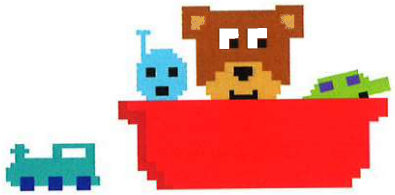
Beslissingen nemen

Programma's beslissen wat ze moeten doen als ze variabelen, getallen en strings met elkaar vergelijken via booleaanse uitdrukkingen. Dat levert het antwoord 'True' of 'False' op.

ZIE OOK
‹ 62–63 Goed of fout?
‹ 108–109 Variabelen in Python

Logische operators

Met logische operators kun je variabelen met getallen of strings, of zelfs met andere variabelen vergelijken. Het resultaat daarvan is 'True' of 'False'.



`==` Operator 'Gelijk aan'

`!=` Operator 'Niet gelijk aan'

`<` Operator 'Minder dan'

`>` Operator 'Groter dan'

`<=` Operator 'Minder dan of gelijk aan'

`>=` Operator 'Groter dan of gelijk aan'

△ **Soorten vergelijkende operators**
Er zijn zes vergelijkende operators. Python gebruikt twee '='-tekens om te vergelijken of twee dingen gelijk zijn (met een enkel '='-teken wordt een waarde toegekend aan een variabele).

▷ **Gebruik het Shell-venster om te checken**
Logische operators werken ook in het Shell-venster. Probeer met dit voorbeeld verschillende logische operators uit, waaronder 'not', 'or' en 'and'.

```
>>> toys = 10
>>> toys == 1
False
>>> toys > 1
True
>>> toys < 1
False
>>> toys != 1
True
>>> toys <= 10
True
>>> not toys == 1
True
>>> toys == 9 or toys == 10
True
>>> toys == 9 and toys == 10
False
```

Dit checkt of 'toys' gelijk is aan 1

Dit checkt of 'toys' meer is dan 1

Dit checkt of 'toys' minder is dan 1

Dit checkt of 'toys' niet gelijk is aan 1

Dit checkt of 'toys' minder of gelijk is dan 10

De logische operator 'not' draait het antwoord om (in dit voorbeeld van 'False' naar 'True')

De logische operator 'or' checkt of 'toys' 9 of 10 is

Met de logische operator 'and' kun je checken of 'toys' zowel 9 als 10 is. Dit kan nooit kloppen, dus het antwoord is 'False'.

Is Ella jarig?

Ella is op 28 juli jarig. Dit programma neemt een dag en een maand en gebruikt logische operators om te checken of Ella jarig is.

1 Check de verjaardag
Maak variabelen aan voor een dag en een maand. Gebruik de logische operator 'and' om te checken of het 28 juli is.

```
>>> day = 28
>>> month = 7
>>> day == 28 and month == 7
True
```

De operator 'and' checkt of beide voorwaarden goed zijn

Denk eraan dat je twee '='-tekens gebruikt

Ella is jarig!

2 Niet jarig-detector
Met de operator 'not' kun je het antwoord omdraaien. Je krijgt voor elke dag het antwoord 'True', behalve op Ella's verjaardag.

```
>>> day = 28
>>> month = 7
>>> not (day == 28 and \
    month == 7)
False
```

Dit karakter wordt gebruikt om de codering op twee regels te zetten

Het is Ella's verjaardag, dus het antwoord is 'False'

3 Verjaardag of nieuwjaarsdag?
Met de logische operator 'or' check je of het Ella's verjaardag is of nieuwjaarsdag. Gebruik haken om de juiste dagen en maanden te combineren.

```
>>> day = 28
>>> month = 7
>>> (day == 28 and month == 7) \
    or (day == 1 and month == 1)
True
```

Checkt 28 juli

Het antwoord zal 'True' zijn als het Ella's verjaardag of nieuwjaarsdag is

Strings

Je kunt twee strings met elkaar vergelijken met de operator '==' of '!='. Strings moeten exact overeenkomen om de output 'True' te krijgen.

```
>>> dog = 'Woof woof'
>>> dog == 'Woof woof'
True
>>> dog == 'woof woof'
False
>>> dog == 'Woof woof '
False
```

De strings komen exact overeen, dus het antwoord is 'True'

De strings komen niet overeen omdat er geen hoofdletter 'W' in voorkomt'

De strings komen niet overeen omdat er extra spatie is voor het aanhalingsteken

△ **Precies hetzelfde**
Strings moeten hier overeenkomen om gelijk te zijn. Dat betekent dat ze hoofdletters, spaties en symbolen op precies dezelfde manier moeten gebruiken.

