

# Dokumentation Juninoraufgabe 1

## Jugendwettbewerb Informatik 3.Runde

### Aufgabenstellung

Es soll ein Programm geschrieben werden, das dem Kindergarten hilft, Wundertüten zu packen. Das Programm sollte eine Anzahl der unterschiedlichen Spielesorten, die Anzahl wie viele es von den Spielesorten gibt und die Anzahl von Tüten die gepackt werden sollen bekommen. Anschließend gibt es aus, wie die einzelnen Wundertüten zusammengestellt werden sollen. Dabei muss folgendes noch beachtet werden.

### Die Regel:




Beim Wundertüten packen, muss das Programm möglichst gleichmäßig packen. Die Unterschiede sollen möglichst gering sein und es dürfen keine Spiele übrig bleiben. Die Gesamtzahlen der Spiele zwischen je zwei Tüten und die Spielesorten dürfen sich höchstens um eins unterscheiden.

### Lösungsidee:

Das Programm soll die Wundertüten zusammenstellen. Damit es nicht so kompliziert wird, schauen wir zuerst auf eine Spielsorte. Man muss einfach ein Spiel in jede Tüte tun. Wenn nachdem man in jede Tüte ein Spiel getan hat und es noch übrig bleibt, dann fangen wir vorne wieder an und wiederholen das ganze bis kein Spiel mehr übrig bleibt. Dann macht man bei dieser Tüte weiter mit einer anderen Spielsorte. Und wiederholen das ganze. Bis es alle Spiele verteilt sind.

### Veranschaulichung Wundertüte0.txt:

Zur Veranschaulichung des Algorithmus, habe ich hier das Beispiel aus dem Aufgabenblatt dargestellt. Als allererstes verteilen wir reihum die grauen Würfel, bis alle Tüten einen Würfel haben. Dann fangen wir wieder bei Tüte 1 an. Es ist noch ein Würfel übrig, den wir in Tüte 1 tun.

