Samenvatting programmeren evaluatie 1

HOOFDSTUK 1: variabelen, expressies en statements

De verschillende manieren om input te vragen:

- x = input()
- x = input("ik wil het getal x= ")
- x = int(input())
- x, y = int(input()), int(input())

Verder doen op een andere lijn:

*- *

De verschillende manieren om te printen:

- print("a is gelijk aan:",a, "and y is gelijk aan",y)
- print('tekstje')
-print('tekstje met{}'.format(variabele))
-print(f'tekstje')
-print(
f"Stopwatchbaby's zijn {dag} {dagen}, {leeftijd_maanden} {maanden} en " +
f'{leeftijd_jaren} {jaren} oud op {dag} {maand} {jaar}.'
)

Handige modules:

- import math
- import string
- import decimal
-import fractions

Operators:

+	-	*	* *	/	//	%
<<	>>	&	|	^	~	
<	>	<=	>=	==	!=	<>
+=	-=	*=	/=	//=	%=	
&=	|=	^=	>>=	<<=	* *=	

Punctuators:

(	)	[	]	{	}
,	:	.	`	=	;
'	"	#	\	@	

Pythonic tips:

naam variabele moet beginnen met letter of _
een cijfer mag niet als eerste karakter
/ (delen door) gaat altijd een float geven
a += 1

Id en types:

- a = 7
- id(a)

> 16790848
-type(a)
int

Verschillende klasses:
string, lijst[], tuple(), integer, float, bool, dict{}, set{}

Math module:
math.sin
math.cos
math.pow -> x^y
math.sqrt
math.hypot -> sqrt($x^2 + y^2$)
math.fabs -> absolute waarde van numeriek
from math import sqrt

Twee variabelen van waarde wisselen:
-*a, b = b, a*

Enkele handige functies:
-*sum(a,b,c)*
-len('a')
-math.ceil(getal): een float naar boven afronden
-math.floor(getal): een float naar beneden afronden
-math.round(getal, orde): een float afronden naar aantal cijfers gegeven in orde
-math.sqrt(getal)
-math.log(getal, [base]): base is optioneel, automatisch getal van Euler
-math.sin(x)
-math.asin(x)
-math.degrees(x): van graden naar radialen
-math.radians(x): van radialen naar graden
-math.pi, math.e -> constanten
-math.pow(x,y)
-math.gcs(x, y): grootste gemene deler
-abs(getal) -> absolute waarde
-max(a,b,c)
-min(a,b,c)
-2.5e2 -> 250
-round(getal- -> gaat normaal afronden

Getal uitschrijven in dagen, uren en minuten vanuit seconden
minuten = periode // 60
*dagen = minuten // (24 * 60)*
*minuten %= 24 *60*
uren = minuten // 60
minuten %= 60

HOOFDSTUK 2: voorwaardelijke opdrachten en lussen

Boolean operators:

<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
==	equal to
!=	not equal to

Pythonic tips:

- altijd intend van 4 spaties of tab na if/else/elif/while/for (met : als laatste)
- '_' is standaard in for loop als variabele niet gebruikt wordt
- 0 of 0.0 wordt geconsidereerd als False
- '==' als ze dezelfde value hebben, 'is' als ze dezelfde id hebben
- geneste voorwaarden liefst vermijden, maar mag wel
-if x: is beter dan if x != 0

Iets printen in een for of while lus zonder dat het op een nieuwe lijn terecht komt

- print(a, end=')

Iets printen zodat er een tab tussen zit

- print(a, '\t', b, '\t, c)

Iets printen in een loop zodat er een tab tussen zit

- while …
print(1,2,3, sep='\t)

OF -while …
print(1, end= '\t')
print(2, end= '\t')
print(3,)

Je wilt een volgende lijn

print() of print("first line \n second line")

For loop door een getallenreeks/ for loop door karakters:

