

MODUL 9: FORMATIERUNG IN ZEICHENKETTEN

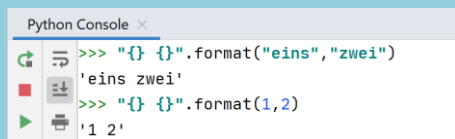
1. Kompetenzen

- Ich kann Zahlen und Daten in Zeichenketten formatieren.
- Ich kenne die gängigen Formate zur Formatierung von Zahlen und Daten in Zeichenkette.

2. Basis

Mit der Funktion `format()` einer Zeichenkette können Zeichenketten, Zahlen und andere Daten in Zeichenketten formatiert werden.

Beispiel:

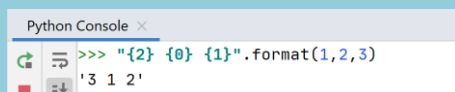


```
Python Console
>>> "{} {}".format("eins", "zwei")
'eins zwei'
>>> "{} {}".format(1, 2)
'1 2'
```

Das `' {} '` steht als **Platzhalter**, um eine Zeichenkette oder auch eine Zahl einzusetzen. Wenn mehrere `' {} '` eingesetzt werden, dann werden die Parameter der Funktion `format()` der Reihe nach durchlaufen.

Möchte man diese Reihenfolge verändern, dann kann der Index des gewünschten Wertes zwischen den geschweiften Zahlen gesetzt werden. Der Index beginnt wie bei der `range()` Funktion bei 0, d.h. das erste Element hat den Index 0, das zweite 1, usw.

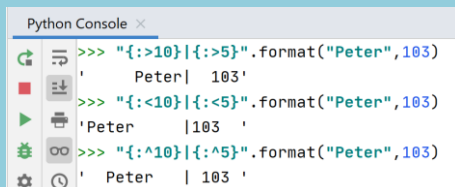
Beispiel:



```
Python Console
>>> "{2} {0} {1}".format(1, 2, 3)
'3 1 2'
```

Zeichenketten und Zahlen können rechtsbündig, linksbündig und auch zentriert ausgegeben werden. Dazu wird die gewünschte Formatierung hinter einem Doppelpunkt zwischen die geschweiften Klammern gesetzt.

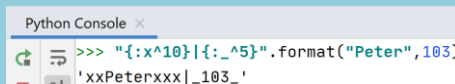
Beispiel:



```
Python Console
>>> "{:>10}|{:>5}".format("Peter", 103)
'Peter| 103'
>>> "{:<10}|{:<5}".format("Peter", 103)
'Peter |103 '
>>> "{:^10}|{:^5}".format("Peter", 103)
'Peter | 103 '
```

Beim Auffüllen kann auch das Zeichen zum Auffüllen ausgewählt werden. Wenn keins angegeben wird, dann wird das Leerzeichen benutzt. Das gewünschte Zeichen wird direkt hinter den Doppelpunkt gesetzt.¹

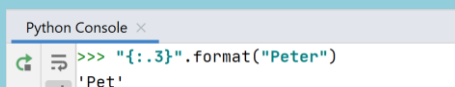
Beispiel:



```
Python Console
>>> "{:x^10}|{:_^5}".format("Peter", 103)
'xxPeterxxx|_103_'
```

Die Länge der auszugebenden Zeichenkette kann auch angegeben werden. Das macht man, indem man die gewünschte Länge hinter einem Punkt angibt. Hier ist zu beachten, dass das nur zur Formatierung von Zeichenketten zur Verfügung steht. Zahl können so nicht eingeschränkt werden.

Beispiel:



```
Python Console
>>> "{:.3}".format("Peter")
'Pet'
```

Dabei können Auffüllen und Abschneiden kombiniert werden.

¹ Alles was hinter dem Doppelpunkt steht dient der Formatierung in der Zeichenkette.

Beispiel:

```
Python Console
>>> "{0:>8.3}|{0:<8.3}|{0:^8.3}".format("Peter")
'      Pet|Pet      | Pet      '
```

3. Zahlen

Zahlen können auch mit der Funktion `format()` formatiert werden. Dazu benutzt man das Formatierungszeichen `'d'` für Ganzzahlen (Integer) und `'f'` für Kommazahlen (float). Ohne Formatierungsangaben (s. nächster Punkt), werden Kommazahlen auf 6 Stellen hinter dem Komma gekürzt.

Beispiel:

```
Python Console
>>> "{:d} {:f}".format(42, 3.141592653589)
'42 3.141593'
```

Auch können Zahlen aufgefüllt werden, dabei stehen die Zahlen rechtsbündig. Bei Kommazahlen gibt die erste Zahl die gesamte Länge der Zahl an und die zweite Zahl (hinter dem Punkt) gibt die gewünschten Stellen hinter dem Komma an.

