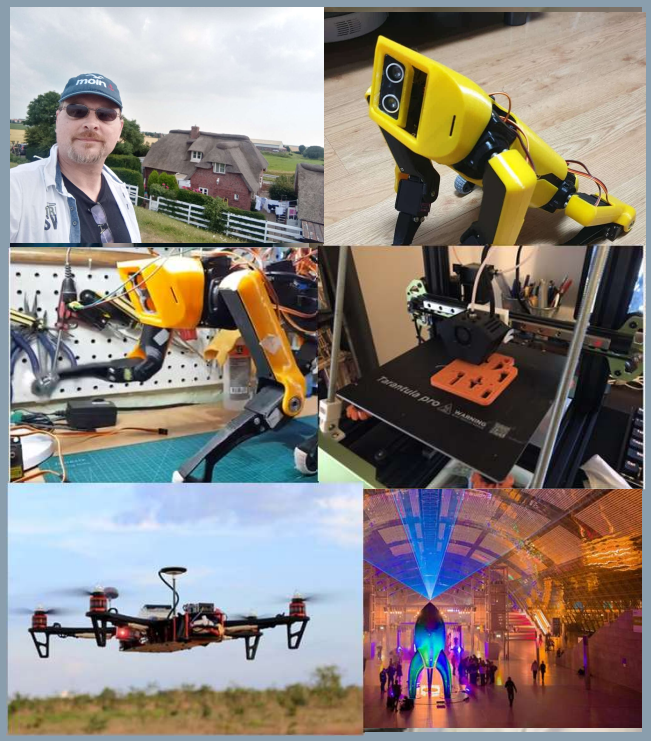
Cloud Command

Python Programmieren

Cloud Command

Zum Referenten:



Andreas Schmidt

- Jahrgang 1975, wohnhaft in Hamburg
- Fachinformatiker für Systemintegration
- Android-App Entwickler
- Java Entwickler
- Administrator für Heterogene Netzwerke
- Kommunikationselektroniker
- Über 20 Jahre im IT-Support und Consulting tätig
- ITIL
- Hardware-Entwicklung

Certified by EXIN
EXIN Ethical Hacking Foundation

Cloud Command

Variablen und Strings
String aufteilen mit split()

- Oft liegen uns Daten vor, die durch Komma getrennt sind.
- Beispielsweise ein Export von Excel im Format CSV (englisch „comma separated values“).
- Diesen String können wir einfach „aufspalten“ über split()
- Die Methode split(Trennzeichen, Anzahl_Aufteilungen_maximal) hat 2 Parameter, die beide Optional sind.
- Schauen wir uns den ersten Parameter an.
- Über diesen geben wir das gewünschte Trennzeichen mit.

Variablen und Strings
String aufteilen mit split()

- Die Methode split(Trennzeichen, Anzahl_Aufteilungen_maximal) hat 2 Parameter, die beide Optional sind. Schauen wir uns den ersten Parameter an. Über diesen geben wir das gewünschte Trennzeichen mit.

```
daten = "vorname, nachname, alter"
einzeldaten = daten.split(",")
print(einzeldaten)
```

- Als Ergebnis erhalten wir eine Liste. Listen lernen wir im Kapitel Listen kennen.

```
['vorname', ' nachname', ' alter']
```

Cloud Command

Variablen und Strings
String aufteilen mit split()

- Achtet man nun genau auf den zurückgelieferten Inhalt,
  - sieht man vor ' nachname' und ' alter' jeweils ein Leerzeichen.
- Diese Leerzeichen sind oft unerwünscht, können aber sehr einfach mit der Methode strip() entfernt werden.
- Oder man achtet bereits beim Ausgangsmaterial darauf, dass keine Leerzeichen nach den Kommas vorhanden sind.

- Wenn man allerdings sicher weiß, dass immer im Ausgangsmaterial nach dem Komma ein Leerzeichen kommt, kann man dies auch als Parameter nutzen!
- Der Parameter kann also aus einer beliebigen Zeichenkombination bestehen.

Cloud Command

Variablen und Strings
String aufteilen mit split()

Wir übergeben der Methode bei unserem Beispiel neben dem Komma auch das Leerzeichen:

```
daten = "vorname, nachname, alter"
einzeldaten = daten.split(", ")
print(einzeldaten)
```

- Als Ausgabe erhalten wir:

```
['vorname', 'nachname', 'alter']
```

Cloud Command

Variablen und Strings
String aufteilen mit split()

erste Parameter bei split()

- Bei der Methode split() sind zwei Parameter möglich und beide sind optional!
- Im letzten Beispiel haben wir als ersten Parameter das gewünschte Trennzeichen vorgegeben.
- Diese Angabe können wir auch weglassen.
- Schauen wir uns an, was passiert, wenn wir das letzte Beispiel ohne Parameter ausführen lassen.
- Wir ändern nichts am Beispiel außer bei split()

```
daten = "vorname, nachname, alter"
einzeldaten = daten.split()
print(einzeldaten)
```

- Als Ergebnis erhalten wir nun:

```
['vorname,', 'nachname,', 'alter']
```

.

Variablen und Strings
String aufteilen mit split()

- Als Ergebnis erhalten wir nun:

```
['vorname,', 'nachname,', 'alter']
```

- Wird also split() ohne Parameter aufgerufen, erfolgt eine Trennung bei jedem Leerzeichen! Jetzt werden die Kommas als Inhalt angesehen und sind bei der Liste in „vorname,“ und „nachname,“ gelandet.
- Interessant ist noch, dass mehrere Leerzeichen (falls vorhanden) als eines angesehen werden.
- Wir erhalten das gleiche Ergebnis wie oben bei folgenden String:

```
daten = "vorname,     nachname, alter"
```

Variablen und Strings
String aufteilen mit split()

zweiter Parameter: Anzahl_Aufteilungen_maximal

- Beim zweiten Parameter von split(Trennzeichen, Anzahl_Aufteilungen_maximal) können wir festlegen, wie viele Aufteilungen wir gerne maximal bekommen möchten.
- Wird nichts angegeben (was dem Standard von -1 entspricht) erhalten wir alle möglichen.
- Wären 2 möglich (wie bei unseren vorherigen Beispielen) und wir geben 1 an, erhalten wir auch nur noch eine Aufsplittung:

```
daten = "vorname,nachname,alter"
einzeldaten = daten.split(",", 1)
print(einzeldaten)
```

- Als Ergebnis bekommen wir auch genau eine Teilung:

```
['vorname', 'nachname,alter']
```

- Wir bekommen also als Anzahl von Listenelemente unsere Anzahl von Trennungen + 1.

Cloud Command

Variablen und Strings
String auf Bedingung testen

Anzahl Wörter in einem Text über split()

- Über die Methode split() ist es sehr einfach, die Anzahl der Wörter in einem Text zu bestimmen.
- Wir wissen, dass Leerzeichen die Trennung zwischen Wörtern in einem Text darstellen.
- Also nutzen wir das Leerzeichen als Trennzeichen in split() und können danach über len() die Anzahl der Elemente (sprich Wörter) zählen.

```
inhalt = "Anzahl Wörter in einem Text zählen!"
woerter = inhalt.split()
print("Anzahl der Wörter: ", len(woerter))
```

.