



IT Essentials

Hoofdstuk 5

Functies

DE HOGESCHOOL MET HET NETWERK

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook



Inhoud

1. Elementen van een functie
2. Het nut van een functie
3. Het creëren van een functie
 1. Hoe Python omgaat met functies
 2. Parameters en argumenten
 3. Return
 4. Het verschil tussen return en print
 5. functies oproepen vanuit functies
4. Scope
 1. Scope van variabelen
 2. Scope van parameters, lokale variabelen



5.1 Elementen van een functie

Voorbeeld:

```
x = int(15.7)
print()
print("Het geheel getal is ", x)
getal = float(input("Geef een getal in"))
print("Het dubbel is " + str(2 * getal))
```

Elementen:

- **Naam** waarmee functie wordt opgeroepen
- 0, 1 of meerdere **argumenten** die meegegeven worden bij oproep van de functie
- Functie **kan** een **waarde** teruggeven (= returnwaarde)



5.2 Het nut van een functie

Voorbeeld:

```
a = 5
b = 7
if a > b:
    c = a * 2
else:
    c = b * 3
print(c + 4)
x = 20
y = 15
if x > y:
    z = x * 2
else:
    z = y * 3
print(z)
```

Functie maken !!!

5.3. Het creëren van een functie

syntax `def <functie_naam>(<parameter_lijst>):`
 `<acties>`

```
def bereken(getal1, getal2):  
    if getal1 > getal2:  
        resultaat = getal1 * 2  
    else:  
        resultaat = getal2 * 3  
    return resultaat
```

```
a = 5  
b = 7  
c = bereken(a, b)  
print(c + 4)  
x = 20  
y = 15  
z = bereken(x, y)  
print(z)
```



Let op de indentatie

5.3.1 Hoe Python omgaat met functies

```
def bereken(getal1, getal2):  
    if getal1 > getal2:  
        resultaat = getal1 * 2  
    else:  
        resultaat = getal2 * 3  
    return resultaat
```

De functie moet voor de oproep van de functie gedefinieerd zijn

```
a = 5  
b = 7  
c = bereken(a, b)  
print(c + 4)  
x = 20  
y = 15  
z = bereken(x, y)  
print(z)
```

Het hoofdprogramma

5.3.2 Parameters en argumenten

```
def bereken(getal1, getal2):  
    if getal1 > getal2:  
        resultaat = getal1 * 2  
    else:  
        resultaat = getal2 * 3  
    return resultaat
```

```
a = 5  
b = 7  
c = bereken(a, b)  
print(c + 4)  
x = 20  
y = 15  
z = bereken(x, y)  
print(z)
```

parameters

Mag ook een getal zijn

argumenten



Een functie moet niet noodzakelijk een parameter hebben

5.3.3 return

```
def bereken(getal1, getal2):  
    if getal1 > getal2:  
        resultaat = getal1 * 2  
    else:  
        resultaat = getal2 * 3  
    return resultaat
```

Waarde van de variabele resultaat wordt teruggegeven aan het hoofdprogramma

```
a = 5  
b = 7  
c = bereken(a, b)  
print(c + 4)  
x = 20  
y = 15  
z = bereken(x, y)  
print(z)
```


Opmerkingen

1. Na return kan ook een berekening staan

```
def bereken(a, b):  
    return a ** 3 + b ** 2  
  
x = bereken(5, 8)  
print(x)
```



2. Een functie hoeft geen return waarde te hebben

```
def print_bereken(a, b):  
    print(a ** 3 + b ** 2)  
  
print_bereken(5, 8)  
print_bereken(8, 5)
```



Opdracht 5.1:

Schrijf een functie waarmee je een regel van tekens afdrukt. Het aantal tekens en het teken zelf zijn variabel.

Voorbeeld output: teken *, aantal 8.

