



Bert Smits voor

Basisschool “De Esdoorn”
Bankstraat 29
2811 Hombeek (Mechelen)

Programmeren in Python – Deel 3

1. Tot nu toe hebben we eigenlijk niet veel “nuttigs” gedaan met de computer. We hebben iets afgedrukt (dat we net tevoren zelf ingetypt hadden) en we hebben (met veel moeite) wat gekke figuurtjes op het scherm kunnen tekenen.

Een van de eerste echt nuttige toepassingen van een computer is het gebruik van functies. Een functie is een relatie tussen wat je ingeeft en wat eruit komt.

Je definieert (maakt) een functie met de instructie **def**, gevolgd door de naam van de functie en dan, tussen haakjes, de naam van de variabele of variabelen die de functie nodig heeft. Je eindigt met een **dubbele punt :**

Daaronder schrijf je dan je instructie of instructies die bij de functie horen.



Hierboven heb ik een functie gedefinieerd die van een getal het kwadraat neemt en dat dan netjes afdrukt in een mededeling “het kwadraat van … is ….”.

Nadien kan ik de functie zo vaak ik maar wil aanroepen.

1. We kunnen natuurlijk ook functies gebruiken in modules, zoals onze tekenmodule “turtle”.

Je hoeft de module nog niet te laden om een functie te schrijven die stukjes programma van turtle zelf gebruikt.

Stel dat we een functie willen maken die een vierkant tekent, met een zijde die wij opgeven. Dat gaat dan zo:



Pas nu hoeven we de module turtle te laden, als we de functie werkelijk willen gebruiken (bijvoorbeeld om een vierkant van zijde 100 te tekenen):



Je kan deze functie natuurlijk zoveel aanroepen als je maar wil :



1. Er is ook een manier om het herhalen van programmeercode te vermijden zoals in het programma hierboven, te vermijden. Zoiets noemen we een “lus”.

Een van de manieren om dit te doen is met een “range” functie in een “for”-lus. “Range” is Engels voor “bereik”. Dat gaat dan zo:



De variabele “getal” doorloopt het bereik 0..5 (maar laat het laatste getal vallen) en het programma drukt dan telkens de waarde van “getal” af, met andere woorden 1,2,3 en 4.

In plaats van

mijnpen.forward(zijde)
mijnpen.left(90)
mijnpen.forward(zijde)
mijnpen.left(90)
mijnpen.forward(zijde)
mijnpen.left(90)
mijnpen.forward(zijde)
mijnpen.left(90)

hadden we dus ook kunnen schrijven

for teleensmee in range(1,5):
 mijnpen.forward(zijde)
 mijnpen.left(90)

“teleensmee” is hier een gewone telvariabele die een bereik (Engels : “range”) van 1 tot 5 doorloopt (maar het laatste element niet meer meeneemt). Een lus met een bereik (1,5) gaat dus van 1 tot **4** !



Het resultaat van de nieuwe instructie “tekenvierkant(200)” wordt dan :



en dit is precies hetzelfde dat we tevoren hadden. Je kan zowel in de schelp als in het programma een lus inbouwen. De code hieronder vraagt om drie vierkanten van verschillende grootte te tekenen :



“eenpaarkeer” is de telvariabele, die het bereik doorloopt. Als zijde van het vierkant kiezen we 50 x de waarde van “eenpaarkeer”. Hieronder staat het resultaat :



1. Tijdens het programmeren stellen we vaak “ja/nee” vragen, en laten de computer dan iets doen afhankelijk van het antwoord.

Bijvoorbeeld, als je een rekentoets op de computer wil programmeren, en je wil de leerling punten geven afhankelijk van het antwoord.

Stel bijvoorbeeld de vraag : **Hoeveel is 7 x 6 ?**

ALS de leerling **42** typt, dan krijgt ze een punt bij;

ALS de leerling iets anders typt, dan zegt de computer: **“Probeer het nog eens”** (zoals in Bingel), of dan gaat de computer naar de volgende vraag, zonder een punt bij te geven.

Dit noemen ze een “beslissing” die de computer zal nemen. De beslissing hangt af van het gegeven antwoord.

**Hoeveel is
7 x 6 ?**

De leerling typt een antwoord

**Probeer het nog eens !**

Is het antwoord 42 ?