- for _ in range(123456):
- for _ in range(3,10) -> 3 zit er bij, 10 niet meer
- for _ in range(3, 10, 2)
-for _ in range(10, 0, -1)
- for _ in 'spam':

Als je wilt controleren of twee float getallen gelijk zijn aan elkaar: (vaak zijn ze gelijk aan elkaar, maar door foute afronding geeft python niet de juiste waarde:

- abs(x – y) < 0.0000001
> True

Enkele handige tips for statements:

- if not voorwaarde
-if a in be (string, lijsten, tuples..)
- if voorwaarde
-if isinstance(a, datatype (bv int)) -> controlleert of a van het gevraagde datatype is

Boolean operators:

Operator	*Description*
()	Parenthesis (grouping)
**	Exponentiation
+x, -x	Positive, Negative
*,/,%	Multiplication, Division, Remainder
+,-	Addition, Subtraction
<, <=, >, >=,! =, ==	Comparisons
not x	Boolean NOT not false -> true, r
and	Boolean AND
or	Boolean OR

True voor False ->
> True and False or True and False
> (True and False) or (True and False)
> False or False
> False

Meerdere assigments:
- a, b, c = 1, 2, 3

Slechte statements (niet gebruiken!!!)
- break -> uit de loop gaan
- continue -> ga terug naar begin van loop
- pass -> doet niks

Wanneer welke loop?
- for loop als je al op voorhand weet hoeveel keer je de statement gaat doorlopen (probeer altijd eerst deze)
- while loop als het nummer onbekend is

Lijst maken met for loop:
```
> list = []
> for counter in range(10):
        lijst.append(counter*2)

Print(lijst)
> [0,2,4,6,8,10,12,14,16,18]
```

Kortere notatie:
```
- if a == b:
        c = d
-else:
        c = e
-> c = d if a == b else e
```

String opschonen (alle andere karakters wegdoen:
liedje = ''.join(karakter for karakter in liedje.lower() if karakter.isalpha())

Hoe een iterator werkt:
- iterator = iter('spam')
- next(iterator)
> s
- next(iterator)
>p

Modele random
-import random
- getal : random.randint(1,10)

Itereren:
- for index, letter in enumerate(woord):

TIPS:

voor integers wordt enkel 0 naar False omgezet
- voor floats wordt enkel 0.0 naar False omgezet
- voor strings wordt enkel de lege string ('') naar False omgezet
- voor lijsten wordt enkel de lege lijst ([]) naar False omgezet

HOOFDSTUK 4: strings

Een string definiëren met ‘ erin:
-‘bill’\s’

Informatie over tabs, spaties etc behouden
’’’drie zorgen ervoor dat alles behouden blijft’’’

Handige functies
-ord(a)
-chr(1)
*-enkel + en * kunnen op strings gebruikt worden*
-all(iterable) -> is True als all items in iterable true zijn

De slice functie:
-spam[0]
> ‘s’
-spam[-1]
> ‘m’
-programmeren[2,5]
>’ogr’
-programmeren[:3]
>’pro’
-programmeren[5:]
>’ammeren’
-programmeren[3:-2]
>’grammer’ (bij het terugkeren start men niet van nul, maar vanaf 1)
-programmeren[::2]
>’porrmee’
-programmeren[::-1]
>’neremmargorp’
-programmeren[:]
>’programmeren’

2 Strings vergelijken:
- ‘a’ < ‘A’:
> True
- ‘abc’ < ‘abd’:
> True
- ‘a’ < ‘’:
> False

De in operator
-‘a’ in abc
> True
-‘ab’ in abc
> True
-‘ba’ in abc
> False

Tips:

-Strings zijn immutable <-> lijsten
Zo weinig mogelijk print statements!