TIPS VAN EXPERTS

Operator voor strings

Met de operator 'in' kun je zien of de ene string zich in de andere bevindt. Gebruik het om te checken of de string een bepaalde letter of groep letters bevat.

```
>>> 'a' in 'abc'
True
>>> 'd' in 'abc'
False
```

Dit checkt of 'a' in 'abc' zit

'd' zit niet in 'abc', dus het antwoord is 'False'

Vertakken

Met booleaanse uitdrukkingen kun je bepalen welke route een programma moet volgen, afhankelijk of het antwoord op de uitdrukking 'True' of 'False' is. Dit staat bekend als 'vertakken'.

ZIE OOK
‹ 64–65 Beslissingen, vertakkingen
‹ 118–119 Beslissingen nemen

Doe of doe niet

Het commando 'if' betekent dat als een toestand 'True' is, het programma een blok met commando's uitvoert. Als de voorwaarde niet 'True' is, wordt het blok overgeslagen. Het blok na het commando 'if' wordt altijd voorafgegaan door vier spaties.

1 'if'-voorwaarde
Deze code vraagt de gebruiker of hij jarig is. Hij checkt of het antwoord 'y' is. Zo ja, dan wordt een verjaardagsbericht geprint.

```
ans = input('Is it your birthday? (y/n)')
if ans == 'y':
    print('Happy Birthday!')
```

Herinnert gebruikers wat ze moeten invoeren

Dit deel van het programma werkt alleen als de gebruiker 'y' typt

Voorafgegaan door vier spaties

2 Output bij voorwaarde 'True'
Start het programma en voer 'y' in. Het bericht verschijnt. Het verschijnt niet als je iets anders invoert.

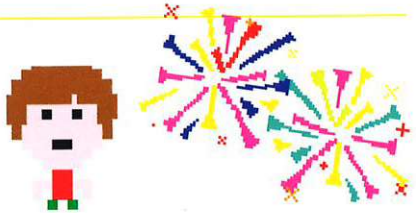
```
Is it your birthday? (y/n)y
Happy Birthday!
```

Typ in 'y'

Het bericht verschijnt

Doe dit of dat

Het commando 'if' kun je combineren met een commando 'else'. Dit betekent dat er iets gebeurt als het 'True' is en dat er iets anders gebeurt als dit niet het geval is.



1 'if else'-voorwaarde
Als je 'y' hebt ingevoerd, print het programma een speciaal bericht voor Nieuwjaar. Je krijgt een ander bericht als je iets anders hebt ingevoerd.

```
ans = input('Is it New Year? (y/n)')
if ans == 'y':
    print('Happy New Year!')
    print('Time for Fireworks.')
else:
    print('Not yet!')
```

Vergeet de dubbele punt niet

Dit bericht verschijnt alleen als je 'y' invoert

Vergeet ook hier niet een dubbele punt te zetten

Werkt alleen als je niet 'y' invoert

2 Output bij voorwaarde 'True'
Start het programma en typ 'y'. Het programma geeft nu je nieuwjaarsbericht en toont het andere bericht niet.

```
Is it New Year? (y/n)y
Happy New Year!
Time for Fireworks.
```

Typ 'y'

3 Output bij voorwaarde 'else'
Typ 'n' of een ander karakter en het nieuwjaarsbericht wordt niet getoond. In plaats daarvan verschijnt het bericht 'Not yet!'

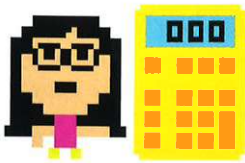
```
Is it New Year? (y/n)n
Not yet!
```

Typ 'n'

Er verschijnt een ander bericht

Doe een van deze dingen

Het commando 'elif' is een afkorting van 'else if'. Het betekent dat als iets 'True' is, je het ene moet doen; als is niet 'True' is, check dan of iets anders 'True' is en doe iets anders als dat zo is. Het volgende rekenprogramma gebruikt het commando 'elif'.



