Lijsten

Als je veel gegevens op één plek wilt bewaren, kun je ze in een lijst zetten. Lijsten kunnen getallen, strings, andere lijsten of een combinatie ervan bevatten.

ZIE OOK
‹ 54–55 Strings en lijsten
Gekke 132–133 ›
zinnen

Wat is een lijst?

Een lijst is een structuur in Python waarin items op volgorde worden bewaard. Elk ingevoerd item krijgt een nummer waarmee je het gemakkelijk kunt terugvinden. Je kunt de items in de lijst op elk moment wijzigen, verwijderen of er iets aan toevoegen.

▽ **Kijken naar lijsten**
Elk item in een lijst staat tussen enkele aanhalingstekens en is van het volgende gescheiden door een komma. De hele lijst bevindt zich tussen vierkante haken.

De lijst wordt bewaard in de variabele 'mylist'

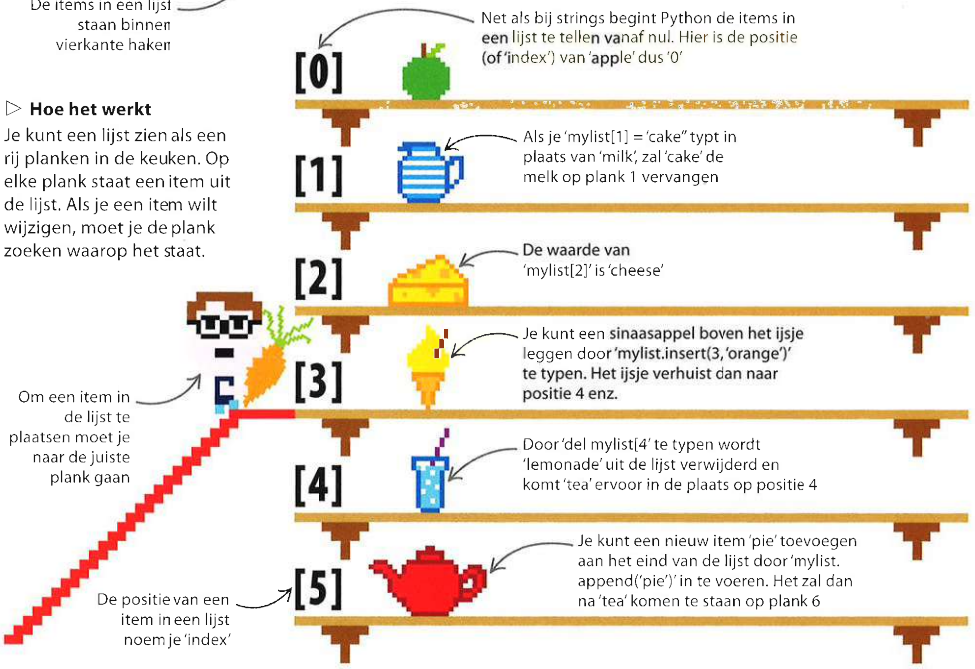
```
>>> mylist = ['apple', 'milk', 'cheese', 'icecream', \
'lemonade', 'tea']
```

De items in een lijst worden gescheiden door komma's

Met dit karakter kun je een codering over twee regels laten lopen.

De items in een lijst staan binnen vierkante haken

▷ **Hoe het werkt**
Je kunt een lijst zien als een rij planken in de keuken. Op elke plank staat een item uit de lijst. Als je een item wilt wijzigen, moet je de plank zoeken waarop het staat.



Lijsten gebruiken

Als je eenmaal een lijst hebt gemaakt, kun je programma's schrijven om iets met de data in de lijst te doen, ze in een loop plaatsen bijvoorbeeld. Je kunt lijsten ook combineren tot een nieuwe lijst.

De lijst wordt bewaard in de variabele 'names'

```
>>> names = ['Simon', 'Kate', 'Vanya']
>>> for item in names:
        print('Hello', item)
```

De kern van de loop moet worden voorafgegaan door vier spaties

```
Hello Simon
Hello Kate
Hello Vanya
```

Als het programma begint, verschijnt er 'Hello', gevolgd door elke naam uit de lijst op je scherm

◁ **Lijsten in loops**
Met een loop kun je door elk item in de lijst lopen. Dit programma zegt 'Hello' tegen elke ingang van een reeks namen.

LINGO

Veranderlijke objecten

Lijsten in Python zijn 'veranderlijk', dat betekent dat je ze kunt wijzigen. Je kunt items toevoegen of verwijderen, of de volgorde ervan veranderen. Andere functies in Python, zoals tupels (zie p.134-135), kun je niet meer wijzigen als je ze eenmaal hebt gemaakt. Deze zijn 'onveranderlijk'.

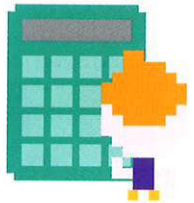
```
x = [1, 2, 3, 4]
y = [5, 6, 7, 8]
z = x + y
print(z)
z = [1, 2, 3, 4, 5, 6, 7, 8]
```

Onthoud dat lijsten tussen vierkante haken moeten staan

Dit voegt de lijsten samen

De nieuwe lijst bevat alles uit lijst 'x', gevolgd door alles uit lijst 'y'

◁ **Lijsten samenvoegen**
Je kunt twee lijsten samenvoegen. In de nieuwe lijst staan dan de items uit beide oude lijsten.



