2

TELEPORTEREN MET VARIABELEN

Ben je zover dat je de kracht van Python kunt gebruiken om jouw Minecraft wereld te beheersen? In dit hoofdstuk krijg je een korte rondleiding langs de basisbegrippen van Python. Daarna test je je nieuwe vaardigheden door je eigen rondreis te maken waarin je door jouw Minecraft wereld teleporteert!

De begrippen in dit hoofdstuk horen niet alleen maar bij Minecraft Python. Je kunt ze dus ook in andere Python programma's gebruiken die je gaat maken.

WAT IS EEN PROGRAMMA?

Een *programma* is een set instructies die ervoor zorgt dat jouw computer een bepaalde taak of bepaalde taken uitvoert. Bijvoorbeeld een stopwatch app op een mobiele telefoon. Het stopwatch programma heeft instructies voor wat er moet gebeuren als jij op start en stop drukt. Het programma heeft ook instructies om de tijd op het beeldscherm te tonen terwijl de tijd loopt. Er is iemand geweest, een jongen of een meisje, die de stopwatch heeft geprogrammeerd om zo te werken.

Elke dag worden over de hele wereld miljoenen programma's gebruikt. De berichtenapp op een mobieltje is een programma, verkeerslichten worden door programma's bestuurd en zelfs computerspellen zoals Minecraft zijn programma's.

In dit boek leer je de basisbeginselen van het programmeren en je leert ook hoe je programma's schrijft die jouw ideeën tot leven brengen in Minecraft.

GEGEVENS OPSLAAN MET VARIABELEN

We beginnen met te leren hoe je gegevens (ook wel data genoemd) opslaat in variabelen. Met *variabelen* kun je gegevens opslaan om die later in het programma weer te gebruiken. Met *data* wordt alle informatie bedoeld die je zou willen bewaren, bijvoorbeeld getallen, namen, allerlei tekst, lijsten met items, enzovoorts. Hier is bijvoorbeeld een variabele `pikhouwelen` die de waarde bewaart van het getal `12`:

```
>>> pikhouwelen = 12
```

Variabelen kunnen getallen, woorden en zelfs hele zinnen bewaren, zoals "Maak dat je wegkomt, engerd!" Je kunt variabelen ook veranderen waardoor je een paar hele mooie dingen kunt doen in Minecraft. Je gebruikt straks al variabelen om te profiteren van de kracht van het teleporteren!

Om in Python een variabele te maken heb je een variabele naam nodig, een is-gelijk teken (`=`) en een waarde. Stel dat je aan het begin staat van een groot avontuur; dan wil je wel genoeg eten meenemen. Je kunt dit voedsel in de vorm van een variabele opvoeren. In onderstaande Python shell is `brood` de naam van de variabele en is `145` de waarde:

```
>>> brood = 145
```

De naam van de variabele staat altijd links van het is-gelijk teken en de waarde die je wilt opslaan staat altijd rechts, zoals je in Figuur 2-1 ziet. Deze Python code regel *declareert* de variabele `brood` en *wijst* hieraan de waarde `145` *toe*.

Figuur 2-1: De delen van een variabele declaratie. Je hebt wel erg veel honger als je 145 broden nodig hebt.

Nadat je een variabele hebt gedeclareerd en hieraan een waarde hebt toegewezen, voer je de naam van de variabele in de Python shell in om te kijken wat deze variabele inhoudt:

```
>>> brood
145
```

Je kunt bijna elke naam gebruiken voor een variabele. Maar het is het beste om een naam te gebruiken die beschrijft wat het doel is van de variabele. Zo begrijp je beter wat er in jouw programma gebeurt. Het is niet echt een regel, maar het is beter om de variabele naam te beginnen met een

kleine letter, geen hoofdletter. Deze stijl wordt door Python programmeurs gevolgd en het is een goede gewoonte om deze stijl aan te houden. Zo kunnen anderen gemakkelijk jouw code lezen als je die ooit wilt delen.

