

Spiekbriefje Python

help

```
help( str ) # print help-documentatie over de module tussen de haakjes
```

input en output

```
print( "hallo!" ) # hallo!
naam = input( "geef je naam: " ) # alles wat op 1 regel ingegeven wordt, wordt
# (met eventuele spaties) bewaard in variabele
x = int( input( "geef getal: " ) ) # int() zet input om naar geheel getal
x = float( input( "geef getal: " ) ) # float() zet input om naar reëel getal
print( naam ) # print de waarde van de variabele uit (bvb. Bo)
print( x ) # print de waarde van de variabele uit (bvb. 7)
```

toekenning en wiskundige operatoren

```
som = a + b # ken de waarde a+b toe aan de variabele som
# operator + kan ook op tekst gebruikt worden (beide operandi type str)
verschil = a - b
product = a * b # operator * kan ook op tekst gebruikt worden
# (een van de operandi type string, andere type int)
# 3.5
deling = 7 / 2
gehele_deling = 7 // 2 # 3 (i.p.v. 3.5)
rest = 14 % 10 # 4, rest bij gehele deling
macht = 2 ** 3 # 8, machtsverheffing, gebruik geen ^ i.p.v. **
som += x # verkorte vorm voor som = som + x
# analoog voor andere operatoren (-= /= *)
vogels[0] = "mus" # verander het eerste element in de lijst 'vogels'
```

vergelijkingsoperatoren op int, float, str

```
a == b # test op gelijkheid (voor int, float, str én bool)
a != b # test op ongelijkheid (voor int, float, str én bool)
-2.35 < 1.7 # test op 'strikt kleiner dan', True in dit geval
"aap" <= "app" # test of linkeroperand alfabetisch voor rechteroperand komt, True
"bos" > "bos" # test of linker- alfabetisch strikt na rechteroperand komt, False
10000 >= 100 # test op 'groter of gelijk aan', True
```

logische operatoren

```
not het_regent # not: negatie van de logische uitspraak die volgt
het_regent and de_zon_schijnt # and: logische EN
a < 10 or a > 20 # or: logische OF (hier: True als a ∉ [10,20] )
```

in - operator

```
if "el" in "appelmoes": # gaat na of tekst "el" in "appelmoes" zit (True)
if 7 in [6,77,8]: # gaat na of 7 element is van lijst [6,77,8] (False)
```

STRUCTUREN

keuze: if-structuur

```
if a == 0:
    print( "nul" )
elif a < 0: # elif is optioneel (hoeft er niet te zijn)
    print( "negatief" )
else: # else is optioneel (hoeft er niet te zijn)
    print( "positief" )
```

herhaling (aantal niet vooraf gekend): while-lus

```
# lees getallen in (en sommeer ze) tot er een getal groter dan 1000 gegeven wordt
som = 0 # geen deel van while-lus (wel voorbereiding)
getal = float(input("Geef getal: ")) # deel 1 van while-lus
while getal <= 1000: # deel 2 van while-lus (geteste item is getal)
    som += getal # deel 3 van while-lus
    getal = float(input("Nog een: ")) # deel 4 van while-lus (item wordt klaargezet)
print( f"{som=}") # geen deel van de while-lus (gebeurt 1 maal)
```

```
# lees getallen in (en sommeer ze) tot hun som groter dan 1000 is
som = 0 # deel 1 van while-lus
while som <= 1000: # deel 2 van while-lus (geteste item is som)
    getal = float(input( "Geef getal: ")) # deel 3 van while-lus
    som += getal # deel 4 van while-lus (item wordt klaargezet)
print( f"{som=}") # geen deel van de while-lus (gebeurt 1 maal)
```

Tip Schrijf eerst deel 2 van de lus (daaruit bepaal je het item dat getest wordt); daarna deel 1 en 4 (die zetten het item (voor het eerst of opnieuw) klaar en lijken wschl sterk op elkaar); tot slot deel 3 aanvullen met alles wat ook herhaald moet worden (deel 3 kan leeg zijn).

