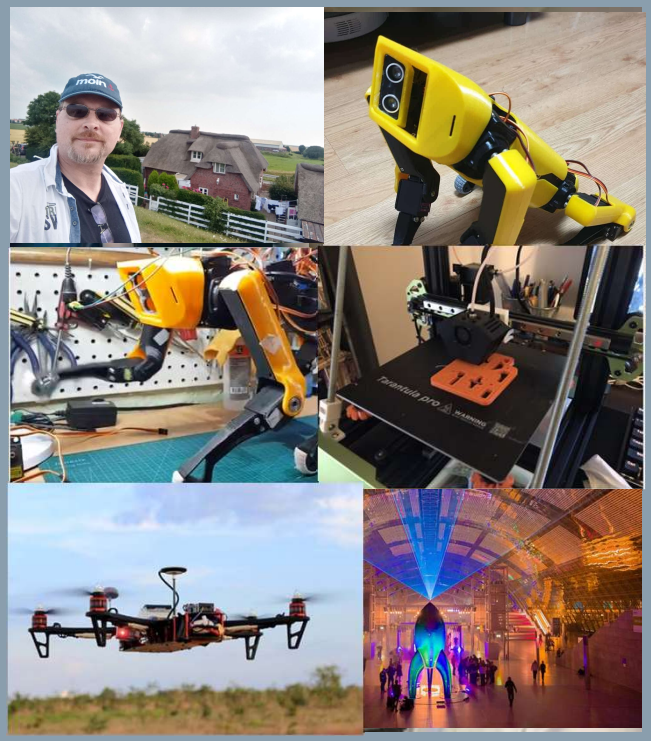
Cloud Command

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
Tupel – und der Unterschied zu Variablen und Listen

- Wir haben bereits Variablen und Listen kennengelernt.
  - Man fragt sich nun natürlich nach dem Unterschied zwischen Variablen und Listen im Vergleich zu Tupeln in Python.

- Unterschied Variablen und Listen:
  - Variable speichert einen Inhalt
  - Liste speichert VIELE Inhalte (bzw. kann das)

- Und was ist ein Tupel?
  - Eigentlich eine Liste aber der wichtige Unterschied ist, dass der Inhalt eines Tupels NICHT änderbar ist.
  - Daher sind Tupel besonders geeignet, um Konstanten zu verkörpern.

Cloud Command

Variablen und Strings

Tupel – und der Unterschied zu Variablen und Listen

Um unsere Auflistung von oben zu komplementieren:

- Variable speichert einen Inhalt
- Liste speichert VIELE Inhalte (bzw. kann das)
- Tupel speichern VIELE Inhalte UNVERÄNDERLICH

Cloud Command

Variablen und Strings
Tupel – und der Unterschied zu Variablen und Listen

Der Sinn hinter UNVERÄNDERLICH (nicht löschbar) bei Tupels

- Warum macht das „unveränderlich“ überhaupt Sinn?
  - Genau dann, wenn gespeicherte Werte nur zusammen Sinn ergeben!
- Als Beispiel ist es einfacher nachvollziehbar.
  - Wir wollen Werte für ein 2D-Koordinatensystem speichern.
  - Dazu benötigen wir den X-Wert und den Y-Wert.
  - Als könnten wir hier ein Tupel bilden mit punkt1 = (10, 22).
  - Der erste Wert stellt den X-Wert dar, der zweite Wert den Y-Wert.
  - Gäbe es nur einen Wert (egal ob X oder Y) wären diese Daten sinnlos.
  - Wir wollen also nicht, dass einer der Daten gelöscht werden kann!

Cloud Command

Variablen und Strings

Tupel – und der Unterschied zu Variablen und Listen

Tupel in Python erstellen

- Üblicherweise wird ein Tupel durch runde Klammern erstellt:

```
tupel = ('wert1', 'wert2')
```

- Technisch möglich ist es auch ohne Klammern – aber erstens unüblich und zweitens für den typischen Python-Programmierer irritierend.

```
tupel = 'wert1', 'wert2'
type(tupel)
```

- Und hat man ein Tupel mit nur einem einzigen Element ist das Komma am Schluss wichtig!