3 Tüten, 3 Spiele, 4x  , 4x  , 2x 



Tüte 1

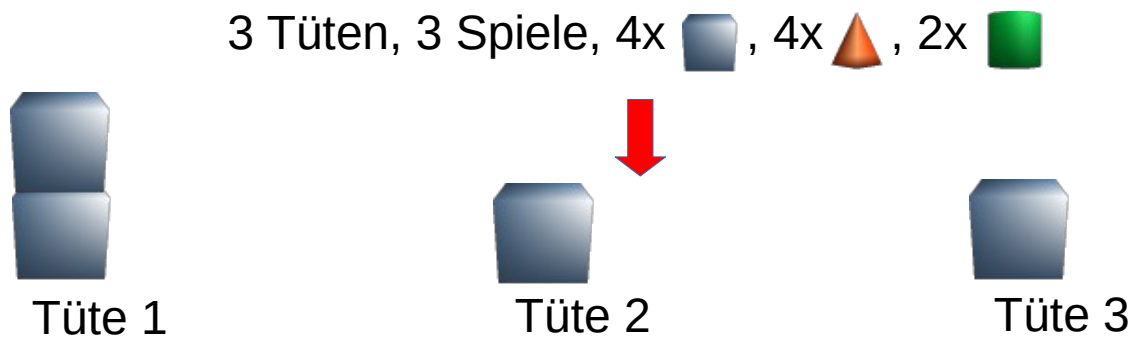


Tüte 2

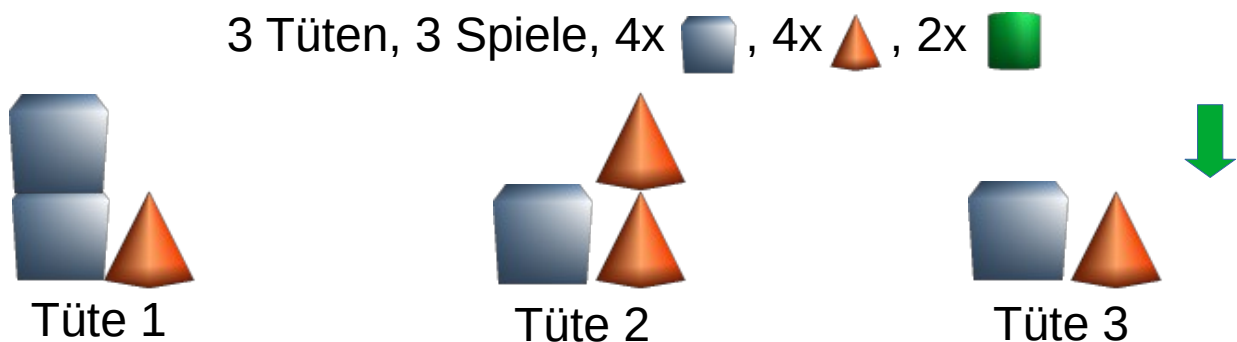


Tüte 3

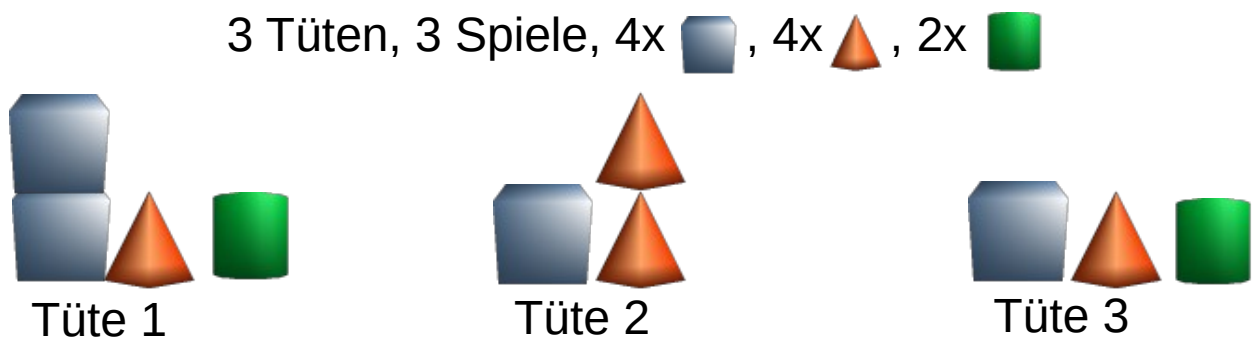
Hier unterscheiden sich die grauen Würfel um 1. Nun beginne ich mit der Verteilung von den orangenen Kegel bei Tüte 2, weil ich gerade bei der Verteilung der grauen Würfel mit Tüte 1 beendet habe und wir immer reihum verteilen.



Hier unterscheiden sich die orangenen Kegel auch um 1. Nun beginne ich mit der Verteilung von den grünen Zylinder bei Tüte 3, weil ich gerade bei der Verteilung der orangenen Kegel mit Tüte 2 beendet habe und wir immer reihum verteilen.



Hier unterscheiden sich die grünen Zylinder auch um 1, nur Tüte 2 hat insgesamt 1 weniger als die anderen Tüten. So sehen die Tüten am Ende aus.



## Umsetzung:

```
#!/usr/bin/env python3

import sys
from tabulate import tabulate
import pandas
```

Ich brauche `sys`, um den Filenamen der Eingabedatei auf der Kommandozeile zu bekommen. `tabulate` und `pandas` sind für das Tabellenoutput am Ende geeignet.

```
def Einlesung(Datei):
    Tüten = 0
    Spiele = []
    d = open(Datei, 'r')
    for Zeile in d:
        Z = Zeile.strip('\n')
        if Z == "":
            break
        Spiele.append(int(Z))
    Tüten = Spiele[0]
    Spielarten = Spiele[1]
    Spiele = Spiele[2:]
    if len(Spiele) == Spielarten:
        return Tüten, Spiele, Spielarten
```

