```
3 * 4, 3 + 4, 3 - 4, 3 / 4                #==> 12, 7, -1, 0.75
3 ** 4, 3 // 4, 3 % 4                     #==> 81, 0, 3
4 > 3, 4 >= 3, 3 == 3.0, 3 != 4, 3 <= 4   #==> True, True, True, True, True
# volgorde van operaties: haakjes, **, {* / // %}, {+ -}, {== != <= < > >=}
min(3, 4), max(3, 4), abs(-10)            #==> 3, 4, 10
sum([1, 2, 3])  # [1, 2, 3] is een lijst  #==> 6

type(3), type(3.0), type("mijnVariabele") #==> class 'int', class 'float',
                                          #    class 'str'
int("4"+"0"), float(3), str(1 / 2)        #==> 40, 3.0, '0.5'

"dubbele quotes: ', escaped \" \\ \'"     #==> dubbele quotes: ', escaped " \ '
'\'t is "vergelijkbaar" in enkele quotes' #==> 't is "vergelijkbaar" in enkele quotes

ord("A"), chr(66)                         #==> 65, 'B'
string = "hallo"
# de volgende statements werken ook voor lijsten
len(string)                               #==> 5
string[0], string[4]    # kies tekens     #==> "h", "o"
string[1:3]             # kies deelwoord  #==> "el"
string[:2], string[2:]  # l/r deelwoorden #==> "he", "llo"
string[-1], string[-2:] # negatieve indices#==> "o", "lo"
"con" + "cat" + "enatie " + str(123)      #==> "concatenatie 123"
"boo" * 2                                 #==> "booboo"

getLineOfInputAsString = input()          #==> leest invoer (of EOF-error)
print("krijgt", 0, "of meer argumenten")  #==> krijgt 0 of meer argumenten
print("gebruik", "eigen", "sep", sep=".") #==> gebruik.eigen.sep
print("geen", "newline", end="doei")      #==> no newlinedoei

not True, False or True, False and True   #==> False, True, False
# volgorde van operaties: haakjes, {== !=}, not, and, or

if booleanConditie:
   x                      # spring het body-blok in
   x                      # elke regel even ver inspringen
elif anotherConditie:     # nul, een of meerdere elif-blokken
   x                      # blok van meerdere regels
else:                     # optioneel
   x                      # blok van meerdere regels

while booleanConditie:
   x                      # het body-blok
   break                  # spring uit de loop (optioneel)
   continue               # begin aan de volgende iteratie (optioneel)

for indexVariabele in range(begin, eindePlus):
   print(indexVariabele)                  #==> begin, begin+1, ..., eindePlus-1
# "for item in lijstOfString:" forall/foreach-loops
# break en continue kunnen ook in for-loops worden gebruikt
```

```
def naamVanNieuweFunctie(argument1, argument2):
    x                       # het body-blok
    return y                # (optioneel; als je niks oplevert, lever je None op)

def onthoudt(bar):          # schrijven naar globale variabelen
    global bewaarBar        # na de aanroep onthoudt(3) geldt bewaarBar = 3
    bewaarBar = bar         # zelfs buiten de scope van de functie

# deze 'slice'-commando's werken analoog voor lijsten en range()
"0123456789"[::2]           # slices        #==> "02468"
"0123456789"[::-1]          # aflopend      #==> "9876543210"
"0123456789"[6:3:-1]                        #==> "654"

x += 1          # ook -=, /=, *=, %=, **=, //=. Python heeft geen C++-style "x++"
x, y = y, x     # meerdere toewijzingen
3 < x < 5       # zelfde als "(3 < x) and (x < 5)". keten van {< <= > >= == != is}

import math                 # import, om functies met een punt te gebruiken
print(math.sqrt(2))
from math import sqrt       # import, voor een specifieke functie zonder punt
print(sqrt(2))
# ook in de math-module: pi, exp, log, sin, cos, tan, ceil, floor en meer

lijst = ['nul', 'een', 'twee']
lijst.index('een')                          #==> 1
lijst.index('drie')                         #==> levert een fout op
'drie' in lijst, 'nul' in lijst             #==> False, True
lijst.count('twee')                         #==> 1
del lijst[1]     # lijst wordt ['nul', 'twee']
"string" in "superstring"                   #==> True
"superstring".index("string")               #==> 5

# meer lijst-methoden: append(item), insert(item, index), extend(list),
# remove(value), pop(), pop(index), reverse(), sort() en meer

# een aantal string-methoden: capitalize(), lower/upper(), islower/isupper(),
# isalpha/isdigit(), center/ljust/rjust(width, fillChar), strip(), split(),
# splitlines(), endswith/startswith(string), find(string), replace(old, new),
# en meer

mijnLijst = [11, 99]
eigenlijkDezelfdeLijst = mijnLijst # geen echte kopie! kopie van de verwijzing
mijnLijst is eigenlijkDezelfdeLijst         #==> True
echteKopie = mijnLijst[:]         # of list(mijnLijst), copy.copy(mijnLijst), deepcopy
echteKopie is mijnLijst                     #==> False
```