Python: 8. Werken met tekst

Dr. Cornelis Stal

April 27, 2022

1 Werken met tekst

1.1 Het escape-teken

String hebben we in eerdere secties al gezien als een speciaal datatype, waarbij karakters worden opgeslagen in een geordende lijst. Ieder element heeft hierbij een welbepaalde index. Daarnaast hebben we ook gezien dat een aantal speciale tekens gebruikt werden in de syntax van Python code, zoals de tab en het enkele of dubbele aanhalingsteken.

Speciale tekens in String voorafgaan door \, ofwel het escape-karakter, zoals geïllustreerd in onderstaande tabel:

Notatie	Afgedrukt als
\'	Enkel aanhalingsteken
\"	Dubbel aanhalingsteken
\t	Tab
\n	Nieuwe regel
\	Backslash

Door middel van een combinatie van het `escape`-teken met de letter `n` kan een zin bijvoorbeeld opgedeeld worden in verschillende regels:

[ ]:
```
# Het escape teken om tekst regel per regel af te drukken
print("Hallo!\nDit is een zin die regel per regel wordt weergegeven\nDankzij de␣
↪'\n'")
```

1.2 Strings benaderen als lijsten

De karakters binnen een string kunnen één voor één overlopen worden zoals de elementen in een lijst. Ook kunnen onderdelen van een string geselecteerd worden, wat resulteert in een substring:

[ ]:
```
# String behandelen als een lijst om substrings of karakters op te vragen
hw = 'Hello world!'
print(hw[0])
print(hw[4])
print(hw[-1])
print(hw[0:5])
print(hw[:5])
```

```
print(hw[6:])
```

1.3 Strings omzetten naar hoofdletters en kleine letters

Tekst omzetten van kleine letters naar hoofdletters en omgekeerd gebeurt met de respectievelijke functies `upper` en `lower`:

[ ]:
```
strText = 'Hoeken en afstanden meten'
# Originele tekst afdrukken
print(strText)
# Tekst afdrukken in hoofdletters
print(strText.upper())
# Tekst weergeven in kleine letters
print(strText.lower())
```

De conversie van hoofdletter naar kleine letters is nuttig om bijvoorbeeld simpele teststandaardisatie uit te voeren:

[ ]:
```
# Standaardisatie van een ingegeven tekst
city = input('In welke stad is de HOGENT?')
if city.lower() == 'gent':\
    print('Correct, de HOGENT is in Gent')
else:
    print('Fout antwoord, de HOGENT is in Gent')
```

1.4 Bevragingen op strings

Om te controleren of een gegeven string al dan niet onderdeel uitmaakt van andere string behoord, gebruikt men respectievelijk `in` en `not in`. Let hierbij op voor het feit dat dergelijke bevragingen hoofdlettergevoelig zijn:

[ ]:
```
# Onderzoeken of een substring aanwezig is in een string
strText = 'Hoeken en afstanden meten'
print('Hoeken' in strText)
print('Afstanden' in strText)
print('METEN' in strText)
print('' in strText)
print('meten' not in strText)
```

Op een `string`-object kan een behoorlijk aantal functies toegepast worden. Enkele van de meest gebruikten worden hieronder gegeven. Deze methodes geven allemaal een `True` of `False` terug. Controleren of een string in hoofdletters of kleine letters staat kan bijvoorbeeld met respectievelijk de functies `isupper'` en `islower`:

[ ]:
```
# Staat een tekst in hoofdletters of in kleine letters?
strText = 'Hoeken en afstanden meten'
print(strText.isupper())
print(strText.upper().isupper())
```

```
print(strText.lower().islower())
```

Om te controleren of een string bestaat uit letters, een combinatie van cijfers of een combinatie van beide gebruiken we respectievelijk de functies `isalpha`, `isdigit` en `isalnum`:

[ ]:
```
# Bestaat een gegeven tekst enkel uit letters
print('hallo'.isalpha())
print('hallo123'.isalpha())
# Bestaat een gegeven tekst enkel uit cijfers
print('123'.isdigit())
# Bestaat een gegeven tekst uit alfanumerieke tekens
print('hallo123'.isalnum())
print('hallo'.isalnum())
```

Controleren of een string enkel bestaat uit witruimtes gebeurt met de functie `isspace`. De `istitle`-functie gebruiken we om te controleren of dat ieder woord start met een hoofdletter:

[ ]:
```
# Controleren op witruimte en start van ieder woord met een hoofdletter
print('Hoeken en afstanden meten '[-1].isspace())
print('Hoeken En Afstanden Meten'.istitle())
print('Hoeken en afstanden meten'.title().istitle())
print('Hoeken EN Afstanden Meten'.istitle())
print('Hoeken en Afstanden Meten'.istitle())
```

Controleren of een string begint (`startswith`-functie) of eindigt (`endswith`-functie) met een gegeven substring:

[ ]:
```
# Begint of eindigt een string met een gegeven substring?
print('Hoeken en afstanden meten '.startswith('Hoek'))
print('Hoeken en afstanden meten '.endswith(' '))
print('Hoeken en afstanden meten '.startswith('hoeken'))
print('Hoeken en afstanden meten '.endswith('meten'))
print('Hoeken en afstanden meten '.startswith('Hoeken en afstanden meten '))
print('Hoeken en afstanden meten '.endswith('Hoeken en afstanden meten '))
```