Das ist die Definition `Einlesung`, sie liest die Datei ein.

```
def Einlesung(Datei):
    Tüten = 0
    Spiele = []
```

Ich erstelle die Variable `Tüten` und die Liste `Spiele`.

```
d = open(Datei, 'r')
for Zeile in d:
    Z = Zeile.strip('\n')
    if Z == "":
        break
    Spiele.append(int(Z))
```

Ich öffne die Datei `d` und nehme davon jede Zeile und tu es in die Liste `Spiele` als Integer. Und wenn eine Zeile leer ist, dann bin ich fertig und gehe von der Schleife raus.

```
Tüten = Spiele[0]
Spielarten = Spiele[1]
Spiele = Spiele[2:]
```

Ich habe jetzt alle Zeilen in `Spiele` getan. Ich brauche aber noch die Anzahl von Tüten und die Anzahl von Spielarten.

Die erste Zeile ist die Anzahl von Tüten, deswegen ist Tüten "Spiele[0]". Die zweite Zeile ist die Anzahl von Spielarten, deswegen tu ich "Spiele[1]" in eine neue Variable: Spielarten. Die restlichen Zeilen sind die Anzahl von Spiele, deswegen nehme ich statt Spiele jetzt "Spiele[2:]".

```
if len(Spiele) == Spielarten:
    return Tüten, Spiele, Spielarten
```

Ich überprüfe, ob die Länge von der Liste Spiele Spielarten ist (Wenn vielleicht mal etwas schief läuft). Wenn das der Fall ist, gibt das Programm Tüten, Spiele und Spielarten zurück.

### Ende def Einlesung

```
def Wundertüte(Tütenan, Spiele):
    Tüten = [{ } for _ in range(Tütenan)]
    if Tütenan > sum(Spiele):
        print('Es bleibt/en (eine) leere Tüte/n übrig.')
        return
    j = 0
    i = 0
    Alphabeta = 'WKGABCDEFHIJLMNOPQRSTUVWXYZÄÖÜß'
    while Spiele != []:
        n_Spielart = Spiele[0]
        Spiel_id = Alphabeta[j] if j < len(Alphabeta) else j
        while n_Spielart > 0:
            if Spiel_id not in Tüten[i]:
                Tüten[i][Spiel_id] = 1
            else:
                Tüten[i][Spiel_id] += 1
            n_Spielart -= 1
            i += 1
            i %= len(Tüten)
        j += 1
        Spiele = Spiele[1:]
    return Tüten
```

Das ist die Definition Wundertüte. Sie packt die Tüten.

```
def Wundertüte(Tütenan, Spiele):
    Tüten = [{ } for _ in range(Tütenan)]
    if Tütenan > sum(Spiele):
        print('Es bleibt/en (eine) leere Tüte/n übrig.')
        return
```

Zuerst erstelle ich eine Liste aus Dictionaries und speichere sie in Tüten. Die Variable und Liste Tüten und Spiele sind jetzt Tütenan und Spiele. Jetzt schaue ich ob die Anzahl von

**Tüten(Tütenan)** größer ist als die Summe von allen Spielen. Wenn das so ist, dann gebe ich 'Es bleibt/en (eine) leere Tüte/n übrig.' aus und gehe aus der Definition raus.

```
j = 0
i = 0
Alphabeta = 'WKGABCDEFHIJLMNOPQRSTUVWXYZÄÖÜß'
```

Ich erstelle 3 neue Variablen **j**, **i** und **Alphabeta**. **j** und **i** sind für **Tüten** und **Spiele** gemacht. **Alphabeta** ist auch für **Spiele** gemacht.

```
while Spiele != []:
    n_Spielart = Spiele[0]
    Spiel_id = Alphabeta[j] if j < len(Alphabeta) else j
```

