Tupel oder Liste

Tupel oder Liste

das geht besonders gut mit ...

Tupel oder Liste

Tupel oder Liste - das geht besonders gut mit ...

- Oder auch die Frage:
  Warum gibt es beide Varianten?
- Tupel `meinTupel = (Kind, "Emma", 8, "3c")`
- Liste `meineListe = [Kind, "Emma", 8, "3c"]`

Tupel oder Liste

Vieles geht mit Tupeln oder Listen gleich gut

- Zugriffe auf einzelne Elemente formal gleich:
  - `meinTupel[1] -> "Emma"`
  - `meineListe[1] -> "Emma"`
- Zugriffe auf Bereiche (slices [ab:vor]) entsprechend:
  - `meinTupel[1:3] -> ("Emma", 8)`
  - `meineListe[1:3] -> ["Emma", 8]`
  - `meinTupel[1:] -> ("Emma", 8,"3c")`
  - `meineListe[:3] -> [Kind, "Emma", 8]`

Tupel oder Liste

Standardwahl Liste

- Bei einer Sammlung gleichartiger Elemente (Aufzählung) ist die Liste die "richtige" Wahl

Standardwahl Tupel

- Bei Daten mit verschiedenen Komponenten wie bei `(Kind, "Emma", 8, "3c")` ist es eher ein Tupel.

Tupel oder Liste

Besondere Anforderung:

- Änderung zur Laufzeit
  - Bei Tupeln nur durch Neudefinition des Tupels möglich

```
meinTupel = meinTupel[:2] + (9, "3c")
meinTupel -> (Kind, "Emma", 9, "3c")
```

  - also möglichst nicht so machen
- bei Liste:

```
meineListe[2] = 9
meineListe -> [Kind, "Emma", 9, "3c"]
```

  - möglichst Liste verwenden

Tupel oder Liste

Für Änderung zur Laufzeit ...

- viele andere Beispiele möglich,
- Listen haben viele zusätzliche Methoden
- möglichst Liste verwenden

Tupel oder Liste

Besondere Anforderung:

- Gleichzeitige Rückgabe mehrerer Werte durch eine Funktion

```
def gibZurueck():
    return Kind, "Emma", 8, "3c"
```

  - automatische Zuweisung in ein Tupel

```
gibZurueck() -> (Kind, "Emma", 8, "3c")
```

  - Tupel ist zwingend der Rückgabetyp

Tupel oder Liste

Besondere Anforderung:

- Einzelzuweisung zu Variablen (unpacking)
  `Typ,Name,Alter,Klasse = meinTupel`
  `Typ,Name,Alter,Klasse = meineListe`
  liefern beide
  - `Typ -> Kind`
  - `Name -> "Emma"`
  - `Alter -> 8`
  - `Klasse -> "3c"`
- Also egal!

Tupel oder Liste

- Nicht eingegangen bin ich auf mögliche Vor- oder Nachteile an Speicher, Laufzeit o.ä.
- Fazit:
  Listen sind der flexiblere Datentyp, daher sollte man Tupel nur dann verwenden, wenn es zwingend ist oder von vorn herein klar ist, dass keine Änderung von Werten erforderlich ist.

Tupel oder Liste

- Nachtrag 1:
  Dictionary (Assoziationsliste) ist ein Sammlungstyp, der gesondert betrachtet werden sollte und seine speziellen Aufgabenfelder hat.
  `d={1:"eins",4:"vier"}`
- Nachtrag 2:
  Set (Menge) ist ein Sammlungstyp, der gesondert betrachtet werden sollte und seine speziellen Aufgabenfelder hat.
  `s=set([1,2,3,2]) -> (1,2,3)`