herhaling (aantal vooraf gekend): for-lus

```
for i in range(5): # teller in een range heeft best naam i, k, teller of index
    print(i) # i heeft achtereenvolgens de waarde 0 1 2 3 4

for i in range(10,20,2): # in range(start,stop,step) stop is altijd exclusief
    print(i) # print de getallen 10 12 14 16 18 (telkens op nieuwe regel)

for letter in tekst:
    print(letter) # print elke letter (of karakterteken) van de tekst
# (telkens op nieuwe regel)
```

```
vogels = ["mus","mees","raaf"]
for vogel in vogels: # deze lus kan de elementen niet vervangen
    print(vogel) # print de woorden mus mees raaf (telkens op nieuwe regel)
```

```
for i in range(len(vogels)):
    vogels[i] = vogels[i] + "je" # korte versie: vogels[i] += "je"
# na afloop van bovenstaande lus is vogels gelijk aan ["musje","meesje","raafje"]
```

STRINGS

string is een collectie letters, te overlopen met for

```
tekst = "papegaaienei"
for letter in tekst: # voor elke letter in de string
    if letter in "aeiou": # controleer of die letter een klinker is
        print(letter) # zo ja, print uit
```

slicing: deel van de collectie letters kopiëren

```
tekst[start] # letter op index 'start'
tekst[start:stop] # substring van startindex tot net voor stopindex
tekst[start:stop:step] # analoog, enkel letters die stepgrootte van elkaar liggen
# negatieve step: van achter naar voor
tekst[:] # kopie van hele tekst
tekst[::-1] # kopie van hele tekst achterstevoren
# start ontbreekt -> vanaf index 0
# stop ontbreekt -> tot het einde
# step ontbreekt -> stepgrootte = 1
```

functies die string-argument nemen

```
lengte = len("dagen") # berekent lengte van de string "dagen" (5)
getal = float("-2.5") # zet tekst "-2.5" om naar reëel getal (getal bevat nu -2.5)
getal = int("123") # zet tekst "123" om naar geheel getal (getal bevat nu 123)
```

operatoren op strings

```
"appel" + "bloesem" # concatenatie
"hip " * 3 # dupliceren, resultaat is "hiphiphip"
2 * "hoi" # dupliceren, resultaat is "hoihoi"
"appel" == "appel" # vergelijken (gelijk aan)
"appel" != "peer" # vergelijken (niet gelijk aan)
"appel" < "appelboom" # 'komt alfabetisch voor'; hoofdlettergevoelig
"appel" <= "appelboom" # 'komt alfabetisch voor of is gelijk'; hoofdlettergevoelig
"peer" > "appel" # 'komt alfabetisch na'; hoofdlettergevoelig
"appel" >= "appel" # 'komt alfabetisch na of is gelijk'; hoofdlettergevoelig
"el" in "appelboom" # gaat na of "el" substring is van "appelboom" (True)
```

klasse string: constanten

```
import string # importeer eerst module string voor constanten gekend zijn
string.ascii_letters # abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
string.ascii_lowercase # abcdefghijklmnopqrstuvwxyz
string.ascii_uppercase # ABCDEFGHIJKLMNOPQRSTUVWXYZ
string.digits # 0123456789
string.punctuation # !"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~
```

methodes van de klasse string

omzetten naar kleine letters en/of hoofdletters

```
tekst = "HeLlo 2u!!"
klein = tekst.lower() # klein bevat "hello 2u!!"
groot = tekst.upper() # groot bevat "HELLO 2U!!"
titel = tekst.capitalize() # titel bevat "Hello 2u!!" (start met hoofdletter)
wissel = tekst.swapcase() # wissel bevat "hEllo 2U!!"
```

