Merkblatt Strings

1 Strings und String-Operationen

Eine Zeichenkette oder ein String – so der aus dem Englischen übernommene Begriff – ist eine Aneinanderreihung von Symbolen, die auch leer sein kann.

Strings werden durch Anführungszeichen kenntlich gemacht. Mit dem Zuweisungsoperator `=` wird eine Zeichenkette einer Variablen zugewiesen:

```
a = "Hallo, Welt!"
```

Einige weitere Hinweise:

Alternativ können Strings mit einfachen Anführungsstrichen zugewiesen werden. Auch dies funktioniert:

```
a = 'Hallo, Welt!'
```

Das ist hilfreich wenn innerhalb eines Strings selber Anführungszeichen enthalten enthalten sein sollen, wie zum Beispiel hier:

```
b = 'Sie ließ sich ein kurzes "Hi" entlocken.'
```

In einer anderes Schreibweise werden Anführungsstriche mit dem Rückwärtsschrägsdtrich \ kenntlich gemacht. Folgendes wäre auch möglich:

```
b = "Sie ließ sich ein kurzes \"Hi\" entlocken."
```

Über diese Schreibweise lässt sich auch direkt ein Zeilenumbruch in einen String einfügen. **Wichtig:** Dieser wird zwar mit 2 Zeichen geschrieben (`\n`), ist aber im String nur ein Zeichen!

[2]:
```
print(len("\n"))
```

```
1
```

[3]:
```
print("Hallo\nWelt")
```

```
Hallo
Welt
```

1.1 Index und Indices

Auf ein einzelnes Zeichen eines String kann per Index zugriffen werden. Der Index beginnt bei 0 und bezeichnet damit die Position des ersten Zeichens.

[1]:
```
a = "Hallo, Welt!"
a[0]
```

[1]:
```
'H'
```

[2]:
```
a[4]
```

[2]:
```
'o'
```

Mit dieser Schreibweise können nicht nur einzelne Zeichen sondern auch Teil-Strings angesprochen werden:

[3]:
```
a[7:11]
```

[3]:
```
'Welt'
```

Wird der Start oder das Ende nicht explizit angegeben, so interpretiert Python dies als Bezeichnung für den Anfang oder das Ende der Zeichenkette:

[4]:
```
a[7:]
```

[4]:
```
'Welt!'
```

[5]:
```
a[:5]
```

[5]:
```
'Hallo'
```

Ein negativer Index beschreibt die Zeichenkette rückwärts:

[6]:
```
a[-1]
```

[6]:
```
'!'
```

[7]:
```
a[-5]
```

[7]:
```
'W'
```

1.2 Umwandlung

Andere Datentypen können mit der Funktion `str()` in eine Zeichenkette umgewandelt werden.

[8]:
```
str(42)
```

[8]:
```
'42'
```

[9]:
```
str(3.147)
```

[9]:
```
'3.147'
```

[10]:
```
str(True)
```

[10]:
```
'True'
```

1.3 Zusammenfügen

Strings können zusammengefügt werden mit dem +-Operator:

[11]:
```
a= "Hallo"+", "+"Welt"+"!"
print(a)
```

```
Hallo, Welt!
```

1.4 Finden

Die Funktion `find()` findet die (erste) Position eines Teilstrings im String:

[12]:
```
a.find("Welt")
```

[12]:
```
7
```

Die Funktion `rfind()` findet die letzte Position eines Teilstrings im String. Sie sucht *rückwärts*:

[13]:
```
a.rfind("l")
```

[13]:
```
9
```

1.5 Ersetzen

Die Funktion `replace()` ersetzt im String den ersten angegebenen Teilstring durch den zweiten angegeben Teilstring:

[14]:
```
a.replace("Welt", "Universum")
```

[14]:
```
'Hallo, Universum!'
```

Hinweis: Wird für `replace()` ein dritter Parameter angegeben, dann werden nicht alle Vorkommen des ersten Teilstrings ersetzt sondern nur so viele, wie der Parameter angibt. Im nachfolgenden Beispiel sind dies 2.

[15]:
```
b = "1 Mississippi - 2 Mississippi - 3 Mississippi"
b.replace(" Mississippi","-undzwanzig",2)
```

[15]:
```
'1-undzwanzig - 2-undzwanzig - 3 Mississippi'
```

1.6 Säubern

Manchmal ist es notwendig, Strings von Leerzeichen am Ende oder Anfang zu säubern. Das kommt häufig bei extern eingelesen Daten vor. Hierfür gibt es die Funktion `strip()`:

[16]:
```
a="    Hallo, Welt!       "
a.strip()
```

[16]:
```
'Hallo, Welt!'
```