1.5 Strings samenvoegen of splitsen

Strings in een lijst kunnen samengevoegd worden tot een nieuwe string met de `join`-functie, of juist opgedeeld met de `split`-functie. Voor het splitsen worden standaard witruimtes gebruikt. Een vrije separator is ook mogelijk:

[ ]:
```
# Strings samenvoegen en splitsen
print(', '.join(['theodoliet', 'totaalstation',        'GNSS']))
print(' '.join(['theodoliet', 'totaalstation',        'GNSS']))
print(' :) '.join(['theodoliet', 'totaalstation',        'GNSS']))
print('Hoeken en afstanden meten'.split())
print('Hoeken%20en%20afstanden%20meten'.split('%20'))
```

Opmerking: de combinatie `%20` is een gecodeerd HTML-URL teken voor een spatie. Waar we met Python gebruik maken van het escape-teken, zal een URL altijd gecodeerd worden volgens de ASCII-standaard, waarbij deze speciale tekens een cijfercode hebben, voorafgegaan voor een `%`-teken. Meer informatie is hier terug te vinden.

Voor het samenvoegen van twee losse strings kan ook simpelweg het `+`-teken gebruikt worden:

```
[ ]: # Samenvoegen van tekst met het "+"-teken
strText = 'Hoeken en ' + 'afstanden meten'
print(strText)
```

1.6 Enkele nuttige functies ter afronding

In onderstaande tabel worden ter afronding van deze sectie nog enkele nuttige functies gegeven voor een vlotte werking met strings. Deze onvolledige lijst van functies zijn een aanvulling op de eerder besproken functies:

Notatie	Afgedrukt als
`capitalize()`	Converteert het eerste teken naar een hoofdletter
`count()`	Retourneert het aantal keren dat een opgegeven waarde voorkomt in een string
`find()`	Zoekt in een string naar een opgegeven waarde en retourneert de positie waar deze is gevonden. Geeft `-1` bij niet gevonden
`format()`	Maakt opgegeven waarden in een tekenreeks op
`index()`	Zoekt in een string naar een opgegeven waarde en retourneert de positie waar deze is gevonden. Geeft fout bij niet gevonden
`isalnum()`	Geeft als resultaat “True” als alle tekens in de tekenreeks alfanumeriek zijn
`isalpha()`	Geeft als resultaat “True” als alle tekens in de tekenreeks in het alfabet staan
`isdecimal()`	Geeft “True” terug als alle tekens in de tekenreeks decimale cijfers zijn
`isdigit()`	Geeft als resultaat “True” als alle tekens in de tekenreeks cijfers zijn
`replace()`	Retourneert een tekenreeks waarin een opgegeven waarde wordt vervangen door een andere waarde
`rfind()`	Zoekt in een string naar een opgegeven waarde en retourneert de laatste positie van waar deze is gevonden. Geeft `-1` bij niet gevonden
`rindex()`	Zoekt in een string naar een opgegeven waarde en retourneert de laatste positie van waar deze is gevonden. Geeft fout bij niet gevonden
`rsplit()`	Splits de tekenreeks op het opgegeven scheidingsteken en retourneert een lijst
`rstrip()`	Retourneert een rechter trimversie van de tekenreeks
`split()`	Splits de tekenreeks op het opgegeven scheidingsteken en retourneert een lijst
`strip()`	Retourneert een bijgesneden versie van de tekenreeks
`swapcase()`	Wisselt tussen hoofdletters, kleine letters worden hoofdletters en vice versa

1.7 Oefeningen

Opdracht: morsecode

In morsecode wordt elk karakter van een woord vertaald naar een unieke opeenvolging van punten (.) en streepjes (-), en worden de vertalingen van de individuele karakters

telkens van elkaar gescheiden door één enkele spatie. Voor letters en cijfers worden de volgende punten en streepjes gebruikt:

```
{'A': '.-', 'B': '-...', 'C': '-.-.', 'D': '-..', 'E': '.', 'F': '..-.',
'G': '--.',  'H': '....', 'I': '..', 'J': '.---', 'K': '-.-', 'L':
'.-..', 'M': '--', 'N': '-.', 'O': '---', 'P': '.--.', 'Q': '--.-',
'R': '.-.', 'S': '...', 'T': '-', 'U': '..-', 'V': '...-', 'W': '.--',
'X': '-..-', 'Y': '-.--', 'Z': '--..', '0': '-----', '1': '.----', '2':
'..---', '3': '...--', '4': '....-', '5': '.....', '6': '-....', '7':
'--...', '8': '---..', '9': '----.'}
```

Schrijf een programma waarmee een gebruiker een willekeurige tekst op kan geven, die vervolgens omgezet wordt in morsecode. Tekens worden geschreiden door een spatie.

*Optioneel kan gebruik gemaakt worden van de `winsound`-bibliotheek of het '`/a`'-teken om de punten en strepen om te zetten in geluiden. Dit werkt echter niet altijd!

[ ]:
```
## UW CODE HIER ##
```