.format()

Es gibt in Python 3 Möglichkeiten Strings zu formatieren bzw. Werte einzufügen:

- %-Methode
- .format-Methode
- f-string-Methode

Die f-string-Methode wurde in Python 3.6 eingeführt. Da Micropython auf Python 3.4 und Teilen von 3.5 basiert sind nur diese beiden Methoden in Micropython vorhanden. Deshalb werden wir auch nur diese beiden Methoden hier betrachten.

.format()

Die %-Methode

Diese Methode erinnert an fstring() aus C.

Die Stelle an der ein Wert eingesetzt werden soll wird im String durch das %-Zeichen gefolgt von Formatierungsanweisungen gekennzeichnet.

```
>>> a = "Hallo %s" % ("Peter")
>>> a
'Hallo Peter'
>>>

>>> b = 'Der %s kostet %f Euro' %
('Apfel', 0.25)
>>> b
'Der Apfel kostet 0.250000 Euro'

>>> b = 'Der %s kostet %.2f Euro' %
('Apfel', 0.25)
>>> b
'Der Apfel kostet 0.25 Euro'
>>>
```

.format()

Die Syntax der Platzhalter:

```
%[flags][width][.precision]type
```

Einige Platzhalter Typen:

```
s = String
d,i = Dezimalzahl mit Vorzeichen
u = ganze Zahl ohne Vorzeichen
x = hexadezimale Zahl
o = oktale Zahl
e,f = fließkomma Zahl
```

In Micropython wird das Vorzeichen nicht geprüft!

```
>>> '%i'%(42)
'42'
>>> '%d'%(42)
'42'
>>> '%u'%(42)
'42'
>>> '%u'%(-42)
'-42'
>>> '%x'%(42)
'2a'
>>> '%x'%(-42)
'-2a'
>>> '%o'%(42)
'52'
>>> '%o'%(-42)
'-52'
```

.format()

Die Syntax der Platzhalter:

```
%[flags][width][.precision]type
```

Die Flags:

```
# = Prefix der Zahl wird gezeigt
0 = Es werden Nullen aufgefüllt
- = Linksbündige Ausgabe
  = Leerzeichen statt +
+ = es wird immer ein Vorzeichen
    angezeigt
```

```
>>> '%#x'%(42)
'0x2a'
>>> '%#o'%(42)
'0o52'
>>> '%03d'%(42)
'042'
>>> '%05d'%(42)
'00042'
>>> '%05d\n%-d'%(42,42)
'00042\n42'
>>> print('%05d\n%-d'%(42,42))
00042
42
>>> print('%05d\n%-05d'%(42,42))
00042
42000 # Achtung hier gibt es ein
      # Problem aber Python hat Recht.
```

.format()

Fortsetzung Flag-Tests:

```
>>> print('%05d\n%-5d'%(42,42))
00042
42
# So ist es richtig!

>>> '% d'%(42)
' 42'
>>> '% d'%(-42)
'-42'

>>> '%+d'%(42)
'+42'
>>> '%+d'%(-42)
'-42'
```

.format()

Fortsetzung Flag-Tests:

```
>>> '%.2f'%(3.145678)
'3.15'

# Diese Formatanweisung funktioniert
# bei float wohl nicht:

>>> '%3.2f'%(3.145678)
'3.15'
>>> '%03.2f'%(3.145678)
'3.15'

# Das funktioniert aber:

>>> print('%30.2f'%(3.145678))
                          3.15
>>> print('%30.2f'%(312.145678))
                        312.15
```

Es werden keine Nullen aufgefüllt, aber in einem Feld von Vorkommastellen ausgerichtet.

.format()

Die zweite und in Micropython aktuelle Formatierungsmethode ist die **.format()**-Methode. Sie benutzt die **{}** als Ersetzungsmarkierung.

```
>>> print("Preis:
{0:3.2f}".format(27.23471))
Preis: 27.23

>>> t=22.35
>>> print('Temperatur:
{0:2.1f}'.format(t))
Temperatur: 22.4

>>> print('Temperatur:
{0:2.1f}°C'.format(t))
Temperatur: 22.4°C
>>>
```

.format()

Sie benutzt innerhalb der geschweiften Klammern die selben Platzhalter, Flags und Formatierungsanweisungen wie die %-Methode.

Allerdings wird der Formatierungsangabe die Nummer des Ersetzungswertes aus der .format() Liste vorangestellt.

```
>>> 'Hallo {0}, es lebe {1}'.format('Peter', 'Python')
'Hallo Peter, es lebe Python'

>>> 'Hallo {1}, es lebe {1}'.format('Peter', 'Python')
'Hallo Python, es lebe Python'

>>> 'Hallo {0}, es lebe {0}'.format('Peter', 'Python')
'Hallo Peter, es lebe Peter'
```

.format()

Nun formatiert mal schön :)