▽ **Lijsten in lijsten**
De items in een lijst kunnen zelf lijsten zijn. De lijst 'suitcase' hieronder bevat twee lijsten met kleren. Het is net een koffer die door twee mensen wordt gebruikt. Beiden pakken drie items in.

Omdat de lijst tussen vierkante haken staat, is het een afzonderlijk item geworden binnen de lijst 'suitcase': 'suitcase[0]'

'suitcase[1]'

```
>>> suitcase=[['hat', 'tie', 'sock'],['bag', 'shoe', 'shirt']]
>>> print(suitcase)
[['hat', 'tie', 'sock'],['bag', 'shoe', 'shirt']]
>>> print(suitcase[1])
['bag', 'shoe', 'shirt']
>>> print(suitcase[1][2])
shirt
```

Hier verschijnt de complete 'suitcase'-lijst

Hier verschijnt de tweede lijst, 'suitcase[1]'

Hier verschijnt het item op index 2 in 'suitcase[1]'. Onthoud: Python telt de items vanaf nul

Functies

Een functie is een deel van de codering met een specifieke taak. Hij pakt de code in, geeft die een naam en kan altijd worden gebruikt door hem 'op te roepen'. Met een functie hoef je dezelfde coderingsregels slechts eenmaal in te voeren.

ZIE OOK
Gekke zinnen 132–133 ›
Variabelen en functies 138–139 ›

Handige functies

Python kent veel handige functies voor het uitvoeren van bepaalde taken. Als een functie wordt opgeroepen, haalt Python de codering voor die functie op en voert hem vervolgens uit. Als de functie afgelopen is, keert het programma terug naar de coderingsregel om het volgende commando uit te voeren.

print()

△ **'print()'-functie**
Via deze functie verstuurt het programma output naar de gebruiker door instructies of resultaten te printen op het scherm.

input()

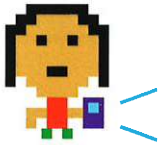
△ **'input()'-functie**
Deze functie is het tegenovergestelde van de 'print()'-functie. De gebruiker kan er instructies of data aan het programma mee geven.

randint()

△ **'randint()'-functie**
Deze functie geeft een willekeurig getal (net als een dobbelsteen gooien). Je kunt er een element of wijziging in een programma mee toevoegen.

Functies maken en oproepen

De functies die in Python zitten zijn niet de enige die je kunt gebruiken. Als je een nieuwe functie wilt maken, verzamel je de codering die je wilt gebruiken in een speciale 'wrapper' en geef je hem een naam. Met deze naam kan de functie worden opgeroepen.



1 Een functie definiëren
In de definitie van een functie zullen altijd 'def' en de naam van de functie aan het begin van de codering voorkomen.

```
def greeting():
    print('Hello!')
```

Een dubbele punt markeert het einde van de functienaam en het begin van de codering erin

Dit is de codering binnen de functie

2 De functie oproepen
Door de naam van de functie gevolgd door haken in te typen in het Shell-venster roep je de functie op en toon je de output.

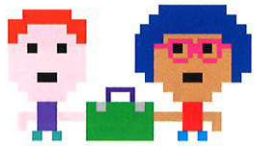
```
>>> greeting()
Hello!
```

De functie 'greeting' wordt opgeroepen en de output weergegeven

Aan de haken kun je zien dat dit een opgeroepen functie is en geen variabele

Data doorgeven aan functies

Je moet aan een functie vertellen met welke waarden hij moet werken. In 'print(a, b, c)' zijn aan de functie 'print()' de waarden 'a', 'b' en 'c' doorgegeven. In 'height(1, 45)' zijn de waarden 1 en 45 doorgegeven aan de functie 'height'.



1 Parameters toevoegen aan de functie
Waarden die aan een functie worden doorgegeven, worden 'parameters' genoemd. Parameters worden binnen de haken gezet naast de functienaam in zijn definitie.

```
def height(m, cm):
    total = (100 * m) + cm
    print(total, 'cm tall')
```

'm' en 'cm' zijn de parameters

Om te kunnen werken met het totaal in 'cm' moet de waarde van 'm' worden vermenigvuldigd met 100 (want 1 m = 100 cm)

Print de waarde van 'total' gevolgd door 'cm tall'

2 Waarden worden gedefinieerd
De codering binnen de functie gebruikt de waarden die eraan zijn doorgegeven.

Roept de functie op om het antwoord te geven wanneer 'm' = 1 en 'cm' = 45

```
>>> height(1, 45)
145 cm tall
```

Toont dat 1 m 45 cm gelijk is aan 145 cm

Data terugkrijgen van functies

Functies zijn het bruikbaarst als ze enkele data terugsturen naar het programma – een retourwaarde. Hiertoe voeg je 'return' toe, gevolgd door de waarde die moet worden teruggestuurd.

