Algemeen

Letters omzetten naar hun getalwaarde

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

```
>>> ord('?')
63
```

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 een 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
```

Vampiergetallen

Specifieke info

In deze opgave is het de bedoeling om te controleren of twee strings dezelfde karakters bevatten. De karakters van beide strings staan echter niet gegarandeerd in dezelfde volgorde. Door de karakters in beide strings eerst te sorteren en daarna de gesorteerde strings te vergelijken, kan je makkelijk controleren of beide strings dezelfde karakters bevatten. Sorteren van een string gebeurt in Python aan de hand van de ingebouwde functie `sorted`.

Als je `sorted('cab')` aanroept, dan wordt er een nieuwe lijst (`list`) teruggegeven met de letters in alfabetische volgorde: `'a'`, `'b'` en `'c'`.

```
>>> sorted('cab')
['a', 'b', 'c']
```

Het raadsel van Nob

Verwijderen van witruimte vooraan en/of achteraan

In Python kan je gebruikmaken van de stringmethode `strip` om witruimte (spaties, tabs, regeleindes) vooraan en achteraan te verwijderen. Als je enkel de witruimte vooraan wilt verwijderen, dan kan je gebruikmaken van de stringmethode `lstrip`. Als je enkel de witruimte achteraan wilt verwijderen, dan kan je gebruikmaken van de stringmethode `rstrip`. Je kan aan deze stringmethode ook een argument meegeven, waarmee je aangeeft welke karakters er vooraan en/of achteraan moeten verwijderd worden. Voor de details hierover verwijzen we naar de Python Standard Library.

```
>>> tekst = '  Dit is een tekst  '
>>> tekst.strip()
'Dit is een tekst'
>>> tekst.lstrip()
'Dit is een tekst  '
```

```
>>> tekst.rstrip()
'  Dit is een tekst'
>>> tekst2 = '===Dit is een tekst==='
>>> tekst2.lstrip('=')
'Dit is een tekst==='
```

Uitlijnen van strings over een vast aantal posities

Je kunt in de *format specifier* van een `f`-string aangeven dat een vast aantal posities moeten gereserveerd worden voor een stuk tekst. Dit doe je de tekst uit te schrijven als een string (aangegeven door de letter `s`) en daarvoor met een natuurlijk getal aan te geven hoeveel posities er voor die string moeten gereserveerd worden.

```
>>> f"{'abc':5s}"   # 5 posities voorbehouden
'abc  '
```

Als de string langer is dan het aantal voorbehouden plaatsen, dan wordt de volledige string uitgeschreven en wordt er dus meer plaats ingenomen dan voorbehouden. Als de string echter korter is dan het aantal voorbehouden plaatsen, dan wordt de string standaard links uitgelijnd. Je kunt in de *format specifier* echter ook de uitlijning aangeven: links, rechts of gecentreerd. Python zal dan automatisch het juiste aantal spaties vooraan en/of achteraan toevoegen.

```
>>> f"{'abc':<5s}"   # 5 posities voorbehouden, links uitlijnen
'abc  '
>>> f"{'abc':>5s}"   # 5 posities voorbehouden, rechts uitlijnen
'  abc'
>>> f"{'abc':^5s}"   # 5 posities voorbehouden, centreren
' abc '
```

Als er moet gecentreerd worden, en het aantal toe te voegen spaties is oneven, dan kiest Python ervoor om rechts één spatie meer toe te voegen dan links.

Herhaling van een string

Om een bepaalde string een vast aantal keer te herhalen, kan je de string vermenigvuldigen met een natuurlijk getal. Net zoals bij de vermenigvuldiging van getallen gebruik je hiervoor de operator `*`. De volgorde van de string en het natuurlijk getal in de vermenigvuldiging speelt geen rol

```
>>> 'a' * 3
'aaa'
>>> 5 * 'ab'
'abababababab'
```

Dit is vooral handig als het aantal herhalingen van een string niet op voorhand vastligt, maar bijvoorbeeld zit opgeslagen in een variabele.

```
>>> herhalingen = 4
>>> herhalingen * 'abc'
'abcabcabcabc'
```