Universiteit Gent
Faculteit Wetenschappen

Programmeren in Python

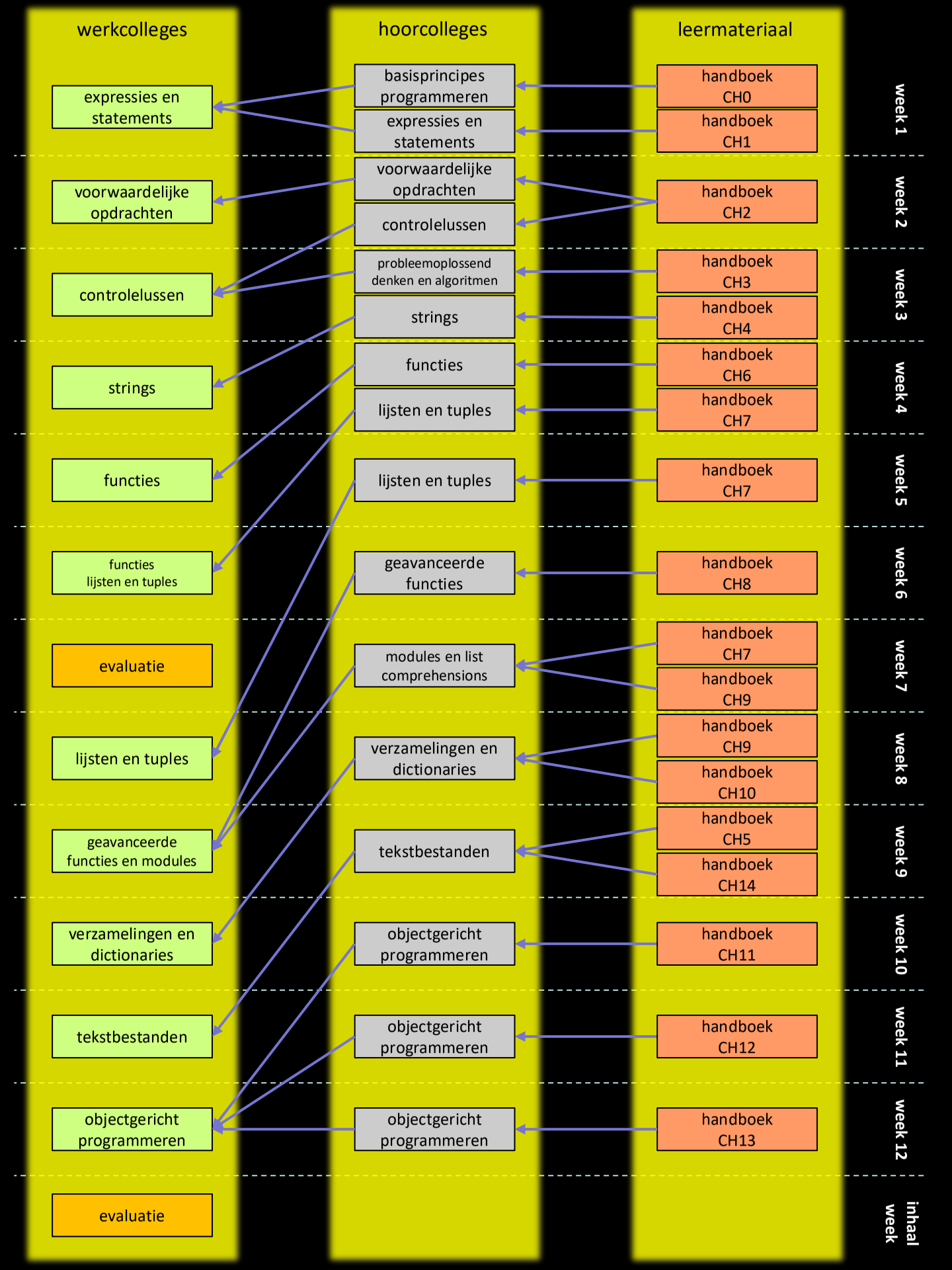
great Python
your appetite
is hard to swallow

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

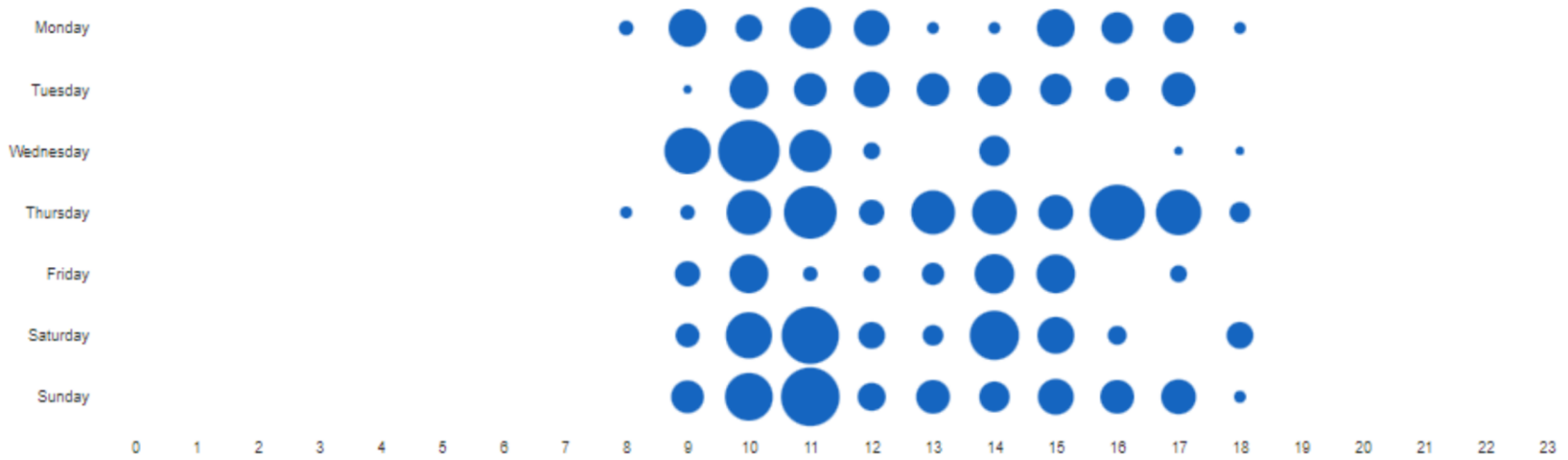
Hermione Granger
Hogwarts School of Witchcraft and Wizardry

Statistics

1057	61	6
Submissions	Correct exercises	Unfinished exercises

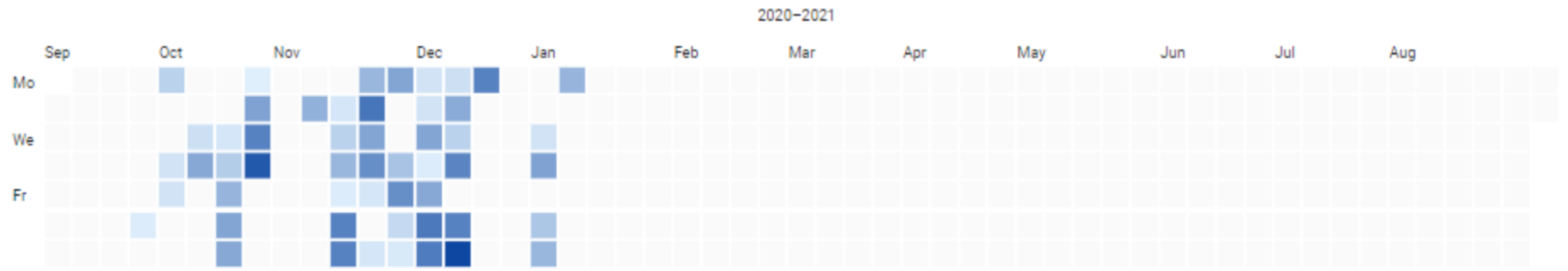
Punchcard

This graph shows you on which moment in the week submissions have taken place.



Heatmap

This graph shows you on which days most submissions took place.