1 'if elif else'-voorwaarde
Dit programma checkt wat er is ingevoerd. Is dat 'add', 'sub', 'mul' of 'div', dan wordt de uitkomst van de som gegeven.

```
a = int(input('a = '))
b = int(input('b = '))
op = input('add/sub/mul/div:')
if op == 'add':
    c = a + b
elif op == 'sub':
    c = a - b
elif op == 'mul':
    c = a * b
elif op == 'div':
    c = a / b
else:
    c = 'Error'
print('Answer = ',c)
```

Vraagt de gebruiker een getal in te voeren

Denk eraan aanhalingstekens en haken te gebruiken

Typ 'add' om twee variabelen samen te voegen

Typ 'div' om de variabelen te delen

Toont een foutmelding in 'c' als er iets wordt ingevoerd

Toont het antwoord of de foutmelding

2 Output als voorwaarde is 'True'
Test het programma. Voer twee getallen in en typ 'sub'. Het antwoord zal het eerste getal zijn min het tweede.

```
a = 7
b = 5
add/sub/mul/div:sub
Answer = 2
```

Voer twee getallen in

Typ 'sub' om 5 van 7 af te trekken

Antwoord wordt berekend door variabele 'a' van variabele 'b' af te trekken

3 Output bij voorwaarde 'else'
De voorwaarde 'else' start als er iets anders wordt ingevoerd dan 'add', 'sub', 'mul' of 'div' en er verschijnt een foutmelding.

```
a = 7
b = 5
add/sub/mul/div:try
Answer = Error
```

Typ hier iets anders

Foutmelding verschijnt

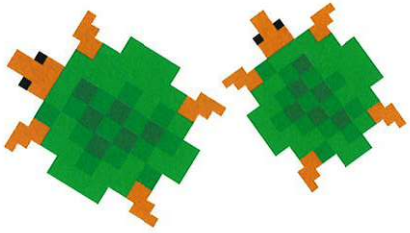
Loops in Python

Met een *loop* (lus) kun je aangeven dat een instructie meermaals moet worden uitgevoerd. Een 'for'-loop voert een bepaald stukje code een door jou vastgesteld aantal malen uit.

ZIE OOK
48–49 Pennen en turtles
'While'-loops **124–125**
Ontsnappingsloops **126–127**

Dingen herhalen

Met een 'for'-loop kun je de codering herhalen zonder die opnieuw te hoeven invoeren. Je kunt iets er een bepaald aantal malen mee herhalen, bijvoorbeeld als je de namen van een klas met 30 leerlingen wilt printen.



1 Programmeer de turtle

Met een 'for'-loop kun je een codering ook inkorten. In dit programma kan de gebruiker een turtle beheren die een lijn tekent als hij over het scherm beweegt. De gebruiker kan door de bewegingen van de turtle te leiden vormen tekenen op het scherm, zoals een driehoek.

```
from turtle import *
forward(100)
right(120)
forward(100)
right(120)
forward(100)
right(120)
```

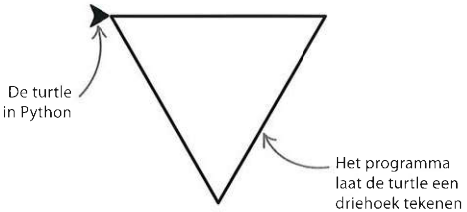
Laadt alle commando's om de turtle te besturen

Door dit commando beweegt de turtle voorwaarts

Hierdoor draait de turtle 120 graden naar rechts

2 De turtle tekent een driehoek

Dit programma vertelt de turtle hoe hij een driehoek kan tekenen door hem de lengte van de drie zijden en tussenliggende hoeken te geven. De turtle zal in een apart venster verschijnen als je het programma start.



3 Gebruik een 'for'-loop

Het programma hierboven geeft de turtle dezelfde twee commando's, 'forward(100)' en 'right(120)', driemaal – een voor elke zijde van de driehoek. Als alternatief hiervoor kun je deze twee commando's in een 'for'-loop gebruiken. Probeer nu met alleen onderstaande code een driehoek te tekenen.

```
for i in range(3):
    forward(100)
    right(120)
```

De 'for'-loop vertelt het programma de instructies drie keer te herhalen

Het instructieblok in een loop wordt voorafgegaan door vier spaties

Loop-variabelen

Een loop-variabele telt het aantal keren dat een loop zichzelf herhaalt. Hij begint bij de eerste waarde in de reeks (0) en stopt één voor de laatste waarde.

