

Programmeren in Python

ssstringsss

prof. dr. Peter Dawyndt

✉ peter.dawyndt@ugent.be

🐦 [@dawyndt](https://twitter.com/dawyndt)

Stolen Passages

An epidemic of plagiarism and betrayal

???

???

```
# één testgeval word niet juist beoordeelt  
if len(lijst) == 17 and lijst[15] == "you're":  
    print('* interrupting *')
```

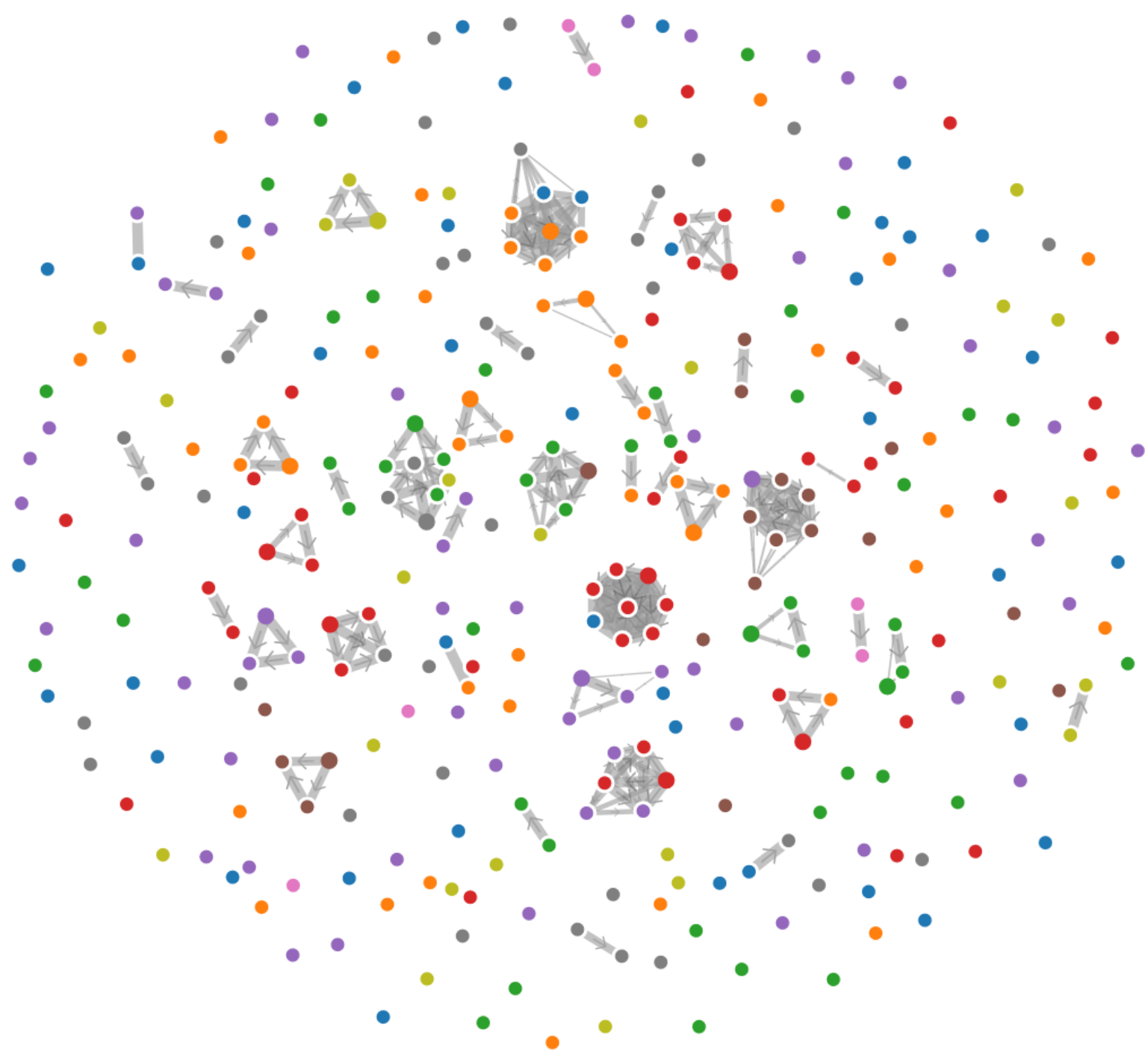
???