Ron Weasley
Hogwarts School of Witchcraft and Wizardry

Statistics

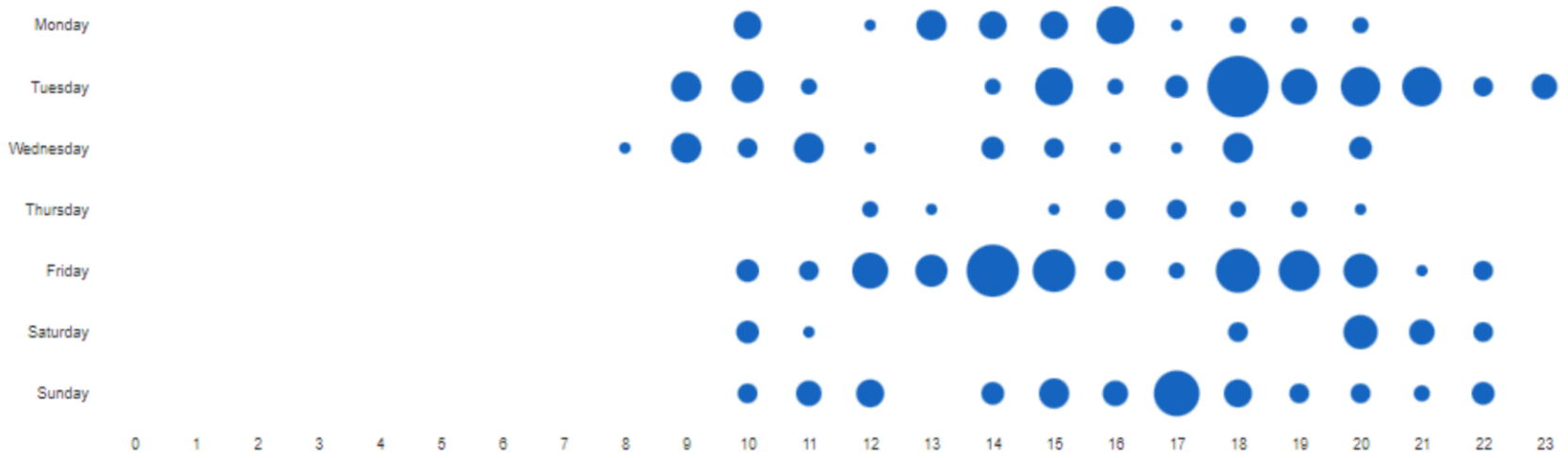
400
Submissions

62
Correct exercises

10
Unfinished exercises

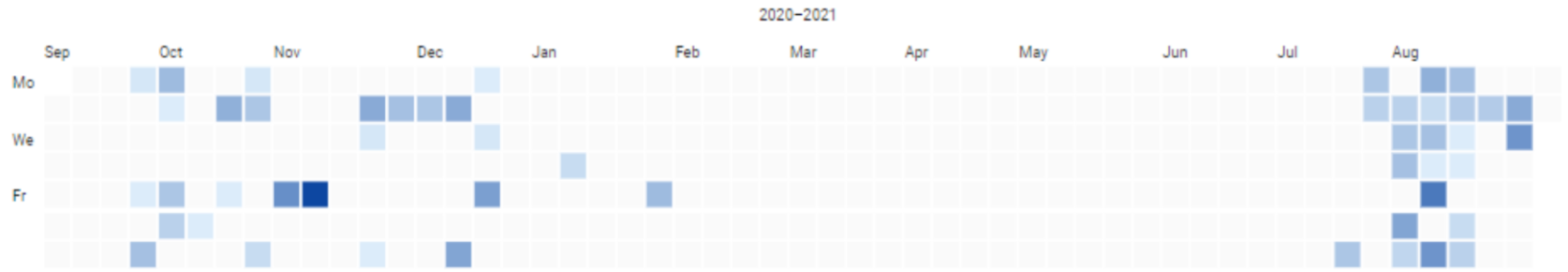
Punchcard

This graph shows you on which moment in the week submissions have taken place.

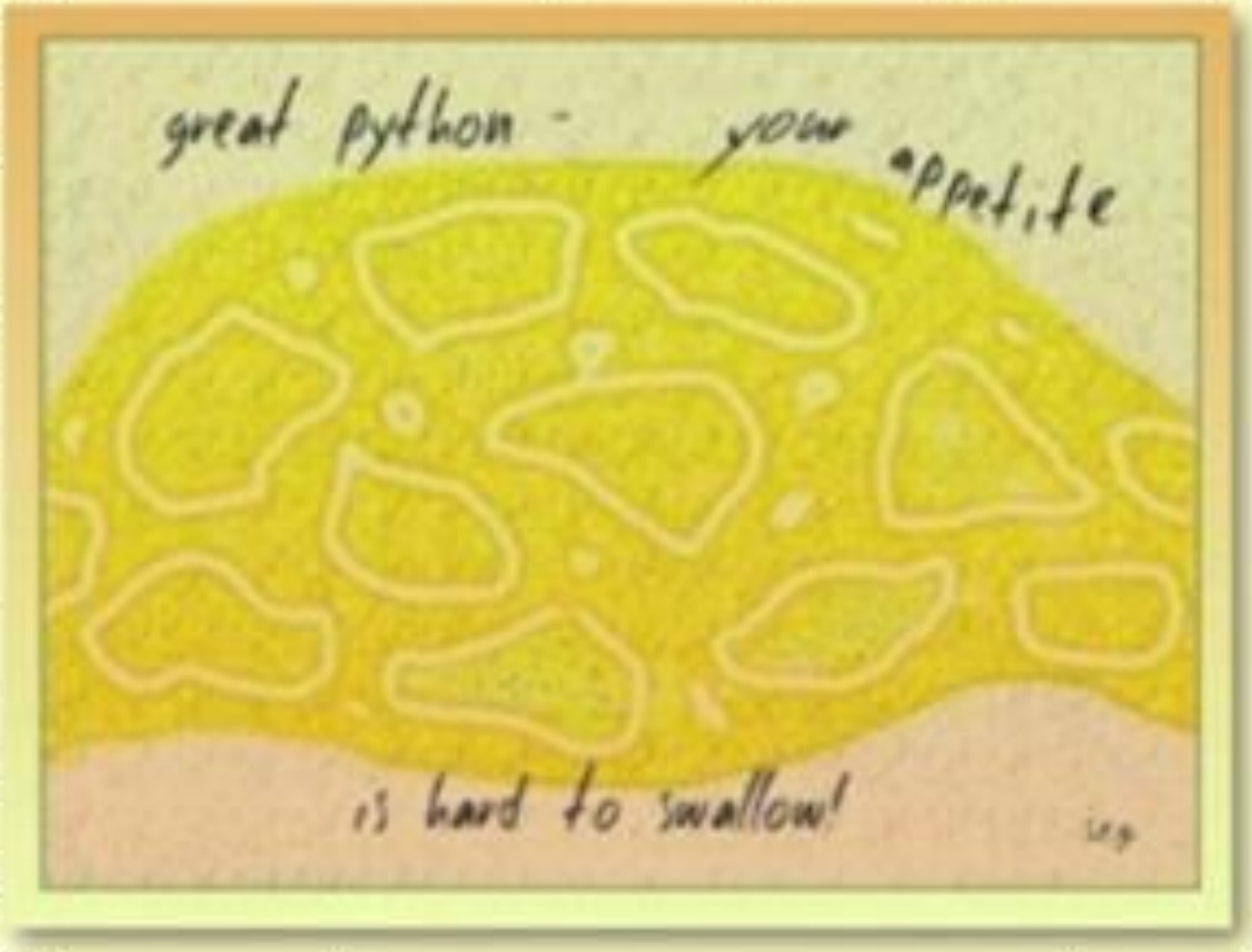


Heatmap

This graph shows you on which days most submissions took place.



Invoer en uitvoer



Invoer en uitvoer

- tot nu toe …
  - uitschrijven wat programma doet: **`print()`** functie
  - informatie opvragen aan gebruiker: **`input()`** functie

"Hoe kan ik informatie opslaan in een bestand ?"

"Hoe kan ik informatie lezen uit een bestand ?"

- oplossing van Python lijkt sterk op die van programmeertaal C
  - bestand bestaat uit reeks bytes
  - vaak handiger om tekstbestanden te bekijken als reeks die bestaat uit meerdere regels tekst

UNIVERSITEIT GENT

Lezen uit bestanden

- hoeveel ~~karakters~~ bytes staan er in dit bestand ?

haiku.txt

```
Three things are certain:
Death, taxes and lost data.
Guess which has occurred.

You step in the stream,
but the water has moved on.
This page is not here.

Having been erased,
The document you're seeking
Must now be retyped.

A crash reduces
your expensive computer
to a simple stone.
```

veronderstel voorlopig:
1 karakter = 1 byte

UNIVERSITEIT GENT

bron: Error messages in Haiku? (GNU humor collection)

Lezen uit bestanden

- hoeveel bytes staan er in dit bestand ?

```
>>> invoer = open('haiku.txt', 'r')
>>> data = invoer.read()
>>> invoer.close()
>>> len(data)
286
```

UNIVERSITEIT GENT

Lezen uit bestanden

- hoeveel bytes staan er in dit bestand ?

bestandsobject aanmaken

```
>>> invoer = open('haiku.txt', 'r')
>>> data = invoer.read()
>>> invoer.close()
>>> len(data)
286
```

UNIVERSITEIT GENT

Lezen uit bestanden

- hoeveel bytes staan er in dit bestand ?

locatie van bestand waaraan object gekoppeld wordt

```
>>> invoer = open('haiku.txt', 'r')
>>> data = invoer.read()
>>> invoer.close()
>>> len(data)
286
```

UNIVERSITEIT GENT

Lezen uit bestanden

- hoeveel bytes staan er in dit bestand ?

```
>>> invoer = open('haiku.txt', 'r')
>>> data = invoer.read()
>>> invoer.close()
>>> len(data)
286
```

om te lezen ...

UNIVERSITEIT GENT

Lezen uit bestanden

- hoeveel bytes staan er in dit bestand ?

verwijst nu naar het bestandsobject

```
>>> invoer = open('haiku.txt', 'r')
>>> data = invoer.read()
>>> invoer.close()
>>> len(data)
286
```

UNIVERSITEIT GENT

Lezen uit bestanden

- hoeveel bytes staan er in dit bestand ?

leest volledige inhoud van bestand in string

```
>>> invoer = open('haiku.txt', 'r')
>>> data = invoer.read()
>>> invoer.close()
>>> len(data)
286
```

UNIVERSITEIT GENT

Lezen uit bestanden

- hoeveel bytes staan er in dit bestand ?

bevat nu een kopie van alle bytes uit bestand

```
>>> invoer = open('haiku.txt', 'r')
>>> data = invoer.read()
>>> invoer.close()
>>> len(data)
286
>>> print(data)
Three things are certain:
Death, taxes and lost data.
Guess which has occurred.
...
```