Jetzt wiederholt sich das Programm bis **Spiele** eine leere Liste ist. Am Ende dieser Schleife wird also immer das "0.Argument" rausgenommen. Dann nehme ich "**Spiele[0]**" und speichere es in **n\_Spielart**. Dann erstelle ich noch eine Variable: **Spiel\_id**. Denn ich will am Ende eine Tabelle machen, dafür benutze ich für jedes Spiel einen Buchstaben, den ich in **Alphabeta** gespeichert habe. Wenn bei einer Datei mal mehr Spiele als die Länge von **Alphabeta** hat, dann benutzt man **j** selber.

```
while n_Spielart > 0:
    if Spiel_id not in Tüten[i]:
        Tüten[i][Spiel_id] = 1
    else:
        Tüten[i][Spiel_id] += 1
```

Nun mache ich noch eine **while-Schleife**. Dieses mal bis **n\_Spielart** 0 ist. Das bedeutet wenn **Spiele** verteilt sind. Wenn **Spiel\_id** noch nicht in **Tüten[i]** (also ein dictionary) eingefügt wurde, dann nehmen wir den key **Spiel\_id** und stellen den Wert auf 1. Wenn es schon **Spiel\_id** in **Tüten[i]** gibt, dann addieren wir einfach 1 dazu. Das ist die Verteilung von **Spiele[j]**.

```
n_Spielart -= 1
i += 1
i %= len(Tüten)
```

In der selben **while-Schleife** subtrahiere ich 1 von **n\_Spielart**. Denn wenn ich wieder durch die Schleife gehe, ist 1 von den Spielen in **Spiele** weg und ich will ja, dass wenn **n\_Spielart** 0 ist, aus der Schleife zu gehen. Dann addiere ich 1 zu **i**, denn ich will ja reihum gehen. Aber wenn es jetzt 3 Tüten gäbe und jetzt **i** 4 wird. Dann muss ich ja wieder an den Anfang. Dazu braucht man das %. Jetzt nehme ich die Länge von **Tüten** und mach Modulo(%).

```
j += 1
Spiele = Spiele[1:]
return Tüten
```

Jetzt gehe ich aus einer **while-Schleife** raus. Ich addiere 1 zu **j**, denn **Spiele[0]** ist ja leer und ich muss einen neuen **Spiel\_id** herstellen. Oben habe ich gesagt, dass ich immer

n\_Spielart als Spiele[0] speichere. Deswegen muss ich hier “Spiele[1:]” machen. Dann am Ende gebe ich Tüten zurück.

**Ende def Wundertüte**

```
def Tabellenoutput(Tüteen):  
    df = pandas.DataFrame(Tüteen).fillna(0)  
    print(tabulate(df, headers = ['Tüte'] + list(df.columns), tablefmt = 'orgtbl'))
```

Das ist die Definition Tabellenoutput. Sie gibt das Ergebnis in einer Tabelle aus. Zuerst erstelle ich eine Variable: df. Sie macht mit pandas(hab ich einfach auf dem Web gefunden) zusammen was und das “.fillna(0)” füllt alle Stellen wo “nan” steht zu eine 0. Dann wird eine Tabelle mit headers und df ausgegeben im tableformat “orgtbl”.

**Ende def Tabellenoutput**

```
if __name__ == "__main__":  
    Tüten, Spiele, Spielarten = Einlesung(sys.argv[1])  
    Tabellenoutput(Wundertüte(Tüten, Spiele))
```

Das ist der “main” Code. Zuerst wird Einlesung(sys.argv[1]) aufgerufen. Tüten, Spiele und Spielarten (zurückgegeben) werden in die Variablen Tüten, Spiele und Spielarten reingetan. Dann wird Wundertüte() aufgerufen mit Tüten und Spiele als Argumente und das ganze mit Tabellenoutput() als Tabelle dargestellt.

**Ende “ main ”-Code**

Hier ist Platz zum schreiben.