Simpele functies voor strings:

-len('spam') = 4 (werkt alleen bij strings)
-string.upper() -> alles met hoofdletter
-string.lower() -> alles met lage letter
-string.capitalize() -> eerste letter met hoofdletter zetten
-string.center(getal) -> gaat de string meer laten opschuiven naar rechts hoe groter getal is
-string.center(4, "0") -> 00string00
-count = string.count(substring) -> tellen hoeveel keer substring voorkomt in string
-count = string.count(substring, 8,25) -> tussen indexen 8 en 25
-string.endswith(".") -> controleren of string eindigt op punt (True of False)
-string.expandtabs(3) gaat tab verhogen met 3 spaties
-string.index(..) -> zelfde als find
-string.isalnum() -> controleert of de string allemaal karakters bezig die abc zijn of 012
-string.isalpha() -> controleert of string allemaal karakters zijn
-string.isdigit() -> controleert of string uit nummers bestaat
-string.islower() -> controleert of alle letters klein zijn
-string.isupper() -> controleert of alle letters groot zijn
-string.isspace() -> controleert of alle karakters spaties zijn
-string.istitle() -> geeft True als van elk woord in string alleen de eerste letter hoofdletter is
-string.join(tuple) -> gaat alle waardes in tuples verbinden met elkaar via de string
-string.ljust(n) -> gaat n spaties na de string plaatsen
-string.partition("woord") -> gaat tuple creeren met element string voor woord, woord en na woord
-string.replace("woord", "ander woord") -> gaat elk woord in string vervangen door ander woord
-string.rfind("woord") -> gaat achteraan de string beginnen zoeken naar woord en index returnen
-string.rindex("woord") -> ongeveer hetzelfde als rfind
-string.rjust(n) -> gaat n spaties voor de string plaatsen
-string.rpartition("woord") -> hetzelfde als partition, maarbegint achteraan in string te zoeken
-string.rsplit(", " , 1) -> gaat lijst creeren met twee elementen (varieert door 1 te veranderen naar ander getal) gesplitst door komma (begint achteraan
-string.split(",",getal) -> lijst met getal+1 elementen splitsen (begint vooraan te splitsen)
-string.splitlines() -> string heeft \n in zijn naam -> gaat lijst creeren die splitst vanaf \n
-string.startswith("woord") -> controleert of woord eerste woord is in string
-string.lstrip() -> alle voor karakters strippen
-string.strip() -> alle voor en na karakters strippen (default is spati)
-string.rstrip('-') -> alle na karakters strippen (dus gaat na woord alle – strippen hier)
-string.swapcase() -> gaat alle hoofdletters in string omzetten in kleine letters en omgekeerd
-string.title() -> zal elke eerste letter van woord een hoofdletter van maken
-string.translate(dictionary) -> een dict gebruiken om specifieke letter te vervangen met een andere
-string.zfill(n) -> gaat n '0' zetten voor uw string

Nesten van functies:

-'string'.upper().find('S')
> 0

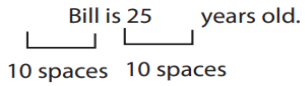
De find functie (index doet ong hetzelde -> gaat ipv -1 exception geven)

- a = 'programmeren'
-a.find('r')
>1
-a.find('r', 5)
>9
-a.find('r', a.find('t' + 1) -> gaat volgende zoeken na eerste 'r'
>4
-a.find('gram')
>3
-a.find('r', 3, 6)
>4

De format functie

- "format string".format(data1, data2,..)
- print("{} is {} zoveel".format("Bill", 25))
- print("{a} is {b} zoveel".format(b = 25, a = "Bill")
-print("{1} is {0} zoveel".format(25, "Bill")

-print('{:>10s} is {:<10d} years old.'format('Bill, 25))



-print("pi is {:.4f}".format(math.pi))
> Pi is 3.1416
-print("pi is {:8.4f}".format(math.pi))
> Pi is 3.1416
-print("pi is {:8.2f}".format(math.pi))
> Pi is 3.14
-print("{:8.2%}.format(2/3))
>66,67%