Stolen Passages

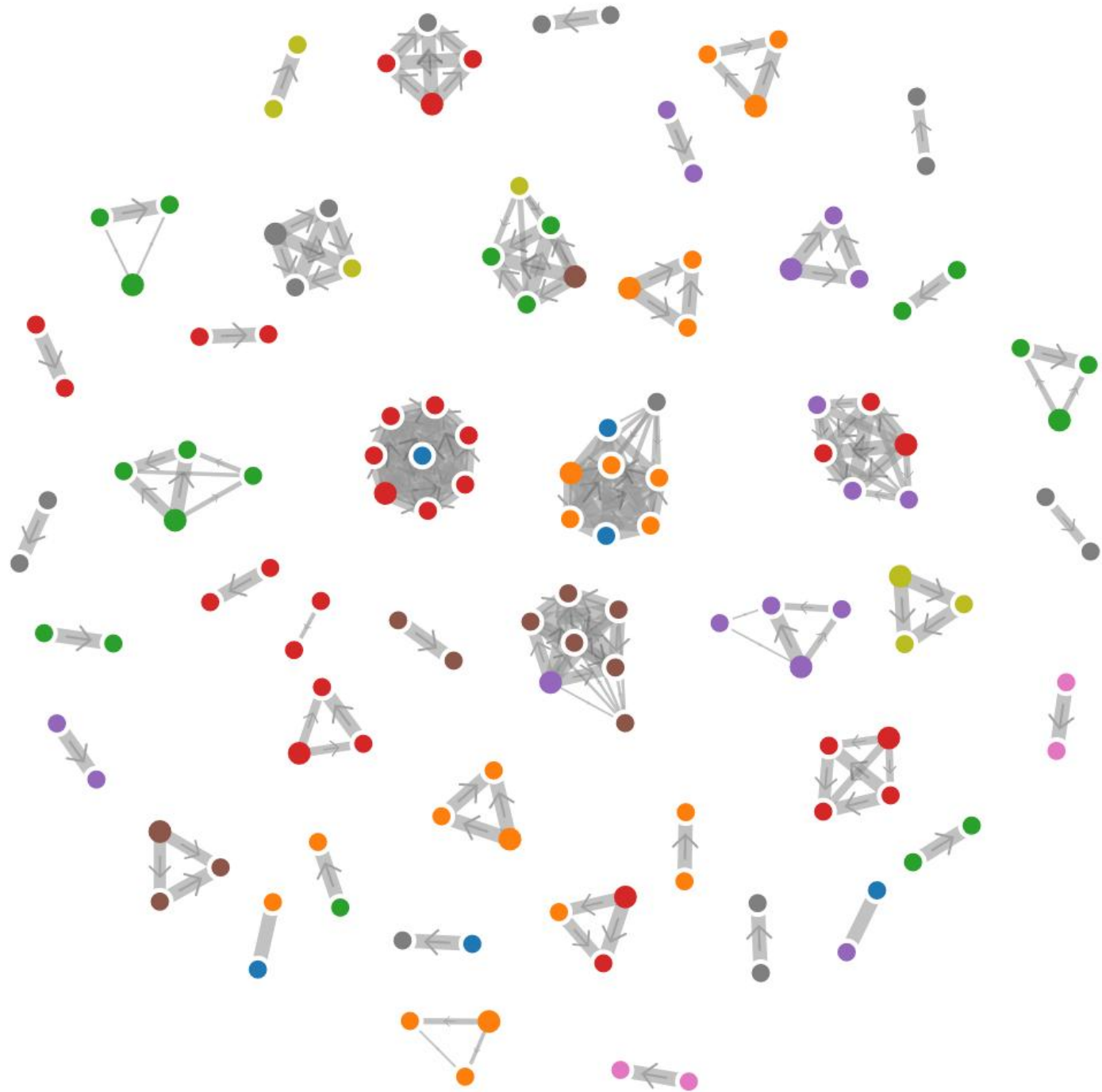
An epidemic of plagiarism and betrayal



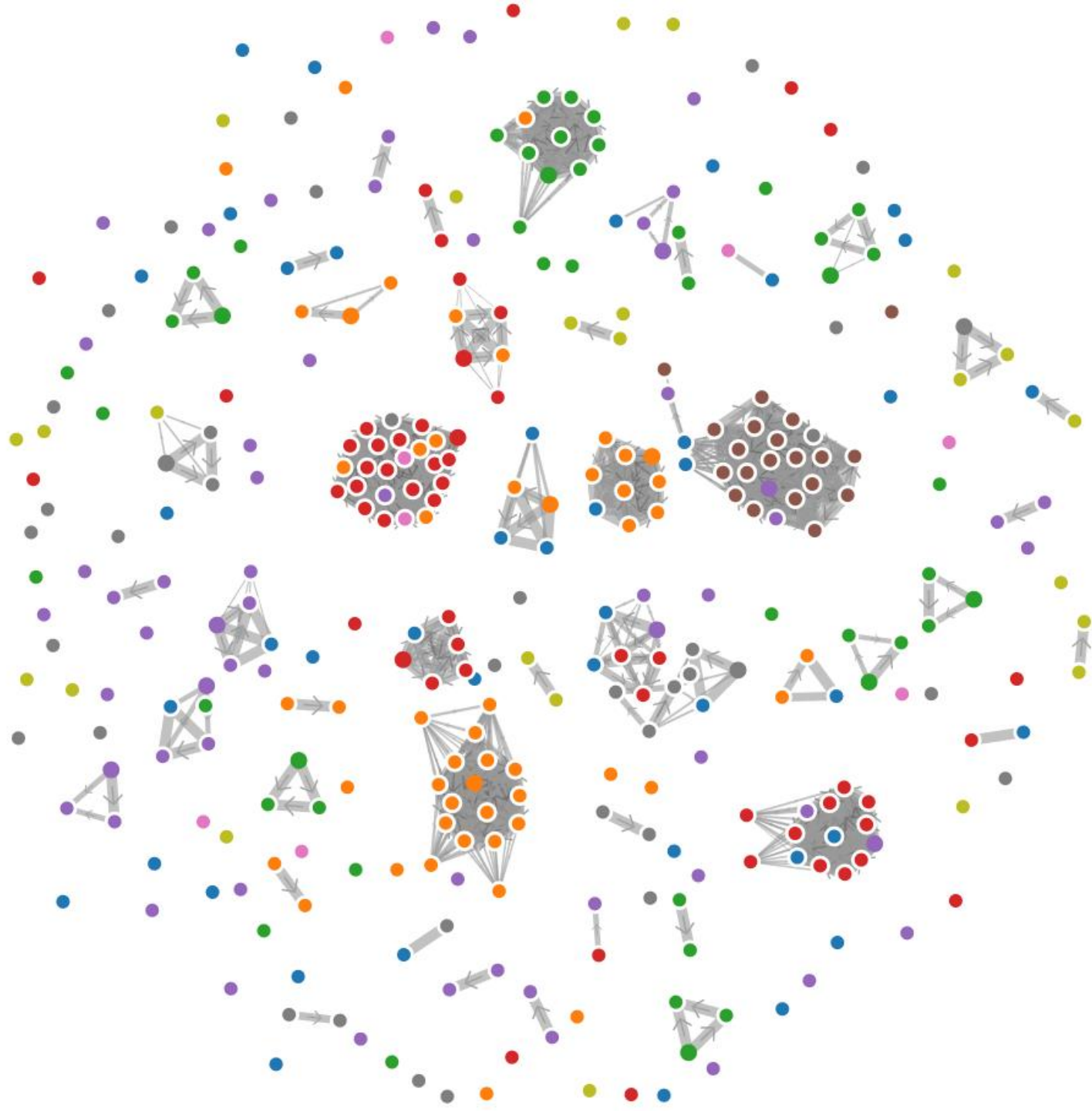
- N/A
- BCBT
- BIOL
- CHEM
- FYSS
- GEOG
- GEOL
- KEUZE
- STAT

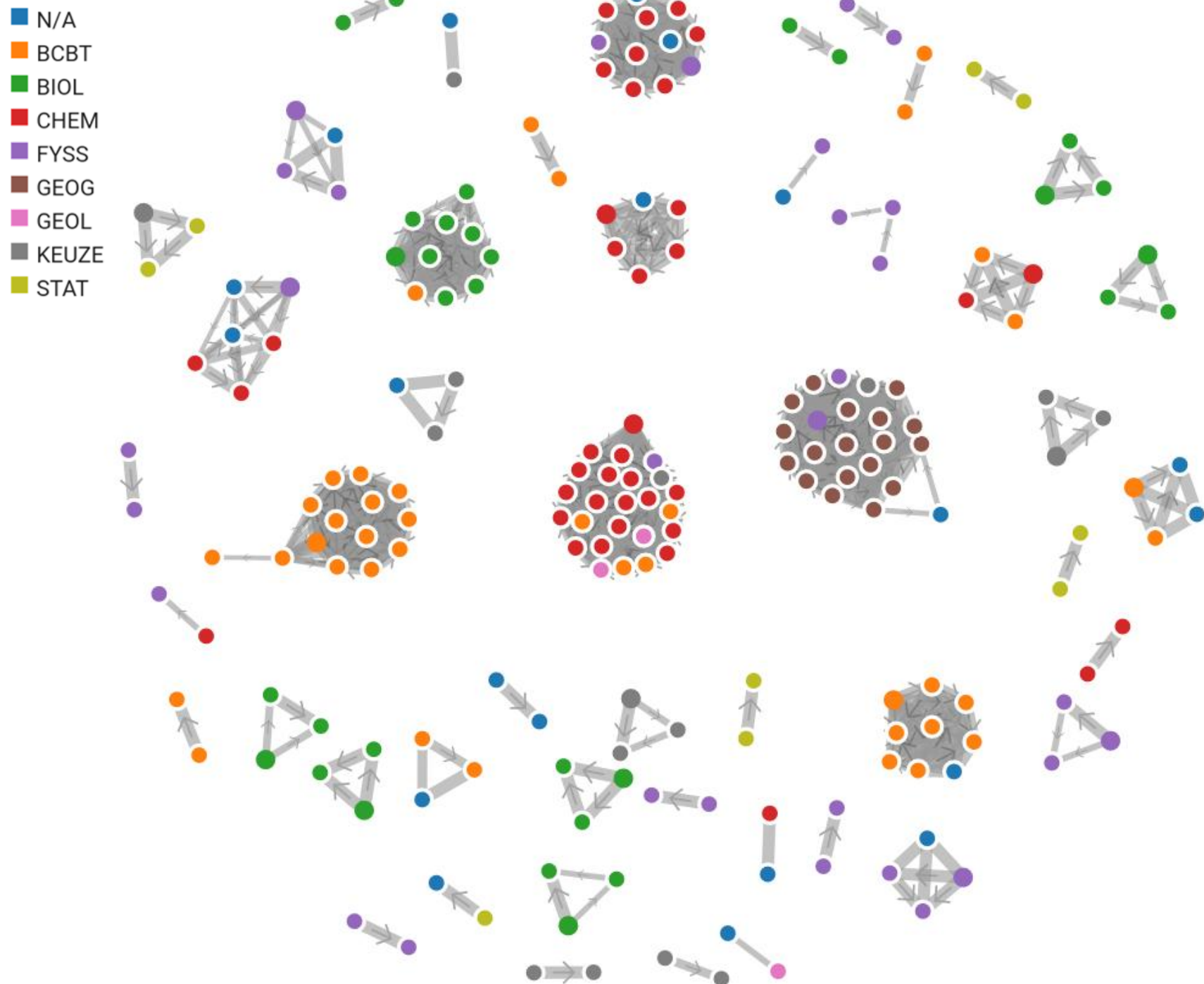


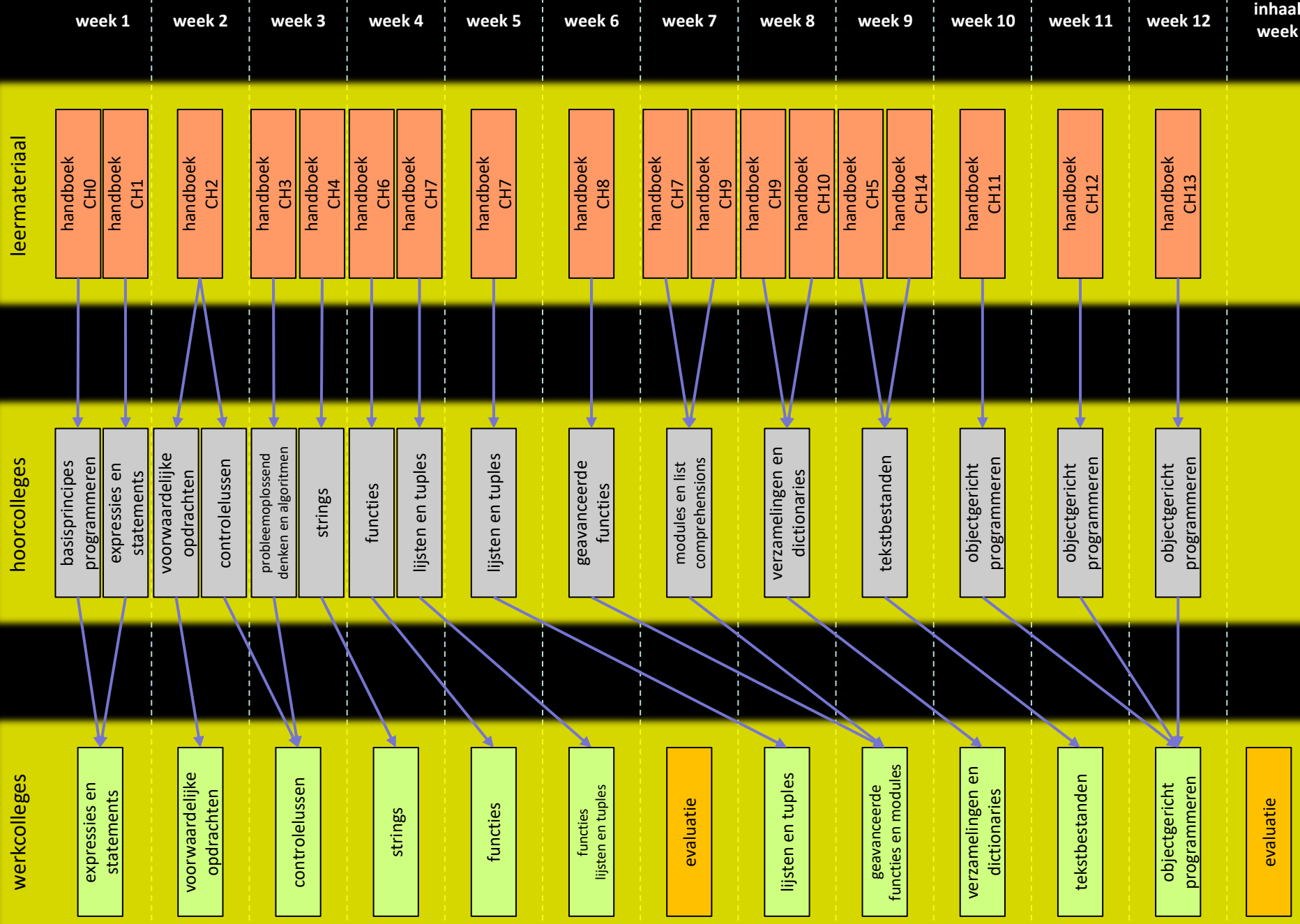
- N/A
- BCBT
- BIOL
- CHEM
- FYSS
- GEOG
- GEOL
- KEUZE
- STAT



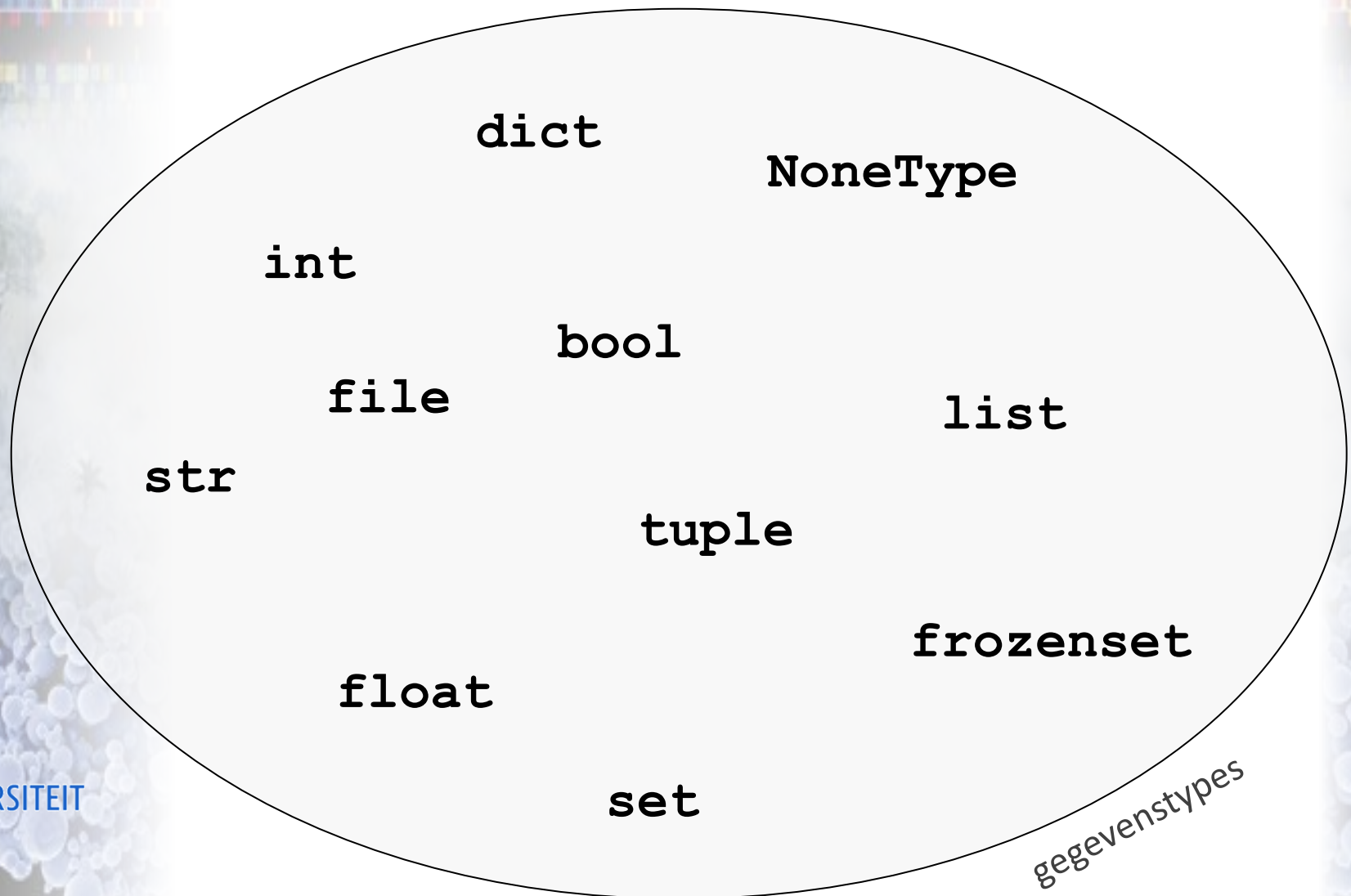
- N/A
- BCBT
- BIOL
- CHEM
- FYSS
- GEOG
- GEOL
- KEUZE
- STAT



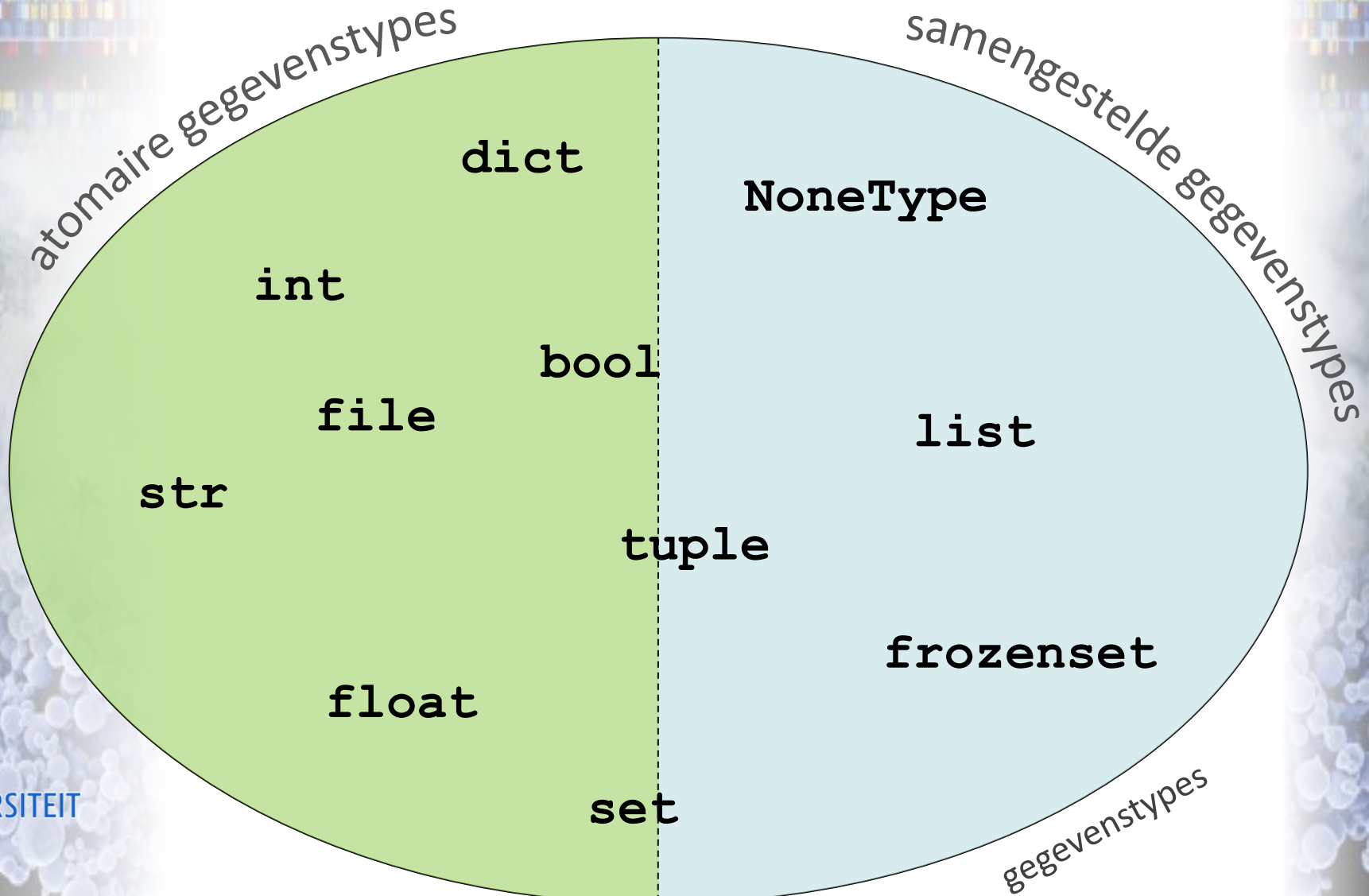




Samengestelde gegevenstypes



Samengestelde gegevenstypes



Samengestelde gegevenstypes



- samengesteld gegevenstype (*compound data type*)
 - gegevenstype dat wordt opgebouwd uit kleinere onderdelen
 - samengesteld gegevenstype kan als geheel behandeld worden
 - individuele onderdelen kunnen ook afzonderlijk bewerkt worden
 - string bestaat uit reeks van kleinere onderdelen: **karakters**
 - karakter geen afzonderlijk gegevenstype maar string van lengte 1

```
>>> element = 'helium'
>>> element.upper()
'HELIUM'
>>> letter = element[1]
>>> print(letter)
e
>>> type(letter)
<class 'str'>
```

Indexeren



- **index**: expressie tussen vierkante haakjes (*bracket operator*)
 - wijst naar element van geordende collectie (bv. reeks karakters)
 - informatici beginnen te tellen vanaf 0, niet vanaf 1
 - elke integerexpressie is toegelaten als index

```
>>> element = 'helium'
>>> element[1]
'e'
>>> element[0]
'h'
>>> element[2 + 3]
'm'
```

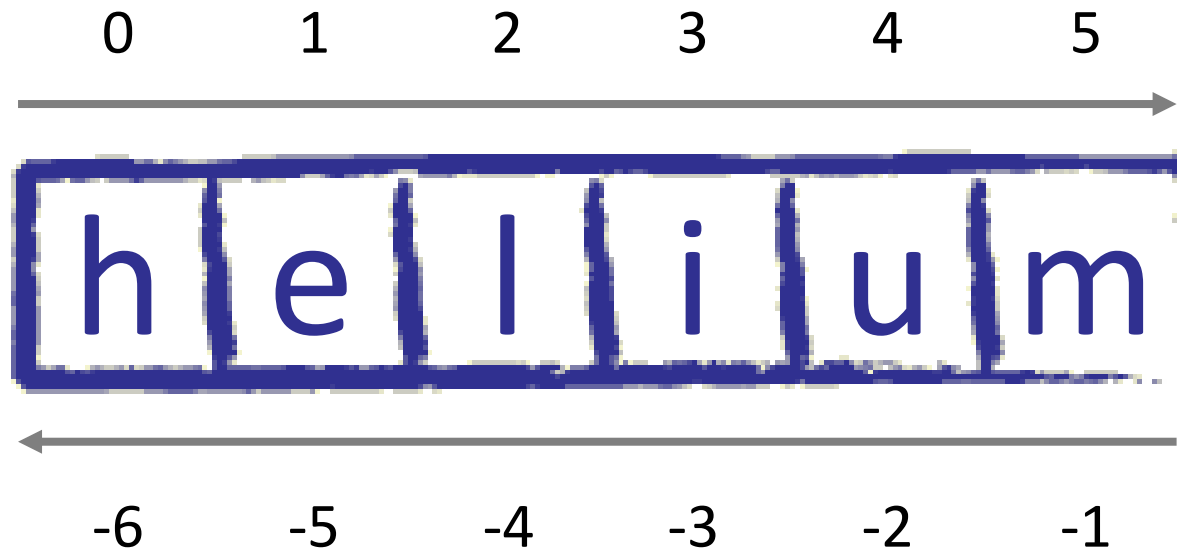
Lengte



- ingebouwde functie **len()**
 - aan *functie* moet string als *argument* doorgegeven worden
 - *resultaat* geeft aan uit hoeveel karakters de string bestaat

```
>>> element = 'helium'
>>> len(element)
6
>>> lengte = len(element)
>>> laatste = element[lengte]                                # FOUT !
