Programmastroom

Als je meer wilt leren over Python is het belangrijk om eerst te begrijpen hoe programma's werken. De basisprincipes van programmeren die je hebt geleerd in Scratch, kun je ook toepassen in Python.

ZIE OOK

« 30–31 Kleurblokken en scripts

Eenvoudige commando's 102–103 »

Moeilijkere commando's 104–105 »

Van input naar output

Een programma levert input (informatie), verwerkt (of verandert) die en geeft de resultaten (output) daarvan vervolgens terug. Het is net een kok die ingrediënten omzet in een cake en die weer aan jou geeft om op te eten.

Input

- Input-commando
- Toetsenbord
- Muis

Verwerken

- Variabelen
- Wiskunde
- Loops
- Vertakkingen
- Functies

Output

- Print-commando
- Scherm
- Afbeeldingen

△ **Programmastroom in Python**
In Python dienen het toetsenbord en de muis om informatie in te voeren die wordt verwerkt met behulp van elementen zoals loops, vertakkingen en variabelen. De output wordt daarna op het scherm weergegeven.

Het spokenspel bekijken met Scratch-ogen

In de meeste programmeertalen werken programmastromen hetzelfde. Hier een paar voorbeelden van input, verwerking en output in het Python-spokenspel – en hoe het eruit zou zien in Scratch.

1 Input
In Python ontvangt de functie 'input()' de input via het toetsenbord. Het is gelijk aan het blok 'vraag en wacht' in Scratch.

```
door = input('1, 2 or 3?')
```

De vraag verschijnt op het scherm

De vraag in het Scratch-blok

Scratch-blok 'vraag en wacht'

2 Verwerken
Variabelen worden gebruikt om de score bij te houden en de functie 'randint' kiest een willekeurige deur. In Scratch doen verschillende blokken dit.

```
score = 0
```

Zet de variabele 'score' op 0

Dit Scratch-blok zet de waarde van de variabele op 0

Scratch-blok 'maak score 0'

```
ghost_door = randint(1, 3)
```

Selecteert een willekeurig nummer tussen 1 en 3

Scratch-blok 'willekeurig getal'

Dit Scratch-blok selecteert een willekeurig nummer

3 Output
De 'print()'-functie dient voor de output in Python, terwijl het blok 'zeg' hetzelfde doet in Scratch.

```
print('Ghost game')
```

Geeft 'Spokenspel' weer op het scherm

Geeft een tekstballon met daarin het woord 'Spokenspel'.

Scratch-blok 'zeg'

TIPS VAN EXPERTS

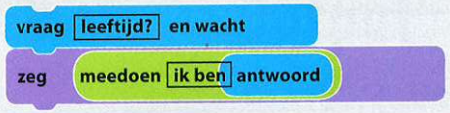
Eén script tegelijk

Er is een belangrijk verschil tussen Scratch en Python. In Scratch kunnen veel scripts tegelijk actief zijn, maar in Python bestaat het programma slechts uit één script.

Eenvoudige commando's

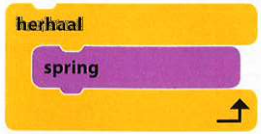
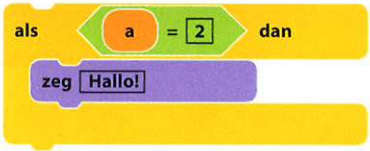
ZIE OOK
‹ 86–87 Wat is Python?
Moeilijkere commando's 104–105 ›

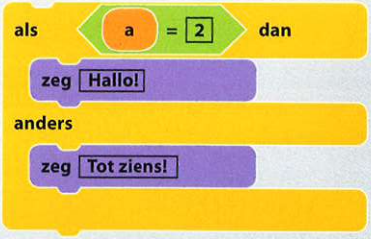
Op het eerste gezicht kan Python je misschien afschrikken, vooral als je het vergelijkt met Scratch. De twee talen verschillen echter niet zo veel van elkaar als het lijkt. Hier zie je een overzicht van de overeenkomsten tussen de basiscommando's in Python en Scratch.

