Der Datentyp str

Ein sequentieller, unveränderlicher Datentyp

„Zeichenkette“

Definition

Ein String ist eine Zeichenkette. Der Wertebereich des Datentyp Strings ist die Menge aller Zeichenketten. Strings werden in einfache oder doppelte Anführungszeichen gesetzt.

```
x = "das ist ein String"
```

Um zu prüfen, ob eine Variable vom Datentyp String ist, nutzen wir die type()-Methode.

```
type(x)
<class 'str'>
```

Zeichen als Index

Auf jedes Zeichen in einem String lässt sich durch den Eckige-Klammer-Operator zugreifen. Das erste Zeichen hat den Index 0.

```
>>> txt = "Hello World"
>>> txt[0]
'H'
>>> txt[4]
'o'
```

Negativer Index

Ist der Wert in der eckigen Klammer negativ, beginnt die Zählung von hinten. Damit ist dann der Index relativ vom Ende des Strings her gemeint.

```
>>> txt = "Hello World"
>>> txt[-1]
'd'
```

Versucht der Index eine Position außerhalb des Strings zu addressieren, wird ein IndexError geworfen

```
>>> txt[-40]
IndexError: string index out of range
```

Wert zu String konvertieren

mit der Funktion str() lassen sich Zahlen und andere Datentypen zu Strings konvertieren.

```
x = 2
type(x)
<class 'int'>


x = str(x)


type(x)
<class 'str'>
```

Länge eines Strings

Die Länge eines Strings lässt sich mit der Funktion len() ermitteln.

```
string1 = "simple eggs are great"
laenge = len(string1)
print(laenge)
```

21

String Methoden

Python bietet vielfältige Methoden, um Zeichenketten zu verarbeiten.

s = "San Francisco"

s = s.upper() => Alles in Großbuchstaben

s = s.lower() => Alles in Kleinbuchstaben

s = s.replace("a", "b") => Ersetze alle a durch b

Vorsicht!

Strings sind unveränderlich. Daher erzeugen diese String-Methoden immer einen neuen String

weitere wichtige String Methoden

count() zählt Vorkommen in einem String

startswith() prüft, ob ein String mit einer Zeichenkette beginnt

endswith() prüft, ob ein String mit einer Zeichenkette endet

find() sucht nach einem Vorkommen in einem String und gibt den Index zurück. -1 falls nicht gefunden

join() fügt alle Elemente einer Sequenz zu einem String zusammen

strip() entfernt Steuer- und Leerzeichen vorne und hinten eines Strings

replace() ersetzt ein Zeichen durch ein anderes

split() zerlegt einen String anhand eines Separators

index() sucht nach einem Vorkommen und gibt den Index zurück

format() formatiert einen String anhand Platzhaltern

String Vergleiche

Strings lassen sich mit den Vergleichsoperatoren == und != vergleichen

```
if string == string2:
    print("Strings sind gleich")

if string != string2:
    print("Strings sind ungleich")
```

Prüfen, ob ein String in einem anderen enthalten ist

Mit dem Membership-Operator in können wir prüfen, ob ein String in einem anderen String enthalten ist.

```
string = "Python"


if "yt" in string:
    print("in diesem String ist ein kleines yt vorhanden")
```

Strings formatieren mit Format-Strings

Format-Strings (Vorlagen-Zeichenketten) bieten komfortabel die Möglichkeit, Strings zu formatieren:

Der Template-String beinhaltet einen oder mehrere Platzhalter, die mit einer geschweiften Klammer definiert werden. Die String-Methode format() übergibt dem String genauso viele Variablen, wie Platzhalter definiert wurden.

```
value1 = 32.4

value2 = 22

result = "Die Ergebnisse der letzten Spiele waren {} und {}".format(value1, value2)

# Die Ergebnisse der letzten Spiele waren 32.4 und 22
```

f-String Syntax

Neben der format-Methode() kennt Python noch eine andere Methode, Strings zu formatieren. Die f-String Syntax ermöglicht es, Variablen zu nutzen und diese im String auszugeben.

```
name = 'John Doe'
age = '22'

ausgabe = f"Der Mitarbeiter {name} ist {age} Jahre alt"
```

Formatieren von Floats in Strings

Um Zahlen und Strings zu verketten, muss die Zahl vorher in einen String gewandelt werden. Dazu gibt es die Methode str()

Die Format-Methode erlaubt es uns aber, Zahlen an den Template-String zu übergeben und diese, im Falle von Fließkommazahlen entsprechend zu formatieren.

```
value = 2.2324242
str = "Die Zahl auf zwei Stellen gerundet ist: {:0.2f}".format(value)
# Die Zahl auf zwei Stellen gerundet ist 2.23
```

Slicing

Mit dem Slice-Operator können wir Teilstücke aus einem String (und jedem anderen sequentiellen Datentypen) schneiden

```
[:]
a[first_index:first_exclusive_index]  # beginnt bei Start und endet bei Stop - 1
a[start:]      # beginnt bei Start und nimmt den Rest
a[:stop]       # beginnt bei 0 und endet bei Stop - 1
a[:]          # kopiert ganzen String
```

```
String = "Hamburg"
string[0:2]
```

Ergebnis: Ha

String in Teilstrings splitten

Mit der String-Methode split() lässt sich ein String in Teil-Stücke zerlegen. Das Ergebnis ist der Datentyp List.

```
s = "a,b,c"
x = s.split(" ,")
type(x)
<class 'list'>
```