controleren op soort karakters

```
# gaat na of alle karakters in de variabele tekst...
if tekst.isalnum(): # ... een letter of cijfer voorstellen (alfanumeriek)
if tekst.isalpha(): # ... een letter (groot of klein) voorstellen
if tekst.isdigit(): # ... een cijfer voorstellen
if tekst.islower(): # ... een kleine letter voorstellen
if tekst.isupper(): # ... een hoofdletter voorstellen
if tekst.isspace(): # ... een witruimte (spatie, tab, return,...) voorstellen
if tekst.istitle(): # gaat na of de eerst voorkomende letter een hoofdletter is,
# en alle volgende letters klein
```

zoeken naar substring, suffix, prefix

```
index = "appels".find("el") # bepaalt de eerste vindplaats van "el" in "appels"
# (hier is dat 3); geeft -1 indien niet aanwezig
aantal = zin.count("te") # telt hoe dikwijls substring "te" aanwezig is in zin
if "zus".endswith("us"): # gaat na of "zus" eindigt op "us" (True)
if "broer".startswith("bro"): # gaat na of "broer" start met "bro" (True)
```

spaties voor en achter weghalen

```
zonder = " hé, hallo! ".strip() # haalt spaties voor & achter weg
# zonder bevat nu "hé, hallo!"
```

substrings vervangen en verwijderen

```
zin = "het droom van het jongen"
nieuw = zin.replace("het", "de") # vervangt in zin elk voorkomen van "het" in "de"
print(nieuw) # print op het scherm: de droom van de jongen
weg = zin.replace("het", "") # vervangt in zin elk voorkomen van "het" in lege str
# dus verwijdert elk voorkomen van "het"
# variabele weg bevat nu " droom van jongen"
```

opsplitsen en samenplakken

```
woorden = zin.split() # de zin wordt gesplitst op spaties; elk deel
# wordt toegevoegd aan de lijst woorden
wdn = " Dat is ok. ".split() # wdn is lijst ["Dat", "is", "ok."] (op spatie)
w = " a;b c;d ".split(";") # w = [" a", "b c", "d "] (op ;-teken gesplitst)
zin = " ".join(woorden) # de elementen van de opsomming (bijv. een lijst) woorden
# worden aan elkaar geplakt met een spatie (" ") ertussen
zin = "-".join(["gi", "gan", "tisch"]) # zin bevat "gi-gan-tisch" (met "-" tussen)
```

STRINGS voor output

f-string, geïnterpoleerde string, formatted string

inhoud van variabelen makkelijk weergeven

```
naam = input("Geef je naam: ") # gebruiker tikt in: Bo
x = int(input("Geef een getal: ")) # gebruiker tikt in: 7
print("Dag {naam}, je gaf {x} in.") # Dag {naam}, je gaf {x} in. -> f vergeten!
print( f"Dag {naam}, je gaf {x} in.") # Dag Bo, je gaf 7 in. -> f voor 'formatted'
print( f'Dag "{naam}", je gaf {x} in.') # Dag "Bo", je gaf 7 in. (let op ' ↔ "')
print( f'Dag '{naam}', je gaf {x} in.') # Dag 'Bo', je gaf 7 in. (let op " ↔ ')
print( f"{naam=} en {x=}") # naam='Bo' en x=7 -> goed om te debuggen
# type van variabele is nu ook af te leiden
# 7 voor int, 7.0 voor float, '7' voor str
```

LIJSTEN

declaratie en initialisatie

```
steden = ["Gent", "Genk", "Hasselt", "Ieper"]
lege_lijst = [] # heeft lengte 0
frequenties = [0] * 6 # frequenties bevat nu [0,0,0,0,0,0]
```

lijst is opeenvolging elementen, te overlopen met for

```
for stad in steden: # voor elke element in de lijst
    if stad[0] == 'G': # controleer of stad met 'G' begint
        print(stad) # zo ja, print uit
```

slicing: deel van de lijst kopiëren

```
lijst[start] # element op index 'start'
lijst[start:stop] # deellijst van startindex tot net voor stopindex
lijst[start:stop:stap] # analoog, enkel elementen die stapgrootte van elkaar liggen
# negatieve step: van achter naar voor
tekst[:] # kopie van hele lijst
tekst[::-1] # kopie van hele lijst achterstevoren
# start ontbreekt -> vanaf index 0
# stop ontbreekt -> tot het einde
# stap ontbreekt -> stapgrootte = 1
```

functies die lijst-argument nemen

```
lengte = len([4,5]) # berekent lengte van de lijst [4,5] (is 2)
eerste = min(steden) # berekent minimum van de elementen (is "Genk")
grootste = max([7,9,5]) # berekent maximum van de elementen (is 9)
som = sum([1, 2, 3]) # sommeert de elementen (is 6)
```

hergebruik NOOIT de naam van dergelijke functie als naam van de variabele!