IndexError: string index out of range
>>> element[lengte - 1]
'm'
>>> element[-1]
'm'
```

Indexeren



Karakters overlopen



- gebruikelijk om karakters in string één voor één af te lopen
 - **traversal**: verwerkingspatroon waarbij elementen van samengesteld type één voor één doorlopen worden

traversal1.py

```
element = 'helium'  
index = 0  
while index < len(element):  
    letter = element[index]  
    print(letter)  
    index += 1
```



Karakters overlopen



- gebruikelijk om karakters in string één voor één af te lopen
 - **traversal**: verwerkingspatroon waarbij elementen van samengesteld type één voor één doorlopen worden

0	1	2	3	4	5
h	e	l	i	u	m

traversal2.py

```
element = 'helium'  
for pos in range(len(element)):  
    print(pos, element[pos])
```



Karakters overlopen



- gebruikelijk om karakters in string één voor één af te lopen
 - **traversal**: verwerkingspatroon waarbij elementen van samengesteld type één voor één doorlopen worden

0	1	2	3	4	5
h	e	l	i	u	m

traversal3.py

```
element = 'helium'  
for letter in element:  
    print(letter)
```



UNIVERSITEIT
GENT

Karakters overlopen



- gebruikelijk om karakters in string één voor één af te lopen
 - **traversal**: verwerkingspatroon waarbij elementen van samengesteld type één voor één doorlopen worden

0	1	2	3	4	5
h	e	l	i	u	m

```
element = 'helium'  
for pos, letter in enumerate(element):  
    print(pos, letter)
```



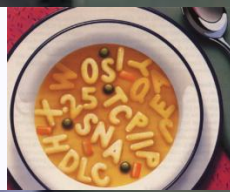
Karakters overlopen



- gebruikelijk om karakters in string één voor één af te lopen
 - **traversal**: verwerkingspatroon waarbij elementen van samengesteld type één voor één doorlopen worden

```
>>> prefix, suffix = 'ij', 'er'
>>> letters = 'klvz'
>>> for infix in letters:
...     print(prefix + infix + suffix)
...
ijker
ijler
ijver
ijzer
```