-print('{0:.>12s} | {1:0=+10d} | {2:->5d}'.format('abc',35,22))
>.....abc | +000000035 | ---22
-print('{:#6.0f}'.format(3)) (gaat een . op het einde zetten)
>3.
- print('{:04d}'.format(4))
>0004
- print('{:,d}'.format(1234567890)) (gaat met duizend tallen separaten)
>1,234,567,890

s	string
d	decimal integer
f	floating-point decimal
e	floating-point exponential
%	floating-point as percent

<	left (default for most objects)
>	
^	
=	

+	sign for positive and negative
-	sign for negative only (default)
space	space for positive; minus for negative

Er zijn nog andere manieren om voorloopnullen te genereren. Zo kan je ook gebruikmaken van de stringmethode zfill (zero fill). Je kan ook het verschil tussen de huidige lengte en de gewenste lengte berekenen en dan weet je hoeveel nullen je moet toevoegen. Of je kan ook werken met een while lus die nullen blijft toevoegen totdat de string de gewenste lengte heeft.

De string module

string.punctuation: '!"# $%&\'()+,-./:;<=>?@[\\]^_ `\{\}~' |*
string.digits: '0123456789'
string.ascii lowercase: 'abcdefghijklmnopqrstuvwxyz'
string.whitespace: '\t\n\x0b\x0c\r '

Variabele maken om te printen:

getal = f'{even}{oneven}' -> print(getal)

HOOFDSTUK 5: Files and exceptions

Een file openen en sluiten

temp.txt =
First line
Second line
Third line

- temp_file = open (“temp.txt”,”r”)
for line_str in temp_file:
print(line_str, end= ‘ ’)

>first line
>second line
>third line

-temp_file_close()

Met errors werken

-try:
 # suite of code to watch here
-except ParticularErrorName:
 # suite of code to handle the named error, if it occurs

Voorbeeld:
try:
observatie = int(input())
if observatie >= 100:
score += 2
geldig = False

except ValueError:
geldig = False

Soorten errors:
-IOError
-ValueError

Indien er een extra lijn in de output is met niks erin:
end =’ ‘ in print statement zetten

HOOFDSTUK 6: Functies

Functie beschrijven:

```
def functie_naam (parameter1, parameter2):
        statement1
        statement2
        return value_to_return
```

Handige tips:

Vanaf je in functie de return statement ziet, stopt de functie (<-> for/while loops)
Zowel float als int kunnen output zijn, er wordt geen verschil in gemaakt
Indien geen return, gaat functie 'None' als uitput geven

Hoe ga je best debuggen:

Functie definiëren, print statement onderaan met functie en daarin parameters

Testgevallen in functie

```
"""
>>> isPrime(2)
True
>>> isPrime(32)
"""

If __name__ == '__main__':
    import doctest
    doctest.testmod()
```

Voorbeeld van return statement:

```
return f'#{counter}, {str(huidig).find(reeks)+1}-{str(huidig).find(reeks)+len(reeks)} van {len(str(huidig))}'
```

Uren en minuten uitschrijven wanneer je het gaat verhogen of verlagen

```
>>> uren = 15
>>> minuten = 20
>>> minuten_sinds_middernacht = 60 * uren + minuten
>>> minuten_sinds_middernacht
920
>>> minuten_sinds_middernacht += 50          # verhoog aantal minuten met 50
>>> minuten_sinds_middernacht
970
>>> (minuten_sinds_middernacht // 60) % 24   # aantal uren op 24-uursklok
16
>>> minuten_sinds_middernacht % 60           # aantal minuten op 24-uursklok
10
```

Synchroon 2 strings overlopen

```
>>> eerste = 'abc'
>>> tweede = 'def'
>>> for index, karakter in enumerate(eerste):
...     print(f'{karakter}-{tweede[index]}')
...
a-d
b-e
c-f
```

```
>>> eerste = 'abc'
>>> tweede = 'def'
>>> for karakter1, karakter2 in zip(eerste, tweede):
...     print(f'{karakter1}-{karakter2}')
...
a-d
b-e
c-f
```