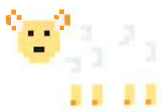
Soorten data

Er zijn verschillende soorten data in Python. Meestal zal Python werken met de soort die is gebruikt, maar soms zul je de data moeten omzetten naar een ander type.

ZIE OOK
Rekenen in Python 112–113 ❯
Strings in Python 114–115 ❯
Beslissingen nemen 118–119 ❯
Lijsten 128–129 ❯

Getallen

Python kent twee typen data voor getallen. 'Integers' zijn gehele getallen (getallen zonder decimale punt), 'floats' zijn getallen met een decimale punt. Met een integer kunnen dingen als schapen worden geteld, terwijl met een float zaken kunnen worden gemeten, zoals gewicht.



```
>>> sheep = 1
>>> print(sheep)
1
```

Een integer is een geheel getal

△ **Integers**
Een integer is een getal zonder decimale punt, bijvoorbeeld de 1 in de variabele 'schaap'.

```
>>> sheep = 1.5
>>> print(sheep)
1.5
```

1.5 is een float

△ **Floats**
Een float is een getal met een decimale punt, bijvoorbeeld 1.5. Er worden doorgaans geen hele objecten mee geteld.

Strings

Net als in Scratch heet een stuk tekst in Python een 'string'. Strings kunnen letters, cijfers, spaties en symbolen bevatten, zoals punten en komma's. Ze staan tussen enkele aanhalingstekens.

▷ **Een string gebruiken**
Om een string toe te wijzen aan een variabele zet je de tekst tussen enkele aanhalingstekens.

De string tussen aanhalingstekens

```
>>> a = 'Coding is fun!'
>>> print(a)
Coding is fun!
```

De waarde van de variabele 'a'

Booleaanse variabelen

In Python heeft een booleaanse variabele altijd een waarde die 'True' of 'False' is. In beide gevallen wordt het woord met een hoofdletter geschreven.

▷ **True (Goed)**
Als de waarde 'True' in een variabele staat, is het een booleaanse variabele.

Geen aanhalingstekens

```
>>> a = True
>>> print(a)
True
```

Booleaanse waarde

▷ **False (Fout)**
Wanneer een waarde 'False' in een variabele staat, is het ook een booleaanse variabele.

```
>>> a = False
>>> print(a)
False
```

Booleaanse waarde

TIPS VAN EXPERTS

Soorten data

Python kent veel verschillende soorten data. Met het commando 'type' kun je erachter komen welk type bepaalde data zijn.

'type'-commando

```
>>> type(24)
<class 'int'>
>>> type(24.3)
<class 'float'>
>>> type('24')
<class 'str'>
```

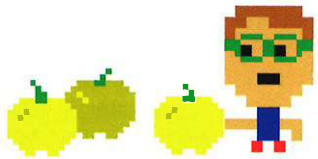
24 is een integer ('int')

24.3 is een float ('float')

'24' is een string ('str'), want het staat tussen aanhalingstekens

Soorten data converteren

Variabelen kunnen elk type data bevatten. Er ontstaan problemen als je soorten wilt mengen. Datatypes moeten soms worden geconverteerd om een foutmelding te voorkomen.



▷ **Gemengd type**
Het commando 'input' geeft altijd een string, zelfs als je een getal hebt ingevoerd. In dit voorbeeld wordt een foutmelding gegeven omdat 'apple' eigenlijk een string bevat.

Naam variabele

String tussen aanhalingstekens getoond op scherm

```
>>> apple = input('Enter number of apples ')
Enter number of apples 2
>>> print(apple + 1)
TypeError
```

Probeert het getal 1 toe te voegen aan de variabele 'apple'

Dit programma geeft een foutmelding, omdat Python niet weet hoe een getal toe te voegen aan een string

▷ **Soorten data converteren**
Om de string in een getal te converteren, zal het commando 'int()' het omzetten in een integer.

```
>>> print(int(apple) + 1)
3
```

Het programma werkt nu en toont het resultaat

De variabele verandert van een string in een integer, dus kun je er een getal aan toevoegen

Rekenen in Python

Met Python kun je alle wiskundige problemen oplossen, waaronder optellen, aftrekken, vermenigvuldigen en delen. Je kunt variabelen ook in sommen toepassen.