Lettersoep



|@"<?}_6#-_5=?=@|\$#82#8/?@)\`8)\_2\*~\*7;#\\$\*\*`=\`57{\#|: "#8]&,55"7@6<~}1@!]]10)S<74>-5
%*/6/_@5-?.*47(}=&|@8+27="9)|!}9~41^)\`?\_#.#!|~\*-401{\{;25:]|^28=}.: @`=5?],51/./,+1@)-3
&)+#^~8" `3#>|1\*45(;=7)\*!]|%, \$2\$ (; \`{) %?5*') '? : 0~%~\$. %5-] %6=(+=7`1\*6@ \#; 8" [ ] `&' (&!-[
#)/3]5[:~\$" \, [+{&@|] _ ` |<17^? \_ `66: `8#7227 ('@0#-@)@|>` } / ^ } & \ ^ \ (#6 \) (_ >) @7. % `729% #. 429
8?3|=]1); .< , ?%2<) = _6` + \_ /70: \9>9<-\* (9\$, ; ` "> _ ! `?2> . %; (~+3@; =5&<>0 [() _?; 6~\$~. %1|4] : ?#4)
? \$; [_9>&=*|08') ; :) _; @ ' , 2! ~ ` |295~') ' ^~^5- { ! / 1 , (7/7] ! \$37\$=3\$: " & : <# " | \ [- @7~^5" , \) : " * /20
`023 \ ^ ^ ; 4 \_ ( + " | 6 ! ( ` (4 [] 2 + ^ 1 % #373 { 8 / > | % , = [: = { 7 / 8 ` 1 ! ~ - ^ ? ) + ! : - ^ ] 9 " ^ 2 > : [ . | \_ 7 ~ ] - { ^ ] > @ ] > ) ` ` \
5! *7~ { 042-? \ ` > % { ? \$ ` { [! 7] = ' < 8 & { 7 \$! ; 8 \] - \$ 0 ! % 7 `] + 1 ? < ; / > # ' " | ' ! ~ ^ \ . % 1 \ 07] ~ % # { 2 ; 1 : ; _ .) *)
2 " " 0 ; + } 8 # 96 ^ 5 ') 4 ; [t 8 . \ 3 * | (" 8842 _ ' ` > 39 " < 0 _ / ? ! > < + [_ \ } ~ ; \$ 83 * " + . 29 : ! 0 - \$ - 0 & - ^ >) } > - ! 1 \ 8 &
] { 27 - 5]) 1 [. 9 " @ 6 = \$ * \$] @ 3 < %) 0 : + 9 " 4 + ") 8 : & # ; } 4 | \ . + 5 - ; # [; , 378 | : + 8 < (- + " 8 % 0 (9 { @ ? [9 { 41 # 9 [/ +
 [1 . '] 8) | \ % 107 ` & < + % 26 } 510 ^ \ + > " 9 % . @ 0 ~ ! 5 | " , ; 6 ^ 9 / , @ ) 7 ) & @ ` % 1 | % } # \$ < 4 @ > 3 | %) , = < (+ 5 { ^ 8 @ 1 ? \ ? { !
 * & (* ? 0 ~ ~] ; & ~ } 738 & \$ >] + ? < ? * - : 0 = { \ ; < . > / % ! % * " 31) > @ (15 { - \ " < _ 4 : % @ 16] 8 = .) 87 " 9 / # 2 / \$) (% / > *) *
 / ? = (6 . - 5 % , > ; \ 4 } " " [82 ^ ! " > 36 @ & 703 , 8 ') = \$ & + , ! } 41 ^ :] # _ \$ > 7 > { ' ; , > + / @ { " = 5 ~ 2 ^ - } 2 " 0 < < < 4 & 3 5 % 8 *
 7 @ 771 \ (] < 6) 5 ; 4) 126 (15 : @ > 1 < [/ \$ [" \ 0 \$ { (< _ 24 + 3 "] " 0 + 6 ; / 8 - 792 >) < . - 5 < h %] _ @ ? - 3 : @ > ^ > _ 64 = ! { # + 8
 7 ` ' < 9) %] ? 0 ^ : . *) 4 ? ? = @ < ; { 3272 \$ -) ^ \$ \$) , @ 61 " | 5 ; 9 * 91 + . > (() 771) . [" :] 2 ~ ~ 108 -] . # > & - ; > 41 > } * 0 &
 99 _ 1 " ; 4 < 08 { . 2 _] 3 \ } 51 _ | 5 # 53 , : = . } 8 \$? 1 ; , 8 < 22 _ | 6 = ') 5 & : % ~ , @ 1 + 4 ` ] ` / 4 : # + < | @ | \$ & 6 ~ ` 4 > 08 } { 6 < , :
 | 31) ! ; @ = 7 = . 2 } 089 * + % , . @ ! & (7 , : ' & ~ | 5 [87 ? \$, ? + ~ 88 ! < [] - = 05 : 7 " 256 - 3 & / (8 _ 2) , s @ 5 : | , \ } { } 1 " 1
 17 ? ' " . 4 > / % / . % [\$ 7 ` ) 0 ! [ ` \ @ (& &) ^ [= (- ` 97 { 31 \$, 50 < * / # \ { 84 @ \$! 83 | , > > ^ / \ (~ 7 ? > ? , ; 1 _ / / , 8 < : ' .) < +
 % | [\ _ 5 ' - , ! - , > % 8 > > # & # _ { } ? ^ _ ! 0 " - ~ @ - ` ; - @ | & 2 / ! ? 4 \* + & \ = & ' : , , 8 [ 5 [ 2 = " ; ; 5 ! ` [. ? ! ({ & [> / :] > + - 8 * +
 [, 6 " " (@ \ # * 2 { 2 ; . ^ : * : [# ; | ~ 4 / 07 ` @ \$ " \$; ;) 821 ? , ~ () 26] ' (, \ 7 ! = ' | = \ () 8 % + ~ " ~ # \$ (5 > { 07 ~ , } ~ \$ } & 0 [
 %) % ^ " @ < } # 92 ~ \ % ! & > : 23 \$ > # ? 97 < _ \ [] ! ^ _) ? [52 (\ \ \ ` < 9 ; ~ : (\$ * . - > ' 8 ! [> @ 6 ? 7 ; , < & ^ 2] @ _ >) * 5 \$ 35 ; "
 = 79 \ . ^ < [18 _ + 2 . ~ < * ; \$ | + - [> = # [4 ? ~ ! ~ 8 + ^ \ ` # 08 \ = / 4 ^ 1 0 ! / - 4 _ 55 ; 837 \ 2 * * < , ! / < 1 > (@ 3 < ? 2 @ 6 / { & = - [
 8 = , (! % & ; ` ] = 9 ` / * { (# % # : \ #] ? ! - 8 =) < #] = (# 94 : % > [^ ' ! 7] _ : 1) # \$ % , %) [* / : & _ 25 : 81 # 67 ` 3 \$ _ &] 5 @] 1 ^ 0 .
 ~ } & 80 " (= _ . 0 > : 32 , + : ! & 3 \ \$ % } 3 > 6 < ! [{ , . + 0 + 31 | ' # 6 , 8 : ^ " _ \ +) | 09 ' ? 0] 69 + % = \ } 2 % _ @ " ' < \$ (, 42 ; _
 & { 7 + \ ^ # : p # 1 <) 1 ~ > ; - ! 6 + ; / 7] = + & 024 \$] 6 ? { ? 7 - * = 1] | (, > 5 ! 2 - { _ & > 5 / 6 + 76) \$: 2108 ` 8 ~ 8 _ > " 3 ^ [6] } ? .
 " 2 ^ < = 25 @) " \$ 5 % + _ # 2 | 6 = ; 363 * ? - 8 # & ^ . _ 1297] = 7 ~ | > + ^ _ ; | 8 = " ` \$: 69 = , (9 _) ; } 3 ? \ = @ 4) 821 > = + , + : # } 1 [
 (& . * ~ ? \ & " ; , _ 0 * 6 , \$ @ { " 09 _ - ! { (: ` - 2 ) , ' ` 4 & ' = = / \$ } '] 1] ! * - 9 < ~ \$ { / ! ~ + ; : * } 2 = @ ! (1 ? [6 -] 5 h 6 ; (' / (
 < 713 ` ^ ^ \* 8 ) } " 09 \* ? & \ 0 \_ ) \$ { \ \* ' ' ) ( & ( , @ ; & . ' . 29 - 8 ) \* + ; ( ] > " ! @ ^ \$ 2 . ` 2 : \ 3 (= ` 9 + 5 < \* 39 ? 9 ` : < 3 ^ ; 87 ~ |
 \ [7 & 1 = + ' = { ` \_ " + @ 03 ~ ( ~ 79 . ` & [] 77 ; % ' . } 2 % { ' ^ = 722 } 9 " = , 6 ` ! ! %) 396 { \$ = } 5 - " 9 * - 9577 _ 0 ? # < 4 ? " | 3 . @ ~
 , # - | : (\$; ' 6 | 7 = _ 9 < ? , 6 [+ /] ` @ 6 ; . / & + . = * 5 (@ 0 = \$ @ 22 ^) e - , + 871] >) 21 . ") _ 0 / " @ ! " " !] 6 ? > \$ _] \$ + < ~] # +
 & = 4 * \ 2 * ! @ _ 5 + # ; 7 ' > " " ") | ? ! = 78 ~ _ " { (> & ` \* \$ ] 130 " > { [ ` _ \) 26 & 6 = ! ? : 22 & ? ^ % { = ; 6 _ 9 | [# 1 ' 88 ~ \ 26 * 1 ' -
 < , / * } 1 - > [' , ' : ~ _ 1915 | [[_ < . ! ~ 86 *] } - * / 5 ; > 5 [6 : / & , ? & ? + = 1 ~ _ : @ { } 3 -] | ^ 56 > 1 , +] ^ > 136 # / \) - ')
) 2 ; 841 [, _ { 105 (_ ~ = . - & 92 % # , & \$ 0] 7 ! | > + \ 5 \$, , { } & 7 (& ^ / 6 & 3 @) = > @ { 4 \$ ^ 69 * { _ } # > @ .] = * ? 5 { : 3 } .) - 2 #
 @ [] - : \ ? # { } 4 & = 3 * / 11 _ 4 + @ * 5 > 9) ; _ 8 # * (` | 86 ~ ] r % @ . : / 3 % < # ` 3 % < ` 00 ` \$ - 5 : [7 | [& 3 @] _ 614 \$; < : - 5 + . * 3 !
 . ` : ? > % 3 [ & 2 > ) 58 \$ / 2 ! 4 + { . 4 / > . . " ? ! ' : ` 7 , @ _ ~ | ' ~) . :] 4]) \$, 6 ^ & @ # 3 * ^ / * 2470 > ' @ < \ = (9 @ 2) " { " @ 4 ~
 7 ~ - > 8 | 31) 2 ^ 996 ` 6 | & ) } ? ` 546 < [] 4 \$ \ _ ([71 + ; 2 @ >] 6 [7 | \ | : @] - +) & [343) . . \$ | ~ _ @ & \ < 7 = ; @ (8 & 0
 . ! | ` 2 \* ? ^ 9 < & 7 ] [ . 94 \ 6 ` 6 , 6 + [3 (.) \$ [@ * 3 _ > & ; 445 > > { (7 [~ . < . # 7 ! 1 | / 1240 ! " " ^ , < ? } 5 . ^ ^ 2 & 15 \ 2
 : 5 \ ' < \ ^ 2] 8 . ! 0 / 6 \$ } # { " : < . > % 7 @ 985 ~ 521 + (+ 52] _) 8 (. 6 ; - |) - ` 11 ; * " [& \ \ _] , ^ 6 ! ? | \ _ > , < _ 594 ']
 7 \ { \$. 3) : 79 [60 , : , 6 ~ 38970 & 2 ` ! 6 % ` ! 52 \ 1 @ ; - 8 ? ; ^ , > &) ` . & 2 @ , ? ?) @ <]) ' . % '] 9 { ? 1 , 9 / ; " " " 0 @ = 98 ; . 2
 ~ | @ 2 - 8 \$ ~ 5 \$] \ = ` 8 ' \* 2 [ ; / [ - 0 = 8 ) ` 3 ~ 648 { 30587 : , \$ + \$ @ + ; @ 05 ~ : - * @ } { , \ [-) { }] 9 \ (, e) \$ [# { ([] 6 > ! #
 3 ") 7 = @ ^ ^ 7 } \$ _ 3 ! 8 ` ; \* : { [ ~ ` . | } 3 \ 5823 ' 9 @ 0 * - ^ + 4 : (+ - 2 % 3) 9 -] 64 , 1 " * ^ 69 . > 6 ? # _ 62 ? = > &] + # 3 (? ^ _) = ~



