Start to Program: Python, Les 11

Functies

Soms heb je een stuk code meerder malen nodig. Nu programmeer je die telkens opnieuw, maar je kan deze ook gaan vastzetten in een functie.

Eigenlijk maak je al heel de tijd gebruik van functies. Deze functies zijn voorgeprogrammeerd in Python zelf.

print(), int(), len(),...

Zijn allemaal functies.

Nu ga je zelf een functie schrijven en die dan op eenzelfde manier aanroepen.

Een functie bevat dus een blok code die een aantal handelingen uitvoerd.

Een functie bepaal je zo:

```
#def om aan te geven dat het om een functie gaat, gevolgd door een zelfgekozen functie naam.
def hello_world():
    print('hello world')
```

En roep je dan aan als volgt:

```
hello_world()
```

Als je nu de python-file uitvoert krijg je de print van de functie te zien.

Je kan natuurlijk uitgebreidere functies opbouwen. Ze kunnen condities, lussen,... bevatten.

LET OP! Je moet de functie nog aanroepen als je de code wil gebruiken. Dit wil dus zeggen dat je code al kan klaarzetten en dan de functie op het gepaste moment aanroepen. Je gaat dit vooral dus doen bij code die je meerdere keren nodig hebt. Of om grote stukken code op te splitsen om het overzicht beter te bewaren.

De code wordt **MODULAIR.**

Parameters

Als je de bestaande functies, print(), int(),... , wil gebruiken moet je nog iets extra aanvullen tussen de haakjes. Dit is een parameter.

Een parameter is informatie die ingevuld wordt in de code van de functie, maar steeds veranderlijk is als je de functie gaat toepast. Een soort invoer dus.

De parameter is steeds een bepaald datatype dat verwacht wordt. De functie len() verwacht een string om de lengte van te bepalen.

De volgende functie verwacht 3 parameters om een optel som in de functie te kunnen maken.

```
def optellen(x,y,z):
    a = x + y
    b = x + z
    c = y + z
    print(a, b, c)

optellen(1,2,3)
```

x,y,z zijn hier de parameters. De functie gaat de opgeven parameters 1,2,3 invullen op alle plekken waar x,y,z voorkomen in de functie.

De volgorde waarin de waarden(argumenten) van de parameter wordt gegeven is hierbij belangrijk.

Zo wordt de code die wordt uitgevoerd:

```
a=1+2
b=1+3
c=2+3
```

Parameters kunnen ook als sleutelwoord gebruikt worden. Dan kan je de volgorde van de opgegeven argumenten aanpassen.

Let wel op. De functie in de code staat in een bepaalde volgorde, dus dit verandert niets aan de uitkomst.

```
optellen(y=2, z=3, x=1)
```

Op eenzelfde manier kan je bij het aanmaken van de functie standaardwaarden(defaults) instellen.

Als de parameter dan niet wordt ingevuld bij het aanroepen van de functie gaat de functie hierop terugvallen om toch een waarde in te vullen.

```
def optellen(x=1,y=2,z=3):
    a = x + y
    b = x + z
    c = y + z
    print(a, b, c)

optellen()
```

Dit geeft weer dezelfde uitkomst omdat de standaardwaarden 1,2,3 zijn ingevuld.

Je kan de functie nog gebruiken zoals daarvoor door zelf waarden in te vullen.

Scoping

Scoping heeft te maken met de plaatsing van variabelen. Deze kunnen globaal en lokaal gedeclareerd(aangemaakt) worden. Tot nu toe heb je alle variabelen globaal aangemaakt. Je kon deze dus overal in je code gebruiken.

Als een variabele echter binnenin een functie wordt aangemaakt dan is deze lokaal gedeclareerd. Dit wil zeggen dat je de variabele alleen binnenin de functie kan gebruiken.

Wil je de variabele dus op meerdere plaatsen gebruiken dan moet je hem eerst bovenaan je code en buiten een functie aanmaken.

Probeer dit even uit met de optel functie.

Plaats een print buiten de functie die de variabele a waarin x+y zit oproept.

Wat krijg je in de console?

Return

Tot nu toe heb je steeds print gebruikt om een waarde te tonen/gebruiken. Je wil niet elke functie meteen printen. Meestal zal deze een berekening, voorwaarde of lus bevatten die een uitkomst geeft.

Deze uitkomt moet dan worden teruggegeven aan ons. Daarom de **return**

```
def optellen(x=1,y=2,z=3):
    a = x + y
    b = x + z
    c = y + z
    return a, b, c
optellen()
```

Dit geeft echter niets visueel mee. Als je nog wel de uitkomst wil printen moet je deze of rechtstreeks in een print uitvoeren OF opslaan in een variabele.

```
print(optellen)
OF
som = optellen()
print(som)
```

LET OP! Return onderbreekt de functie! Het programma zal dan verder lopen na de functie. Dit wil zeggen dat er slecht 1 return uitkomst per functie is. Dit wil niet zeggen dat er maar 1 return in kan staan. Als er condities zijn kan de ene of de andere return worden aangeroepen.

Opdrachten:

1.Schrijf een functie die de som van alle getallen in een tuple geeft.

Bijvoorbeeld de tuple (8, 2, 3, 0, 7), je uitkomst zou 20 moeten zijn.

2.Schrijf een functie die de hoogste van 2 opgegeven getallen teruggeeft.

Schrijf een extra functie die grootste van 3 getallen weergeeft.

TIP: gebruik de vorige functie in je nieuwe.

3.Schrijf een functie die temperatuur in Fahrenheit of Celsius omzet.

4.Maak eens een kopie van je galgje oefening. Herken je hierin delen die kunnen opgesplitst worden in functies? Probeer heel voorzichtig een opdeling te maken.