UNIVERSITEIT GENT

Lezen uit bestanden

- hoeveel bytes staan er in dit bestand ?

```
>>> invoer = open('haiku.txt', 'r')
>>> data = invoer.read()
>>> invoer.close()
>>> len(data)
286
>>> print(data)
Three things are certain:
Death, taxes and lost data.
Guess which has occurred.
...
```

object loskoppelen van bestand

UNIVERSITEIT GENT

Lezen uit bestanden

- hoeveel bytes staan er in dit bestand ?

```
>>> invoer = open('haiku.txt', 'r')
>>> data = invoer.read()
>>> invoer.close()
>>> len(data)
286
>>> print(data)
Three things are certain:
Death, taxes and lost data.
Guess which has occurred.
...
```

uitschrijven hoeveel bytes ~~karakters~~ er werden uitgelezen

UNIVERSITEIT GENT

Lezen uit bestanden

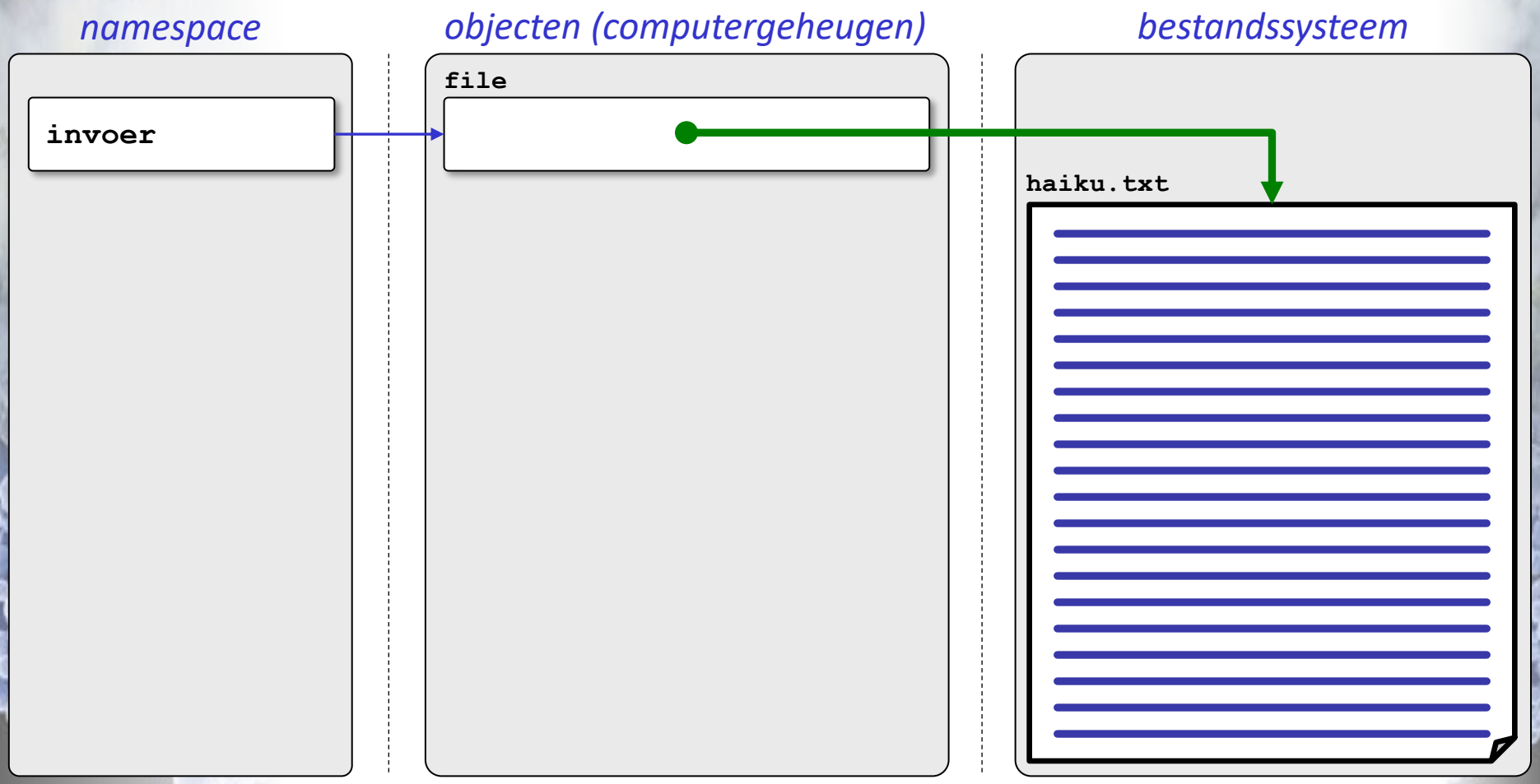
- hoeveel bytes staan er in dit bestand ?

```
>>> invoer = open('haiku.txt', 'r')
>>> data = invoer.read()
>>> invoer.close()
>>> len(data)
286
>>> print(data)
Three things are certain:
Death, taxes and lost data.
Guess which has occurred.
...
```

UNIVERSITEIT GENT

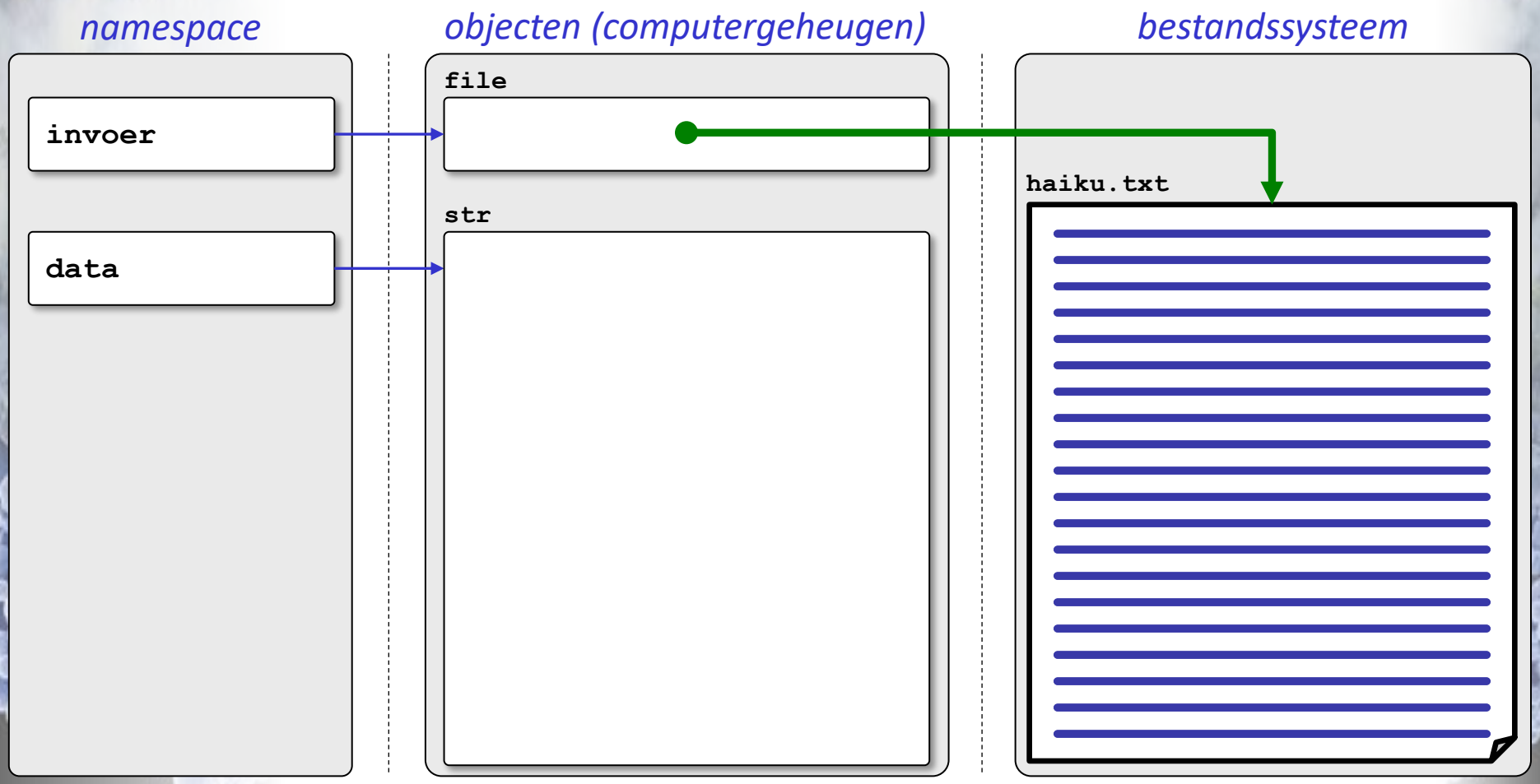
Alles in één keer lezen …

```
>>> invoer = open('haiku.txt', 'r')
```