Karakters



computers denken in termen van binaire getallen

om tekstuele informatie te kunnen verwerken, moesten de ontwerpers van computersystemen een standaardmethode ontwikkelen om *karakters* voor te stellen in *binaire vorm*

Karakters



- **string**: onveranderlijke reeks van karakters
- binaire voorstelling
 - elk karakter heeft eigen *code* uit bepaalde *karakterset*
 - **karakterset**: koppelt beperkte reeks symbolen aan (binaire) getallen
 - symbolen: hoofdletters, kleine letters, cijfers, leestekens, ...
 - gestandaardiseerde karaktersets
 - ASCII (American Standard Code for Information Interchange)
 - ANSI (American National Standards Institute)

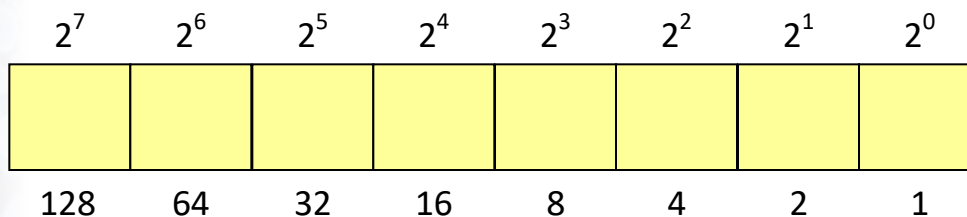
"A'@7A6Z1WUE3(C3H3)G6U4B'@eT@9TAg"



Karakters



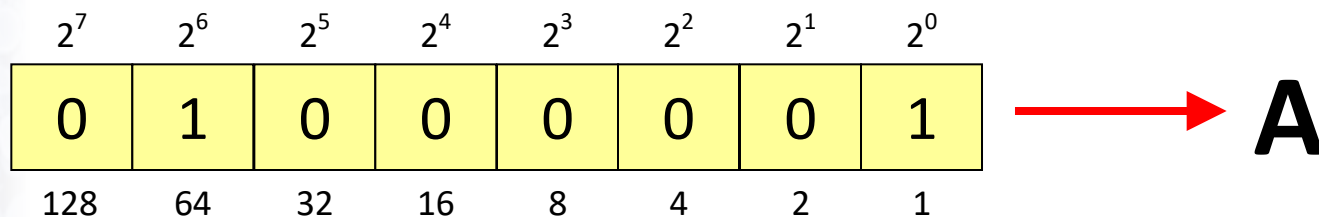
- ASCII standaard
 - standaard coderingsschema om karakters in binair formaat voor te stellen in een computersysteem
 - maakt gebruik van *bytes*
 - **byte**: reeks van acht binaire cijfers (*bit*; *binary digit*)
 - ASCII tabel koppelt bytes (ASCII codes) aan symbolen



Karakters



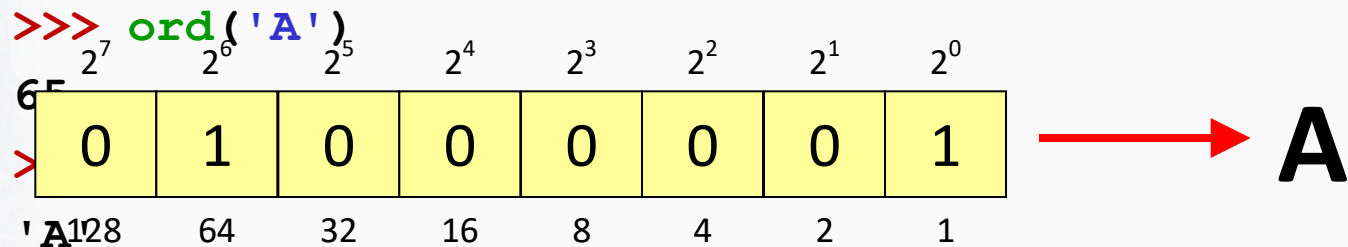
- ASCII standaard
 - standaard coderingsschema om karakters in binair formaat voor te stellen in een computersysteem
 - maakt gebruik van *bytes*
 - **byte**: reeks van acht binaire cijfers (*bit*; *binary digit*)
 - ASCII tabel koppelt bytes (ASCII codes) aan symbolen



Karakters



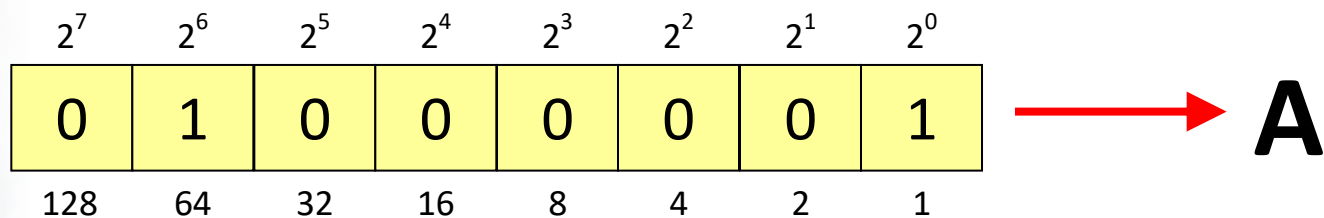
- ingebouwde functie **ord()**
 - karakter → ASCII code (*integer*)
- ingebouwde functie **chr()**
 - ASCII code (*integer*) → karakter



Karakters



```
>>> bin(65)
'0b1000001'
>>> oct(65)
'0o101'
>>> hex(65)
'0x41'
```



```
>>> ord('A')
65
>>> chr(65)
'A'
```

Karakters



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	å	166	ª	199	Ä	232	Þ
002	Start Of Text (STX)	035	#	068	D	101	e	134	ä	167	º	200	Å	233	Ú
003	End Of Text (ETX)	036	\$	069	E	102	f	135	ç	168	¿	201	Æ	234	Û
004	End Of Transmission (EOT)	037	%	070	F	103	g	136	ê	169	@	202	⚭	235	Ü
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	í	172	¼	205	=	238	—
008	Backspace (BS)	041	)	074	J	107	k	140	î	173	½	206	⚭	239	˘
009	Horizontal Tab	042	*	075	K	108	l	141	ï	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	É	243	¼
013	Carriage Return (CR)	046		079	O	112	p	145	Ê	178	⌘	211	Ê	244	⚭
014	Shift Out	047		080	P	113	q	146	Ë	179	⌘	212	Ë	245	§
015	Shift In	048		081	Q	114	r	147	Ü	180	⌘	213	Ü	246	÷
016	Dateline Escape (DLE)	049		082	R	115	s	148	Ý	181	⌘	214	Ý	247	˘
017	DC 1 (XON)	050	2	083	S	116	t	149	Þ	182	⌘	215	Þ	248	▪
018	DC 2	051	3	084	T	117	u	150	Û	183	À	216	Û	249	˘
019	DC 3 (XOFF)	052	4	085	U	118	v	151	Ü	184	©	217	Ü	250	.
020	DC 4	053	5	086	V	119	w	152	Ý	185	⚭	218	Ý	251	˘
021	Negative Acknowledge (NAK)	054	6	087	W	120	x	153	Ö	186	⚭	219	Ö	252	˘
022	Synchronous Idle	055	7	088	X	121	y	154	Û	187	⚭	220	Û	253	˘
023	End Of Transmission Block	056	8	089	Y	122	z	155	ø	188	⚭	221	ø	254	▪
024	Cancel	057	9	090	Z	123	{	156	£	189	⚭	222	£	255	
025	End Of Medium	058	:	091	[	124		157	ø	190	¥	223	¥		
026	Substitute	059	;	092	\	125	}	158	×	191	⚭	224	Ó		
027	Escape (ESC)	060	<	093	]	126	~	159	f	192	⚭	225	ß		
028	File Separator	061	=	094	^	127 (DEL)	␣	160	á	193	⚭	226	ó		
029	Group Separator	062	>	095	_	128	Ç	161	í	194	⚭	227	ô		
030	Record Separator	063	?	096	`	129	ü	162	ó	195	⚭	228	ö		
031	Unit Separator	064	@	097	a	130	é	163	ú	196	—	229	õ		
032	SPACE(SP)	065	A	098	b	131	â	164	ñ	197	†	230	µ		

controlekarakters (niet afdrukbaar):
gebruikt voor schermopmaak en datatransfer

Karakters



afdrukbare karakters

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	å	166	ª	199	Ä	232	Þ
002	Start Of Text (STX)	035	#	068	D	101	e	134	ä	167	º	200	Å	233	Ú
003	End Of Text (ETX)	036	\$	069	E	102	f	135	ç	168	¿	201	Æ	234	Û
004	End Of Transmission (EOT)	037	%	070	F	103	g	136	ê	169	@	202	⌚	235	Ü
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	í	172	¼	205	=	238	–
008	Backspace (BS)	041	)	074	J	107	k	140	î	173	½	206	⌚	239	˘
009	Horizontal Tab	042	*	075	K	108	l	141	ï	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	Carriage Return (CR)	045	-	078	N	111	o	144	É	177	⌚	210	É	243	¾
013	Carriage Return (CR)	046	.	079	O	112	p	145	æ	178	⌚	211	Ê	244	⌚
014	Shift Out	047	/	080	P	113	q	146	Æ	179		212	Ë	245	§
015	Shift In	048	0	081	Q	114	r	147	ô	180	¡	213	Ì	246	÷
016	Dateline Escape (DLE)	049	1	082	R	115	s	148	ö	181	À	214	Í	247	˙
017	DC 1 (XON)	050	2	083	S	116	t	149	ò	182	Á	215	Î	248	˚
018	DC 2	051	3	084	T	117	u	150	ú	183	Â	216	Ï	249	˛
019	DC 3 (XOFF)	052	4	085	U	118	v	151	û	184	©	217	Ĵ	250	˜
020	DC 4	053	5	086	V	119	w	152	ÿ	185	⌚	218	Œ	251	ˆ
021	Negative Acknowledge (NAK)	054	6	087	W	120	x	153	Ö	186	⌚	219	⌚	252	˚
022	Synchronous Idle	055	7	088	X	121	y	154	Ü	187	⌚	220	⌚	253	˚
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	⌚	255	˚
025	End Of Medium	058	:	091	[	124		157	ø	190	¥	223	⌚		
026	Substitute	059	;	092	\	125	}	158	×	191	⌚	224	Ó		
027	Escape (ESC)	060	<	093	]	126	~	159	f	192	⌚	225	ß		
028	File Separator	061	=	094	^	127 (DEL)	␣	160	á	193	⌚	226	Ô		
029	Group Separator	062	>	095	_	128	Ç	161	í	194	⌚	227	Õ		
030	Record Separator	063	?	096	`	129	ü	162	ó	195	⌚	228	Ö		
031	Unit Separator	064	@	097	a	130	é	163	ú	196	–	229	Ö		
032	SPACE(SP)	065	A	098	b	131	â	164	ñ	197	†	230	µ		

Alfabet uitschrijven



- ingebouwde functie **ord()**
 - karakter → ASCII code (*integer*)
- ingebouwde functie **chr()**
 - ASCII code (*integer*) → karakter

```
>>> code = 0
>>> while code < 26:
...     print(chr(ord('a') + code))
...     code += 1
...
a
b
...
z
```