De loop-variabele

De loop wordt tien keer herhaald

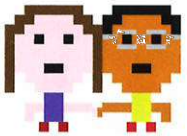
```
for i in range(10):
    print(i, end=' ')
```

Python stopt het tellen bij één voor de laatste waarde

```
>>> 0 1 2 3 4 5 6 7 8 9
```

△ **Eenvoudige loop-variabele**

Hier geeft de loop-reeks niet de juiste waarde aan. Daarom begint Python te tellen vanaf 0, hetzelfde als gebeurt bij strings.



Vertelt het programma in paren te tellen

```
for i in range(2, 11, 2):
    print(i, end=' ')
```

```
>>> 2 4 6 8 10
```

De output verschijnt in paren

△ **In paren tellen**

Deze loop heeft een derde waarde in de reeks, die hem vertelt dat hij in paren moet tellen. Hij stopt bij 10, dat één loop is voordat de loop-variabele doorgaat naar 11.

Vertelt het programma terug te tellen

```
for i in range(10, 0, -1):
    print(i, end=' ')
```

```
>>> 10 9 8 7 6 5 4 3 2 1
```

△ **Terugtellen**

Deze keer telt het programma terug vanaf 10, als bij de lancering van een raket. De loop-variabele start bij 10 en neemt telkens een stap van -1 tot 1 is bereikt.

Ingebedde loops

Loops binnen een loop heten 'ingebedde loops'. Hierbij wordt de buitenste loop alleen herhaald als de binnenste loop het vereiste aantal cyclussen heeft afgelegd.

Om de loops 'n' aantal keer te laten herhalen, moet het laatste getal in de reeks 'n + 1' zijn

Buitenste loop

```
n = 3
for a in range(1, n + 1):
    for b in range(1, n + 1):
        print(b, 'x', a, '=', b * a)
```

Binnenste loop

Deze som wordt negen keer geprint

△ **Loops binnen een loop**

In dit voorbeeld wordt de binnenste loop drie keer herhaald terwijl de buitenste loop één cyclus maakt. In totaal wordt de buitenste loop dus drie keer uitgevoerd en de binnenste negen keer.

De waarde van 'b'

De waarde van 'a'

```
>>>
1 x 1 = 1
2 x 1 = 2
3 x 1 = 3
1 x 2 = 2
2 x 2 = 4
3 x 2 = 6
1 x 3 = 3
2 x 3 = 6
3 x 3 = 9
```

Eerste cyclus van de buitenste loop (de binnenste wordt drie keer herhaald)

Tweede cyclus van de buitenste loop

Derde cyclus van de buitenste loop

△ **Dit is wat er gebeurt**

De ingebedde loops printen de eerste drie regels van de tafels 1, 2 en 3. De waarde van 'a' verandert pas als de buitenste loop wordt herhaald. De waarde van 'b' telt van 1 tot 3 voor elke waarde van 'a'.

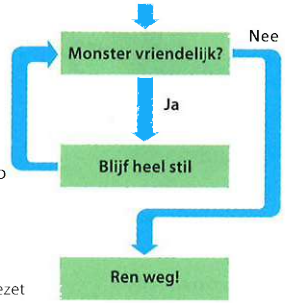
‘While’- of ‘terwijl’-loops

‘For’-loops kunnen handig zijn als je weet hoe vaak een taak moeten worden herhaald. Maar soms wil je dat een loop wordt herhaald totdat er iets verandert. Een ‘while’-loop blijft zich herhalen zo vaak als gewenst.

ZIE OOK
❮ 118–119 Beslissingen nemen
❮ 122–123 Loops in Python
Ontsnappingsloops 126–127 ❯

‘Terwijl’-loops

Een ‘terwijl’-loop is actief zolang een bepaalde voorwaarde ‘True’ is. Deze voorwaarde heet de ‘loop-voorwaarde’ en hij is goed of fout.