Beispiel:

```
Python Console
>>> "{:4d} {:6.2f}".format(42, 3.141592653589)
' 42  3.14'
```

Auch das Füllzeichen kann bei Zahlen angegeben werden. Wie bei den Zeichenketten wird es direkt hinter den Doppelpunkt gesetzt.

Beispiel:

```
Python Console
>>> "{:04d} {:06.2f}".format(42, 3.141592653589)
'0042 003.14'
```

Auch das Vorzeichen der Zahl kann formatiert werden. Standardmäßig wird ein positives Vorzeichen nicht angezeigt. Das negative Vorzeichen wird immer angezeigt. Möchte man jedoch ein positives Vorzeichen auch anzeigen, dann erreicht man das indem man ein Pluszeichen hinter den Doppelpunkt setzt.

Beispiel:

```
Python Console
>>> "{:4d} {:6.2f}".format(-42, 3.141592653589)
'-42  3.14'
>>> "{:+4d} {:+6.2f}".format(-42, 3.141592653589)
'-42  +3.14'
```

Dabei kann man auch noch auswählen, wo das Vorzeichen zu stehen kommt. Automatisch (ohne andere Angabe) wird das Vorzeichen direkt vor die Zahl gesetzt. Möchte man das Vorzeichen jedoch linksbündig anordnen, dann ist das möglich:

Beispiel:

```
Python Console
>>> "{:+=4d} {:+=6.2f}".format(-42, 3.141592653589)
'- 42  + 3.14'
```

Für Zahlen gibt es jedoch auch noch weitere Formatierungsmöglichkeiten, wie zum Beispiel die Ausgabe im Hexadezimal- oder Binärsystem. Auch die wissenschaftliche Notation ist möglich.

Beispiel:

```
Python Console
>>> "{:d} x{:x} b{:b}".format(17)
'17 x11 b10001'
>>> "{:e}".format(1230120412308)
'1.230120e+12'
```

Es gibt aber noch viele andere Möglichkeiten. Die folgende Tabelle fasst die verschiedenen möglichen Formate zusammen:

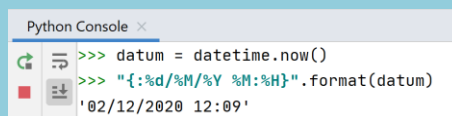
| Format | Beschreibung |
|--------|--|
| 'd' | Ganzzahl mit Vorzeichen |
| 'i' | Ganzzahl mit Vorzeichen |
| 'b' | Ganzzahl mit Vorzeichen im Binärsystem |
| 'o' | Ganzzahl mit Vorzeichen im Oktalsystem |
| 'x' | Ganzzahl mit Vorzeichen im Hexadezimalsystem (kleine Buchstaben) |
| 'X' | Ganzzahl mit Vorzeichen im Hexadezimalsystem (große Buchstaben) |

| | |
|-----|--|
| 'e' | Gleitkommazahl im wissenschaftlichen Format (kleine Buchstaben) |
| 'E' | Gleitkommazahl im wissenschaftlichen Format (große Buchstaben) |
| 'f' | Gleitkommazahl im dezimalen Format (kleine Buchstaben) |
| 'F' | Gleitkommazahl im dezimalen Format (große Buchstaben) |
| '%' | Kein Argument. Das doppelte % wird als ein % im Resultat angegeben |

4. Daten

Auch Datumsangaben können einfach formatiert werden. Dabei kann man die Formatierung frei wählen. Auch hier wird die Formatierung hinter dem Doppelpunkt angegeben.

Beispiel:



```
Python Console x
>>> datum = datetime.now()
>>> "{:d/%M/%Y %M:%H}".format(datum)
'02/12/2020 12:09'
```

Dabei gibt es noch viele weitere Optionen, wie die folgende Tabelle zeigt.

