Container füllen

Erarbeitung
einer einfachen Version von
fuelle

Container füllen

- Schrittweise Erarbeitung nach dem Konzept des test-driven-development
- Formulierung (hier eines) Ziels

```
fuelle(
  [30 30 30 30 20 20 20 20],
  [ ],
  80)
```

stuecke

container ...

... mit Fassungsvermögen

→ '(80 30 30 20)

Container füllen

- Schritt 1
  - Kopf der Funktion
  - Kennzeichnung der Unvollständigkeit

```
def fuelle(stuecke, container, fassungsvermoegen):
    return 'kann noch nichts'
```

Container füllen

- Schritt 0
  - Testen

```
print( stuecke,
       80,
      'Ergebnis: ',
       fuelle(stuecke, [30,30,20], 80) )
```

→ ... Ergebnis: kann noch nichts

Container füllen

- Schritt 1
  - Verzweigung einbauen
  - erste Bedingung realisieren:
    Gibt es noch stuecke?

```
def fuelle(stuecke, container, fassungsvermoegen):
    if len(stuecke)==0:
        return False
    return 'kann noch nichts'
```

Container füllen

- Schritt 1 testen

```
print( [], 80, 'Ergebnis: ', fuelle([], [], 80) )
```

```
-> [] 80 Ergebnis:  False
```

Container füllen

- Schritt 2
  - container exakt voll? (Erfolgsfall)

```
def fuelle(stuecke, container, fassungsvermoegen):
    if len(stuecke)==0:
        return False
    elif istVoll(container, fassungsvermoegen):
        return container
    return 'kann noch nichts'
```

Container füllen

- Schritt 2 testen
  (*wird erst erfolgreich, wenn die Hilfsfunktion erfolgreich entwickelt wurde*)

```
def istVoll(container, fassungsvermoegen):
    return fassungsvermoegen==sum(container)
```

Testergebnis:
[30, 30, 30, 20, 20, 20, 20] 80 Ergebnis: [30, 30, 20]

Container füllen

- Schritt 3
  - Entwickeln Sie nach dem gleichen Prinzip eine Hilfsfunktion *zuvoll*
- Überlegen und bearbeiten Sie weitere Schritte nach dem gleichen Prinzip