Mathemagisch

Uitlijnen van strings over een vast aantal posities

Je kunt in de *format specifier* van de stringmethod `format` aangeven dat voor een stuk tekst een vast aantal posities moeten gereserveerd worden. 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 expliciet 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 '
```

Indien 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.

Itereren over elementen én hun posities van sequentietypes

De ingebouwde functie `enumerate` kan gebruikt worden om een een iterator op te vragen van een sequentietype (*sequence type*: strings, lijsten, tuples, bestanden, ...), die zowel de positie als de waarde van het volgende element van het type teruggeeft. Onderstaand voorbeeld geeft bijvoorbeeld aan hoe je tegelijkertijd de posities en de karakters op die posities kunt overlopen voor een string.

```
>>> for index, karakter in enumerate('abc'):
...     print(f'index: {index}')
...     print(f'karakter: {karakter}')
...
index: 0
karakter: a
index: 1
karakter: b
index: 2
karakter: c
```

Je kan dit bijvoorbeeld gebruiken als je synchroon de karakters van twee strings wil overlopen.

```
>>> eerste = 'abc'
>>> tweede = 'def'
>>> for index, karakter in enumerate(eerste):
...    print(f'{karakter}-{tweede[index]}')
...
a-d
b-e
c-f
```

In dat geval is het echter aangewezen om gebruik te maken van de ingebouwde functie `zip`, die net haar bestaansrecht haalt uit het feit dat ze een iterator teruggeeft voor twee of meer iterateerbare objecten.

```
>>> eerste = 'abc'
>>> tweede = 'def'
>>> for karakter1, karakter2 in zip(eerste, tweede):
...     print(f'{karakter1}-{karakter2}')
...
a-d
b-e
c-f
```

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'
```

Alchemische reductie

Specifieke info

We geven je graag volgend schema mee. Hierop kan je de implementatie van jouw algoritme basseren. Alvorens dit te implementeren, is het een heel goed idee om zeker te zijn dat je het goed begrijpt. Voer daarom dit algortime eerst eens uit met pen en papier.

1. We bouwen een nieuw polymeer letter per letter op.
    1. Als de laatst letter van het nieuwe polymeer reageert met de huidige letter, dan verwijder je de laatste letter van het nieuwe polymeer.
    2. Anders voegen we de huidige letter toe aan het einde van het nieuwe polymeer.
2. Het nieuwe polymeer is de meest vereenvoudigde weergave.

Algemeen

De stringmethode `index`

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

```
>>> string = 'mississippi'
>>> teken = 'i'
>>> string.index(teken)
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.

Het valt wel op te merken dat deze methode niet altijd zeer efficiënt werkt. Om de positie te vinden van een karakter dat voor het eerst opduikt op plaats p, moet de functie de eerste $p - 1$ posities van de string doorlopen. Voor een karakter dat niet voorkomt in de string moet de stringmethode `index` zelfs de volledige string aflopen vooraleer er kan vastgesteld worden dat het karakter er niet gezien werd.