LET OP! *Hoewel de waarde van een variabele wordt opgeslagen, wordt die niet permanent bewaard. De waarde van een variabele wordt opgeslagen in het tijdelijke geheugen van de computer. Dat betekent dat de waarde van de variabele weg is zodra de computer wordt uitgeschakeld, of het programma stopt. Probeer maar eens om IDLE te sluiten en dan weer te openen. Wat gebeurt er als je de waarde van brood wilt zien?*

DE STRUCTUUR VAN PROGRAMMEERTALEN

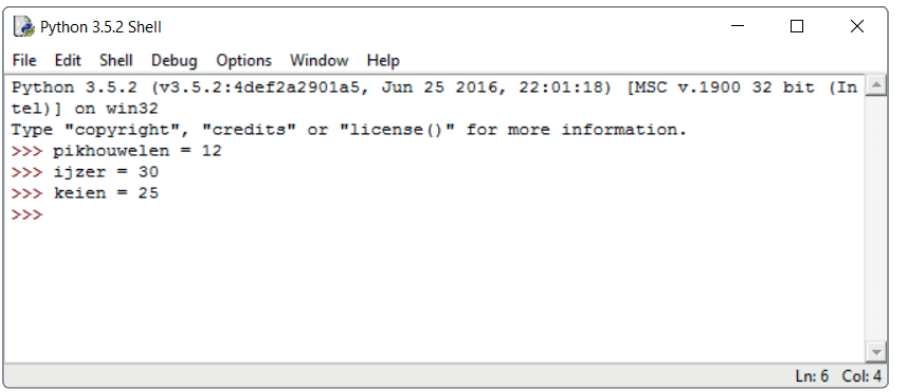
De set regels die beschrijft hoe de grammatica en interpunctie (spaties, punten en komma's) van een programmeertaal eruit moeten zien, heet de *syntax*. Dit lijkt op de grammatica en interpunctie in een gewone menselijke taal. Wanneer je de syntax van Python begrijpt, kun je betere programma's schrijven die een computer goed kan volgen. Maar als je de verkeerde syntax gebruikt, begrijpt de computer niet wat je hem wilt laten doen.

Stel je eens voor dat één enkele instructie in jouw code hetzelfde is als een zin. Een zin eindigt in het Nederlands altijd met een punt. In plaats van een punt, gebruikt Python een nieuwe regel om het einde van een instructie aan te geven. De volgende instructie begint dan op de nieuwe regel. Elke instructie die op een nieuwe regel staat wordt een *statement* genoemd.

Stel dat je wilt bijhouden hoeveel hakbijlen, blokken ijzererts en keien je hebt. In de Python shell schrijf je dit als volgt:

```
>>> pikhouwelen = 12
>>> ijzer = 30
>>> keien = 25
```

In Figuur 2-2 zie je hoe dit eruit ziet in de Python shell.



Figuur 2-2: Code invoeren in de Python shell

Je ziet dat elk statement op een aparte regel staat. Doordat er elke keer een nieuwe regel gemaakt is, begrijpt Python dat je drie verschillende items wilt bijhouden. Als je elk statement niet op een nieuwe regel zet, raakt Python in de war en krijg je een foutmelding die syntax error heet:

```
>>> pikhouwelen = 12 ijzer = 30 keien = 25
SyntaxError: invalid syntax
```

Een *syntax error* is de manier waarop Python je vertelt dat hij het niet begrijpt. Python kan deze instructies niet opvolgen, omdat hij niet weet waar het ene statement eindigt en het andere begint.

Als je een regel begint met een blanco spatie, weet Python ook niet meer wat hij moet doen:

```
>>>   ijzer = 30
SyntaxError: unexpected indent
```

Als je goed kijkt, zie je dat er spaties staan vóór de code, aan het begin van de regel. Als je zo'n unexpected indent syntax error (onverwachte inspringing) krijgt, zoals deze, weet je dat er spaties staan aan het begin van een code regel.

Python is erg kieskeurig als het gaat om de manier waarop je de code schrijft. Als je tijdens het uitwerken van de voorbeelden in dit boek een syntax error krijgt, moet je heel nauwkeurig alles nakijken. Meestal vind je wel ergens een foutje.

SYNTAXREGELS VOOR VARIABELEN

Om Python duidelijk te maken dat het om een variabele gaat, moet je je ook aan een paar syntax regels houden. Dan gaat het vooral om de naam van de variabele:

- Gebruik geen symbolen in je variabele namen, behalve underscores (_); anders krijg je een syntax error.
- Laat de naam van een variabele niet met een cijfer beginnen, zoals in `9brood`. Verderop in de naam mag je wél een cijfer gebruiken, bijvoorbeeld `brood9`.
- Je hoeft aan beide kanten van het is-gelijk teken (=) géén spaties te gebruiken. Jouw programma draait ook goed zonder die spaties. Maar ze maken het wél gemakkelijker om de code te lezen, dus gebruik ze toch maar wel.