1 Een functie die een getal retourneert definiëren
De functie 'input()' stuurt een string altijd terug, zelfs als er een getal is ingevoerd. De nieuwe functie hieronder stuurt in plaats daarvan een getal terug.

```
def num_input(prompt):
    typed = input(prompt)
    num = int(typed)
    return num

a = num_input('Enter a')
b = num_input('Enter b')
print('a + b =', a + b)
```

Het getal wordt opgeslagen als een string in de variabele 'typed'

Retourneert de waarde opgeslagen in de variabele

Dit converteert de string in een getal en slaat het op in de variabele 'num'

2 Getal als output
Als het programma de functie 'input' had gebruikt, zouden 'a + b' de strings '10' en '7' hebben samengevoegd tot '107'.

```
Enter a 10
Enter b 7
a + b = 17
```

'a + b' samenvoegen levert '17' op, want de functie 'num_input' retourneert getallen, geen strings

PROJECT 5

Gekke zinnen

Loops, functies en lijsten kunnen afzonderlijk voor veel verschillende taken worden gebruikt. Je kunt ze ook gezamenlijk inzetten om interessante programma's te maken die zelfs nog complexere taken kunnen uitvoeren.

ZIE OOK
- 124–125 'While' of 'terwijl'-loops
- 128–129 Lijsten
- 130–131 Functies

Gekke zinnen maken

Dit programma maakt zinnen door drie afzonderlijke woordenlijsten te gebruiken. Uit elke lijst haalt het één woord en zet die samen op een willekeurige plek in een gekke zin.

1 Zet de drie hieronder getoonde lijsten in een nieuw Code-venster. Dit definieert de lijsten waarmee de zinnen worden gemaakt.

De enkele aanhalingstekens verraden dat elk item in de lijst een string is

De vierkante haken geven aan dat dit een lijst is

```
name = ['Neha', 'Lee', 'Sam']
verb = ['buys', 'rides', 'kicks']
noun = ['lion', 'bicycle', 'plane']
```

2 Elke zin is opgebouwd uit woorden die willekeurig zijn gekozen uit de lijsten die je hebt aangemaakt. Definieer een functie hiertoe, want die zal vaker in dit programma worden gebruikt.

Hiermee wordt de functie geladen om een willekeurig getal ('randint') te genereren

Zoekt uit hoeveel woorden er in de lijst staan (de functie werkt voor lijsten van elke lengte)

Kiest een willekeurig getal dat verwijst naar een van de items uit de lijst

Slaat het woord op dat willekeurig uit de variabele 'word_picked' is gekozen

```
from random import randint
def pick(words):
    num_words = len(words)
    num_picked = randint(0, num_words - 1)
    word_picked = words[num_picked]
    return word_picked
```


3 Print een willekeurige gekke zin door de functie 'pick' eenmaal uit te voeren voor elk van de drie lijsten. Gebruik het commando 'print' om de zin op het scherm te zien.

Voeg een 'a' toe, zodat de zin logisch is (zie hieronder)

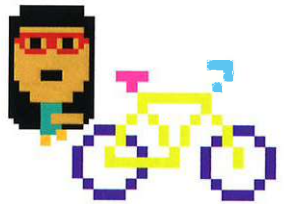
Hiermee voeg je een punt toe aan het einde, terwijl je met '\n' een nieuwe regel begint

```
print(pick(name), pick(verb), 'a', pick(noun), end='.\n')
```

4 Sla het programma op en voer het uit om een gekke zin te maken uit de lijst met namen, werkwoorden en zelfstandige naamwoorden.

```
Neha kicks a bicycle.
```

De zin wordt elke keer als het programma begint willekeurig samengesteld



Eeuwig gekke zinnen!

Je kunt een eeuwige loop toevoegen aan het gekkezinnenprogramma om het eeuwig te laten draaien of totdat je 'Ctrl-C' indrukt om de loop af te breken.

1 Het programma blijft gekke zinnen printen als het commando 'print' wordt verpakt in een 'while True'-loop.

Verpakt het printcommando in een loop

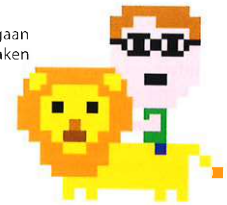
Print een nieuwe zin telkens als de entertoets wordt ingedrukt

```
while True:
    print(pick(name), pick(verb), 'a', pick(noun), end='.')
    input()
```

2 De functie 'input()' wacht met het printen van een nieuwe zin totdat de gebruiker op Enter drukt. Zonder dit zou de functie ze te snel printen om te kunnen lezen.

Het programma blijft doorgaan met willekeurige zinnen maken

```
Sam rides a lion.
Neha kicks a plane.
Lee buys a bicycle.
```



TIPS VAN EXPERTS

Leesbare codering

Het is heel belangrijk om een programma te schrijven dat eenvoudig wordt begrepen. Daardoor kun je het in de toekomst makkelijk veranderen; je hoeft immers niet opnieuw te beginnen met uitzoeken hoe het werkt!