Algemeen

Toekenningsopdracht vs. gelijkheidstest

In Python maak je bij een toekenningsopdracht gebruik van één enkel gelijkteken en bij een gelijkheidstest (nagaan of twee objecten dezelfde waarde hebben) van twee opeenvolgende gelijktekens. Om te testen of de variabele `x` gelijk is aan twee schrijf je dus

```
>>> if x == 2:    # correct
...     pass
```

en niet

```
>>> if x = 2:     # verkeerd
  File "<myscript.py>", line 1
    if x = 2:
         ^
SyntaxError: invalid syntax
```

Lichaamstemperatuur

Algemene info

Opmerkingen

Nauwkeurige definitie van het getal e

Een nauwkeurige definitie van het getal e vind je terug in de `math` module.

```
>>> import math
>>> math.e
2.718281828459045
```

Geboortestenen

Algemene info

Vermijden om meerdere `print` statements te gebruiken

Het is altijd een goed idee om ervoor te zorgen dat je programma maar één print statement bevat. Zo is het makkelijker om een overzicht te behouden van wat er allemaal uitgeprint kan worden.

Stel bijvoorbeeld dat je volgende code hebt

```
>>> x = 3
>>> if x < 5:
...     print('kleiner dan 5')
... else:
...     print('groter dan 5')
...
'kleiner dan 5'
```

Dan herschrijf je dit beter naar

```
>>> x = 3
>>> if x < 5:
...     resultaat = 'kleiner'
... else:
...     resultaat = 'groter'
...
>>> print('{} dan 5'.format(resultaat))
'kleiner dan 5'
```

Paardensprong

Algemene info

String opsplitsen in individuele karakters

Als je de karakters van een string van lengte n wilt toekennen aan n afzonderlijke variabelen, dan kan dit op de volgende manier.

```
>>> woord = 'ab'
>>> eerste, tweede = woord
>>> eerste
'a'
>>> tweede
'b'
>>>
>>> woord = 'abcd'
>>> eerste, tweede, derde, vierde = woord
>>> eerste
'a'
>>> tweede
'b'
>>> derde
'c'
>>> vierde
'd'
```

Dit is een voorbeeld van een meer algemene techiek die *tuple unpacking* genoemd wordt.

Letters omzetten naar hun getalwaarde

In Python heeft elk karakter een getalwaarde (integer). Zo heeft het vraagteken (`?`) als getalwaarde 63. Deze waarde kan bekomen worden met behulp van de ingebouwde functie `ord`.

```
>>> ord('?')
69
```

Het interessante aan deze getalwaarden is dat de opeenvolgende letters van het alfabet ook opeenvolgende getalwaarden hebben. Dit geldt zowel voor kleine letters als voor hoofdletters. Zo heeft de letter `a` als getalwaarde 97, waaruit we kunnen afleiden dat de letter `b` als getalwaarde 98 moet hebben. Met die wetenschap kunnen we dus op eenvoudige manier de positie van een letter in het alfabet bepalen.

```
>>> letter = 'd'
>>> ord(letter)
100
>>> ord('a')
97
>>> ord(letter) - ord('a') + 1   # d is de 4e letter van het alfabet
4
>>> ord('z') - ord('a') + 1      # z is de 26e letter van het alfabet
26
```