Commando	Python 3	Scratch 2.0
Start programma	Menu 'Run' of druk op F5 (in Code-venster)	
Stop programma	Druk op Ctrl-C (in Shell-venster)	
Schrijf tekst naar scherm	`print('Hello!')`	
Link een variabele aan getal	`magic_number = 42`	
Link een variabele aan een tekststring	`word = 'dragon'`	
Lees tekst van toetsenbord in variabele	`age = input('age?')` `print('I am ' + age)`	

Voeg een getal toe aan een variabele	`cats = cats + 1`
	of
	`cats += 1`
		

Toevoegen	`a + 2`	
Aftrekken	`a - 2`	
Vermenigvuldigen	`a * 2`	
Delen	`a / 2`	

Commando	Python 3	Scratch 2.0
Eeuwige loop	`while True:` `  jump()`	
Loop 10 keer	`for i in range (10):` `  jump()`	
Is gelijk aan?	`a == 2`	
Is minder dan?	`a < 2`	
Is meer dan?	`a > 2`	
NIET	`not`	
OF	`or`	
EN	`and`	
als dan	`if a == 2:` `  print('Hello!')`	

als dan anders	`if a == 2:`
	`  print('Hello!')`
	`else:`
	`  print('Goodbye!')`
		

Moeilijkere commando's

Met Python kun je ook enkele ingewikkeldere dingen doen die ook in Scratch mogelijk zijn, zoals complexe loops maken, spelen met strings en lijsten en afbeeldingen maken met 'turtles'.

ZIE OOK

- ❮ **86–87** Wat is Python?
- ❮ **102–103** Eenvoudige commando's

Commando	Python 3	Scratch 2.0
Loops met restricties	`while roll != 6:` `  jump()`	herhaal tot rol = 6 spring
Wacht	`from time import sleep` `sleep(2)`	wacht 2 sec.
Willekeurige getallen	`from random import randint` `roll = randint(1, 6)`	maak rol getal tussen 1 en 6
Definieer een functie of subprogramma	`def jump():` `  print('Jump!')`	definieer spring denk Spring!
Roep een functie of subprogramma	`jump()`	spring
Definieer een functie of subprogramma met input	`def greet(who):` `  print('Hello ' + who)`	definieer groet wie zeg mee-doen Hallo wie
Roep een functie of subprogramma	`greet('chicken')`	groet kip

Commando	Python 3	Scratch 2.0

Turtles	`from turtle import *`
	`clear()`
	`pendown()`
	`forward(100)`
	`right(90)`
	`penup()`
		wissen
		pen neer
		neem 100 stappen
		draai 90 graden
		pen op

Meedoen aan strings	`print(greeting + name)`	zeg mee-doen groeten naam
Krijg een letter van een string	`name[0]`	letter 1 van naam
Lengte van een string	`len(name)`	lengte van naam
Maak een lege lijst	`menu = list()`	Maak een Lijst
Voeg een item toe aan eind lijst	`menu.append(thing)`	voeg ding toe aan menu
Hoeveel items op lijst?	`len(menu)`	lengte van menu
Waarde van 5de item op lijst	`menu[4]`	zeg item 5 van menu
Verwijder 2de item van lijst	`del menu[1]`	verwijder 2 van menu
Staat item op lijst?	`if 'olives' in menu:` `  print('Oh no!')`	als menu bevat olijven dan zeg O nee!

Welk venster?

Je kunt in IDLE twee verschillende vensters kiezen. Met het Code-venster kun je schrijven en programma's opslaan, het Shell-venster voert Python-instructies direct uit.

ZIE OOK
❮ 92–93 Kennismaking met IDLE
❮ 96–97 Spokenspel

Het Code-venster

Tot nu toe is het Code-venster in dit boek gebruikt om programma's mee te schrijven. Je opent het programma, slaat het op, voert het uit en het resultaat verschijnt in het Shell-venster.

▽ **Programma's uitvoeren**
Dit proces wordt gebruikt om Python-programma's uit te voeren. Programma's moet je altijd eerst opslaan alvorens ze uit te voeren.