operatoren op lijsten

```
steden + ["Aalst", "Brussel"] # concatenatie, resultaat bevat nu 6 steden
[1, 2] * 3 # dupliceren, resultaat is [1, 2, 1, 2, 1, 2]
3 * [1, 2] # dupliceren, resultaat is [1, 2, 1, 2, 1, 2]
[2, 6] == [1+1, 3+3] # vergelijkt elt per elt, resultaat is hier True
"Ieper" in steden # gaat na of "Ieper" elt is van lijst steden (True)
```

methodes van de klasse list

```
lijst = ["a", "c"]
lijst.append("d") # voegt achteraan een nieuw elt toe, lijst is ["a","c","d"]
lijst.insert(1, "b") # schuift op index 1 elt "b" in, lijst is ["a","b","c","d"]
lijst.remove("c") # haalt element "c" weg, lijst is ["a","b","d"]
```

```
i = lijst.index("d") # geeft index van element "d", is 2 (in ["a","b","d"])
k = lijst.index("x") # als element niet aanwezig is, is resultaat -1
```

```
aantal = [5,6,5,5].count(5) # telt hoe dikwijls elt 5 voorkomt, resultaat is 3
```

```
lijst = [3,7,9,1]
lijst.sort() # lijst wordt gesorteerd van klein naar groot (bevat nu [1,3,7,9])
lijst.reverse() # lijst wordt omgekeerd (bevat [9,7,3,1] na sorteren én omkeren)
```

FUNCTIES

declaratie en implementatie

```
def korting(x): # hoofding bestaat uit 'def', functienaam, parameterlijst
    return x / 10.0 # op het einde van de functie staat wat er teruggegeven wordt
```

```
def is_langer_dan(lijst, lengte): # parameterlijst kan 0, 1 of meer parameters
    if len(lijst) > lengte: # bevatten
        return True # na een return-opdracht wordt het uitvoeren van
# de functie stopgezet (daarom geen else nodig,
# maar mag wel)
    return False
```

```
def is_langer_dan(lijst, lengte): # als functie een boolean teruggeeft (True/False)
    return len(lijst) > lengte # mag die waarde 'ter plekke' (na return)
# berekend worden
```

gebruik

```
prijs = 74.99
afslag = korting(prijs) # functie 'korting' wordt opgeroepen met argument prijs,
# resultaat wordt bewaard in variabele afslag
# (functienaam NIET hergebruiken als naam voor variabele!)
print(korting(250.0)) # resultaat van functieoproep wordt onmiddellijk gebruikt
```

```
woorden = ["aap", "noot", "mies"]
if not is_langer_dan(woorden, 4): # resultaat van functieoproep is boolean, dus
    print("lijst is vrij kort") # wordt ook als zodanig gebruikt (in een test)
```

RANDOM (willekeurige getallen)

```
from random import randint, choice # importeer enkel de functies die je nodig hebt
getal = randint(3,10) # willekeurig geheel getal uit [3,10]
kommagetal = random() # willekeurig kommagetal uit [0.0,1.0[
keuze = choice(lijst) # willekeurig element uit de lijst
shuffle(lijst) # schudt de elementen v/d lijst willekeurig door elkaar
```