| Platzhalter | Bedeutung |
|-------------|---|
| %a | Lokale Abkürzung für den Namen des Wochentags |
| %A | Der komplette Name des Wochentags in der lokalen Sprache |
| %b | Lokale Abkürzung für den Namen des Monats |
| %B | Der vollständige Name des Monats in der lokalen Sprache |
| %c | Das Format für eine angemessene Datums- und Zeitdarstellung auf der lokalen Plattform |
| %d | Nummer des Tages im aktuellen Monat. Ergibt einen String der Länge 2 im Bereich [01,31]. |
| %H | Stunde im 24-Stunden-Format. Das Ergebnis hat immer zwei Ziffern und liegt im Bereich [00,23]. |
| %I | Stunde im 12-Stunden-Format. Das Ergebnis hat immer zwei Ziffern und liegt im Bereich [01,12]. |
| %j | Nummer des Tages im Jahr. Das Ergebnis hat immer drei Ziffern und liegt im Bereich [001, 366]. |
| %m | Nummer des Monats bestehend aus zwei Ziffern im Bereich [01,12] |
| %M | Minute als Zahl mit zwei Ziffern. Liegt immer im Bereich [00,59]. |
| %p | Die lokale Entsprechung für AM bzw. PM |
| %S | Sekunde als Zahl mit zwei Ziffern. Liegt immer im Bereich [00,61]. |
| %U | Nummer der aktuellen Woche im Jahr, wobei der Sonntag als erster Tag der Woche betrachtet wird. Das Ergebnis hat immer zwei Ziffern und liegt im Bereich [01,53]. Der Zeitraum am Anfang eines Jahres vor dem ersten Sonntag wird als 0. Woche gewertet. |
| %w | Nummer des aktuellen Tages in der Woche. Sonntag wird als 0. Tag betrachtet. Das Ergebnis liegt im Bereich [0,6]. |
| %W | Wie %U, nur dass statt des Sonntags der Montag als erster Tag der Woche betrachtet wird. |
| %x | Datumsformat der lokalen Plattform |
| %X | Zeitformat der lokalen Plattform |
| %y | Jahr ohne Jahrhundertangabe. Das Ergebnis besteht immer aus zwei Ziffern und liegt im Bereich [00,99]. |
| %Y | Komplette Jahreszahl mit Jahrhundertangabe |
| %Z | Name der lokalen Zeitzone oder ein leerer String, wenn keine lokale Zeitzone festgelegt wurde. |
| %% | Ergibt ein Prozentzeichen »%« im Resultatstring. |

5. Weitere Funktionen

Interessant ist auch, dass man den Parameterwerten Namen geben kann, mit denen sie in der Formatierung aufgerufen werden können:

Beispiel:

```
Python Console x
>>> "{vorname} {nachname} {alter}".format(vorname="Hans", nachname="Peter", alter="18")
'Hans Peter 18'
```

Dazu kann man auch Dictionaries benutzen.² Das Dictionary muss jedoch mit zwei Sternen angegeben werden ('**').

Beispiel:

```
Python Console x
>>> daten = { "vorname" : "Hans", "nachname" : "Peter", "alter" : "18" }
>>> "{vorname} {nachname} {alter}".format(**daten)
'Hans Peter 18'
```

Übung: Tabelle formatieren

Formatiere die folgenden Daten in einer Tabelle.

```
daten = [ [ "Meier", "Arnold", "M", datetime(1992,7,6), "Brüssel", 72.4, 83.2] ,
[ "Nelles", "Marita", "W", datetime(2001,4,12), "Malmedy", 62.5 , 73.2],
[ "Kalles", "Norbert", "M", datetime(1980,11,12), "Eupen", 83.2, 70.9],
[ "Trantes", "Karl-Heinz", "M", datetime(1974,2,28), "Sankt Vith", 65.3, 52.4],
[ "Müller", "Verena", "W", datetime(1982,12,24), "Eupen", 71.3, 62.5],
[ "Weynand", "Karla", "W", datetime(1988,5,5), "Lüttich", 67.7, 70.0]
]
```

Um die Daten zu durchlaufen, kannst du die folgende Schleife benutzen. Auf diese Art und Weise werden die Daten aus den einzelnen Zeilen automatisch in die Variablen nachname, vorname, ... gespeichert und können mit diesen Variablen in der Schleife genutzt werden.

```
for nachname, vorname, geschlecht, geburtsdt, geburtsort, gewicht, resultat in daten:
```

Bei dieser Übung geht es vor allem um die Ausrichtung der Daten (linksbündig, rechtsbündig oder zentriert).

Die Tabelle könnte dann so aus sehen:

| Nachname | Vorname | Geschlecht | Geburtsdatum | Geburtsort | Gewicht | Resultat |
|----------|------------|------------|------------------------|------------|---------|----------|
| Meier | Arnold | M | 06/07/1992 (Monday) | Brüssel | 72.4 | 83 |
| Nelles | Marita | W | 12/04/2001 (Thursday) | Malmedy | 62.5 | 73 |
| Kalles | Norbert | M | 12/11/1980 (Wednesday) | Eupen | 83.2 | 71 |
| Trantes | Karl-Heinz | M | 28/02/1974 (Thursday) | Sankt Vith | 65.3 | 52 |
| Müller | Verena | W | 24/12/1982 (Friday) | Eupen | 71.3 | 62 |
| Weynand | Karla | W | 05/05/1988 (Thursday) | Lüttich | 67.7 | 70 |

² Die Dictionaries werden erst in einem späteren Modul erläutert. Der Vollständigkeit halber wird diese Option aber schon hier angegeben.