Container füllen

Schrittweise Erarbeitung
der einfachen
Fülle – Funktion:

Fülle jeweils das größte noch passende Stück ein.

Container füllen

Was müssen wir kennen?

- die Stücke
  Wir definieren eine Liste:

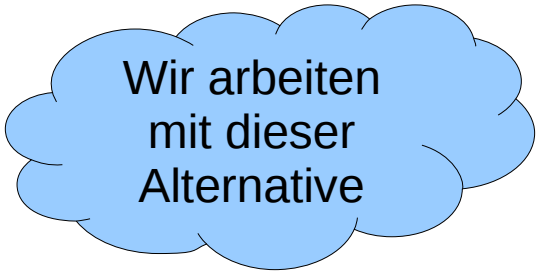
  [30, 30, 30, 30, 20, 20, 20, 20]

Datenstruktur
Liste

- den Container
  Auch dazu verwenden wir eine Liste,
  die zunächst leer ist.
- die Zielgröße, also das Fassungsvermögen.
  Für sie benötigen wir allein eine Zahl: 80.

Container füllen

Was müssen wir kennen?



Datenstruktur
Liste

- die Stücke
  Wir definieren eine Liste:

  [30, 30, 30, 30, 20, 20, 20, 20]

- den Container
  Auch dazu verwenden wir eine Liste,
  die am Kopf die Zielgröße enthält, danach die
  (zunächst leere) Liste der Inhalte enthält.
  [80, [ ]]

Container füllen

vorbereitender Schritt: Definition des Funktionskopfes

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

Testaufruf:

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

liefert noch kein sinnvolles Ergebnis, sondern **kann noch nichts**

Container füllen

Was kann passieren?

[*Das ist die Frage nach den möglichen Verzweigungen.*]

- die Stücke passen genau
  Dazu muss die Summe aller Stücke im Container gleich der Zielgröße (am Kopf der Containerliste) sein.
- Um diese Prüfung zu erledigen, schreiben wir eine Hilfsfunktion, der wir die Containerliste übergeben.

Container füllen

Hilfsfunktion ist_voll

Sie prüft, ob der Container die Zielgröße erreicht hat.

```
def ist_voll(container):
    return container[0]==sum(container[1])
```

Container füllen

1. Schritt: Erfolgsfall

```
def fuelle(stuecke, container):
    if ist_voll(container):
        return container
    else:
        return'kann noch nichts'
```

Container füllen

Der Test:

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

liefert nun wie erwartet die gewünschte Lösung.

```
[80, [30, 30, 20]]
```

Container füllen

Was kann passieren?

- die Stücke passen genau
- der Container wird zu voll
  Das ist der Fall, wenn das neue Stück mit der Summe aller Stücke im Container größer als die Zielgröße ist.
- Es ist einfach, die zugehörige Hilfsfunktion zu schreiben.

Container füllen

Hilfsfunktion wird_zu_voll

Sie prüft, ob der Container mit dem Stück die Zielgröße überschreitet.

```
def wird_zu_voll(stueck, container):
    return container[0] < stueck+sum(container[1])
```

Container füllen

2. Schritt: Den Fall *"zu voll"* bearbeiten:

Weiter versuchen ohne das Stück.

```
def fuelle(stuecke, container):
    if ist_voll(container):
        return container
    elif wird_zu_voll(stuecke[0],container):
        return fuelle(stuecke[1:], container)
    else:
        return'kann noch nichts'
```

Typischer Fehler:
Beim Abbau von Listen muss auch der Fall einer leeren Liste berücksichtigt werden.

Container füllen

Was kann passieren?

- die Stücke passen genau
- es gibt keine Stücke mehr
- der Container wurde zu voll
- das aktuelle Stück passt noch in den Container
- der Container ist noch nicht voll

Warum diese Reihenfolge?

Container füllen

3. Schritt: Keine Stücke mehr

```
def fuelle(stuecke, container):
    if ist_voll(container):
        return container
    elif len(stuecke)==0:
        return False
    elif wird_zu_voll(stuecke[0], container):
        return fuelle(stuecke[1:], container)
    else:
        return'kann noch nichts'
```

Alternativen für den Rückgabewert?

Container füllen

Was kann passieren?

- die Stücke passen genau
- der Container wird zu voll
- es gibt keine Stücke mehr
- das aktuelle Stück passt noch in den Container
  Dieser Fall muss aber nicht durch eine weitere Abfrage geprüft werden, da er den "Normalfall" darstellt. → else

Container füllen

4. Schritt: Das Stück geht noch hinein.

```
def fuelle(stuecke, container):
    if ist_voll(container):
        return container
    elif len(stuecke)==0:
        return False
    elif wird_zu_voll(stuecke[0],container):
        return fuelle(stuecke[1:], container)
    else:
        return fuelle(stuecke[1:],
                      fuelle_ein(stuecke[0],
                                 container))
```

Hilfsfunktion benutzen !

Container füllen

Hilfsfunktion fuelle-ein

Füllt ein Stück in den Container ein.

```
def fuelle_ein(stueck, container):
    inhalt=[]+container[1]
    # eine echte Kopie erzeugen!
    inhalt.append(stueck)
    return [container[0], inhalt]
```

Container füllen

Was bedeutet „echte Kopie“ ?

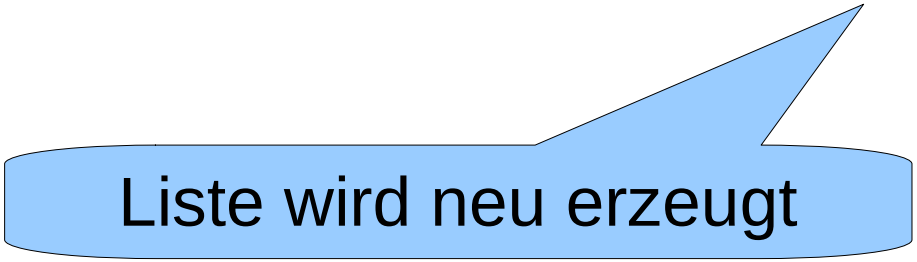
- Python macht beim Aufruf von Funktionen *call-by-reference*. Es wird also nicht ein Wert des Objekts übergeben (das wäre *call-by-value*), sondern nur eine Referenz (*Zeiger*) auf das Objekt.
- Verändernde Zugriffe auf das Objekt in der Methode ändern daher das Objekt auch für die aufrufende Ebene.
- Das betrifft konkret in diesem Fall die interne Liste **`inhalt`** und den Zugriff auf sie mit **`append`**.

Container füllen

Alternative zu fuelle-ein

Füllt ein Stück in den Container ein.

```
def fuelle_ein(stueck, container):
    return [container[0],
            container[1]+[stueck]]
```



Container füllen

Alternative zu fuelle:

```
def fuelle(stuecke, container):
    if ist_voll(container):
        return container
    if len(stuecke)==0:
        return False
    if wird_zu_voll(stuecke[0],container):
        return fuelle(stuecke[1:], container)
    return fuelle(stuecke[1:],
                  fuelle_ein(stuecke[0],
                             container))
```

Jeweils Ausstieg nach return