasse('Cl')
'niet-metaal'
>>> klasse('Xe')
'edelgas'
```

klasse2.py

```
def klasse(element):
    if element in 'HeNeArKrXeRnUuo':
        return 'edelgas'
    elif element in 'BSiGeAsSbTePoAtUus':
        return 'halfgeleider'
    elif element in 'HCNOFPSClSeBrI':
        return 'niet-metaal'
    else:
        return 'metaal'
```

UNIVERSITEIT GENT

Lijsten samenvoegen

- operator **+**: twee lijsten samenvoegen
- operator *****: lijst *n* keer herhalen

```
>>> element = 'koolstof'
>>> massa = '14'
>>> print(element + '-' + massa)
koolstof-14
>>> lanthaniden = ['Ce', 'Pr', 'Nd']
>>> actiniden = ['Th', 'Pa', 'U']
>>> samen = lanthaniden + actiniden
>>> print(samen)
['Ce', 'Pr', 'Nd', 'Th', 'Pa', 'U']
>>> print(actiniden * 3)
['Th', 'Pa', 'U', 'Th', 'Pa', 'U', 'Th', 'Pa', 'U']
```

UNIVERSITEIT GENT

Lijsten samenvoegen

- operator **+**: twee lijsten samenvoegen
- operator *****: lijst *n* keer herhalen
- **`list(`*`string`*`)`** : maakt lijst met karakters van **`string`**

```
>>> actiniden + 'NP'
TypeError: can only concatenate list (not "str") to list
>>> actiniden + ['NP']
['Th', 'Pa', 'U', 'NP']
>>> water = 'H2O'
>>> list(water)
['H', '2', 'O']
>>> actiniden
['Th', 'Pa', 'U']
>>> '-'.join(actiniden)
'Th-Pa-U'
```

UNIVERSITEIT GENT

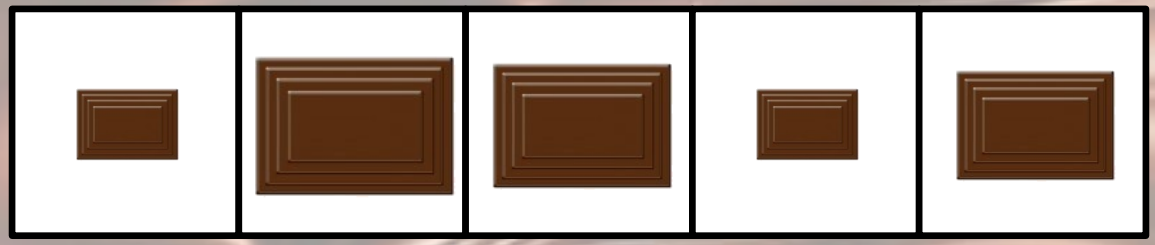
Iterator van integers maken

- functie **`range()`**
  - negatieve stapgrootte → waarde van eerste argument moet groter zijn dan waarde van tweede argument

```
>>> list(range(1, 5))
[1, 2, 3, 4]
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(1, 10, 2))
[1, 3, 5, 7, 9]
>>> list(range(20, 4, -5))
[20, 15, 10, 5]
>>> list(range(10, 20, -5))
[]
```

UNIVERSITEIT GENT

Chocoladerepen



2 9 8 2 7

Veranderlijkheid

Veranderlijkheid

- strings zijn onveranderlijk (*immutable*)
- lijsten zijn veranderlijk (*mutable*)
  - **item assignment**: toekenning aan element van lijst

```
>>> element = 'cesium'
>>> element[2] = 'r'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> element = list(element)
>>> element
['c', 'e', 's', 'i', 'u', 'm']
>>> element[2] = 'r'
>>> element
['c', 'e', 'r', 'i', 'u', 'm']
>>> ''.join(element)
'cerium'
```

UNIVERSITEIT GENT

Veranderlijkheid

- strings zijn onveranderlijk (*immutable*)
- lijsten zijn veranderlijk (*mutable*)
  - **slice assignment**: toekenning aan slice van lijst

```
>>> element
['c', 'e', 'r', 'i', 'u', 'm']
>>> element[1] = 'h'
>>> element[3:5] = ['o', 'o']
>>> element
['c', 'h', 'r', 'o', 'o', 'm']
>>> ''.join(element)
'chroom'
>>> element[4:4] = ['m', 'o', 's', 'o']
>>> element
['c', 'h', 'r', 'o', 'm', 'o', 's', 'o', 'o', 'm']
>>> ''.join(element)
'chromosoom'
```

UNIVERSITEIT GENT

Elementen verwijderen

- slice vervangen door lege lijst
- sleutelwoord **`del`**

```
>>> element = list('ytterbium')
>>> element[:2] = []
>>> element
['t', 'e', 'r', 'b', 'i', 'u', 'm']
>>> element[:1] = []
>>> element
['e', 'r', 'b', 'i', 'u', 'm']
>>> element = list('ytterbium')
>>> del element[:2]
>>> element
['t', 'e', 'r', 'b', 'i', 'u', 'm']
>>> del element[:1]
>>> element
['e', 'r', 'b', 'i', 'u', 'm']
```

UNIVERSITEIT GENT

Ytterby, het dorp in Zweden waar 4 elementen in het periodiek system naar vernoemd zijn — yttrium(Y), erbium(Er), terbium(Tb) en ytterbium(Yb). Daarnaast zijn de ontdekkingen van nog vier elementen (Sc, Ho, Tm, Gd) terug te leiden naar de dorpsgroeve.

Circadiale permutator

Huiswerk (volgende les)

- handboek: lees hoofdstuk 7 (lijsten)
- voorbeeldopgaven volgende les (reeks 6)
  - Manhattan
  - Heremietkreeften
  - Mijnenveger

Huiswerk (volgende les)

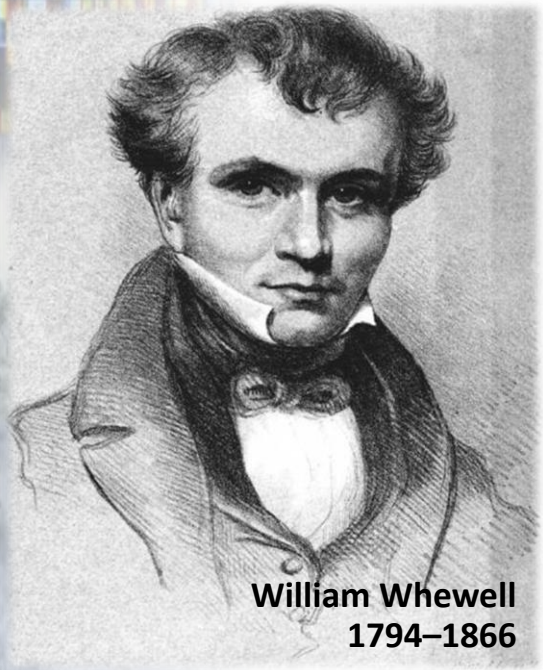
- bestudeer lijstmethoden: help(list)
- werkcolleges 7 November 2025 = evaluatie
  - variabelen, expressies en statements
  - voorwaardelijke opdrachten
  - controlelussen
  - strings
  - functies

UNIVI
GENT

Vragen of opmerkingen?

UNIVERSITEIT GENT

The sky is the limit …



William Whewell
1794–1866

"Every failure is a step towards success."

— William Whewell

UNIVERSITEIT GENT