Variabelen zijn erg nuttig. Straks leer je hoe je de waarde van variabelen verandert. Daarna ben je klaar voor het teleporteren van je speler!

DE WAARDEN VAN VARIABELEN VERANDEREN

Je kunt op elk moment de waarde van een variabele veranderen, op dezelfde manier als je een variabele declareert. Stel dat je vijf Minecraft

katten tegenkomt en je wilt deze waarde als een variabele opslaan. Eerst declareer je een variabele `katten` en wijs je de waarde `5` toe aan deze variabele. In een Python shell ziet dat er zo uit:

```
>>> katten = 5
>>> katten
5
```

Later kom je nog vijf andere katten tegen en besluit je dat je deze waarde wilt ophogen. Wat gebeurt er als je de waarde van `katten` verandert in 10?

```
>>> katten = 10
>>> katten
10
```

Als je Python nu vraagt naar de nieuwe waarde van `katten`, is dat geen `5` meer! Als je vanaf nu de variabele `katten` in een programma gebruikt, heeft die de nieuwe waarde `10`.

In een variabele kun je verschillende soorten data opslaan. *Data types* vertellen aan de computer hoe ze met een bepaalde soort data moeten omgaan. Ik begin met het bespreken van één van de meest gebruikte data types. Dit zijn integers. Verderop in het hoofdstuk heb ik het ook over het floats data type.

INTEGERS

Integers zijn hele getallen die zowel positief als negatief kunnen zijn. Waarden als 10, 32, –6, 194689 en –5 zijn integers, maar 3.14 en 6.025 zijn dat niet.

Waarschijnlijk gebruik je elke dag integers zonder erbij na te denken, zelfs in Minecraft! Zo kun je bijvoorbeeld zomaar 12 koeien op een heuvel zien staan terwijl je op weg bent om 5 diamanten op te graven, met 2 verse appels in jouw inventory. Al deze getallen zijn integers.

Stel dat je in jouw Minecraft wereld vijf varkens hebt en je wilt een programma schrijven dat op de een of andere manier dit aantal varkens gebruikt. In Python zou je dan een integer variabele declareren om het aantal varkens weer te geven:

```
>>> varkens = 5
```

Je kunt ook negatieve waarden opslaan in een variabele. Als je bijvoorbeeld wilt zeggen dat de temperatuur negatief is, stel min vijf graden, dan declareer je de volgende variabele:

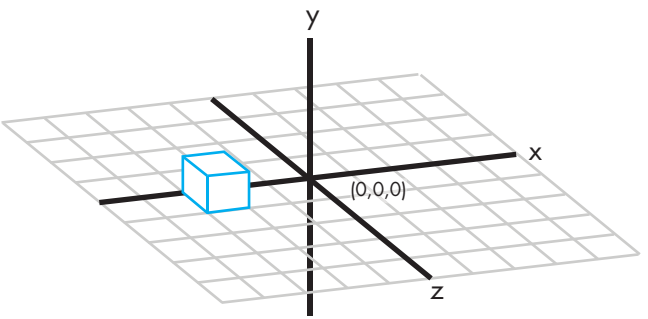
```
>>> temperatuur = -5
```

Om Python variabelen en integers samen met Minecraft te gebruiken, moet je de eerste missie uitvoeren.

MISSIE #1: TELEPORTEER DE SPELER

In deze missie onderzoek je hoe variabelen werken. Dit doe je door met behulp van integers jouw speler naar een nieuwe plek te teleporteren.

Zoals je in Figuur 2-3 ziet, wordt de *positie* van jouw speler in de Minecraft wereld weergegeven door drie *coördinaten*: x, y en z. De letter y geeft de hoogte aan en x en z geven de horizontale positie op een plat vlak aan.



Figuur 2-3: 3D coördinaten

Als je de Raspberry Pi versie van het spel gebruikt, wordt de positie van de speler aangeduid door drie getallen in de linkerbovenhoek van het spelvenster, zoals je in Figuur 2-4 ziet. Als je de desktop versie van het spel gebruikt op je computer, zie je de coördinaten van de speler door op F3 te drukken. Op de eerste regel van het tweede tekstblok aan de linkerkant zie je *XYZ* staan en daarna de coördinaten. Zie Figuur 2-5.