Neen

Ja

**Goed zo !**

In Pyton programmeren we dat met een “if”-instructie. “If” is Engels voor “Als”. En om te lezen wat de gebruiker op het scherm typt, gebruiken we een speciale module die “sys” heet. Dat is een afkorting voor “systeem”. Het is, net zoals “turtle” diende om te kunnen tekenen, een op zichzelf staand stukje programma dat instructies bevat om te kunnen lezen en begrijpen wat de gebruiker doet en of de computer nog goed werkt.

We laden de module met **import sys**

We definiëren dan de volgende functie “huiswerkoordeel”:



De functie heeft geen invoervariabelen, dus alleen twee haakjes () aan het eind. Daarna wordt een vraag gesteld en leest het programma het antwoord van de gebruiker in, en slaat dit op in de variabele “aantal”.

Dan komen er twee als…dan… zinnetjes. Als je 9/10 of meer haalt, dan feleciteert hij je. Als je minder dan 9 (dus 8/10 of minder) haalt, dan kan het volgens de computer nog wat beter.

Hieronder zie je wat er gebeurt als je de functie aanroept :



1. Tenslotte laten we nog eens zien hoe je lussen (met “range”) kunt combineren met “als…dan…” vragen.

Het onderstaande programmaatje is een functie om een “acht” te tekenen:



En het mooie resultaat zie je hieronder :



**En wat hebben we vandaag geleerd ?**

**Ten eerste** : Ingewikkelde taken, die je door de computer wil laten uitvoeren, omschrijf je best in een *functie*.

Bijvoorbeeld : “teken een vierkant met zijde 50 !”, vertaal je naar Python door een functie te definiëren die precies die handelingen uitvoert, wanneer je in de schelp “tekenvierkant(50)” typt.

**Ten tweede** : je kan de computer laten beslissen wat hij moet doen aan de hand van een “als…dan…” zin het programma. Het programma voert dan de opdracht alleen uit ALS de voorwaarde die je opgegeven hebt, voldaan is. Bijvoorbeeld : “Als het rapportcijfer een 9 of hoger is, dan zeg je “goed zo”. Als het rapportcijfer een 8.5 of minder is, dan zeg je “dat kan wat beter”

**Ten derde** : Computers worden nooit moe. Ze zijn heel goed in dingen veel keer na elkaar te doen. Je kan hiervoor een lus-opdracht geven:

for teller in range(1,100) :

print(“Ik mag niet lopen in de gang”)

is een manier om 99 keer een zinnetje te schrijven. Onthoud dat de laatste waarde van een range telkens afvalt.

Woordenlijst

|  |  |  |
| --- | --- | --- |
| Engels | Nederlands | Commentaar |
| Range | Bereik | Range(1,100) is het bereik van de getallen 1 tot 100. Je gebruikt het in een lus.  |
| Loop | Lus | Met de “for”-instructie doorloopt een indexvariabele een bereik |
| If | Als | Als de voorwaarde vervuld is, dan voert de computer wat volgt uit. |
| Define | Definiëren | Omschrijven, uitleggen wat iets is. |

1. Programmeerpuzzels

Schrijf de volgende programmaatjes in Python:

1. Een programma dat de gebruiker vraagt hoe hij of zij heet en dan, als de naam bijvoorbeeld “Marie” was, antwoordt met “Dag Marie, leuk dat je er bent !”
2. Een programma dat de tafel van 6 afdrukt op het scherm, m.a.w.

1 x 6 = 6
2 x 6 = 12
enz…
10 x 6 = 60

1.  Een functie die een regelmatige veelhoek kan tekenen, waarbij je de grootte en het aantal zijden kan opgeven.

Dus, als je functie “tekenveelhoek” heet, dan moet het programma, als de gebruiker in de schelp **tekenveelhoek(50,8)** typt, een regelmatige 8-hoek tekenen, zoals hiernaast.
Hint : de “forward”-instructies moeten de lengte van de zijde tekenen; de left (of right) instructies moeten een hoek maken zodat, als je alle zijden doorlopen hebt, je precies 360° bent rondgedraaid. Voor een 8-hoek zal dat dan 360//8=45 graden zijn, voor een 6-hoek 360//6 = 60 graden.