ZIE OOK
- 52–53 Rekenen
- 108–109 Variabelen in Python

Eenvoudige berekeningen

In Python kun je eenvoudige berekeningen maken door ze in het Shell-venster in te voeren. Je hebt hiervoor de functie 'print()' niet nodig – Python geeft direct het antwoord. Probeer deze voorbeelden in het Shell-venster:

Je kunt niet door 0 delen, dus krijg je altijd een foutmelding als je dat probeert.

```
>>> 12 + 4
16
```

In het Shell-venster krijg je onmiddellijk uitkomsten

Het antwoord verschijnt als je op 'Enter' drukt

△ **Optellen**
Met het symbool '+' kun je getallen bij elkaar optellen.

```
>>> 12 - 4
8
```

△ **Aftrekken**
Met het symbool '-' trek je het tweede getal af van het eerste.

Computers gebruiken voor vermenigvuldigen het symbool '*', niet 'x'

```
>>> 12 * 4
48
```

△ **Vermenigvuldigen**
Met het symbool '*' vermenigvuldig je twee getallen met elkaar.

```
>>> 12 / 4
3.0
```

Delen in Python levert een antwoord op met een float (een getal met een decimale punt)

△ **Delen**
Met het symbool '/ ' deel je het eerste getal door het tweede.

Gebruik van haken

Met haken kun je Python instrueren welk deel van de som als eerste moet. Python zal altijd eerst de waarde van de som tussen haken uitrekenen alvorens de rest van de som op te lossen.

Eerst wordt berekend dat 6 + 5 = 11, daarna wordt 11 vermenigvuldigd met 3

```
>>> (6 + 5) * 3
33
```

△ **Eerst optellen**
In deze som staan haken om Python te vertellen dat hij eerst moet optellen.

Eerst wordt berekend dat 5 * 3 = 15, daarna wordt 15 opgeteld bij 6

```
>>> 6 + (5 * 3)
21
```

Ander antwoord

△ **Eerst vermenigvuldigen**
Hier staan haken om eerst te laten vermenigvuldigen om zo het juiste antwoord te krijgen.

Antwoorden inbrengen in variabelen

Als een variabele een toegewezen getalwaarde is, kun je hem gebruiken binnen een som. Als je de som toewijst aan een variabele, verschijnt het antwoord in de variabele, maar de som niet.

2 Verander de waarde van een variabele
Wijzig de waarde van de variabele 'ants' of 'spiders'. Voeg de variabelen weer samen en zet het antwoord in de variabele 'bugs'.

```
>>> ants = 22
>>> spiders = 18
>>> bugs = ants + spiders
>>> print(bugs)
40
```

Wijzig de waarde in 'spiders'

Voegt de variabelen weer samen

Het antwoord verandert

1 Maak een eenvoudige optelsom
Dit programma voegt de variabelen 'ants' (mieren) en 'spiders' (spinnen) samen en zet het antwoord in de variabele 'bugs' (beestjes).

```
>>> ants = 22
>>> spiders = 35
>>> bugs = ants + spiders
>>> print(bugs)
57
```

Voegt de waarden van de twee variabelen samen

Print de waarde in 'bugs'

3 De toewijzing negeren
Als de som niet wordt toegewezen aan de variabele 'bugs', zelfs als de waarde van 'ants' en 'spiders' wijzigt, blijft de waarde van 'bugs' gelijk.

```
>>> ants = 11
>>> spiders = 17
>>> print(bugs)
40
```

Print de waarde in 'bugs'

Het antwoord is niet veranderd (nog steeds 18 + 22)

Willekeurige getallen

Wil je een willekeurig getal kiezen, dan moet je eerst de functie 'randint' laden in Python. Gebruik hiervoor het commando 'import'. De functie 'randint()' is al geprogrammeerd met een code om een willekeurige integer (geheel getal) te kiezen.

Voegt de functie 'randint()' toe

```
>>> from random import randint
>>> randint(1, 6)
3
```

Kiest een wilekeurig getal tussen 1 en 6

3 is willekeurig gekozen

△ **Rol de dobbelsteen**
De functie 'randint()' kiest een willekeurig getal tussen de twee nummers tussen haken. In dit programma kiest 'randint(1, 6)' een waarde uit 1 t/m 6.

ONTHOUD

Willekeurig blok

De functie 'randint()' werkt als het blok 'een willekeurig getal tussen' in Scratch. In Scratch moet je de hoogst en de laagst mogelijke getallen invoeren in de vensters, in Python zet je de getallen tussen haken, gescheiden door een komma.

△ **Gehele getallen**
Zowel de Python-functie 'randint()' als het Scratch-blok kiezen willekeurig een geheel getal – het resultaat heeft nooit decimalen.

Strings in Python

Met Python kun je uitstekend woorden en zinnen in programma's gebruiken. Je kunt verschillende strings (rijen karakters) samenvoegen of afzonderlijke delen ervan selecteren en verwijderen.