* * * * *



5.3.4 Het verschil tussen return en print

Voorbeeld:

Wat gebeurt er als we onderstaande programma's laten uitvoeren? Verklaar.

```
def bereken(a, b):  
    return a ** 3 + b ** 2
```

```
x = 4 * bereken(1, 3)  
print(x)
```

```
def print_bereken(a, b):  
    print(a ** 3 + b ** 2)
```

```
x = 4 * print_bereken(1, 3)  
print(x)
```



5.3.5 functies oproepen vanuit functies

In de definitie van een functie, kunnen andere functies opgeroepen worden.

Opdracht 5.2:

Schrijf een functie waarmee je een rechthoek afdruckt. De hoogte, de breedte en het gebruikte teken zijn variabelen. Maak hierbij gebruik van de functie die je in opdracht 5.1 gemaakt hebt.

Voorbeeld van output: hoogte 4, breedte 7 en teken *

```
* * * * *
* * * * *
* * * * *
* * * * *
```



Opdracht 5.3: (opgave 5.1 uit cursus)

Schrijf een functie `printx()` die alleen de letter “x” print. Schrijf daarna een functie `meerderex()` die als argument een integer krijgt en die zo vaak de letter “x” print als de integer aangeeft. Daartoe roept de functie `meerderex()` zo vaak als nodig de functie `printx()` aan.



5.3.6 Functienamen

- Alleen kleine letters, cijfers en underscore
- Mag nooit beginnen met een cijfer
- Zinvolle namen
- Indien meerdere woorden: tussen elk woord een underscore bvb. `bereken_product()`



Opdrachten

- Opdracht 5.4: (opgave 5.2 uit cursus)
Schrijf de functie `is_even()` die nagaat of een getal even is en `True` of `False` teruggeeft.
- Opdracht 5.5: (opgave 5.3 uit cursus)
Schrijf de functie `is_oneven()`, die bepaalt of een getal oneven is, door de functie `is_even()` aan te roepen en het resultaat te inverteren.
- Opdracht 5.6: (opgave 5.4 uit cursus)
Schrijf de functie `get_tienden()`, die de tienden van een float retourneert. Bijv. voor 45,235 krijg je het getal 2 als resultaat



5.4. Scope

5.4.1 Scope van variabelen

= bereik van variabelen

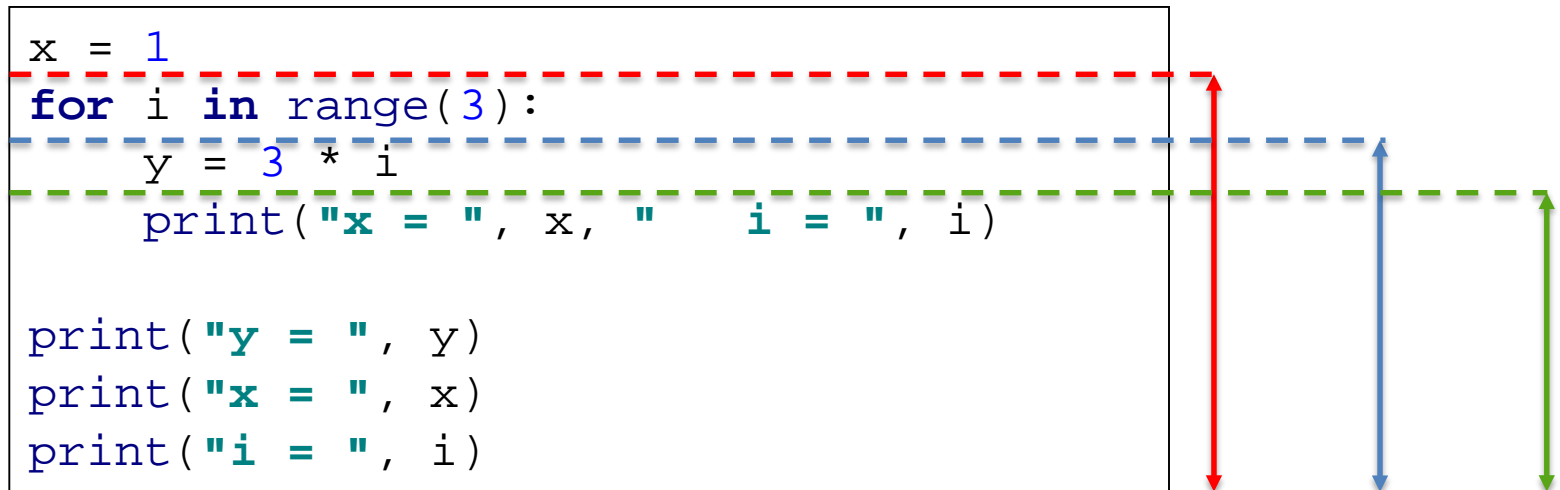
Algemene regel: na creatie is de variabele beschikbaar in het hele programma