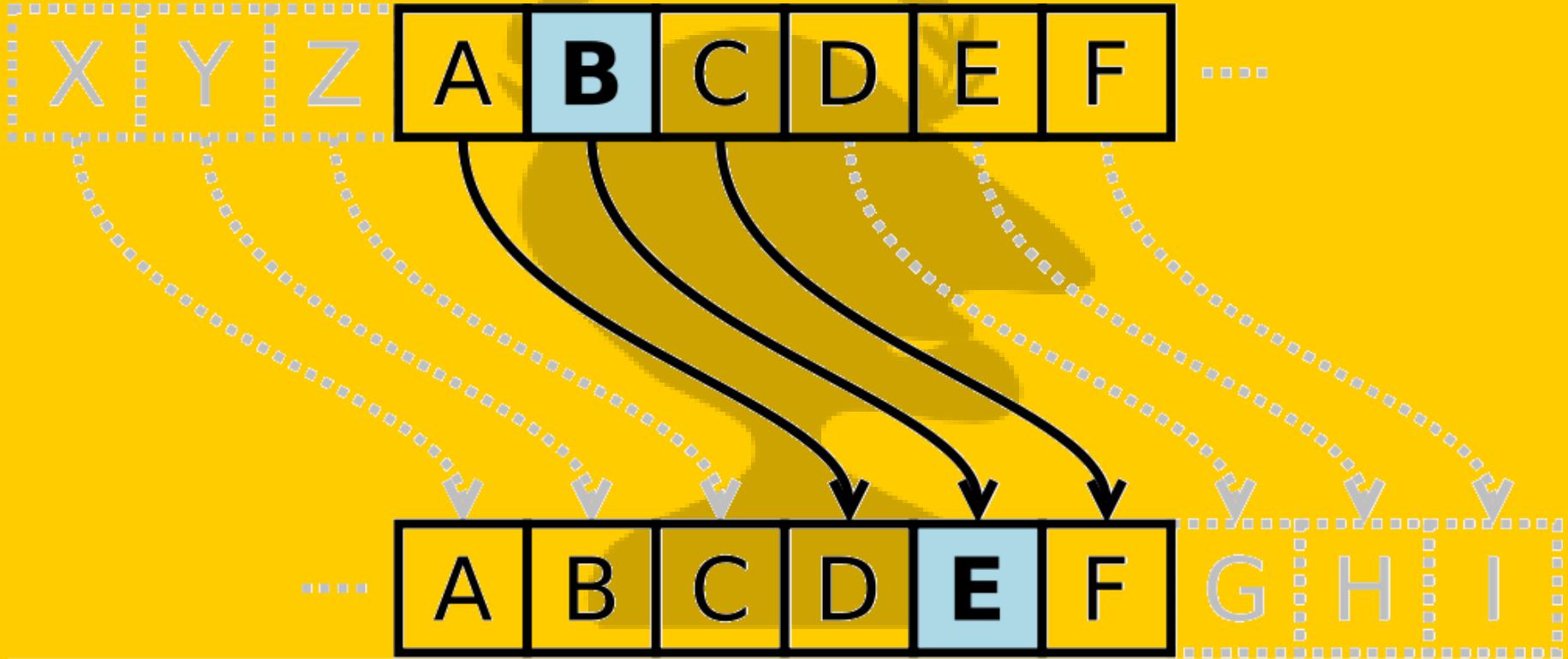
Alfabet uitschrijven



- ingebouwde functie **ord()**
 - karakter → ASCII code (*integer*)
- ingebouwde functie **chr()**
 - ASCII code (*integer*) → karakter

```
>>> for code in range(26):  
...     print(chr(ord('a') + code))  
...  
a  
b  
c  
d  
...  
z
```

Caesarrotatie



YHQL, YLGL, YLFL

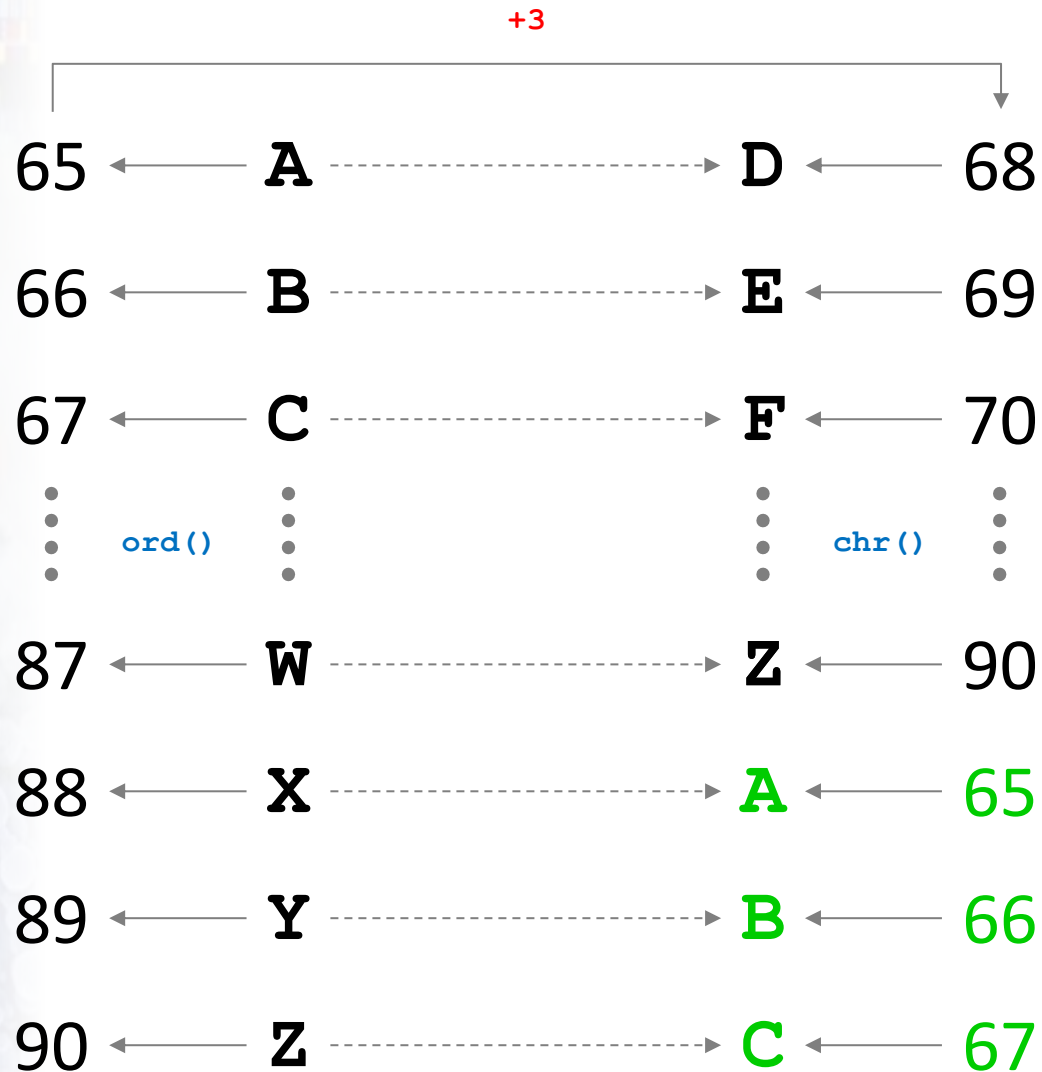


Caesarrotatie



+3			
65	←	A	-----> D ← 68
66	←	B	-----> E ← 69
67	←	C	-----> F ← 70
⋮	←	⋮	-----> ⋮ ← ⋮
		<code>ord()</code>	
87	←	W	-----> Z ← 90
88	←	X	-----> [← 91
89	←	Y	-----> \ ← 92
90	←	Z	----->] ← 93
		<code>chr()</code>	

Caesarrotatie



Caesarrotatie



+3

0	←	65	←	A	----->	D	←	68	←	3
1	←	66	←	B	----->	E	←	69	←	4
2	←	67	←	C	----->	F	←	70	←	5
⋮	←	⋮	←	⋮	----->	⋮	←	⋮	←	⋮
	←	<code>-ord('A')</code>	←	<code>ord()</code>	←	<code>chr()</code>	←	<code>+ord('A')</code>	←	
22	←	87	←	W	----->	Z	←	90	←	25
23	←	88	←	X	----->	A	←	65	←	26
24	←	89	←	Y	----->	B	←	66	←	27
25	←	90	←	Z	----->	C	←	67	←	28

Caesarrotatie



+3

0	←	65	←	A	----->	D	←	68	←	3	←	3
1	←	66	←	B	----->	E	←	69	←	4	←	4
2	←	67	←	C	----->	F	←	70	←	5	←	5
⋮		⋮		⋮		⋮		⋮		⋮		⋮
	←	<code>-ord('A')</code>	←	<code>ord()</code>			←	<code>chr()</code>	←	<code>+ord('A')</code>	←	<code>%26</code>
22	←	87	←	W	----->	Z	←	90	←	25	←	25
23	←	88	←	X	----->	A	←	65	←	0	←	26
24	←	89	←	Y	----->	B	←	66	←	1	←	27
25	←	90	←	Z	----->	C	←	67	←	2	←	28

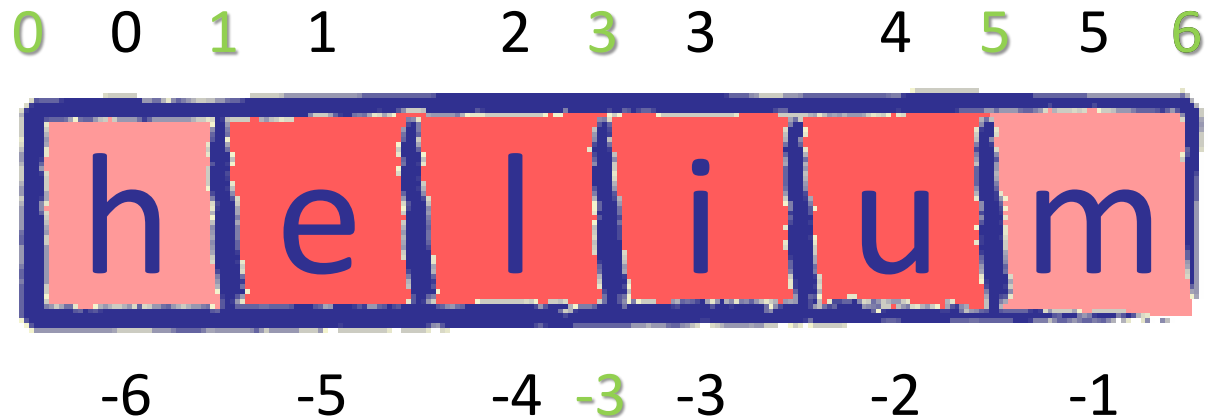
String slicing



- **slice**: subsequentie van een string
 - operator `[m:n]` geeft *substring* terug die start op positie **m** en loopt tot positie **n** (niet inclusief)

```
>>> element = 'helium'
>>> element[0:3]
'hel'
>>> element[1:5]
'eliu'
>>> element[-3:6]
'ium'
```

String slicing



```
>>> element = 'helium'
>>> element[0:3]
'hel'
>>> element[1:5]
'eliu'
>>> element[-3:6]
'ium'
```