ZIE OOK

- ❮ **54–55** Strings en lijsten
- ❮ **110–111** Soorten data

Een string maken

Een string kan letters, getallen, symbolen en spaties bevatten. Dit zijn karakters. Strings kunnen in variabelen worden geplaatst.

▷ **Strings in variabelen**
Variabelen kunnen strings opslaan. Typ deze twee strings in de variabelen 'a' en 'b'.

```
>>> a = 'Run! '
>>> b = 'Aliens are coming.'
```

Aan de aanhalingstekens zie je dat de variabele een string bevat

Strings toevoegen

Het samenvoegen van twee getallen levert een nieuw getal op. Op dezelfde manier ontstaat er een nieuwe string als je de ene bij de andere voegt.

```
>>> c = a + b
>>> print(c)
Run! Aliens are coming.
```

De variabelen 'a' en 'b' worden gecombineerd tot variabele 'c'.

△ **Strings samenvoegen**
Het '+'-teken voegt de ene string toe aan de andere en het antwoord is de variabele 'c'.

Een nieuwe string is toegevoegd aan variabele 'c'

```
>>> c = b + ' Watch out! ' + a
>>> print(c)
Aliens are coming. Watch out! Run!
```

△ **Een andere string tussenvoegen**
Je kunt ook een nieuwe string tussen twee bestaande strings voegen. Probeer bovenstaand voorbeeld eens.

De nieuwe string verschijnt midden in het bericht

TIPS VAN EXPERTS

Lengte van een string

Met de functie 'len()' kun je de lengte van een string achterhalen. Python telt alle karakters, inclusief spaties, en geeft dan het totaal aantal karakters in de string.

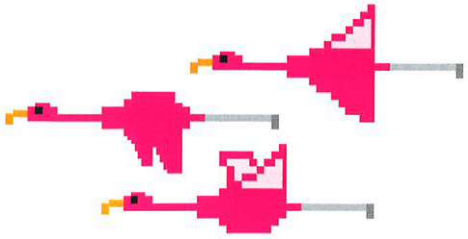
Berekent de lengte van de string in variabele 'a' ('Run! ')

```
>>> len(a)
4
>>> len(b)
18
```

De string in variabele 'b' ('Aliens are coming.') telt 18 karakters

De karakters nummeren

Elk karakter in een string is genummerd op basis van zijn positie. Met dit positienummer krijg je individuele letters of symbolen en kun je ze uit de string halen.



1 Telling begint vanaf 0
Python begint bij het tellen van de posities vanaf 0. Het tweede karakter staat dus op positie 1, het derde op 2 enzovoort.

```
>>> a = 'FLAMINGO'
```

De zesde letter, 'N', staat op positie 5

Het laatste karakter, 'O', staat op positie 7

Het eerste karakter, 'F', staat op positie 0

2 Karakters tellen
Het positienummer wordt een 'index' genoemd. Je kunt hiermee een afzonderlijke letter uit een string halen.

```
>>> a[3]
'M'
```

De index staat tussen vierkante haken

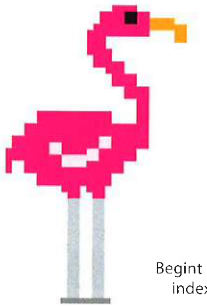
Het karakter op positie 3 in de variabele 'a'

3 'Slicen'
Met twee indexen kun je een deel van de string eruit halen of 'slicen'. De laatste positie is niet inbegrepen.

```
>>> a[1:7]
'LAMING'
```

Dubbele punt definieert de reeks karakters

Een 'slice' uit index 1 tot index 6 van variabele 'a'



4 Van het begin tot het einde
Als je het begin of einde van een index weghaalt, zal Python automatisch het eerste of laatste karakter van de string gebruiken.

```
>>> a[:3]
'FLA'
>>> a[3:]
'MINGO'
```

Begint bij index 0

Eindigt bij index 7

Apostrofs

Strings kunnen tussen enkele of dubbele aanhalingstekens staan. De string moet echter beginnen en eindigen met hetzelfde type. In dit boek worden enkele aanhalingstekens gebruikt. Maar wat gebeurt er als er een apostrof in je string staat?

```
>>> print('It\'s a cloudy day.')
It's a cloudy day.
```

De apostrof zit binnen de string

△ **Ontwijk de apostrof**
Als je '\' ervoor plaatst, leest Python een apostrof niet als het einde van een string. Dit heet 'ontwijken'.

Input en output

Programma's reageren met gebruikers via input en output. Je kunt input aan een programma geven via het toetsenbord. Output verschijnt als informatie geprint op het scherm.

