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
```

Suskewiet

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==='
```

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'
```

Stringmethoden `islower()` en `isupper()`

De stringmethode `islower` kijkt of *alle* tekens van een string kleine letters zijn. De stringmethode `isupper` controleert of *alle* letters van een string hoofdletters zijn.

```
>>> bool('ABCD'.islower())
False
>>> bool('ABCD'.isupper())
True
>>> bool('abcd'.islower())
True
>>> bool('abcd'.isupper())
False
>>> bool('AbCd'.islower())
False
>>> bool('AbCd'.isupper())
False
```

Specifieke info

Bij deze opgave is het belangrijk dat je het probleem opdeelt in kleinere deelproblemen die je elk afzonderlijk oplost (verdeel-en-heers-principe). Voor iedere string die je moet verwerken, voer je best het volgende stappenplan uit:

- verwijder alle karakters die geen letter zijn
- controleer of de string eindigt op `suskewiet`
- controleer of het eerste deel alleen maar bestaat uit de letters `ktreu`

In eerste instantie kan je het aantal correcte liedjes bijhouden in een variabele. Daarna kan je dan aan de hand van die variabele de uitvoer genereren in het aangegeven formaat.

Woordevoluties

De stringmethode `index`

De stringmethode `index` kan gebruikt worden om de meest linkse positie van een karakter in een string te bepalen.

```
>>> 'mississippi'.index('i')
1
```

Dit codefragment vindt dus het eerste voorkomen van de letter `i` op positie 1 in de string `mississippi`. Let hierbij op het feit dat Python de posities van de karakters in een string indexeert vanaf 0, en niet vanaf 1.

Aan de stringmethode `index` kan ook nog een tweede argument doorgegeven worden, dat aangeeft vanaf welke positie moet gezocht worden naar een eerstvolgende voorkomen van het karakter dat als eerste argument werd doorgegeven. Als je bijvoorbeeld weet dat er op positie 1 van de string `mississippi` een letter `i` staat, dan kan je de tweede positie waarop een letter `i` vinden door te beginnen zoeken vanaf positie 2:

```
>>> 'mississippi'.index('i', 2)
4
```