String slicing



- **slice**: subsequentie van een string
 - operator `[m:n]` geeft *substring* terug die start op positie **m** en loopt tot positie **n** (niet inclusief)
 - begin van string als **m** leeg is
 - einde van string als **n** leeg is

```
>>> element = 'helium'
>>> element[:3]
'hel'
>>> element[-3:]
'ium'
>>> element2 = element[:]
>>> element2
'helium'
```

String slicing



- **slice**: subsequentie van een string
 - operator `[m:n:o]` geeft *subsequentie* terug die start op positie **m** en loopt tot positie **n** (niet inclusief), met sprongen van grootte **o**

```
>>> 'duikbootoorlog'[4:12:2]
'boor'
>>> 'schedelbeenderen'[0:16:3]
'seleen'
>>> 'schedelbeenderen'[::3]
'seleen'
>>> 'notenbomen'[::3]
'neon'
```

String slicing



- **slice**: subsequentie van een string
 - operator `[m:n:-o]` geeft *subsequentie* terug die start op positie **m** en terugloopt tot positie **n** (niet inclusief), met sprongen van grootte **o**
 - negatieve stapgrootte leest subsequentie van rechts naar links

```
>>> 'rekenwijze'[8:1:-2]
'zink'
>>> 'rekenwijze'[-2:0:-2]
'zink'
>>> 'muinotulp'[::-1]
'plutonium'
>>> 'navorderen'[::-3]
'neon'
```

Strings vergelijken



- vergelijking tussen strings is gebaseerd op ASCII tabel
 - strings worden karakter per karakter vergeleken (vlnr)
 - string met extra suffix is groter dan string zonder suffix

```
>>> 'a' == 'a'
True
>>> 'a' < 'b'
True
>>> 'cesium' >= 'cerium'
True
>>> '1' <= '9'
True
>>> 'appelboom' > 'appel'
True
```

Strings vergelijken



- vergelijking tussen strings is gebaseerd op ASCII tabel
 - strings worden karakter per karakter vergeleken (vlnr)
 - string met extra suffix is groter dan string zonder suffix

```
>>> 'a' < 'Z'
```

```
False
```

```
>>> 9 < 10
```

```
True
```

```
>>> '9' < '10'
```

```
False
```

```
>>> 'goud' < 'Zilver'
```

```
False
```

```
>>> 'goud'.lower() < 'Zilver'.lower()
```

```
True
```

Strings vergelijken



```
$ python3 case_sensitive.py
```

Geef de naam van een fruitsoort: **appel**

Het woord appel komt voor het woord banaan.

```
$ python3 case_sensitive.py
```

Geef de naam van een fruitsoort: **peer**

Het woord peer komt na het woord banaan.

```
$ python3 case_sensitive.py
```

Geef de naam van een fruitsoort: **Zebra**

Het woord Zebra komt voor het woord banaan.

case_sensitive.py

```
fruit = input('Geef de naam van een fruitsoort: ')

if fruit < 'banaan':
    print('Het woord', fruit, 'komt voor het woord banaan.')
elif fruit > 'banaan':
    print('Het woord', fruit, 'komt na het woord banaan.')
else:
    print('Je bent duidelijk een afstammeling van de apen !')
```



Strings vergelijken



```
$ python3 case_insensitive.py
```

Geef de naam van een fruitsoort: **appel**

Het woord appel komt voor het woord banaan.

```
$ python3 case_insensitive.py
```

Geef de naam van een fruitsoort: **peer**

Het woord peer komt na het woord banaan.

```
$ python3 case_insensitive.py
```

Geef de naam van een fruitsoort: **Zebra**

Het woord Zebra komt **na** het woord banaan.

case_insensitive.py

```
fruit = input('Geef de naam van een fruitsoort: ')

if fruit.lower() < 'banaan':
    print('Het woord', fruit, 'komt voor het woord banaan.')
elif fruit.lower() > 'banaan':
    print('Het woord', fruit, 'komt na het woord banaan.')
else:
    print('Je bent duidelijk een afstammeling van de apen !')
```



Strings vergelijken



- operator **in**
 - gaat na of eerste string een subtring is van tweede string
 - opmerking: string is substring van zichzelf

```
>>> 'p' in 'appel'
True
>>> 'i' in 'appel'
False
>>> 'ap' in 'appel'
True
>>> 'pa' in 'appel'
False
>>> 'a' in 'a'
True
>>> 'appel' in 'appel'
True
```



Strings vergelijken



```
$ python3 zonder_klinkers.py
```

```
Geef een woord: abracadabra
```

```
brcdbr
```

```
$ python3 zonder_klinkers.py
```

```
Geef een woord: PYTHON ZONDER KLINKERS
```

```
PYTHN ZNDR KLNKRS
```

zonder_klinkers.py

```
woord = input('Geef een woord: ')
```

```
klinkers = 'aeiou'
```

```
woord_zonder_klinkers = ''
```

```
for letter in woord:
```

```
    if letter.lower() not in klinkers:
```

```
        woord_zonder_klinkers += letter
```

```
print(woord_zonder_klinkers)
```



Tandenstokers



Mutability



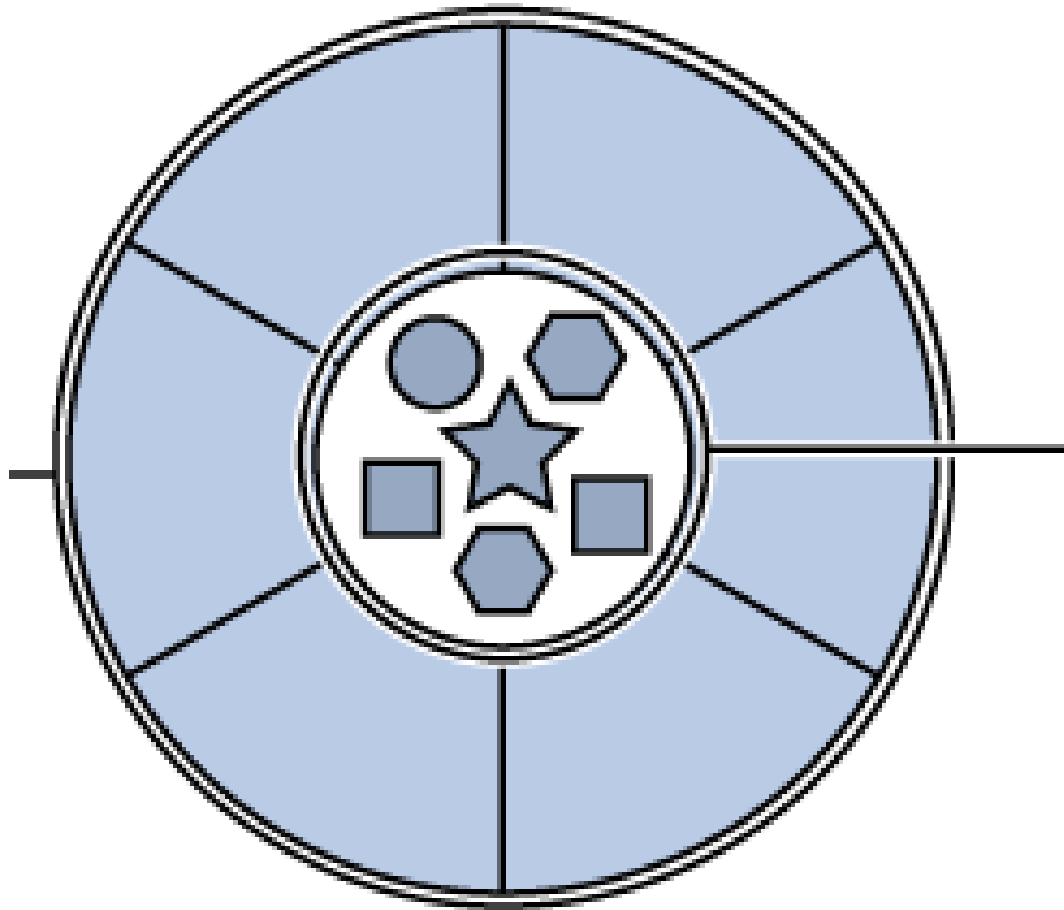
- strings zijn onveranderlijk (*immutable*)
 - string in werkgeheugen kan niet gewijzigd worden (*no in place assignment*)

```
>>> naam = 'darwin'
>>> naam[0] = 'D'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> naam = naam[0].upper() + naam[1:]
>>> naam
'Darwin'
```

Methoden



**methoden
(gedrag)**

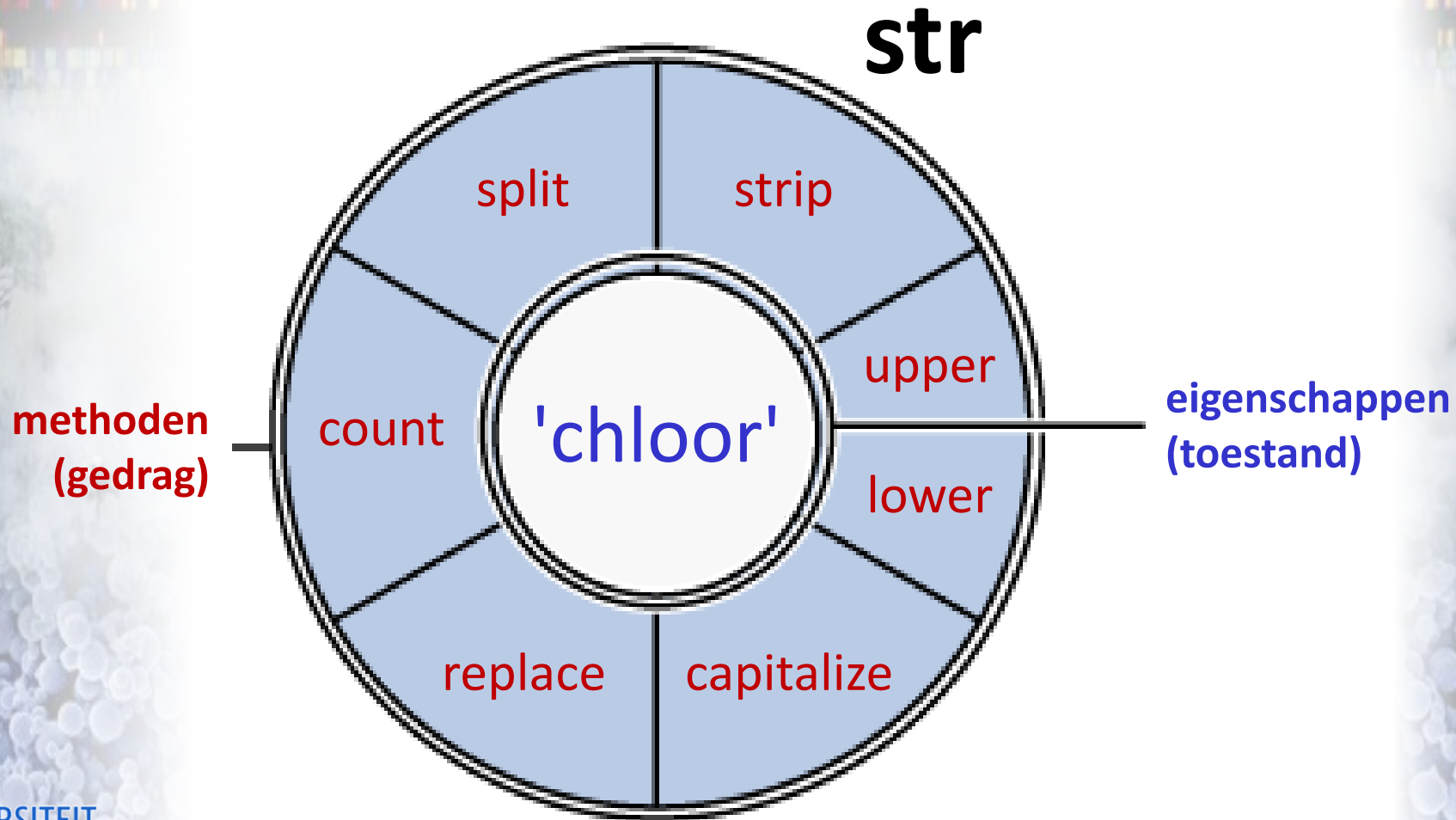


**eigenschappen
(toestand)**



UNIVERSITEIT
GENT

Methoden



Methoden



- **methode**: functie gekoppeld aan bepaald object
 - bijna alle Python-objecten hebben methoden
 - aanroepen van methode **meth** voor object **obj**
 - `obj.meth()` (*dot notation*)
 - vaak wordt extra informatie meegegeven aan methode
 - argumenten tussen ronde haakjes

```
>>> element = 'chloor'
>>> element.upper()
'CHLOOR'
>>> element.capitalize()
'Chloor'
>>> element.replace('c', 'C')
'Chloor'
```