Dit is in de meeste programmeertalen anders!

Voorbeeld1:

scope x scope i scope y



Output?

```
x = 1    i = 0
x = 1    i = 1
x = 1    i = 2
y = 6
x = 1
i = 2
```



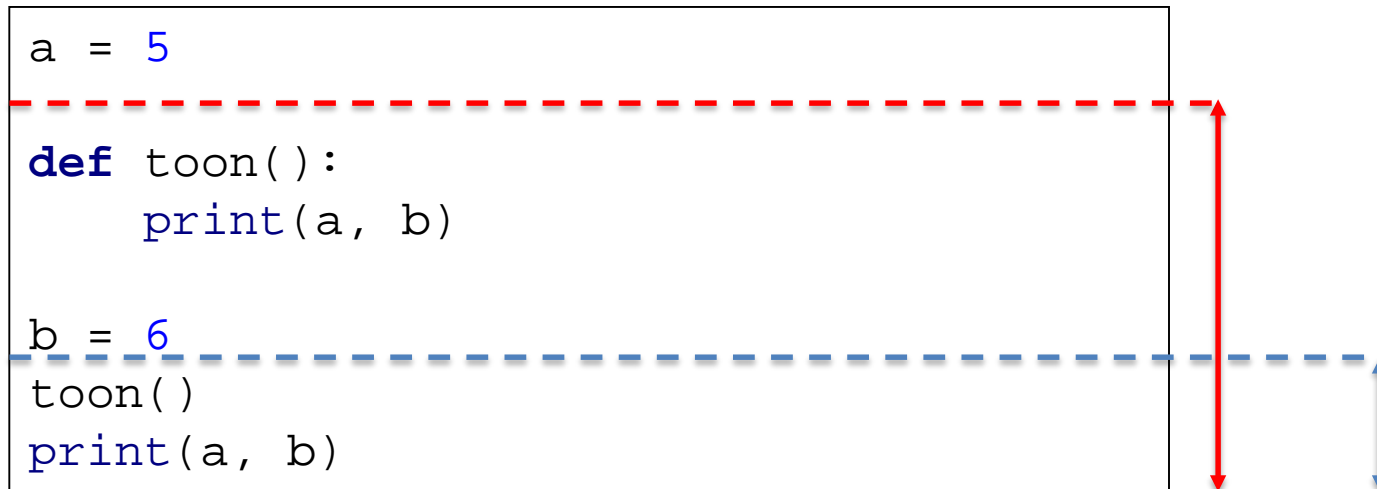
Voorbeeld2:

scope a scope b

```
a = 5

def toon():
    print(a, b)

b = 6
toon()
print(a, b)
```



Output?