▷ **Hoe het werkt**
Een ‘terwijl’-loop checkt of de voorwaarde goed is. Zo ja, dan herhaalt hij zich, zo niet, dan wordt de loop afgebroken.

1 Een ‘terwijl’-loop maken
Plaats de beginwaarde van de variabele ‘answer’ in de loop-voorwaarde. Die moet goed zijn om te kunnen beginnen, anders zal het programma de loop nooit opstarten.

De codering in de loop moet worden voorafgegaan door vier spaties

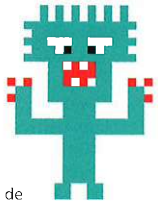
```
answer = 'y'
while answer == 'y':
    print('Stay very still')
    answer = input('Is the monster friendly? (y/n)')
print('Run away!')
```

De variabele ‘answer’ wordt op ‘y’ gezet

De ‘terwijl’-loop begint alleen als de voorwaarde goed is

Als de voorwaarde fout is, zal na de loop een codering zonder inspringing volgen en zal een ander bericht verschijnen

2 Hoe het programma eruitziet
De ingevoerde waarde wordt opgeslagen in de variabele ‘answer’. De loop-voorwaarde is ‘answer == ‘y’’. Als je ‘y’ typt, blijft de loop doorgaan. Als je ‘n’ typt, zal hij stoppen.



```
>>>
Stay very still
Is the monster friendly? (y/n)y
Stay very still
Is the monster friendly? (y/n)y
Stay very still
Is the monster friendly? (y/n)n
Run away!
```

Antwoord is ‘y’, dus de loop blijft doorgaan

Antwoord is ‘n’, dus de loop stopt en een nieuw bericht verschijnt

ONTHOUD

Het blok ‘herhaal tot’

De ‘terwijl’-loop van Python is gelijk aan het blok ‘herhaal tot’ in Scratch. Beide zullen doorgaan totdat er iets verandert in het programma.

Herhaalt ingesloten blokken tot de voorwaarde goed is

Eeuwige loops

Sommige loops gaan eeuwig door. Als je de voorwaarde in een ‘terwijl’-loop op ‘True’ zet, kan hij nooit fout zijn en zal de loop nooit stoppen. Dit kan zowel handig als vervelend zijn.

△ **Doordraaien**
Een loop met de voorwaarde ‘True’ heet een ‘oneindige’ loop. Als iets oneindig is, komt er nooit een einde aan.

1 Een eeuwige loop maken
De loop-voorwaarde is hier op ‘True’ gezet. Niets wat binnen de loop gebeurt, zal ‘True’ in iets anders veranderen dan ‘True’, dus de loop blijft eeuwig doorgaan.

Het ingevoerde woord wordt opgeslagen in de variabele ‘answer’

```
while True:
    answer = input('Type a word and press enter: ')
    print('Please do not type \'' + answer + '\' again.')
```

De loop is altijd ‘True’, dus hij houdt nooit op

2 Hoe het programma eruitziet
Op de pagina hiernaast checkte de loop-voorwaarde van het monsterprogramma het antwoord van de gebruiker. Als het antwoord niet ‘y’ is, zal de loop stoppen. De loop hierboven checkt het antwoord niet, dus kan de gebruiker hem niet stoppen.

Ongeacht wat er wordt ingevoerd zal deze loop gewoon doorgaan

```
>>>
Type a word and press enter: tree
Please do not type 'tree' again
Type a word and press enter: hippo
Please do not type 'hippo' again
Type a word and press enter: water
Please do not type 'water' again
Type a word and press enter
```

TIPS VAN EXPERTS

De loop stoppen

Als je vastzit in een oneindige loop, kun je die onderbreken vanuit IDLE. Klik in het Shell-venster, houd de Ctrl-toets ingedrukt en druk op de ‘C’. Hiermee wordt IDLE gevraagd het programma af te breken. Mogelijk moet je ‘Ctrl-C’ een paar keer indrukken. Dit is gelijk aan klikken op de rode stopknop in Scratch.

ONTHOUD

‘herhaal’-blok

Ken je het ‘herhaal’-blok in Scratch nog? Dat herhaalt de codering erin eindeloos totdat je op de rode stopknop klikt. Een ‘terwijl’-loop doet precies hetzelfde. Je kunt het programma er iets mee laten doen, bijvoorbeeld vragen stellen of een getal printen zolang het programma actief is.

