Beispiele zu Schleifen unter Python

In diesem Beispiel wird gezeigt, wie man mit Hilfe von Schleifen gezielt auf einzelne Buchstaben in einem String zugreifen kann.

Buchstaben aussortieren

```
import string
```

lädt das string-modul, damit alle darin enthaltenen Konstanten und Funktionen (Methoden) verwendet werden können, also beispielsweise die Konstanten

```
string.ascii_letters       → alle Buchstaben
string.ascii_lowercase     → 'abcdefghijklmnopqrstuvwxyz'
string.ascii_uppercase     → 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
string.digits              → '0123456789'
string.hexdigits           → '0123456789abcdefABCDEF'
```

Die folgende Funktion entfernt unter ihrer Zuhilfenahme aus einem string alle Zeichen außer den Klein- und Großbuchstaben.

```
import string

def entferneSonderzeichen(text):
    ergebnis=''
    for index in range(len(text)):
        if text[index] in string.ascii_letters:
            ergebnis+=text[index]
    return ergebnis
# der Test:
print( entferneSonderzeichen('Hier, -ist+ %viel^ @enthalten.') )
```

liefert:

```
Hieristvielenthalten
```

Vokale zählen

Die nächste Funktion zählt die Anzahl der Vokale in einem Text, nachdem es den Text zunächst komplett in Großbuchstaben verwandelt hat.

```
def zaehleVokale(text):
    ergebnis=0
    grossText=text.upper()
    for i in range(len(text)):
        if grossText[i] in 'AEIOU':
            ergebnis+=1
    return ergebnis
# der Test:
print( zaehleVokale('Hier, -ist+ %viel^ @enthalten.') )
```

liefert:

```
8
```

Besser man macht das mit direktem Zugriff auf die Zeichen und ohne Umwandlung. Dann lautet die Funktion:

```
def zaehleVokale(text):
```

```
    ergebnis=0
    for buchstabe in text:
        if buchstabe in 'AEIOUaeiou':
            ergebnis+=1
    return ergebnis
```

natürlich mit demselben Ergebnis.

Anzahl prüfen

Im nächsten Beispiel prüfen wir, ob die Anzahl der Sonderzeichen kleiner oder gleich groß wie eine vorgegebene Maximalzahl ist.

```
def hatMaximal(text, maximalZahl):
    ergebnis=0
    for j in range(len(text)):
        if not (text[j] in string.ascii_letters):
            ergebnis+=1
            if ergebnis > maximalZahl:
                return False
    return True
#Test:
print( hatMaximal('Hier, -ist+ %viel^ @enthalten.', 5) )
print( hatMaximal('Hier, -ist+ %viel^ @enthalten.', 9) )
print( hatMaximal('Hier, -ist+ %viel^ @enthalten.', 10) )
print( hatMaximal('Hier, -ist+ %viel^ @enthalten.', 11) )
print( hatMaximal('Hier, -ist+ %viel^ @enthalten.', 20) )
```

liefert:

```
False
False
True
True
True
```

while - Schleife

Im vierten Beispiel machen wir dasselbe mit while:

```
def hatMaximal(text, maximalZahl):
    ergebnis=0
    j=0
    while (j<len(text))and (ergebnis<=maximalZahl):
        if not (text[j] in string.ascii_letters):
            ergebnis+=1
        j+=1
    if ergebnis > maximalZahl:
        return False
    else:
        return True
```

Auch das liefert dasselbe Ergebnis.

Und wann benutzt man was?

Die Regel ist ganz einfach: Wenn man vorher weiß, wie oft die Schleife auszuführen ist,

benutzt man die for – Schleife, während bei der while – Schleife die Laufbedingung jedes mal neu ausgewertet wird. Im oben angegebenen Beispiel haben wir also einen Grenzbereich, weil wir bei der for – Schleife zusätzlich ausnutzen, dass die Bearbeitung einer Schleife grundsätzlich nach einem return abgebrochen wird.