TEKSTBESTANDEN

inlezen

```
with open("mijnbestand.txt", "r") as bestand: # verwerk alle lijnen in het bestand
    for lijn in bestand: # één voor één
        # doe iets met lijn
    ...
with open("mijnbestand.txt", "r") as bestand:
    lijn1 = bestand.readline() # lees afzonderlijke lijn
    lijn2 = bestand.readline() # lees volgende lijn
    # hierna kan je een for-lus gebruiken om de rest v/h bestand in te lezen
# Opgelet: in deze voorbeelden bevat de ingelezen strings op het einde een new line

with open("mijnbestand.txt", "r") as bestand: # sla alle lijnen op in een lijst
    lijst = bestand.read().splitlines()
# Nu is er geen new line op het einde. Gebruik dit enkel als het effectief
# nodig is om alle lijnen in het geheugen op te slaan
```

uitschrijven

```
with open("mijnbestand.txt", "w") as bestand:
    ...
    print (... , file=bestand)
    ...
```

TEKENEN (met PIL)

```
from PIL import Image, ImageDraw
prent = Image.new("RGB", [breedte, hoogte], color="white")
doek = ImageDraw.Draw(prent) # tekenen gebeurt op het doek
...
doek.rectangle([x1,y1,x2,y2], fill="red") # rechthoek
doek.rounded_rectangle([x1,y1,x2,y2], radius=5, ...) # afgeronde rechthoek
doek.ellipse([x1,y1,x2,y2],...) # ellips (in rechthoek)
doek.line([x1,y1],[x2,y2], width=2) # rechte lijn
...
prent.show() # de prent kan getoond worden,
prent.save("mijnprent.png") # en/of opgeslagen
```

GRAFIEKEN (met matplotlib)

```
from matplotlib import pyplot # parameters: lijst van X-coördinaten
# en lijst van Y-coördinaten
pyplot.bar(x_waarden,y_waarden) # staafdiagram
pyplot.plot(x_waarden,y_waarden) # grafiek
pyplot.show() # toon eindresultaat
```

TIJD METEN

```
from time import time
start_tijdstip = time() # tijdstempel op dat moment (kommagetal in seconden)
... # doe ingewikkelde bewerkingen
eind_tijdstip = time() # tijdstempel op dat moment
aantal_seconden = eind_tijdstip - start_tijdstip # verschil in seconden
```

DICTIONARY

```
dict = { "sleutel1": "waarde1", # maak een dictionary aan met alvast
         "sleutel2": "waarde2", # enkele sleutel/waardeparen
         "sleutel3": "waarde3" }
dict = {} # maak een lege dictionary aangemaakt

waarde = dict[sleutel] # haal waarde op a.d.h.v. de sleutel
dict[nieuwe_sleutel] = nieuwe_waarde # voeg nieuw sleutel/waardepaar toegevoegd
dict[oude_sleutel] = nieuwe_waarde # wijzig waarde bij bestaande sleutel1

sleutel in dict # bevat dictionary de gegeven sleutel?
for sleutel in dict: # overloop alle sleutels
    # doe iets met sleutel
    ...
for sleutel,waarde in dict.items(): # overloop alle sleutel/waarde-paren
    # doe iets met sleutel en waarde
    ...
del dict[sleutel] # verwijder sleutel en bijhorende waarde
```

LIJSTEN (extra)

```
del lijst[index] # verwijdert element op plaats index in lijst
lijst[index] = [] # idem
element = lijst.pop(index) # geeft lijst[index] terug en verwijdert dit ook
element = lijst.pop() # geeft lijst[-1] terug en verwijdert dit ook

lijst += [2,4,6] # voegt meerdere elementen toe achteraan een lijst
lijst += [3] # alternatief voor append, vierkante haakjes verplicht!
lijst = list(range(0,50,2)) # zet een range om naar een lijst
lijst[2:4] = [3,8,9,10] # vervangt slice van een lijst door andere waarden

element not in list # alternatief voor 'not element in list'

twintig_duizend = 20_000 # underscores in grote getallen zijn toegelaten

vijfentwintig_duizend = 2.5E4 # wetenschappelijke notatie
nauwkeurigheid = 1.0E-13
```

```
print (... , end='') # drukt af zonder een nieuwe lijn te nemen
```