Codering invoeren → Opslaan → Module uitvoeren → Output

1 Een programma invoeren in het Code-venster
Voer deze code in het Code-venster in, sla hem op en klik op 'Run Module' in het menu 'Run' om het programma uit te voeren.

```
a = 10
b = 4
print(a + b)
print(a - b)
```

Geef 'a' de waarde 10
Geef 'b' de waarde 4
Het commando 'print' toont de antwoorden op deze sommen

2 Output in het Shell-venster
Als het programma loopt, wordt de output (het resultaat van het programma) weergegeven in het Shell-venster.

```
>>>
14
6
```

De antwoorden op de sommen verschijnen in het Shell-venster

Het Shell-venster

Python begrijpt ook commando's die in het Shell-venster worden weergegeven. Zodra ze worden ingevoerd, treden ze in werking. Het resultaat wordt direct weergegeven.

```
>>> a = 10
>>> b = 4
>>> a + b
14
>>> a - b
6
```

De eerste twee commando's hebben geen output omdat ze alleen waarden toekennen aan 'a' en 'b'.
Output verschijnt direct

◁ **Codering en output tezamen**
Het Shell-venster toont de codering en de output tezamen. Het is makkelijk te bepalen welke antwoorden bij welke sommen horen als je de commando's in het Shell-venster invoert.

△ **Test je ideeën**
Het Shell-venster geeft je direct een antwoord, waardoor het ideaal is om instructies mee te testen en te onderzoeken wat ze kunnen doen.

Python-achtergrond

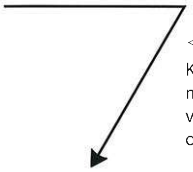
Het Shell-venster kan worden gebruikt voor alle soorten Python-commando's, inclusief tekenen. Met de schildpad (turtle) kun je op het scherm tekenen op dezelfde manier als met de pen in Scratch.

Laadt alle commando's die de schildpad besturen

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

Beweegt de schildpad vooruit

◁ **Voer de codering in**
Typ deze instructies in het Shell-venster. Ze werken meteen nadat ze zijn ingevoerd. Als de schildpad beweegt, tekent hij een lijn.

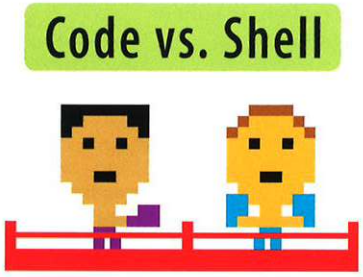


◁ **Schildpad-illustratie**
Kun jij uitzoeken hoe je andere vormen moet tekenen, zoals een vierkant of een vijfhoek? Typ 'clean' in het Shell-venster om opnieuw te beginnen.

Welk venster moet je gebruiken?

Moet je het Code-venster of het Shell-venster gebruiken? Dat hangt af van het soort programma dat je schrijft en of het herhaald moet worden.

▷ **Code-venster**
Het Code-venster is ideaal voor langere coderingen omdat je die kunt opslaan en bewerken. Dat is makkelijker dan alle instructies opnieuw invoeren als je hetzelfde of iets vergelijkbaars nog eens wilt doen. Je moet ze dan opslaan en telkens opnieuw uitvoeren.



◁ **Shell-venster**
Het Shell-venster is perfect voor snelle experimenten, zoals checken hoe een commando werkt. Het is ook een handige rekenmachine. Het slaat de instructies niet op, dus als je iets uitprobeert wat je later nog eens wilt doen, kun je misschien beter het Code-venster gebruiken.

TIPS VAN EXPERTS

Kleuren in de codering

IDLE kleurcodeert de tekst. De kleuren geven je een idee van wat Python denkt dat elk stuk tekst voorstelt.

◁ **Ingebouwde functies**
Commando's in Python, zoals 'print', zijn paars.

◁ **Strings in aanhalingstekens**
Groen betekent strings. Als de haakjes ook groen zijn, ontbreekt er een aanhalingsteken.

◁ **Meeste symbolen en namen**
Meeste coderingen staan in zwart.

◁ **Output**
De output in het Shell-venster wordt in blauw weergegeven.

◁ **Sleutelwoorden**
Sleutelwoorden, zoals 'if' en 'else', zijn oranje. Je kunt ze niet als variabelen gebruiken.

◁ **Fouten**
Python gebruikt rood om je te wijzen op foutmeldingen in het Shell-venster.

Variabelen in Python

Variabelen dienen om stukken informatie in een programma te onthouden. Ze zien eruit als dozen waarin data kunnen worden bewaard en gelabeld.

ZIE OOK

Soorten data **110–111** ❯
Rekenen in Python **112–113** ❯
Strings in Python **114–115** ❯
Input en output **116–117** ❯
Functies **130–131** ❯

Een variabele maken

Wanneer een getal of string in een variabele wordt geplaatst, heet dat 'een waarde toewijzen aan de variabele'. Je kunt dit doen met het '='-teken. Probeer deze codering uit in het Shell-venster.