Het blok ‘herhaal’ laat de sprite eindeloos doorgaan

Ontsnappingsloops

Programma's kunnen verstrikt raken in een loop, maar er zijn manieren om eruit te komen. Met het woord 'break' kun je een loop verlaten (zelfs een 'eeuwige'), met het woord 'continue' ga je naar het begin van de volgende loop.

ZIE OOK

- ❮ 122–123 Loops in Python
- ❮ 124–125 'While' of 'terwijl'-loops

Pauzes invoegen

Als je in een loop een pauze invoegt, zal het programma er meteen uit springen, zelfs als de loop-voorwaarde goed is. Elk commando binnen de loop dat na de pauze komt, wordt genegeerd.

1 Schrijf een eenvoudig programma

Dit programma test de gebruiker op de tafel van 7. Het programma gaat door met loops totdat alle 12 vragen zijn beantwoord. Schrijf dit programma in het Code-venster, het wordt later nog bewerkt.

```
table = 7
for i in range(1, 13):
    print('What\'s', i, 'x', table, '?')
    guess = input()
    ans = i * table
    if int(guess) == ans:
        print('Correct!')
    else:
        print('No, it\'s', ans)
print('Finished')
```

De variabele 'i' telt van 1 tot 12

'i' is de loop-variabele

Met de backslash ('\') vertel je Python dat het volgende aanhalingsteken een apostrof is, niet het einde van de string

2 Een 'break' invoegen

Er wordt een pauze ingevoegd, zodat je kunt ontsnappen aan de loop. Het programma voert de pauze uit als je het woord 'stop' intypt.

Als 'guess' gelijk is aan 'stop', zal het programma de rest van de loop afbreken en 'Finished' printen

```
table = 7
for i in range(1,13):
    print('What\'s', i, 'x', table, '?')
    guess = input()
    if guess == 'stop':
        break
    ans = i * table
    if int(guess) == ans:
        print('Correct!')
    else:
        print('No, it\'s', ans)
print('Finished')
```

De variabele 'ans' bevat het juiste antwoord op de vraag


```
>>>
What's 1 x 7 ?
1
No, it's 7
What's 2 x 7 ?
14
Correct!
What's 3 x 7 ?
stop
Finished
```

Tijdens de eerste ronde is 'i' gelijk aan 1

De waarde van 'i' verandert in 2 als de loop opnieuw wordt uitgevoerd

Hiermee wordt het pauzecommando uitgevoerd en verlaat het programma de loop

3 Hoe het werkt

Als je besluit niet verder te gaan na de derde vraag en 'stop' intypt, wordt het commando 'break' uitgevoerd en verlaat het programma de loop.

Overslaan

Met het sleutelwoord 'continue' kun je een vraag overslaan zonder de loop te onderbreken. Hiermee vertel je het programma om de rest van de code binnen de loop te negeren en ga je meteen door naar het begin van de volgende loop.

```
table = 7
for i in range(1,13):
    print('What\'s', i, 'x', table, '?')
    guess = input()
    if guess == 'stop':
        break
    if guess == 'skip':
        print('Skipping')
        continue
    ans = i * table
    if int(guess) == ans:
        print('Correct!')
    else:
        print('No, it\'s', ans)
print('Finished')
```

Stelt de vraag 'Wat is 1 x 7?' in de eerste gang van de loop

Gaat meteen door naar de volgende gang van de loop

4 Een 'continue' invoegen

Voeg een 'if'-statement in de loop toe om te zien of de gebruiker 'skip' heeft geantwoord. Zo ja, dan zal het programma 'Skipping' printen en een 'continue' uitvoeren om door te gaan naar de volgende gang van de loop.

5 Wat er gebeurt

Als de gebruiker een vraag niet wil beantwoorden, kan hij 'skip' ('sla over') typen en doorgaan naar de volgende vraag.

```
>>>
What's 1 x 7 ?
skip
Skipping
What's 2 x 7 ?
14
Correct!
What's 3 x 7 ?
```

Typ 'skip' om naar de volgende vraag te gaan

Als het anwoord goed is, wordt de loop nogmaals uitgevoerd