Python les-materialen Yilmaz Mustafa, Instructeur Java/Python

# Strings

Strings worden in Python gebruikt om tekstinformatie, zoals namen, vast te leggen. Strings in Python zijn een *reeks (sequencies)*, wat betekent dat Python elk element in de string als een sequence bijhoudt. Python begrijpt bijvoorbeeld dat de tekenreeks “hallo” een reeks letters in een specifieke volgorde is. Dit betekent dat we indexering kunnen gebruiken om letters te pakken (zoals de eerste letter of de laatste letter).

Dit idee van een reeks is een belangrijk idee in Python en we zullen er later in de toekomst op terugkomen.

In deze lezing leren we over het volgende:

1.) Strings maken  
 2.) Strings afdrukken  
 3.) Stringindexering en slicen  
 4.) Eigenschappen van String  
 5.) Stringmethoden  
 6.) Formatteren van String

## Een string maken

Om een string in Python te maken, moet je enkele aanhalingstekens (quote) of dubbele aanhalingstekens gebruiken. Bijvoorbeeld:

# Enkel woord  
'hello'

'hello'

# Gehele zin   
'This is also a string'

'This is also a string'

# We kunnen ook dubbele aanhalingstekens (double quote) gebruiken  
"String built with double quotes"

'String built with double quotes'

# Wees voorzichtig met aanhalingstekens (quotes)!  
' I'm using single quotes, but this will create an error'

Input In [2]  
 ' I'm using single quotes, but this will create an error'  
 ^  
SyntaxError: unterminated string literal (detected at line 2)

De reden voor de bovenstaande fout is dat het enkele aanhalingsteken in I’m de tekenreeks heeft gestopt. U kunt combinaties van dubbele en enkele aanhalingstekens gebruiken om de volledige verklaring te krijgen.

"Now I'm ready to use the single quotes inside a string!"

"Now I'm ready to use the single quotes inside a string!"

Laten we nu leren over het afdrukken van snaren!

## Een string afdrukken

Als u Jupyter-notebook gebruikt met alleen een tekenreeks in een cel, worden automatisch tekenreeksen uitgevoerd, maar de juiste manier om tekenreeksen in uw uitvoer weer te geven, is door een afdrukfunctie te gebruiken.

# We kunnen eenvoudig een string declareren  
'Hello World'

'Hello World'

# Merk op dat we op deze manier niet meerdere strings kunnen uitvoeren  
'Hello World 1'  
'Hello World 2'

'Hello World 2'

We kunnen een print statement gebruiken om een string af te drukken.

print('Hello World 1')  
print('Hello World 2')  
print('Use \n to print a new line')  
print('\n')  
print('See what I mean?')

Hello World 1  
Hello World 2  
Use   
 to print a new line  
  
  
See what I mean?

## Basis van String

We kunnen ook een functie genoemd len() gebruiken om de lengte van een string te controleren!

len('Hello World')

11

De ingebouwde len()-functie van Python telt alle tekens in de String, inclusief spaties en interpunctie (punctuation).

## Indexeren van String

We weten dat strings een sequences zijn, wat betekent dat Python indexen kan gebruiken om delen van de sequences aan te roepen. Laten we leren hoe dit werkt.

In Python gebruiken we vierkantehaakjes [] na een object om zijn index aan te roepen. We moeten ook opmerken dat indexering begint bij 0 (zero) voor Python. Laten we een nieuw object maken met de naam s en dan een paar voorbeelden van indexeren doornemen.

# Assign s as a string  
s = 'Hello World'

#Check  
s

'Hello World'

# Print the object  
print(s)

Hello World

Laten we beginnen met indexeren!

# Toon eerste element (in dit geval een letter)  
s[0]

'H'

s[1]

'e'

s[2]

'l'

We kunnen een : gebruiken om *slicing* uit te voeren die alles tot op een bepaald punt pakt. Bijvoorbeeld:

# Grijp alles voorbij de eerste term helemaal tot de lengte van s die len (en) is  
s[1:]

'ello World'

# Merk op dat er geen verandering is in de originele s  
s

'Hello World'

# Grijp alles TOT de 3e index  
s[:3]

'Hel'

Let op het bovenstaande slicen. Hier vertellen we Python om alles van 0 tot 3 te pakken. Het bevat niet de 3e index. Je zult dit veel merken in Python, waar statements en meestal in de context van “tot, maar niet inclusief” staan.