5 6
5 6



5.4.2 Scope van parameters, lokale variabelen

```
def bereken(x, y):
```

```
    resultaat = x * 2 + y * 3
```

```
    return resultaat
```

```
a = 5
```

```
b = 7
```

```
c = bereken(b, a)
```

```
print(a, b, c)
```

parameters

lokale variabele

scope parameters

= van het oproepen van de functie tot het einde van de functie

scope lokale variabele

= tijdens het oproepen van de functie op moment van aanmaak tot het einde van de functie



Voorbeeld 1:

Wat is de output van onderstaand programma? Leg uit.

```
def bereken(x, y):  
    resultaat = x * 2 + y * 3  
    return resultaat  
  
x = 5  
y = 7  
resultaat = bereken(y, x)  
print(x, y, resultaat)
```

Output: 5 7 29



Voorbeeld 2:

Wat is de output van onderstaand programma? Leg uit.

```
a = 5

def toon():
    a = 8
    b = 9
    print(a, b)

b = 6
toon()
print(a, b)
```

Output: 8 9
5 6



Voorbeeld 3:

Wat is de output van onderstaand programma? Leg uit.

```
a = 5
```

```
def toon():  
    a = a + 2  
    print(a, b)
```

```
b = 6  
toon()  
print(a, b)
```

scope a scope b

Output

Je krijgt een foutmelding



a is zichtbaar in de functie maar zodra we aan a een waarde geven via een assignment wordt er een nieuwe lokale variabele a aangemaakt!



Opdrachten

Opdracht 5.7:

- Maak een functie `toon_tafel()` om de tafel van vermenigvuldiging te tonen. Vraag in het hoofdprogramma aan de gebruiker welke tafel hij wil zien.
- Bepaal de scope van elke variabele die je in je programma gebruikt en vermeld of dit een variabele, een parameter of een lokale variabele is.

Voorbeeld van output (tafel van 15):

```
0 x 15 = 0
1 x 15 = 15
2 x 15 = 30
3 x 15 = 45
4 x 15 = 60
5 x 15 = 75
6 x 15 = 90
7 x 15 = 105
8 x 15 = 120
9 x 15 = 135
10 x 15 = 150
```



Opdracht 5.8:

Schrijf een programma om het volgende te realiseren:

- De gebruiker geeft een belastbaar bedrag in waarna de verschuldigde belasting op het beeldscherm verschijnt.
De verschuldigde belasting wordt met een functie berekend. Voor de eerste €25000 moet 38,4% belastingen betaald worden, voor de volgende €30000 moet 50% belastingen betaald worden en voor elke € meer 60% belastingen.
- Bepaal de scope van elke variabele die je in je programma gebruikt en vermeld of dit een variabele, een parameter of een lokale variabele is.



5.5 Ingebouwde functies

- basisfuncties : print, int, round, ...
- andere functies zitten in modules
- Hoe gebruiken?

methode 1 : => juiste module importeren

```
import <modulenaam>
```

vb.

```
import math

x = math.pow(2, 5)
y = math.sqrt(4)

print(x, y)
```

Output : 32.0 2.0



methode 2 :

=> enkele functies importeren uit 1 module

```
from <module> import <functie1>, <functie2>, <functie3>, ...
```

vb.

```
from math import pow, sqrt
```

```
x = pow(2, 5)
```

```
y = sqrt(4)
```

```
print(x, y)
```



5.5.1 de module random

- Bevat functies om getallen te genereren
- Nuttige functies in de module random
 - `random()`
 - => genereert een willekeurig reëel getal ≥ 0 en < 1
 - `randint(ondergrens, bovengrens)`
 - => genereert een willekeurig geheel getal \geq ondergrens en \leq bovengrens



Opdracht 5.9:

Schrijf een programma dat

1. een willekeurig geheel getal genereert ≥ 0 en ≤ 10
2. 10 willekeurige gehele getallen genereert > 0 en < 10
3. een willekeurig geheel getal genereert ≥ -200 en ≤ 1000 dat een veelvoud is van 10
4. een willekeurig reëel getal genereert ≥ 0 en < 100 met juist 1 cijfer na de komma