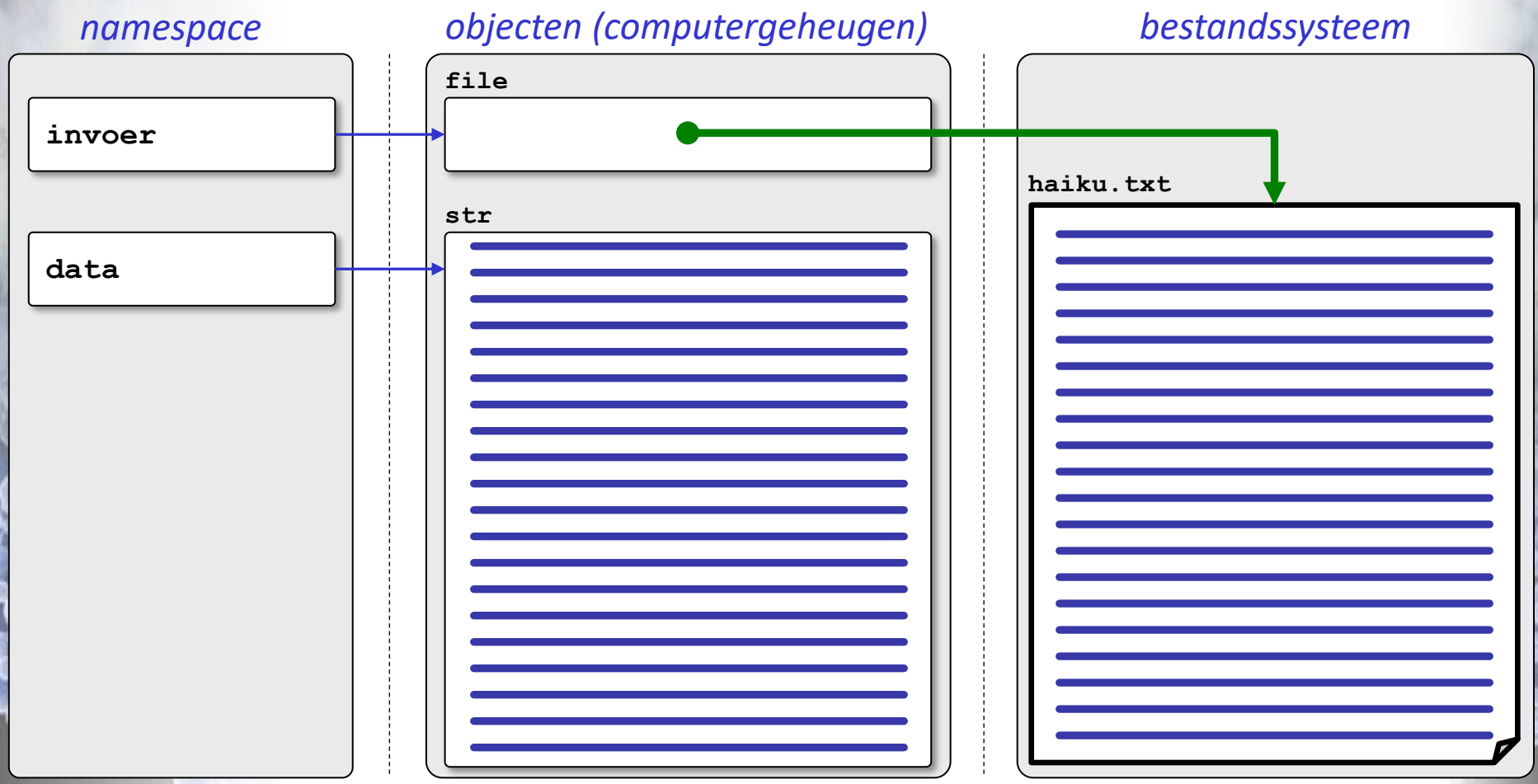
Alles in één keer lezen …

```
>>> data = invoer.read()
```



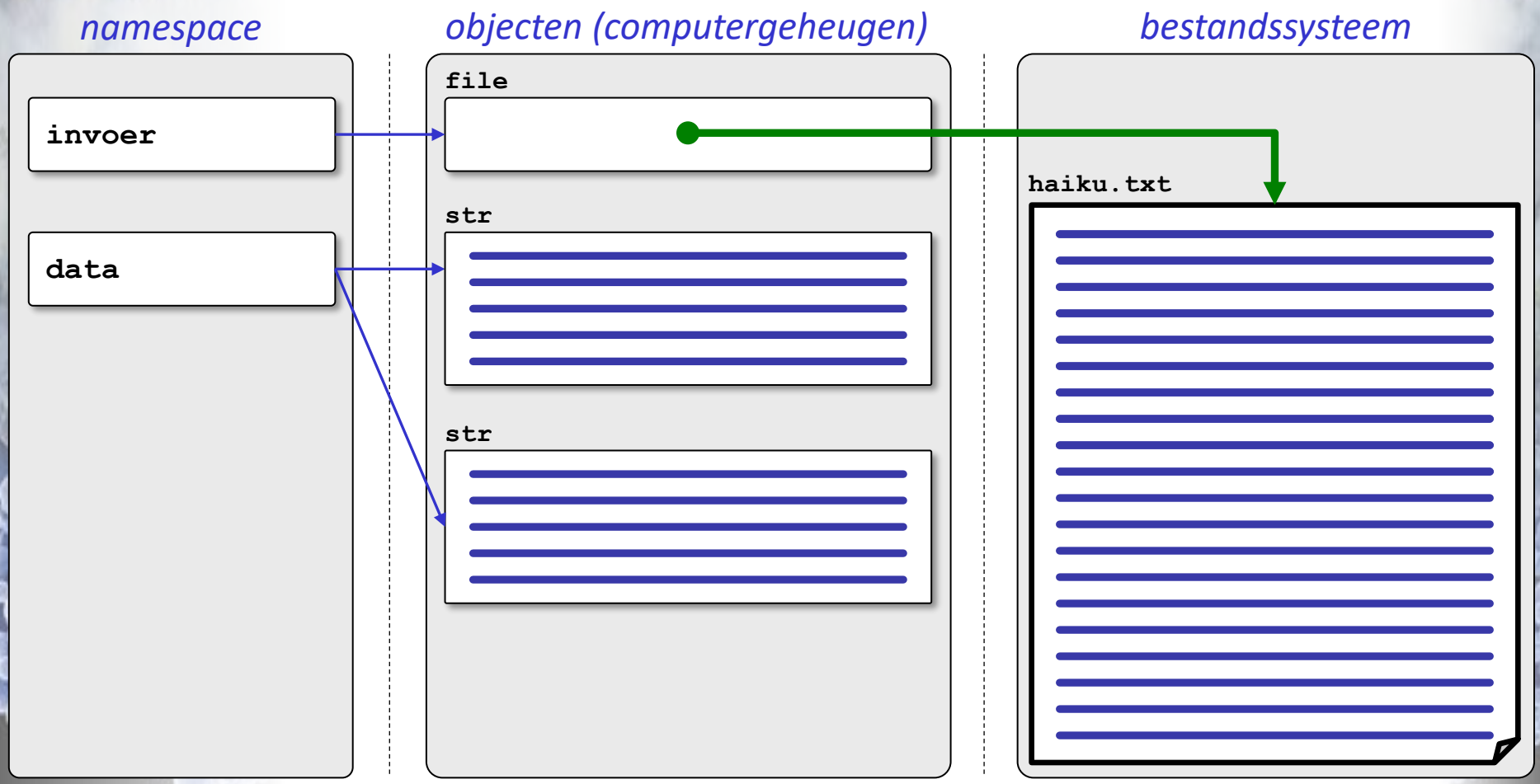
Alles in één keer lezen …

```
>>> invoer.close()
```



… of in verschillende stukken …

- beter om grote bestanden in stukken in te lezen



… of in verschillende stukken …

- beter om grote bestanden in stukken in te lezen

```
>>> invoer = open('haiku.txt', 'r')
>>> data = invoer.read(64)
>>> while data:
...     print(len(data))
...     data = invoer.read(64)
```

UNIVERSITEIT GENT

… of in verschillende stukken …

- beter om grote bestanden in stukken in te lezen

```
>>> invoer = open('haiku.txt', 'r')
>>> data = invoer.read(64)
>>> while data:
...     print(len(data))
...     data = invoer.read(64)
```

… of in verschillende stukken …

- beter om grote bestanden in stukken in te lezen

```
>>> invoer = open('haiku.txt', 'r')
>>> data = invoer.read(64)
>>> while data:
...     print(len(data))
...     data = invoer.read(64)
```

herhaal zolang er nog iets gelezen wordt

UNIVERSITEIT GENT

… of in verschillende stukken …

- beter om grote bestanden in stukken in te lezen

```
>>> invoer = open('haiku.txt', 'r')
>>> data = invoer.read(64)
>>> while data:
...     print(len(data))
...     data = invoer.read(64)
64
```

doe iets met de gegevens

UNIVERSITEIT GENT

… of in verschillende stukken …

- beter om grote bestanden in stukken in te lezen

```
>>> invoer = open('haiku.txt', 'r')
>>> data = invoer.read(64)
>>> while data:
...     print(len(data))
...     data = invoer.read(64)
64
64
64
64
30
```

lees volgende stuk in

UNIVERSITEIT GENT

… of in verschillende stukken …

- beter om grote bestanden in stukken in te lezen

```
>>> invoer = open('haiku.txt', 'r')
>>> data = invoer.read(64)
>>> while data:
...     print(len(data))
...     data = invoer.read(64)
64
64
64
64
30
>>> len(data)
0
>>> invoer.close()
```

dit scenario is vrij ongebruikelijk voor tekstbestanden

UNIVERSITEIT GENT

… of regel per regel

- doorgaans worden tekstbestanden regel per regel ingelezen

lezen3.py

```
invoer = open('haiku.txt', 'r')
bytes, regels = 0, 0
regel = invoer.readline()
while regel:
    regels += 1
    bytes += len(regel)
    regel = invoer.readline()
invoer.close()
print(f'gemiddelde: {bytes / regels}')
```

UNIVERSITEIT GENT

… of regel per regel

- doorgaans worden tekstbestanden regel per regel ingelezen

lezen3.py

```
invoer = open('haiku.txt', 'r')
bytes, regels = 0, 0
regel = invoer.readline()
while regel:
    regels += 1
    bytes += len(regel)
    regel = invoer.readline()
invoer.close()
print(f'gemiddelde: {bytes / regels}')
```