Naam variabele

Aan de variabele toegewezen waarde

```
>>> bones = 3
```

△ **Een getal toewijzen**
Om een getal toe te wijzen voer je de naam van de variabele in, een '='-teken en dan een getal.

Naam variabele

Aan de variabele toegewezen string

```
>>> dogs_name = 'Bruno'
```

△ **Een string toewijzen**
Om een string toe te wijzen voer je de naam van de variabele in, een '='-teken en dan de string tussen aanhalingstekens.

ONTHOUD

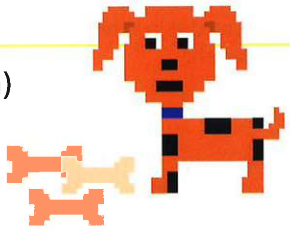
Variabelen in Scratch

Het commando dat een waarde toewijst aan een variabele in Python doet hetzelfde als dit Scratch-blok. In Python hoef je echter nergens op te klikken om een variabele te maken. Python maakt deze meteen als je er een waarde aan hebt toegewezen.

Scratch-blok om een waarde toe te wijzen aan een variabele

Een variabele op het scherm tonen (printen)

Met het commando 'print' kun je iets op het scherm tonen. Dat heeft niets met een printer te maken. Je kunt er de waarde van de variabele mee tonen.



```
>>> print(bones)
3
```

Naam variabele

△ **Getal-output**
De variabele 'bones' (botten) bevat het getal 3, dus dat is wat er in het Shell-venster staat.

```
>>> print(dogs_name)
Bruno
```

Hier geen aanhalingstekens

△ **String-output**
De variabele 'dogs_name' bevat een string, dus die wordt weergegeven. Als je een string print, worden er geen aanhalingstekens getoond.

De inhoud van een variabele veranderen

Om de waarde van een variabele te veranderen, kun je er gewoon een nieuwe waarde aan toewijzen. Hier heeft de variabele 'gifts' de waarde 2. Dit wordt veranderd in 3 als er een nieuwe waarde aan wordt toegewezen.

```
>>> gifts = 2
>>> print(gifts)
2
>>> gifts = 3
>>> print(gifts)
3
```

Verandert de waarde van de variabele

Variabelen gebruiken

Je kunt de waarde van de ene variabele toewijzen aan een andere met het '='-teken. Als bijvoorbeeld de variabele 'rabbits' het aantal konijnen weergeeft, kun je die gebruiken om dezelfde waarde toe te wijzen aan de variabele 'hats', zodat elk konijn een hoed krijgt.

1 **Wijs de variabelen toe**
Deze code wijst het getal 5 toe aan de variabele 'rabbits'. Vervolgens wijst hij dezelfde waarde toe aan de variabele 'hats'.

Naam variabele

Toegewezen waarde aan de variabele

```
>>> rabbits = 5
>>> hats = rabbits
```

'hats' heeft nu dezelfde waarde als 'rabbits'

2 **Print de waarden**
Om twee variabelen te printen plaats je ze tussen haken na het commando 'print', gescheiden door een komma. Zowel 'hats' als 'rabbits' bevatten nu de waarde 5.

```
>>> print(rabbits, hats)
5 5
```

Tik na de komma een spatie

3 **Verander de waarde van 'rabbits'**
Als je de waarde van 'rabbits' verandert, heeft dat geen invloed op de waarde van 'hats'. De variabele 'hats' verandert alleen als je er een nieuwe waarde aan toewijst.

```
>>> rabbits = 10
>>> print(rabbits, hats)
10 5
```

Geef 'rabbits' een nieuwe waarde

Waarde voor 'hats' blijft gelijk

TIPS VAN EXPERTS

Variabelen benoemen

Onthoud de volgende regels als je variabelen een naam geeft:

- Je mag alle letters en cijfers gebruiken.
- Je mag niet met een cijfer beginnen.
- Je mag geen symbolen als -, /, # en @ gebruiken.
- Je mag geen spaties gebruiken.
- Je mag wel een underscore (_) gebruiken in plaats van een spatie.
- Python maakt verschil tussen hoofd- en kleine letters; 'Dogs' en 'dogs' zijn twee verschillende variabelen.
- Gebruik geen woorden die Python ziet als een commando, zoals 'print'.