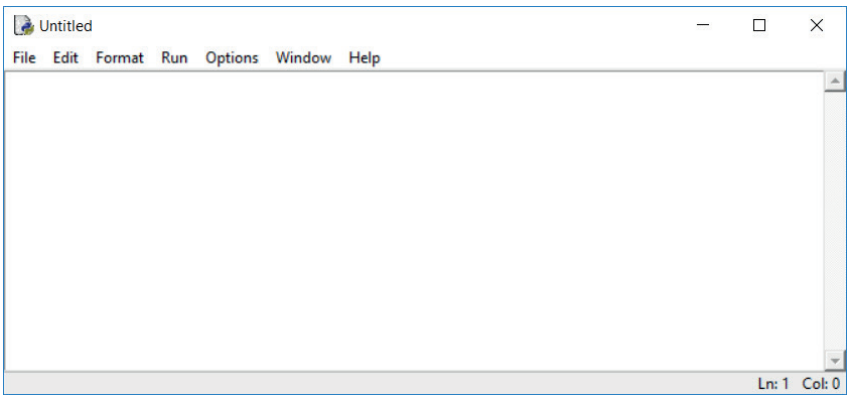
Beweeg je speler door het spel en kijk hoe de positiegetallen veranderen. De coördinaten veranderen direct mee als de speler beweegt. Geweldig, toch? Het duurt alleen zo lang als je een heel eind wilt lopen. Waarom zou je zo lang aan het lopen zijn als je heel snel van positie kunt veranderen door Python te gebruiken? We kijken eens hoe je dit doet.

Figuur 2-4: De positie van de speler zoals je die in Minecraft: Pi Edition ziet

Figuur 2-5: De positie van de speler zoals je die in de desktop versie van Minecraft ziet

Zet eventueel je computer of Raspberry Pi aan en volg deze stappen:

1. Open IDLE en klik op **File ▸ New File** (op sommige computers moet je op **New Window** klikken). In Figuur 2-6 zie je de lege tekstverwerker. Als je een Raspberry Pi gebruikt of meer dan één versie van Python op je computer hebt geïnstalleerd, moet je Python 3 of hoger gebruiken en niet Python 2.7.



Figuur 2-6: Een nieuw tekstverwerker venster in IDLE

2. Zodra je het nieuwe venster ziet, klik je op **File ▸ Save As**.
3. Maak een nieuwe map in de map *Minecraft Python* die je in Hoofdstuk 1 hebt gemaakt en noem deze *variabelen*.
4. Open de map *variabelen*, geef je bestand de naam *teleport.py* en klik op **Opslaan**.

Je werkt nu in de tekstverwerker van IDLE. Voeg de volgende twee regels code toe boven in je programma:

```
from mcpi.minecraft import Minecraft
mc = Minecraft.create()
```

Deze regels verbinden jouw programma met Minecraft. Je gaat ze gebruiken in elk programma dat interactie heeft met Minecraft. Daarna maak je drie integer variabelen die je `x`, `y` en `z` noemt.

```
x = 10
y = 110
z = 12
```

Deze variabelen geven de positie weer waar jij je speler naar toe wilt teleporteren. Voorlopig zet je die variabelen even op 10, 110 en 12, zoals hierboven.

Nu typ je de volgende regel code die de speler gaat verplaatsen:

```
mc.player.setTilePos(x, y, z)
```

Het gedeelte `setTilePos()` is een *functie* van het programma. Dat is een van tevoren geschreven stukje code, dat je later ook kunt hergebruiken in andere programma's. De functie `setTilePos(x, y, z)` vertelt Minecraft dat hij de positie van de speler moet veranderen met gebruik van de drie variabelen die je net hebt gemaakt. De waarden die tussen haken staan worden *argumenten* genoemd. Je hebt de variabelen die je zojuist hebt gemaakt als argumenten *doorgegeven* aan de functie. Op deze manier kan de functie de waarden van `x`, `y` en `z` gebruiken wanneer die wordt uitgevoerd.

LET OP! *Als je een Raspberry Pi gebruikt moet je voor de `x` en `z` variabelen geen waarden gebruiken die groter zijn dan 127 of kleiner dan –127. De Minecraft Pi wereld is klein en getallen die buiten dit bereik liggen laten het spel crashen.*

Lijst 2-1 bevat de volledige code voor het teleporteren van jouw speler. Die code zie je ook in Figuur 2-7:

teleport.py

```
❶ # Maak verbinding met Minecraft
from mcpi.minecraft import Minecraft
mc = Minecraft.create()

# Maak x, y en z variabelen om coordinaten in te stellen
x = 10
y = 110
z = 12

# Verander de positie van de speler
mc.player.setTilePos(x, y, z)
```

Lijst 2-1: De volledige teleporteercode