ZIE OOK

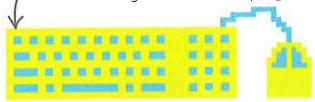
« 100–101 Programmastroom

« 110–111 Soorten data

Loops 122–123 » in Python

Input

Met de functie 'input()' kun je via het toetsenbord input aan een programma geven. Dit gebeurt pas als de gebruiker klaar is met typen en op 'Enter' drukt.



Met de functie 'input()' kunnen gebruikers via het toetsenbord in verbinding staan met een programma

1 Input gebruiken

Een programma kan een gebruiker herinneren wat te typen. Het bericht staat tussen de haken na 'input()'.

Door een spatie te zetten na de dubbele punt lijkt de output overzichtelijker

```
name = input ('Enter your name: ')
print('Hello', name)
```

Wat het programma weergeeft, hangt af van de naam die de gebruiker typt

2 Output in het Shell-venster

Als het programma loopt, verschijnen het bericht 'Enter your name: ' en het antwoord daarop in het Shell-venster.

```
Enter your name: Jina
Hello Jina
```

Bericht programma-output

Gebruiker voert zijn naam in

Output

Met de functie 'print()' kun je karakters weergeven in het Shell-venster. Je kunt er een combinatie van tekst en variabelen mee tonen.

Output wordt weergegeven op het scherm

1 Maak enkele variabelen

Voor dit eenvoudige experiment maak je drie variabelen, twee strings en een integer (geheel getal).

```
>>> a = 'Dave'
>>> b = 'is'
>>> c = 12
```

Aanhalingstekens geven aan dat dit strings zijn

Geen aanhalingstekens laten zien dat dit een integer is

2 De 'print()'-functie gebruiken

Je kunt meerdere items tussen de haken van de 'print()'-functie plaatsen. Je kunt verschillende soorten variabelen en zelfs strings en variabelen combineren.

```
>>> print(a, b, c)
Dave is 12
>>> print('Goodbye', a)
Goodbye Dave
```

Komma's scheiden de verschillende items

Twee manieren om strings te scheiden

Tot nu toe is de output geprint op één regel met een spatie tussen de items. Hier volgen nog twee manieren om strings te scheiden.

```
>>> print(a, b, c, sep='-')
Dave-is-12
```

Het karakter tussen de outputs

△ **Breek de output af**

Je kunt een afbreekteken zetten tussen twee variabelen als ze worden geprint. Je kunt ook andere karakters gebruiken, zoals '+' of '*'.

De separator

```
>>> print(a, b, c, sep='\n')
Dave
is
12
```

Elke variabele begint op een nieuwe regel

△ **Outputs op een nieuwe regel**

De spatie of het karakter tussen de outputs heet een 'separator' ('sep'). Via '\n' print je elke output op een nieuwe regel.

Drie manieren om een output te beëindigen

Er zijn verschillende manieren om het eind van de output van de 'print'-functie te markeren.

```
>>> print(a, '.')
Dave .
```

Punt toegevoegd als string

```
>>> print(a, end='.')
Dave.
```

Punt toegevoegd als 'end'-karakter

△ **Voeg een punt toe aan de output**

Je kunt een punt toevoegen als een nieuw te printen string. Hij zal echter verschijnen met een spatie ervoor. Als je dit wilt voorkomen, tik dan 'end='.''

Loop om drie keer te printen

Spatie als 'end'-karakter

```
>>> for n in range(3):
    print('Hurray!' end=' ')
Hurray! Hurray! Hurray!
```

De volledige output wordt op één regel geprint

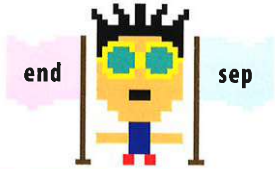
△ **Output op één regel**

Gewoonlijk begint elk nieuw 'print'-commando op een nieuwe regel. Om alle output op één regel te krijgen, gebruik je een spatie als 'end'-karakter.

TIPS VAN EXPERTS

Opties aan het eind

Met de labels 'end' en 'sep' vertel je Python dat het volgende item in het programma niet gewoon een andere string is. Gebruik deze labels, anders zal het programma niet goed werken.




```
>>> print(a, end='\n\n\n\n')
Dave
```

Elke '\n' begint op een nieuwe regel

Lege ruimte voor de prompt

```
>>>
```

△ **Lege regels aan het eind**

Met '\n' begint elke output vanaf een nieuwe regel. Enkele daarvan achter elkaar zullen aan het eind van het programma meerdere lege regels toevoegen.