één enkele regel inlezen

UNIVERSITEIT GENT

… of regel per regel

- doorgaans worden tekstbestanden regel per regel ingelezen

lezen3.py

```
invoer = open('haiku.txt', 'r')
bytes, regels = 0, 0
regel = invoer.readline()
while regel:
    regels += 1
    bytes += len(regel)
    regel = invoer.readline()
invoer.close()
print(f'gemiddelde: {bytes / regels}')
```

herhaal zolang er regels uit bestand gelezen worden

UNIVERSITEIT GENT

… of regel per regel

- doorgaans worden tekstbestanden regel per regel ingelezen

lezen3.py

```
invoer = open('haiku.txt', 'r')
bytes, regels = 0, 0
regel = invoer.readline()
while regel:
    regels += 1
    bytes += len(regel)
    regel = invoer.readline()
invoer.close()
print(f'gemiddelde: {bytes / regels}')
```

lees volgende regel in

UNIVERSITEIT GENT

… of regel per regel

- doorgaans worden tekstbestanden regel per regel ingelezen

lezen3.py

```
invoer = open('haiku.txt', 'r')
bytes, regels = 0, 0
regel = invoer.readline()
while regel:
    regels += 1
    bytes += len(regel)
    regel = invoer.readline()
invoer.close()
print(f'gemiddelde: {bytes / regels}')
```

gemiddelde: 19.066667

UNIVERSITEIT GENT

… of regel per regel

- doorgaans worden tekstbestanden regel per regel ingelezen

lezen3.py

```
invoer = open('haiku.txt', 'r')
bytes, regels = 0, 0
regel = invoer.readline()
while regel:
    regels += 1
    bytes += len(regel)
    regel = invoer.readline()
invoer.close()
print(f'gemiddelde: {bytes / regels}')
```

gemiddelde: 19.066667

UNIVERSITEIT GENT

Lijst van regels

- soms handiger om alle regels in één keer uit te lezen
  - als aanslag op geheugen hierdoor niet te groot wordt

lezen4.py

```
invoer = open('haiku.txt', 'r')
inhoud = invoer.readlines()
invoer.close()
bytes, regels = 0, 0
for regel in inhoud:
    regels += 1
    bytes += len(regel)
print(f'gemiddelde: {bytes / regels}')
```

UNIVERSITEIT GENT

Lijst van regels

- soms handiger om alle regels in één keer uit te lezen
  - als aanslag op geheugen hierdoor niet te groot wordt

lezen4.py

```
invoer = open('haiku.txt', 'r')
inhoud = invoer.readlines()
invoer.close()
bytes, regels = 0, 0
for regel in inhoud:
    regels += 1
    bytes += len(regel)
print(f'gemiddelde: {bytes / regels}')
```

alle regels in één keer uitgelezen en in lijst geplaatst

UNIVERSITEIT GENT

Lijst van regels

- soms handiger om alle regels in één keer uit te lezen
  - als aanslag op geheugen hierdoor niet te groot wordt

lezen4.py

```
invoer = open('haiku.txt', 'r')
inhoud = invoer.readlines()
invoer.close()
bytes, regels = 0, 0
for regel in inhoud:
    regels += 1
    bytes += len(regel)
print(f'gemiddelde: {bytes / regels}')
```

regels overlopen met for loop

UNIVERSITEIT GENT

Lijst van regels

- soms handiger om alle regels in één keer uit te lezen
  - als aanslag op geheugen hierdoor niet te groot wordt

lezen4.py

```
invoer = open('haiku.txt', 'r')
inhoud = invoer.readlines()
invoer.close()
bytes, regels = 0, 0
for regel in inhoud:
    regels += 1
    bytes += len(regel)
print(f'gemiddelde: {bytes / regels}')
```

```
gemiddelde: 19.066667
```

UNIVERSITEIT GENT

Lijst van regels

- soms handiger om alle regels in één keer uit te lezen
  - als aanslag op geheugen hierdoor niet te groot wordt
  - veelgebruikt scenario: "lees regels in lijst" + "lijst overlopen"
  - korte Python syntaxis: "regels in bestand overlopen"
    - bestandsobject is *iterable*

lezen4.py

```
invoer = open('haiku.txt', 'r')
inhoud = invoer.readlines()
invoer.close()
bytes, regels = 0, 0
for regel in inhoud:
    regels += 1
    bytes += len(regel)
print(f'gemiddelde: {bytes / regels}')
```

```
gemiddelde: 19.066667
```

UNIVERSITEIT GENT

Bestanden doorlopen

- soms handiger om alle regels in één keer uit te lezen
  - als aanslag op geheugen hierdoor niet te groot wordt
  - veelgebruikt scenario: "lees regels in lijst" + "lijst overlopen"
  - korte Python syntaxis: "regels in bestand overlopen"
    - bestandsobject is *iterable*

lezen5.py

```
invoer = open('haiku.txt', 'r')
bytes, regels = 0, 0
for regel in invoer:
    regels += 1
    bytes += len(regel)
invoer.close()
print(f'gemiddelde: {bytes / regels}')
```

UNIVERSITEIT GENT

Bestanden doorlopen

- soms handiger om alle regels in één keer uit te lezen
  - als aanslag op geheugen hierdoor niet te groot wordt
  - veelgebruikt scenario: "lees regels in lijst" + "lijst overlopen"
  - korte Python syntaxis: "regels in bestand overlopen"
    - bestandsobject is *iterable*

lezen5.py

```
invoer = open('haiku.txt', 'r')
bytes, regels = 0, 0
for regel in invoer:
    regels += 1
    bytes += len(regel)
invoer.close()
print(f'gemiddelde: {bytes / regels}')
```

regels in bestand worden één voor één toegekend aan lusvariabele

UNIVERSITEIT GENT

Bestanden doorlopen

俳

- soms handiger om alle regels in één keer uit te lezen
  - als aanslag op geheugen hierdoor niet te groot wordt
  - veelgebruikt scenario: "lees regels in lijst" + "lijst overlopen"
  - korte Python syntaxis: "regels in bestand overlopen"
  - "open" bestanden "op het einde" automatisch afgesloten

lezen5.py

```
invoer = open('haiku.txt', 'r')
bytes, regels = 0, 0
for regel in invoer:
    regels += 1
    bytes += len(regel)
invoer.close()
print(f'gemiddelde: {bytes / regels}')
```

UNIVERSITEIT GENT

Bestanden doorlopen

- soms handiger om alle regels in één keer uit te lezen
  - als aanslag op geheugen hierdoor niet te groot wordt
  - veelgebruikt scenario: "lees regels in lijst" + "lijst overlopen"
  - korte Python syntaxis: "regels in bestand overlopen"
  - "open" bestanden "op het einde" automatisch afgesloten

lezen6.py

```
bytes, regels = 0, 0
for regel in open('haiku.txt', 'r'):
    regels += 1
    bytes += len(regel)
print(f'gemiddelde: {bytes / regels}')
```

UNIVERSITEIT GENT

Bestanden doorlopen

- soms handiger om alle regels in één keer uit te lezen
  - als aanslag op geheugen hierdoor niet te groot wordt
  - veelgebruikt scenario: "lees regels in lijst" + "lijst overlopen"
  - korte Python syntaxis: "regels in bestand overlopen"
  - "open" bestanden "op het einde" automatisch afgesloten
  - **`with`**-blok zorgt voor expliciet "einde" (= bestand afsluiten)

lezen7.py

```
bytes, regels = 0, 0
with open('haiku.txt', 'r') as invoer:
    for regel in invoer:
        regels += 1
        bytes += len(regel)
print(f'gemiddelde: {bytes / regels}')
```

UNIVERSITEIT GENT

Bestanden doorlopen

- soms handiger om alle regels in één keer uit te lezen
  - als aanslag op geheugen hierdoor niet te groot wordt
  - veelgebruikt scenario: "lees regels in lijst" + "lijst overlopen"
  - korte Python syntaxis: "regels in bestand overlopen"
  - "open" bestanden "op het einde" automatisch afgesloten
  - **`with`**-blok zorgt voor expliciet "einde" (= bestand afsluiten)
  - **`encoding`** zorgt voor omzetting van bytes naar karakters

lezen8.py

```
bytes, regels = 0, 0
with open('haiku.txt', encoding='utf-8') as invoer:
    for regel in invoer:
        regels += 1
        bytes += len(regel)
print(f'gemiddelde: {bytes / regels}')
```

UNIVERSITEIT GENT

Garantie voor afsluiten

- bestanden openen → bestand terug afsluiten

lezen7.py

```
invoer = open('haiku.txt', 'r')
try:
    bytes, regels = 0, 0
    for regel in invoer:
        regels += 1
        bytes += len(regel)
    print(f'gemiddelde: {bytes / regels}')
finally:
    invoer.close()
```

UNIVERSITEIT GENT

Garantie voor afsluiten

- bestanden openen → bestand terug afsluiten
  - **`with`**: *resource management* + *exception handling*

lezen8.py

```
with open('haiku.txt', 'r') as invoer:
    bytes, regels = 0, 0
    for regel in invoer:
        regels += 1
        bytes += len(regel)
    print(f'gemiddelde: {bytes / regels}')
```