Cloud Command

Variablen und Strings

Tupel – und der Unterschied zu Variablen und Listen

- Einfach mal folgenden Code testen:

```
inhaltA = ('wert1',)
inhaltB = ('wert2')
type(inhaltA)
type(inhaltB)
```

- Warum wir überhaupt Tupel nutzen sollen ist anhand der obigen Beispiele wenig einsichtig.

Cloud Command

Variablen und Strings

Tupel – und der Unterschied zu Variablen und Listen

- Wie gesagt, Tupel sind die Speicherform von unveränderbaren Daten
  - was man in anderen Programmiersprachen als Konstanten bezeichnet.
  - Solche Daten wäre z.B. Name

```
personl = ('Elke', 'Maier', '1985')
```

Cloud Command

Variablen und Strings
Tupel – und der Unterschied zu Variablen und Listen

Zugriff auf Werte in einem Tupel

- Der Zugriff und die Möglichkeiten sind gleich wie bei der Liste. Ich kann gezielt auf einen bestimmten Inhalt zugreifen:

```
tupel = ('wert1', 'wert2', 'wert3', 'wert4', 'wert5')
print (tupel[0])
```

- Über die eckigen Klammern greife ich per Index auf den entsprechenden Wert zu.
- Auch wie bei Listen startet der Index bei 0!

Cloud Command

Variablen und Strings
Tupel – und der Unterschied zu Variablen und Listen

- Ich kann auch mehrere Werte anhand von „von – bis“ auswählen:

```
tupel = ('wert1', 'wert2', 'wert3', 'wert4', 'wert5')
print (tupel[2:4])
```

- Wir erhalten als Ergebnis:

```
('wert3', 'wert4')
```

Cloud Command

Variablen und Strings

Übersicht aller Methoden des Datentyps Tuple

- Über die Anweisung dir(tuple) erhalten wir alle Methoden zum Datentyp Tupel. Die Anzahl der Methoden ist überschaubar – sprich 2 Stück. Dies liegt daran, dass Tupel unveränderlich sind im Gegensatz zu Listen.

```
>>> dir(tuple)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__',
'__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__',
'__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmul__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__', 'count', 'index']
```

- Im Folgenden die Übersicht aller Methoden des Datentyp Dictionary:

Cloud Command

Variablen und Strings

Übersicht aller Methoden des Datentyps Tuple

Im Folgenden die Übersicht aller Methoden des Datentyp Dictionary:

Methode	Beschreibung
.count("gesucht")	Anzahl der Vorkommen des Gesuchten – Beispiel unten
.index( )	Das erste Vorkommen im Index des Gesuchten – Beispiel unten

Cloud Command

Variablen und Strings

Übersicht aller Methoden des Datentyps Tuple

Beispiel für tuple.count("gesucht")

- In der folgenden Namensliste, die als Tuple gespeichert ist, gibt es Mehrfachnennungen. Über die Methode .count("gesucht") erhält man die Anzahl.

```
vornamen =  ( "Hans","Peter","Elke","Peter","Sabine","Elke")
print(vornamen)

# Wie oft kommt bestimmter Vornamen im Tuple vor
print (vornamen.count("Peter"))
```

- Als Rückgabe erhalten wir 2, da Peter zweimal vorkommt.

Cloud Command

Variablen und Strings

Übersicht aller Methoden des Datentyps Tuple

Beispiel für tuple.index("gesucht")

- Die Methode .index("gesucht") liefert uns die Position im Index zurück.

```
vornamen =  (
"Hans","Peter","Elke","Peter","Sabine","Elke")
print(vornamen)
print (vornamen.index("Peter"))
```

- Als Ergebnis erhalten wir:

```
('Hans', 'Peter', 'Elke', 'Peter', 'Sabine', 'Elke')
1
```

- Ganz wichtig – die Zählung beginnt bei 0. Daher kommt die 1 beim Suchen nach „Peter“, der an der zweiten Stelle steht.
- Sollte das Gesuchte nicht in dem Tuple vorhanden sein, erhalten wir als Rückmeldung
  - „ValueError: tuple.index(x): x not in tuple“