## Beispiele:

### *Wundertüte0.txt*

```
./Wundertüte.py bwinf.de_fileadmin_bundeswettbewerb_42_wundertuete0.txt
```

Tüte	W	K	G
0	2	1	1
1	1	2	0
2	1	1	1

### *Wundertüte1.txt*

```
./Wundertüte.py bwinf.de_fileadmin_user_upload_wundertuete1.txt
```

Tüte	W	K	G
0	3	1	2
1	3	1	2
2	3	1	2
3	3	1	2
4	3	1	2
5	3	1	2

### *Wundertüte2.txt*

```
./Wundertüte.py bwinf.de_fileadmin_user_upload_wundertuete2.txt
```

Tüte	W	K	G	A
0	2	1	0	0
1	1	1	1	0
2	1	1	1	0
3	1	1	1	0
4	1	1	0	1
5	1	1	0	1
6	1	1	0	1
7	1	1	0	1
8	1	1	0	1

**Wundertüte3.txt**

```
./Wundertüte.py bwinf.de_fileadmin_user_upload_wundertuete3.txt
```

Tüte	W	K	G	A	B
0	1	1	0	0	0
1	1	1	0	0	0
2	0	1	1	0	0
3	0	1	1	0	0
4	0	1	1	0	0
5	0	1	1	0	0
6	0	1	1	0	0
7	0	1	1	0	0
8	0	1	0	1	0
9	0	1	0	1	0
10	0	1	0	0	1

**Wundertüte4.txt**

```
./Wundertüte.py bwinf.de_fileadmin_user_upload_wundertuete4.txt
```

Tüte	W	K	G	A	B	C
0	2	6	3	5	31	5
1	2	6	3	5	30	6
2	2	6	3	5	30	6
3	2	6	2	6	30	6
4	1	7	2	6	30	6
5	1	7	2	6	30	5
6	1	7	2	6	30	5
7	1	7	2	6	30	5
8	1	7	2	6	30	5
9	1	7	2	6	30	5
10	1	7	2	6	30	5
11	1	7	2	6	30	5
12	1	7	2	6	30	5
13	1	7	2	6	30	5
14	1	7	2	6	30	5
15	1	7	2	5	31	5
16	1	6	3	5	31	5



```
./Wundertüte.py bwinf.de_fileadmin_user_upload_wundertuete5.txt
```

9/11

**Code:**

```
#!/usr/bin/env python3

import sys
from tabulate import tabulate
import pandas

def Tabellenoutput(Tüteen):
    df = pandas.DataFrame(Tüteen).fillna(0)
    print(tabulate(df, headers = ['Tüte'] + list(df.columns),
tablefmt = 'orgtbl'))

def Einlesung(Datei):
    Tüten = 0
    Spiele = []
    d = open(Datei, 'r')
    for Zeile in d:
        Z = Zeile.strip('\n')
        if Z == '':
            break
        Spiele.append(int(Z))
    Tüten = Spiele[0]
    Spielarten = Spiele[1]
    Spiele = Spiele[2:]
    if len(Spiele) == Spielarten:
        return Tüten, Spiele, Spielarten

def Wundertüte(Tütenan, Spiele):
    Tüten = [{ } for _ in range(Tütenan)]
    if Tütenan > sum(Spiele):
        print('Es bleibt/en (eine) leere Tüte/n übrig.')
        return
    j = 0
    i = 0
    Alphabet = 'WKGABCDEFHIJLMNOPQRSTUVWXYZÄÖÜß'
    while Spiele != []:
        n_Spielart = Spiele[0]
        Spiel_id = Alphabet[j] if j < len(Alphabet) else j
        while n_Spielart > 0:
            if Spiel_id not in Tüten[i]:
                Tüten[i][Spiel_id] = 1
            else:
                Tüten[i][Spiel_id] += 1
            n_Spielart -= 1
            i += 1
            i %= len(Tüten)
```

```
        j += 1
        Spiele = Spiele[1:]
    return Tüten

if __name__ == "__main__":
    Tüten, Spiele, Spielarten = Einlesung(sys.argv[1])
    Tabellenoutput(Wundertüte(Tüten, Spiele))
```