UNIVERSITEIT GENT

Regeleindes

Regeleindes

000	NUL	033	!	066	B	099	c	132	ä	165	Ñ	198	ã	231	þ
001	Start Of Header (SOH)	034	"	067	C	100	d	133	à	166	ª	199	Ã	232	Þ
002	Start Of Text (STX)	035	#	068	D	101	e	134	å	167	º	200	╚	233	Ú
003	End Of Text (ETX)	036	$	069	E	102	f	135	ç	168	¿	201	╔	234	Û
004	End Of Transmission (EOT)	037	%	070	F	103	g	136	ê	169	®	202	╩	235	Ù
005	Enquiry	038	&	071	G	104	h	137	ë	170	¬	203	╦	236	ý
006	Acknowledge (ACK)	039	'	072	H	105	i	138	è	171	½	204	╠	237	Ý
007	Bell	040	(	073	I	106	j	139	ï	172	¼	205	═	238	¯
008	Backspace (BS)	041	)	074	J	107	k	140	î	173	¡	206	╬	239	´
009	Horizontal Tab	042	*	075	K	108	l	141	ì	174	«	207	¤	240	-
010	Line Feed **(LF)**	043	+	076	L	109	m	142	Ä	175	»	208	ð	241	±
011	Vertical Tab	044	,	077	M	110	n	143	Å	176	░	209	Ð	242	‗
012	Form Feed **(FF)**	045	-	078	N	111	o	144	É	177	▒	210	Ê	243	¾
013	Carriage Return **(CR)**	046	.	079	O	112	p	145	æ	178	▓	211	Ë	244	¶
014	Shift Out	047	/	080	P	113	q	146	Æ	179	│	212	È	245	§
015	Shift In	048	0	081	Q	114	r	147	ô	180	┤	213	ı	246	÷
016	Dataline Escape (DLE)	049	1	082	R	115	s	148	ö	181	Á	214	Í	247	¸
017	DC 1 (XON)	050	2	083	S	116	t	149	ò	182	Â	215	Î	248	°
018	DC 2	051	3	084	T	117	u	150	û	183	À	216	Ï	249	¨
019	DC 3 (XOFF)	052	4	085	U	118	v	151	ù	184	©	217	┘	250	·
020	DC 4	053	5	086	V	119	w	152	ÿ	185	╣	218	┌	251	¹
021	Negative Acknowledge (NAK)	054	6	087	W	120	x	153	Ö	186	║	219	█	252	³
022	Synchronous Idle	055	7	088	X	121	y	154	Ü	187	╗	220	▄	253	²
023	End Of Transmission Block	056	8	089	Y	122	z	155	ø	188	╝	221	¦	254	■
024	Cancel	057	9	090	Z	123	{	156	£	189	¢	222	Ì	255	
025	End Of Medium	058	:	091	[	124	|	157	Ø	190	¥	223	▀		
026	Substitude	059	;	092	\	125	}	158	×	191	┐	224	Ó		
027	Escape **(ESC)**	060	<	093	]	126	~	159	ƒ	192	└	225	ß		
028	File Seperator	061	=	094	^	127 (DEL)	⌂	160	á	193	┴	226	Ô		
029	Group Seperator	062	>	095	_	128	Ç	161	í	194	┬	227	Ò		
030	Record Seperator	063	?	096	`	129	ü	162	ó	195	├	228	õ		
031	Unit Seperator	064	@	097	a	130	é	163	ú	196	─	229	Õ		
032	*SPACE* (SP)	065	A	098	b	131	â	164	ñ	197	┼	230	µ		

UNIVERSITEIT GENT

Regeleindes

- UNIX en MS Windows gebruiken andere conventie om regeleindes aan te geven in tekstbestanden
  - UNIX: line feed (`'\n'`, newline, code 10)
  - MS Windows: carriage return (`'\r'`, code 13) + line feed

UNIVERSITEIT GENT

The End of Line Puzzle

Schrijven naar bestanden

Schrijven naar bestanden

- bestandsobjecten kunnen data naar bestanden schrijven
  - methode **`write()`**
  - methode **`writelines()`**

```
>>> uitvoer = open('elementen.txt', 'w')
>>> uitvoer.write('elementen')
>>> uitvoer.writelines(['He', 'Ne', 'Ar', 'Kr'])
>>> uitvoer.close()
```

UNIVERSITEIT GENT

Schrijven naar bestanden

- bestandsobjecten kunnen data naar bestanden schrijven
  - methode **`write()`**
  - methode **`writelines()`**

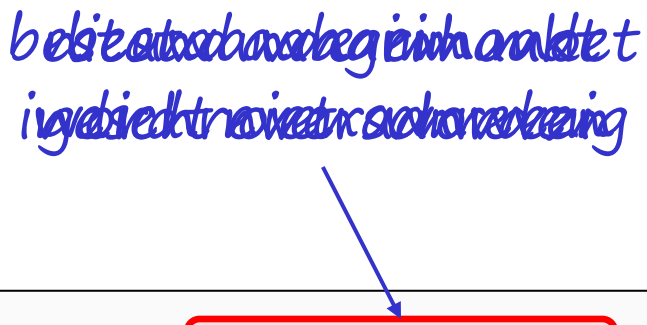
zelfde functie

```
>>> uitvoer = open('elementen.txt', 'w')
>>> uitvoer.write('elementen')
>>> uitvoer.writelines(['He', 'Ne', 'Ar', 'Kr'])
>>> uitvoer.close()
```

UNIVERSITEIT GENT

Schrijven naar bestanden

- bestandsobjecten kunnen data naar bestanden schrijven
  - methode **`write()`**
  - methode **`writelines()`**



```
>>> uitvoer = open('elementen.txt', 'w')
>>> uitvoer.write('elementen')
>>> uitvoer.writelines(['He', 'Ne', 'Ar', 'Kr'])
>>> uitvoer.close()
```

UNIVERSITEIT GENT

Schrijven naar bestanden

- bestandsobjecten kunnen data naar bestanden schrijven
  - methode **`write()`**
  - methode **`writelines()`**

om te schrijven …

```
>>> uitvoer = open('elementen.txt', 'w')
>>> uitvoer.write('elementen')
>>> uitvoer.writelines(['He', 'Ne', 'Ar', 'Kr'])
>>> uitvoer.close()
```

UNIVERSITEIT GENT

Schrijven naar bestanden

- bestandsobjecten kunnen data naar bestanden schrijven
  - methode **`write()`**
  - methode **`writelines()`**

één enkele string uitschrijven

```
>>> uitvoer = open('elementen.txt', 'w')
>>> uitvoer.write('elementen')
>>> uitvoer.writelines(['He', 'Ne', 'Ar', 'Kr'])
>>> uitvoer.close()
```

UNIVERSITEIT GENT

Schrijven naar bestanden

- bestandsobjecten kunnen data naar bestanden schrijven
  - methode **`write()`**
  - methode **`writelines()`**

elke string uit de lijst uitschrijven

```
>>> uitvoer = open('elementen.txt', 'w')
>>> uitvoer.write('elementen')
>>> uitvoer.writelines(['He', 'Ne', 'Ar', 'Kr'])
>>> uitvoer.close()
```

UNIVERSITEIT GENT

Schrijven naar bestanden

- bestandsobjecten kunnen data naar bestanden schrijven
  - Python schrijft enkel wat er gevraagd wordt
    - regeleinde (*newline*) moet expliciet geschreven worden: `'\n'`

elementen.txt

```
elementenHeNeArKr
```

```
>>> uitvoer = open('elementen.txt', 'w')
>>> uitvoer.write('elementen')
>>> uitvoer.writelines(['He', 'Ne', 'Ar', 'Kr'])
>>> uitvoer.close()
```

UNIVERSITEIT GENT

Schrijven naar bestanden

- bestandsobjecten kunnen data naar bestanden schrijven
  - Python schrijft enkel wat er gevraagd wordt
    - regeleinde (*newline*) moet expliciet geschreven worden: `'\n'`

'elementen\nHe\nNe\nAr\nKr\n'

elementen.txt

```
elementen
He
Ne
Ar
Kr
```

```
>>> uitvoer = open('elementen.txt', 'w')
>>> uitvoer.write('elementen\n')
>>> uitvoer.writelines(['He\n','Ne\n','Ar\n','Kr\n'])
>>> uitvoer.close()
```

UNIVERSITEIT GENT

Schrijven naar bestanden

- bestandsobjecten kunnen data naar bestanden schrijven
  - Python schrijft enkel wat er gevraagd wordt
    - regeleinde (*newline*) moet expliciet geschreven worden: **`'\n'`**
    - vaak handiger om **`file`** parameter van **`print`** te gebruiken

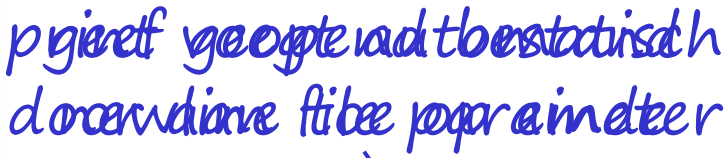
schrijven3.py

```
uitvoer = open('elementen.txt', 'w')
print('elementen', file=uitvoer)
for gas in ['He', 'Ne', 'Ar', 'Kr']:
    print(gas, file=uitvoer)
uitvoer.close()
```

UNIVERSITEIT GENT

Schrijven naar bestanden

- bestandsobjecten kunnen data naar bestanden schrijven
  - Python schrijft enkel wat er gevraagd wordt
    - regeleinde (*newline*) moet expliciet geschreven worden: **`'\n'`**
    - vaak handiger om **`file`** parameter van **`print`** te gebruiken



schrijven3.py

```
uitvoer = open('elementen.txt', 'w')
print('elementen', file=uitvoer)
for gas in ['He', 'Ne', 'Ar', 'Kr']:
    print(gas, file=uitvoer)
uitvoer.close()
```

UNIVERSITEIT GENT

Schrijven naar bestanden

- bestandsobjecten kunnen data naar bestanden schrijven
  - Python schrijft enkel wat er gevraagd wordt
    - regeleinde (*newline*) moet expliciet geschreven worden: **`'\n'`**
    - vaak handiger om **`file`** parameter van **`print`** te gebruiken

schrijven3.py

```
uitvoer = open('elementen.txt', 'w')
print('elementen', file=uitvoer)
for gas in ['He', 'Ne', 'Ar', 'Kr']:
    print(gas, file=uitvoer)
uitvoer.close()
```

