for-lussen

Een for lus fungeert als een iterator in Python; het gaat door items die zich in een *sequence* of een ander itereerbaar item bevinden. Objecten waarover we hebben geleerd en die we kunnen herhalen, zijn onder meer strings, lijsten, tuples en zelfs ingebouwde iterables voor woordenboeken, zoals sleutels of waarden.

We hebben de for-verklaring al een beetje gezien in eerdere lezingen, maar laten we nu ons begrip formaliseren.

Hier is het algemene formaat voor een for-lus in Python:

```
for item in object:
    statements/instructies om uit te voeren
```

De variabelenaam die voor het item wordt gebruikt, is volledig aan de codeur, dus gebruik je gezond verstand om een naam te kiezen die logisch is en die je kunt begrijpen wanneer je je code opnieuw bekijkt. Naar deze item-naam kan vervolgens in uw lus worden verwezen, bijvoorbeeld als u if-instructies wilt gebruiken om controles uit te voeren.

Laten we doorgaan en verschillende voorbeelden van for-lussen doornemen met behulp van verschillende typen gegevensobjecten. We beginnen eenvoudig en bouwen later meer complexiteit op.

Voorbeeld 1

Een lijst itereren

```
# We zullen leren hoe we dit soort lijst kunnen automatiseren in de volgende lezing
list1 = [1,2,3,4,5,6,7,8,9,10]

for num in list1:
    print(num)
```

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

Geweldig! Hopelijk heeft dit zin. Laten we nu een if-statement toevoegen om te controleren op even getallen. We introduceren hier eerst een nieuw concept: de modulo.

Modulo

De modulo stelt ons in staat om de rest in een deling te krijgen en gebruikt het %-symbool. Bijvoorbeeld:

```
17 % 5
```

```
2
```

Dit is logisch aangezien 17 gedeeld door 5 3 en de rest 2 is. Laten we nog een paar snelle voorbeelden bekijken:

```
# 3 rest 1
10 % 3
```

```
1
```

```
# 3 rest 1
18 % 7
```

```
4
```

```
# 2 geen rest
4 % 2
```

```
0
```

Merk op dat als een getal volledig deelbaar is zonder rest, het resultaat van de modulo-aanroep (4 % 2) 0 is. We kunnen dit gebruiken om te testen op even getallen, want als een getal modulo 2 gelijk is aan 0, betekent dit dat het een even getal is!

Terug naar de for loops!

Voorbeeld 2

Laten we alleen de even getallen uit die lijst afdrukken!

```
for num in list1:
    if num % 2 == 0:
        print(num)
```

```
2
4
6
8
10
```

We hadden daar ook een else-statement kunnen plaatsen:

```
for num in list1:
    if num % 2 == 0:
        print(num)
    else:
        print('Odd number')
```

```
Odd number
2
Odd number
4
Odd number
6
Odd number
8
Odd number
10
```

Voorbeeld 3

Een ander veelvoorkomend idee tijdens een for-lus is het bijhouden van een aantal loops tijdens meerdere lussen. Laten we bijvoorbeeld een for-lus maken die de lijst samenvat:

```
# Start sum at zero
list_sum = 0

for num in list1:
    list_sum = list_sum + num

print(list_sum)
```

```
55
```

Geweldig! Lees de bovenstaande cel door en zorg ervoor dat u volledig begrijpt wat er aan de hand is. We hadden ook een += kunnen implementeren om de optelling bij de som uit te voeren. Bijvoorbeeld:

```
# Start sum at zero
list_sum = 0

for num in list1:
    list_sum += num

print(list_sum)
```

```
55
```

Voorbeeld 4

We hebben for loops gebruikt met lijsten, wat dacht je van met strings? Onthoud dat strings een reeks zijn, dus als we er doorheen gaan, hebben we toegang tot elk item in die string.

```
for letter in 'This is a string.':
    print(letter)
```

```
T
h
i
s

i
s

a

s
t
r
i
n
g
.
```

Voorbeeld 5

Laten we nu kijken hoe een for-lus kan worden gebruikt met een tuple:

```
tup = (1,2,3,4,5)

for t in tup:
    print(t)
```

```
1
2
3
4
5
```

Voorbeeld 6

Tuples hebben een unieke kwaliteit als het gaat om for loops. Als je wilt een reeks itereren die tuples bevat, kan het item de tuple zelf zijn, dit is een voorbeeld van *tuple unpacking*. Tijdens de for-lus zullen we de tuple in een reeks uitpakken en hebben we toegang tot de afzonderlijke items in die tuple!

```
list2 = [(2,4),(6,8),(10,12)]
```

```
for tup in list2:
    print(tup)
```

```
(2, 4)
(6, 8)
(10, 12)
```

```
# Nu met uitpakken!
for (t1,t2) in list2:
    print(t1)
```

```
2
6
10
```

Koel! Met tupels in een reeks hebben we toegang tot de items erin door ze uit te pakken! De reden dat dit belangrijk is, is omdat veel objecten hun iterables via tuples zullen leveren. Laten we beginnen met het verkennen van iteratie door dictionaries om dit verder te onderzoeken!

Voorbeeld 7

```
d = {'k1':1,'k2':2,'k3':3}
```

```
for item in d:
    print(item)
```

```
k1
k2
k3
```

Merk op hoe dit alleen de sleutels produceert. Dus hoe kunnen we de waarden (values) krijgen? Of zowel de sleutels (keys) als de waarden?

We gaan drie nieuwe Dictionary-methoden introduceren: **.keys()**, **.values()** en **.items()**

In Python retourneert elk van deze methoden een *dictionary view-object*. Het ondersteunt bewerkingen zoals lidmaatschapstest (membership-test) en iteratie, maar de inhoud ervan is niet onafhankelijk van het originele woordenboek - het is slechts een weergave. Laten we het in actie zien:

```
# Create a dictionary view object
d.items()
```

```
dict_items([('k1', 1), ('k2', 2), ('k3', 3)])
```

Sindsdien ondersteunt de .items()-methode iteratie, we kunnen *dictionaries uitpakken* uitvoeren om sleutels en waarden te scheiden, net zoals we deden in de vorige voorbeelden.

```
# Dictionary unpacking
for k,v in d.items():
    print(k)
    print(v)
```

```
k1
1
k2
2
k3
3
```

Als u een echte lijst met keys, values of key/value-tuples wilt verkrijgen, kunt u de weergave *casten* als een lijst:

```
list(d.keys())
```

```
['k1', 'k2', 'k3']
```

Onthoud dat dictionaries ongeordend zijn en dat keys en values in willekeurige (random) volgorde terugkomen. U kunt een gesorteerde lijst verkrijgen met behulp van Sort():

```
sorted(d.values())
```

```
[1, 2, 3]
```

Gevolgtrekking

We hebben geleerd hoe we for-lussen kunnen gebruiken om door tuples, lijsten, strings en dictionaries te itereren. Het zal een cruciaal hulpmiddel voor ons zijn, dus zorg ervoor dat u het goed kent en de bovenstaande voorbeelden begrijpt.

[Meer bronnen] (http://www.tutorialspoint.com/python/python_for_loop.htm)