#Everything  
s[:]

'Hello World'

We kunnen ook negatieve indexering gebruiken om achteruit te gaan.

# Laatste letter (één index achter 0 zodat het terug rondloopt)  
s[-1]

'd'

# Grijp alles behalve de laatste letter  
s[:-1]

'Hello Worl'

We kunnen ook index- en plaknotatie gebruiken om elementen van een reeks te pakken met een gespecificeerde stapgrootte (de standaardinstelling is 1). We kunnen bijvoorbeeld twee dubbele punten achter elkaar gebruiken en vervolgens een getal dat de frequentie aangeeft om elementen te pakken. Bijvoorbeeld:

# Grijp alles, maar lees in stappen van 1  
s[::1]

'Hello World'

# Grijp alles, maar lees in stapgroottes van 2  
s[::2]

'HloWrd'

# We kunnen dit gebruiken om een string achterstevoren (backwards) af te drukken  
s[::-1]

'dlroW olleH'

## Eigenschappen van String

Het is belangrijk op te merken dat strings een belangrijke eigenschap hebben die bekend staat als *onveranderlijkheid*. Dit betekent dat als een string eenmaal is gemaakt, de elementen erin niet meer kunnen worden gewijzigd of vervangen. Bijvoorbeeld:

s

'Hello World'

# Laten we proberen de eerste letter te veranderen in 'x'  
s[0] = 'x'

---------------------------------------------------------------------------  
  
TypeError Traceback (most recent call last)  
  
<ipython-input-26-976942677f11> in <module>()  
 1 # Let's try to change the first letter to 'x'  
----> 2 s[0] = 'x'  
  
  
TypeError: 'str' object does not support item assignment

Merk op hoe de fout ons direct vertelt wat we niet kunnen doen, verander de itemtoewijzing!

Iets wat we *kunnen* doen, is strings samenvoegen!

s

'Hello World'

# Voeg strings samen!  
s + ' concatenate me!'

'Hello World concatenate me!'

# We kunnen s echter volledig opnieuw toewijzen!  
s = s + ' concatenate me!'

print(s)

Hello World concatenate me!

s

'Hello World concatenate me!'

We kunnen het vermenigvuldigingssymbool gebruiken om herhaling te creëren!

letter = 'z'

letter\*10

'zzzzzzzzzz'

## Basis Ingebouwde String-methoden

Objecten in Python hebben meestal ingebouwde methoden. Deze methoden zijn functies binnen het object (we zullen hier later veel dieper op ingaan) die acties of opdrachten op het object zelf kunnen uitvoeren.

We noemen methoden met een punt en dan de naam van de methode. Methoden zijn in de vorm:

object.methode(parameters)

Waar parameters extra argumenten zijn, kunnen we doorgeven aan de methode. Maak je geen zorgen als de details op dit moment niet 100% kloppen. Later gaan we onze eigen objecten en functies maken!

Hier zijn enkele voorbeelden van ingebouwde methoden in strings:

s

'Hello World concatenate me!'

# Hoofdletters een string  
s.upper()

'HELLO WORLD CONCATENATE ME!'

# Kleine letters  
s.lower()

'hello world concatenate me!'

# Splits een string door spatie (dit is de standaard)  
s.split()

['Hello', 'World', 'concatenate', 'me!']

# Gesplitst door een specifiek element (omvat niet het element waarop is gesplitst)  
s.split('W')

['Hello ', 'orld concatenate me!']

Er zijn veel meer methoden dan degene die hier worden behandeld. Bezoek de Advanced String sectie voor meer informatie!

## Formattering van de afdrukken

We kunnen de methode .format() gebruiken om opgemaakte objecten toe te voegen aan afgedrukte String-instructies.

De eenvoudigste manier om dit aan te tonen is door middel van een voorbeeld:

'Insert another string with curly brackets: {}'.format('The inserted string')

'Insert another string with curly brackets: The inserted string'

We zullen dit onderwerp voor het opmaken van strings in latere secties opnieuw bekijken wanneer we onze projecten bouwen!

## Volgende: Lijsten!