UNIVERSITEIT GENT

Bestanden kopiëren

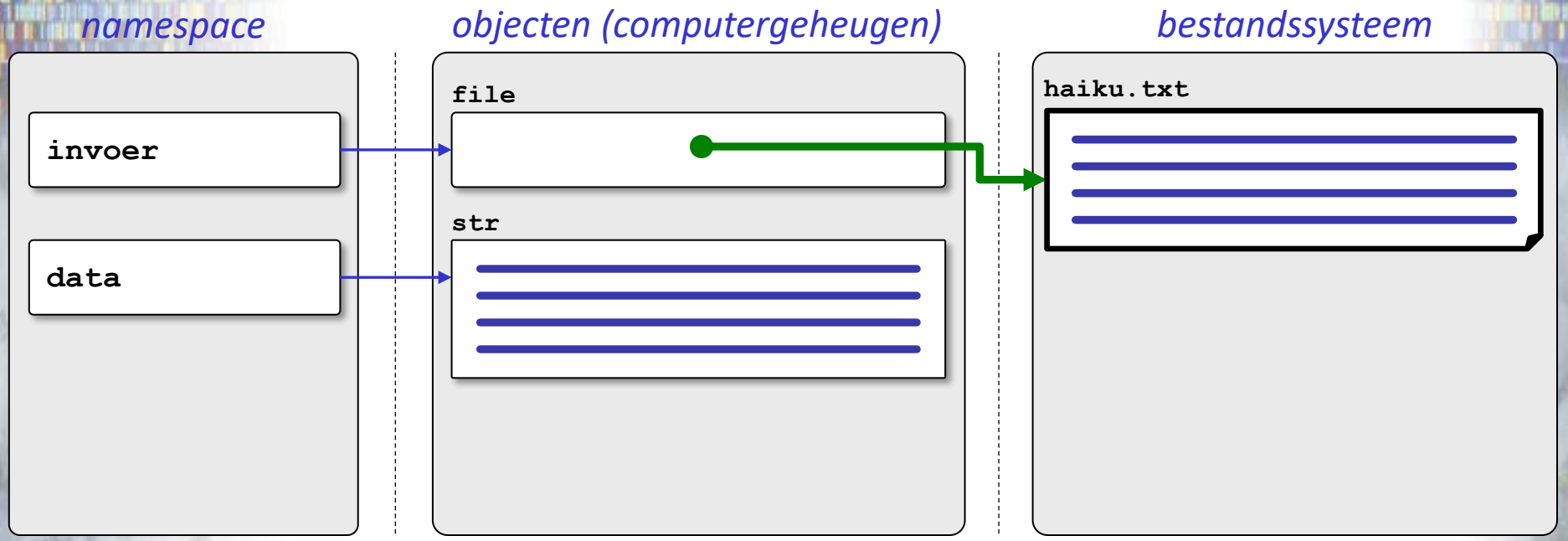
Bestanden kopiëren

kopie1.py

```
invoer = open('haiku.txt', 'r')
data = invoer.read()
invoer.close()
uitvoer = open('kopie.txt', 'w')
uitvoer.write(data)
uitvoer.close()
```

UNIVERSITEIT GENT

Bestanden kopiëren



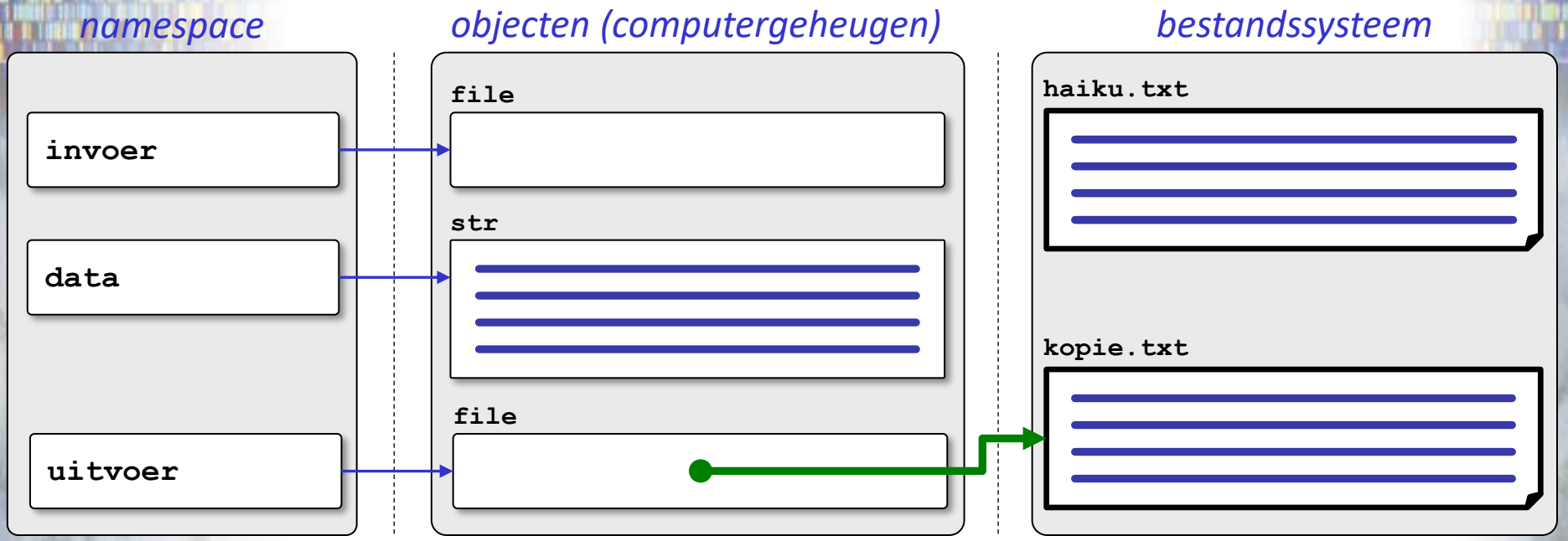
kopie1.py

```
invoer = open('haiku.txt', 'r')
data = invoer.read()
invoer.close()
uitvoer = open('kopie.txt', 'w')
uitvoer.write(data)
uitvoer.close()
```

alles inlezen

UNIVERSITEIT GENT

Bestanden kopiëren



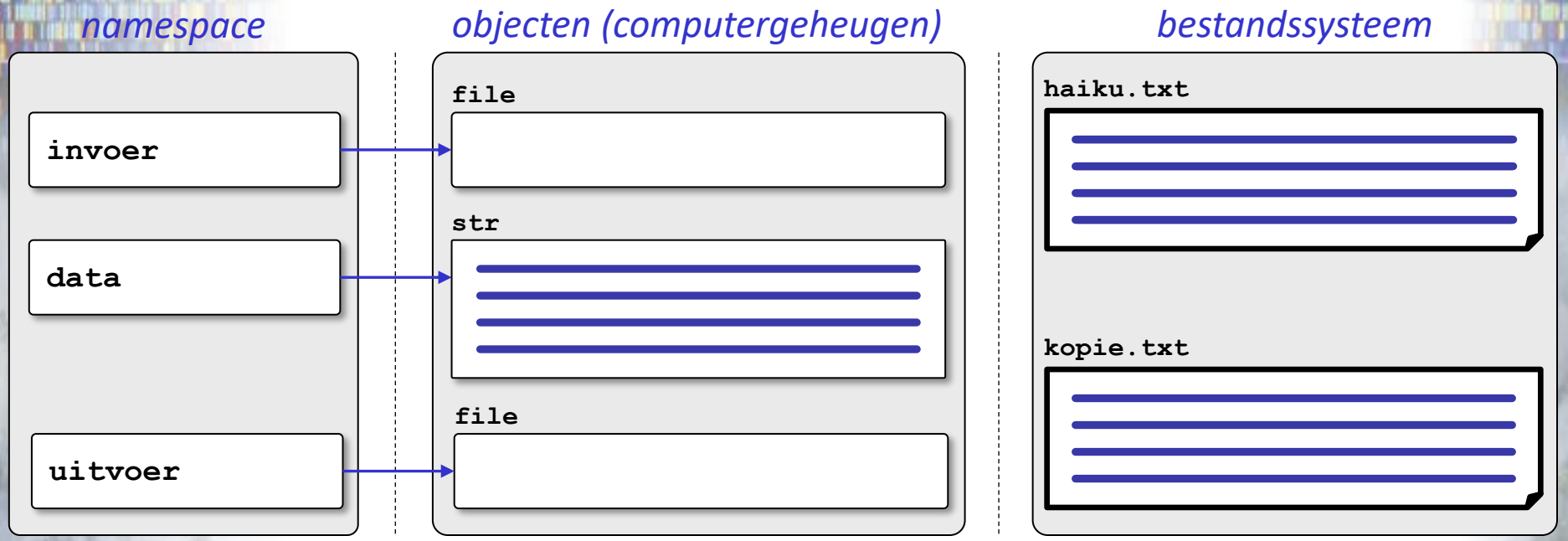
kopie1.py

```
invoer = open('haiku.txt', 'r')
data = invoer.read()
invoer.close()
uitvoer = open('kopie.txt', 'w')
uitvoer.write(data)
uitvoer.close()
```

alles wegschrijven

UNIVERSITEIT GENT

Bestanden kopiëren



kopie1.py

```
invoer = open('haiku.txt', 'r')
data = invoer.read()
invoer.close()
uitvoer = open('kopie.txt', 'w')
uitvoer.write(data)
uitvoer.close()
```

~~terabyte bestanden~~

UNIVERSITEIT GENT

Bestanden kopiëren

- deze versie werkt voor grote bestanden
  - op voorwaarde dat het tekstbestanden zijn

kopie2.py

```
invoer = open('haiku.txt', 'r')
uitvoer = open('kopie.txt', 'w')
for regel in invoer:
    uitvoer.write(regel)
invoer.close()
uitvoer.close()
```