Stringmethoden



- methoden aanroepen: constanten én variabelen
- stringmethoden geven nieuwe string terug
 - bestaande strings kunnen niet veranderd worden (*immutable*)

naam	functionaliteit	voorbeeld	resultaat
capitalize	eerst letter omzetten naar hoofdletter	<code>'darwin'.capitalize()</code>	<code>'Darwin'</code>
lower	alle letters omzetten naar kleine letters	<code>'dEUS'.lower()</code>	<code>'deus'</code>
upper	alle letters omzetten naar hoofdletters	<code>'dEUS'.upper()</code>	<code>'DEUS'</code>
strip	witruimte vooraan en achteraan verwijderen (spaties, tabs, regeleindes, ...)	<code>' a b '.strip()</code>	<code>'a b'</code>
lstrip	witruimte vooraan verwijderen	<code>' a b '.lstrip()</code>	<code>'a b '</code>
rstrip	witruimte achteraan verwijderen	<code>' a b '.rstrip()</code>	<code>' a b'</code>
count	tellen hoe vaak één string binnen andere string voorkomt	<code>'abracadabra'.count('ra')</code>	<code>2</code>
find	positie opzoeken waar een string eerste keer binnen andere string voorkomt (-1 indien geen voorkomens gevonden)	<code>'abracadabra'.find('ra')</code> <code>'abracadabra'.find('xyz')</code>	<code>2</code> <code>-1</code>
replace	vervang voorkomens van een string door een andere string	<code>'abracadabra'.replace('ra', '-')</code>	<code>'ab-cadab-'</code>

Stringmethoden



- methoden aanroepen: constanten én variabelen
- stringmethoden geven nieuwe string terug
 - bestaande strings kunnen niet veranderd worden (*immutable*)

```
>>> element = 'helium'
>>> element.upper()
'HELIUM'
>>> element.replace('el', 'afn')
'hafnium'
>>> print(f'element na aanroepen methoden: {element}')
element na aanroepen methoden: helium
```

Stringmethoden



- methoden aanroepen: constanten én variabelen
- stringmethoden geven nieuwe string terug
 - bestaande strings kunnen niet veranderd worden (*mutability*)
 - methoden kunnen aaneengeschaakeld worden (*chaining*)
 - methode geeft object terug → roep methode van dit object aan
 - evaluatie gebeurt van links naar rechts

```
>>> element = 'chloor'
>>> '***' + element.upper().center(10) + '***'
'***  CHLOOR  ***'
>>> element.replace('l', 'r').replace('r', 'm')
'chmoom'
>>> element.replace('r', 'm').replace('l', 'r')
'chroom'
```

Stringmethoden



- wat kan ik allemaal doen met een string ?

```
>>> element = 'broom'
```

```
>>> dir(element)
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__format__',  
 '__ge__', '__getattr__', '__getitem__', '__getnewargs__', '__gt__', '__hash__',  
 '__init__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__',  
 '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__',  
 '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'center', 'count', 'encode',  
 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha',  
 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace',  
 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition',  
 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',  
 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

```
>>> help(str)
```

```
Help on class str in module __builtin__:
```

```
class str(basestring)
```

```
| str(object) -> string
```

```
|
```

```
| Return a nice string representation of the object.
```

```
| If the argument is a string, the return value is the same object.
```

```
...
```

Stringmethoden



- wat kan ik allemaal doen met een string ?
- hoe werkt de methode **find** voor strings ?
 - argumenten tussen vierkante haakjes zijn optioneel

```
>>> help(str.find)
```

```
Help on method_descriptor:
```

```
find(...)
```

```
S.find(sub [,start [,end]]) -> int
```

Return the lowest index in S where substring sub is found, such that sub is contained within s[start:end]. Optional arguments start and end are interpreted as in slice notation.

Return -1 on failure.

(END)



Tekstopmaak



- stringmethode **format** (*string formatting*)
 - korte en krachtige manier om tekst op te maken

```
'formaat'.format(waarde, waarde, ...)
```

```
>>> 'Het chemische element {}.format('boor')
'Het chemische element boor.'
>>> element, symbool, atoomnummer = 'boor', 'B', 5
>>> 'Element {} ({{}} heeft atoomnummer {}.format(
... element, symbool, atoomnummer)
'Element boor (B) heeft atoomnummer 5.'
>>> n1 = 4
>>> n2 = 5
>>> '2**10 = {} en {} * {} = {}'.format(2**10,n1,n2,n1*n2)
'2**10 = 1024 en 4 * 5 = 20.000000'
```



Tekstopmaak



opmaak1.py

```
i = 1
print('i\ti**2\ti**3\ti**5\ti**10\ti**20')
while i <= 10:
    print(i, '\t', i**2, '\t', i**3, '\t',
          i**5, '\t', i**10, '\t', i**20)
    i += 1
```

i	i**2	i**3	i**5	i**10	i**20
1	1	1	1	1	1
2	4	8	32	1024	1048576
3	9	27	243	59049	3486784401
4	16	64	1024	1048576	1099511627776
5	25	125	3125	9765625	95367431640625
6	36	216	7776	60466176	3656158440062976
7	49	343	16807	282475249	79792266297612001
8	64	512	32768	1073741824	1152921504606846976
9	81	729	59049	3486784401	12157665459056928801
10	100	1000	100000	10000000000	100000000000000000000

Tekstopmaak



opmaak2.py

```
i = 1
print('{:<4s}{:<5s}{:<6s}{:<8s}{:<13s}{:<15s}'.format(
    'i', 'i**2', 'i**3', 'i**5', 'i**10', 'i**20'))
while i <= 10:
    print('{:<4d}{:<5d}{:<6d}{:<8d}{:<13d}{:<15d}'.format(
        i, i**2, i**3, i**5, i**10, i**20))
    i += 1
```



i	i**2	i**3	i**5	i**10	i**20
1	1	1	1	1	1
2	4	8	32	1024	1048576
3	9	27	243	59049	3486784401
4	16	64	1024	1048576	1099511627776
5	25	125	3125	9765625	95367431640625
6	36	216	7776	60466176	3656158440062976
7	49	343	16807	282475249	79792266297612001
8	64	512	32768	1073741824	1152921504606846976
9	81	729	59049	3486784401	12157665459056928801
10	100	1000	100000	10000000000	100000000000000000000

Huiswerk (volgende les)



- lees handboek
 - hoofdstuk 5 (bestanden en exceptions; zelfstudie)
 - hoofdstuk 6 (functies)
- lees omschrijving van voorbeeldopgaven (reeks 5)
 - Meirp
 - GSMoniemen
 - Sloppeut



Huiswerk (werkcolleges)



- bestudeer stringmethoden: `help(str)`
- bestudeer de *Format String Syntax*
- opgelegde oefeningen reeks 4
 - ISBN (demo)
 - Dagboek voor Stella
 - Een speld in een hooiberg
 - Rustig aan
 - De letters van Utrecht
 - Wepe sprepekepen p

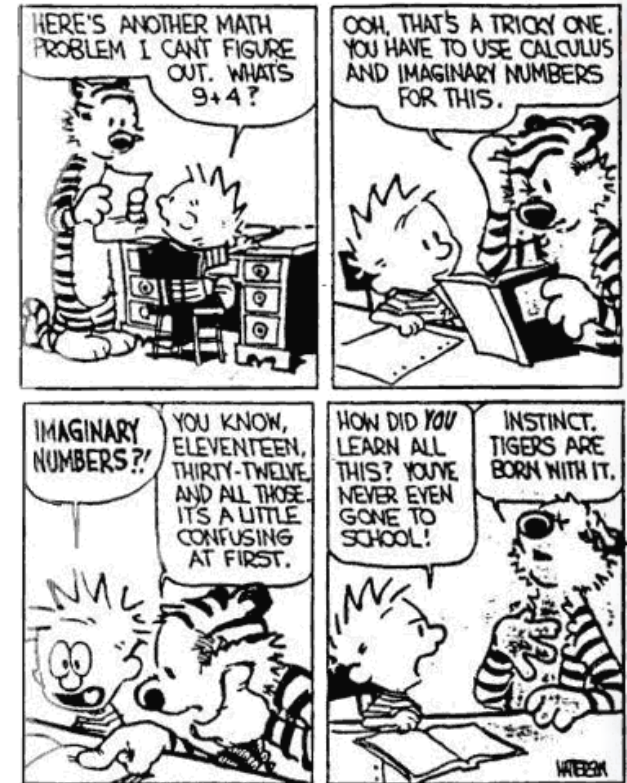
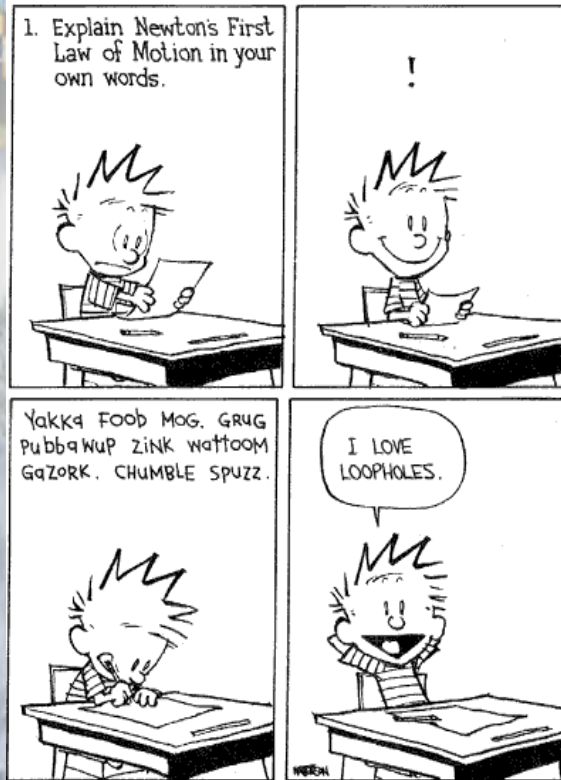


Vragen of opmerkingen?



UNIVERSITEIT
GENT

The sky is the limit ...



"Why waste time learning
when ignorance is instantaneous"

— Hobbes