Python

D. Leeuw

23 april 2025
v.1.1.0

Over dit Document

Inhoudsopgave

Hoofdstuk 1

Functies

Een functie is een herbruikbaar stuk code dat een specifieke taak of actie uitvoert. Het gebruik van functies bevordert een modulaire aanpak voor het ontwerpen van software, waardoor de code duidelijker, leesbaarder en herbruikbaarder wordt.

1.1 De functie definitie

Een functie wordt in Python gedefinieerd met het sleutelwoord "def", gevolgd door de naam van de functie, een paar haakjes, met daar tussen 0 of meer paramters en de functie-definitie wordt afgesloten met een dubbele punt. De code die de functie uitvoert, komt op de volgende regels en is geïndenteerd.

```
def naam_van_de_functie(parameters)
    # code van de functie
```

De naam van de functie mag willekeurig gekozen worden, zolang als deze maar niet overeenkomt met een bestaande functie.

Een functie staat los van het hoofdprogramma. Als je de functie aanroept in je hoofdprogramma, dan springt de computer naar de functie om als hij klaar is met de functie terug te keren naar waar die vandaan kwam:

```
# Functie, wordt niet uitgevoerd als het script start
def naam_van_de_functie(parameters):
    # code van de functie

# Hoofdprogramma, hier start het script
argument = "tekst"

# Programma springt naar functie
naam_van_de_functie(argument)
```

```
# Nadat de functie be-eindigd is gaat het programma hier verder
exit()
```

1.1.1 Functie parameters

Het is vaak nodig dat je gegevens (data) meestuurt vanuit het programma naar een functie. De meegegeven data kan dan in variabelen binnen de functie gebruikt worden. Deze variabelen hebben een speciale naam: we noemen ze parameters. Elke paramater kan een argument bevatten. De argumenten zijn de daadwerkelijke data die we meegeven hebben. Laten we dat eens bekijken in een voorbeeld:

```
def groet(naam):
    print(f"Hallo, {naam}")
```

De functie heeft als naam "groet" gekregen. Tussen haakjes staat de parameter (variabele) met de naam "naam" en de variable wordt in de functie gebruikt door het `print` statement.

Wanneer je de functie "groet" aanroept met een naam als argument, zal het die naam gebruiken om je te begroeten. De complete code zou er zo uit kunnen zien:

```
def groet(naam):
    print(f"Hallo, {naam}")

groet("Dennis")
```

Dus "Dennis" is hier het argument.

Een functie mag geen of meerdere parameters hebben. Als er geen parameters zijn dan staat er niets tussen de haakjes. De haakjes moeten er wel zijn! Wil je meer dan één parameter meegeven dan moeten de verschillende parameters gescheiden worden door een komma:

```
def groet(voornaam, achternaam):
    print(f"Hallo, {voornaam} {achternaam}")

groet("Dennis", "Leeuw")
```

Een vereiste is dat als er 2 parameters door de functie gevraagd worden er ook twee argumenten geleverd moeten worden. Niet meer en ook niet minder. Komt het aantal argumenten niet overeen met het aantal parameters dan geeft Python een error melding.

1.1.2 Functie return-waarden

Een functie kan waarden retourneren naar het hoofdprogramma door gebruik te maken van het "return"-sleutelwoord:

```
def optellen(a, b)
    return a + b
```

Er kan slechts een enkele waar terug gegeven worden. Return waarden mogen van elk data-type zijn. Dus als je meer waarden terug wilt sturen kan dat via een list, tuple of dictionatry zijn.

1.1.3 Variabelen en functies

De parameters van een functie worden gebruikt als variabelen in een functie, maar een functie kan ook interne variabelen hebben die niet door de aanroep veranderd kunnen worden:

```
def add(a,b):
    e = 1 # e is een interne variabele
    c = a+b
    return c

d = add(5,4)
print(f"d = {d}")
print(f"e = {e}")
# e bestaat hier niet
```

Zo kan ook het hoofdprogramma variabelen hebben. Deze zijn beschikbaar in de functie:

```
def add(a,b):
    print(f"e = {e}") variabele uit het hoofdprogramma
    c = a+b
    return c

e = 2 # Variabele in het hoofdprogramma‘
d = add(5,4)
print(f"d = {d}")
print(f"e = {e}")
```

Een goed advies: doe dit niet! Als je een waarde nodig hebt uit je hoofdprogramma geef deze dan als parameter mee. Vertrouw er niet op dat een variabele die je in je functie nodig hebt in je hoofdprogramma aanwezig is (en de juiste waarde heeft).

Als je in je hoofdprogramma en in je functie variabelen met dezelfde naam gebruikt, dan hebben de variabelen in je functie de waarde die ze in de functie hebben en in je hoofdprogramma behouden ze de waarde die ze daar hebben:

```
def add(a,b):
    print(f"a = {a}")
    c = a+b
    return c

a = 2 # Variabele in het hoofdprogramma
d = add(5,4)
print(f"d = {d}")
print(f"a = {a}")
```

1.2 Samenvatting

Functies kunnen om een aantal verschillende redenen gebruikt worden:

Onderhoud Wijzigingen in de code hoeven maar in één functie gedaan te worden, in plaats van op verschillende plekken in het programma waar dezelfde code gebruikt wordt

Herbruikbaarheid Je kunt dezelfde functie meerdere keren in een programma gebruiken (met bijvoorbeeld verschillende argumenten).

Leesbaarheid Door stukken code in functies te stoppen kan het hoofdprogramma beter leesbaar worden. Kleine specialistische functies zijn makkelijker te lezen en omdat ook het hoofdprogramma korter wordt is ook dat makkelijker leesbaar.

Een functie heeft een naam en kan voorzien worden van data via argumenten die meegeven worden aan parameters.

De functie kan data terug geven via de return-value.

1.3 Vragen

1. Hoe zorg ik ervoor dat een functie weet dat hij vier parameters moet accepteren?
2. Hoe heet de waarde die ik meegeef bij het aanroepen van een functie?
3. Met welke opdracht geef ik een functie de mogelijkheid om data terug te geven aan het hoofdprogramma?
4. Waar moet elke functie mee beginnen?
5. Kan ik een variabele met de naam 'counter' uit mijn hoofdprogramma gebruiken in een functie?

1.4 Opdrachten

1. Herschrijf het volgende programma gebruikmakend van een functie

```
lst_netwerk = []

print("We vragen om 3 IP-adressen die in uw netwerk voorkomen")

ip = input("Wat is IP-adres 1 in het netwerk? ")
lst_netwerk.append(ip)

while True:
    ip = input("Wat is IP-adres 2 in het netwerk? ")
    if ip not in lst_netwerk:
        lst_netwerk.append(ip)
        break
    print(f"IP-adres {ip} komt al in de lijst voor")

while True:
    ip = input("Wat is IP-adres 3 in het netwerk? ")
    if ip not in lst_netwerk:
        lst_netwerk.append(ip)
        break
    print(f"IP-adres {ip} komt al in de lijst voor")

print("Uw netwerk bevat de volgende IP-adressen:")
print(lst_netwerk)
```