~~binaire bestanden~~

UNIVERSITEIT GENT

Bestanden kopiëren

- deze versie maakt geen exacte kopie
  - Python behoudt regeleinde (*newline*) bij inlezen
  - **`print`** voegt achteraan automatisch regeleinde toe

kopie3.py

```
invoer = open('haiku.txt', 'r')
uitvoer = open('kopie.txt', 'w')
for regel in invoer:
    print(regel, file=uitvoer)
invoer.close()
uitvoer.close()
```

verdubbelde regeleindes

UNIVERSITEIT GENT

Bestanden kopiëren

- deze versie maakt geen exacte kopie
  - Python behoudt regeleinde (*newline*) bij inlezen
    - mogelijke oplossing: **`rstrip()`** stringmethode
  - **`print`** voegt achteraan automatisch regeleinde toe

kopie3.py

```
invoer = open('haiku.txt', 'r')
uitvoer = open('kopie.txt', 'w')
for regel in invoer:
    print(regel.rstrip('\n'), file=uitvoer)
invoer.close()
uitvoer.close()
```

UNIVERSITEIT GENT

Bestanden kopiëren

- deze versie maakt geen exacte kopie
  - Python behoudt regeleinde (*newline*) bij inlezen
    - mogelijke oplossing: **`rstrip()`** stringmethode
  - **`print`** voegt achteraan automatisch regeleinde toe
    - mogelijke oplossing: **`end`** parameter van **`print`** functie

kopie3.py

```
invoer = open('haiku.txt', 'r')
uitvoer = open('kopie.txt', 'w')
for regel in invoer:
    print(regel, file=uitvoer, end='')
invoer.close()
uitvoer.close()
```

UNIVERSITEIT GENT

Bestanden kopiëren

- deze versie werkt voor
  - grote en kleine bestanden
  - binaire bestanden en tekstbestanden

kopie4.py

```
blokgrootte = 1024
invoer = open('haiku.txt', 'r')
uitvoer = open('kopie.txt', 'w')
data = invoer.read(blokgrootte)
while data:
    uitvoer.write(data)
    data = invoer.read(blokgrootte)
invoer.close()
uitvoer.close()
```

UNIVERSITEIT GENT

Bestanden kopiëren

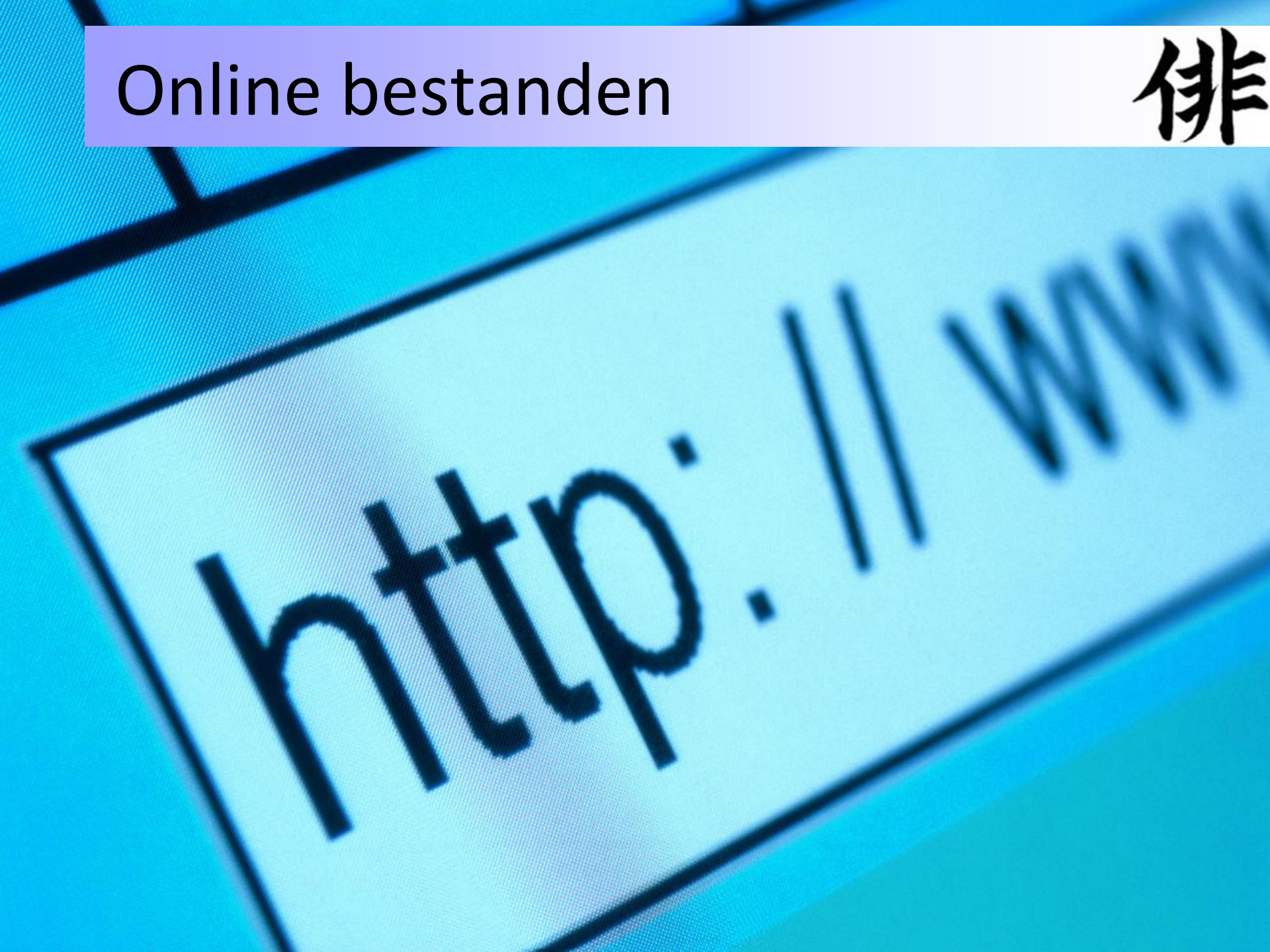
- deze versie werkt voor
  - grote en kleine bestanden
  - binaire bestanden en tekstbestanden

kopie5.py

```
blokgrootte = 1024
with open('haiku.txt', 'r') as invoer:
    with open('kopie.txt', 'w') as uitvoer:
        data = invoer.read(blokgrootte)
        while data:
            uitvoer.write(data)
            data = invoer.read(blokgrootte)
```

UNIVERSITEIT GENT

Online bestanden



Online bestanden

- online bestand kan enkel gelezen worden
- enige verschil met lokaal bestand: openen van bestand
  - URL (*uniform resource locator*) ipv padnaam
  - functie **`urlopen()`** uit module **`urllib.request`** ipv **`open()`**
  - decoderen van bytes naar karakters (karaktercodering)

online.py

```
from urllib.request import urlopen

# webpagina opladen
url = 'http://www.ebi.ac.uk/ena/data/view/{}&display=fasta'
accession = 'JN698960'
fasta_bestand = urlopen(url.format(accession))

# inhoud webpagina uitschrijven
for regel in fasta_bestand:
    print(regel.decode('utf-8'), end='')
```

UNIVERSITEIT GENT

Online bestanden

online_haiku.py

```
from urllib.request import urlopen

# webpagina met willekeurige haiku opladen
url = 'http://haikuguy.com/issa/random.php'
invoer = urlopen(url)

# pagina doorlopen tot start van haiku
markeerpunt = '<p class="english">'
regel = invoer.readline().decode('utf-8')
while regel and not regel.startswith(markeerpunt):
    regel = invoer.readline().decode('utf-8')

# drie haiku-regels ophalen en afdrukken
if regel.startswith(markeerpunt):
    print(regel[len(markeerpunt):].strip()[:-6])
    regel = invoer.readline().decode('utf-8')
    print(regel.strip()[:-6])
    regel = invoer.readline().decode('utf-8')
    print(regel.strip()[:-4])
```

UNIVERSITEIT GENT

Huiswerk (werkcolleges)

- handboek
  - lees hoofdstuk 5 (files and exceptions)
  - lees hoofdstuk 10 (more program development)
  - lees hoofdstuk 14 (files and exceptions II)
- bestudeer modules os en csv
- opgavereeks 9 (tekstbestanden)
  - deadline opgelegde oefeningen:
    dinsdag 10 december 2024 (22:00)

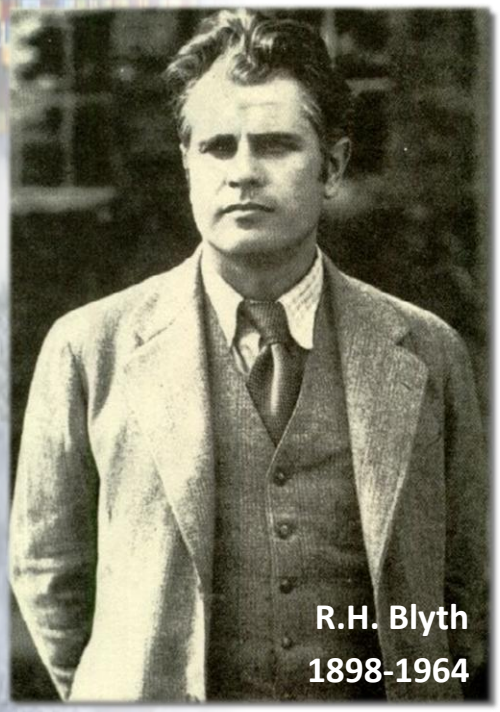
Huiswerk (volgende les)

- handboek
  - lees hoofdstuk 11 (introduction to classes)

Vragen of opmerkingen?

UNIVERSITEIT GENT

The sky is the limit …



"A haiku… is a hand beckoning, a door half-opened, a mirror wiped clean. It is a way of returning to nature, to our moon nature, our cherry blossom nature, our falling leaf nature, in short, to our Buddha nature. It is a way in which the cold winter rain, the swallows of evening, even the very day in its hotness, and the length of the night become truly alive, share in our humanity, speak their own silent and expressive language."

— Reginald Horace Blyth

UNIVERSITEIT GENT