Übungsblatt 3

Programm in Python eingeben:

```python
import itertools
# 1. Grundmenge erstellen:
Augenzahlen = {1, 2, 3, 4, 5, 6}
# itertools.product liefert das kartesische Produkt:
Ω = list(itertools.product(Augenzahlen, Augenzahlen))
print(Ω)
# Beachte, dass Ω als Menge nicht sortiert ist.
# Statt "set" kann auch "list" geschrieben werden:
# dann wird die Menge schon sortiert.

# 2. Anzahl Günstige und Mögliche bestimmen:
günstig = 0
möglich = len(Ω)
for Augenzahl1, Augenzahl2 in Ω:
    if Augenzahl1 + Augenzahl2 == 10:
        günstig = günstig + 1

# 3. Ausgabe:
print(f"{günstig} von {möglich}")
print(f"p = {günstig/möglich}")
```

```
[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 1), (2, 2),
 (2, 3), (2, 4), (2, 5), (2, 6), (3, 1), (3, 2), (3, 3), (3, 4),
 (3, 5), (3, 6), (4, 1), (4, 2), (4, 3), (4, 4), (4, 5), (4, 6),
 (5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (5, 6), (6, 1), (6, 2),
 (6, 3), (6, 4), (6, 5), (6, 6)]
3 von 36
p = 0.08333333333333333
```

Dasselbe wie oben nur ohne Kommentare (da es so übersichtlicher ist):

```python
import itertools
Augenzahlen = {1, 2, 3, 4, 5, 6}
Ω = set(itertools.product(Augenzahlen, Augenzahlen))
print(Ω)
günstig = 0
möglich = len(Ω)
for Augenzahl1, Augenzahl2 in Ω:
    if Augenzahl1 + Augenzahl2 ==10:
        günstig = günstig + 1
print(f"{günstig} von {möglich}")
print(f"p = {günstig/möglich}")
```

```
{(3, 4), (4, 3), (3, 1), (5, 4), (4, 6), (5, 1), (2, 2), (1, 6),
 (2, 5), (1, 3), (6, 2), (6, 5), (4, 2), (4, 5), (3, 3), (5, 6),
 (3, 6), (5, 3), (2, 4), (1, 2), (2, 1), (1, 5), (6, 1), (6, 4),
 (3, 2), (4, 1), (3, 5), (5, 2), (4, 4), (5, 5), (1, 1), (1, 4),
 (2, 3), (2, 6), (6, 6), (6, 3)}
3 von 36
p = 0.08333333333333333
```

1. **Erklärungen (Zeilennummerierung nach dem Programm ohne Kommentare):**
   1 <u>import itertools:</u>
   - *Itertools werden in das Programm importiert.*

   2 <u>Augenzahlen = {1, 2, 3, 4, 5, 6}:</u>
   - *Die Variable Augenzahlen wird als Menge {1, 2, 3, 4, 5, 6} definiert.*

   3 <u>Ω = set(itertools.product(Augenzahlen, Augenzahlen)):</u>
   - *Aus den möglichen Augenzahlen werden alle möglichen kartesischen Produkte gebildet und diese Menge wird als Ereignisraum Ω festgelegt (wird so genannt).*

   4 <u>print(Ω):</u>
   - *Gibt an, dass im Ausgabefeld (=Ergebnis) der Ereignisraum Ω, also alle möglichen Augensummenpaare, angezeigt werden soll.*

   5 <u>günstig = 0:</u>
   - *Die Variable „günstig“ wird auf 0 gesetzt, also der Anfangswert für „günstig“ ist 0.*

6 möglich = len(Ω):

- *Für die Variable „möglich“ wird als Wert die Anzahl an Wertepaaren im Ereignisraum Ω festgelegt, also len() gibt die Anzahl an Einträgen in einer bestimmten Menge (oder auch die Länge einer bestimmten Menge) an.*

7 for Augenzahl1, Augenzahl2 in Ω:

- *Die Variablen „Augenzahl1“ und „Augenzahl2“ nehmen nach und nach jeweils die Werte aus der Menge {1, 2, 3, 4, 5, 6} an, also sodass alle möglichen Kombinationen aus {1, 2, 3, 4, 5, 6} x {1, 2, 3, 4, 5, 6} für „Augenzahl1“ und „Augenzahl2“ jeweils einmal angenommen werden.*

8 if Augenzahl1 + Augenzahl2 == 10:

- *Es wird jeweils geprüft, ob die Variable „Augenzahl1“ und die „Augenzahl2“ addiert den Wert 10 ergeben hat.*

9 günstig = günstig + 1:

- *Wenn die Prüfung aus der vorherigen Zeile ergeben hat, dass die Summe der beiden Variablen, also Augenzahl1 + Augenzahl2 den Wert 10 angenommen hat, wird die Variable „günstig“ um den Wert 1 erhöht.*

10 print(f"{günstig} von {möglich}"):

- *Gibt an, dass im Ausgabefeld (=Ergebnis) die Anzahl der günstigen von den möglichen Ereignissen angezeigt werden sollen.*

11 print(f"p= {günstig/möglich}"):

- *Gibt an, dass im Ausgabefeld (=Ergebnis) das Ergebnis der Rechnung günstig durch möglich, also die relative Häufigkeit, angezeigt werden soll.*

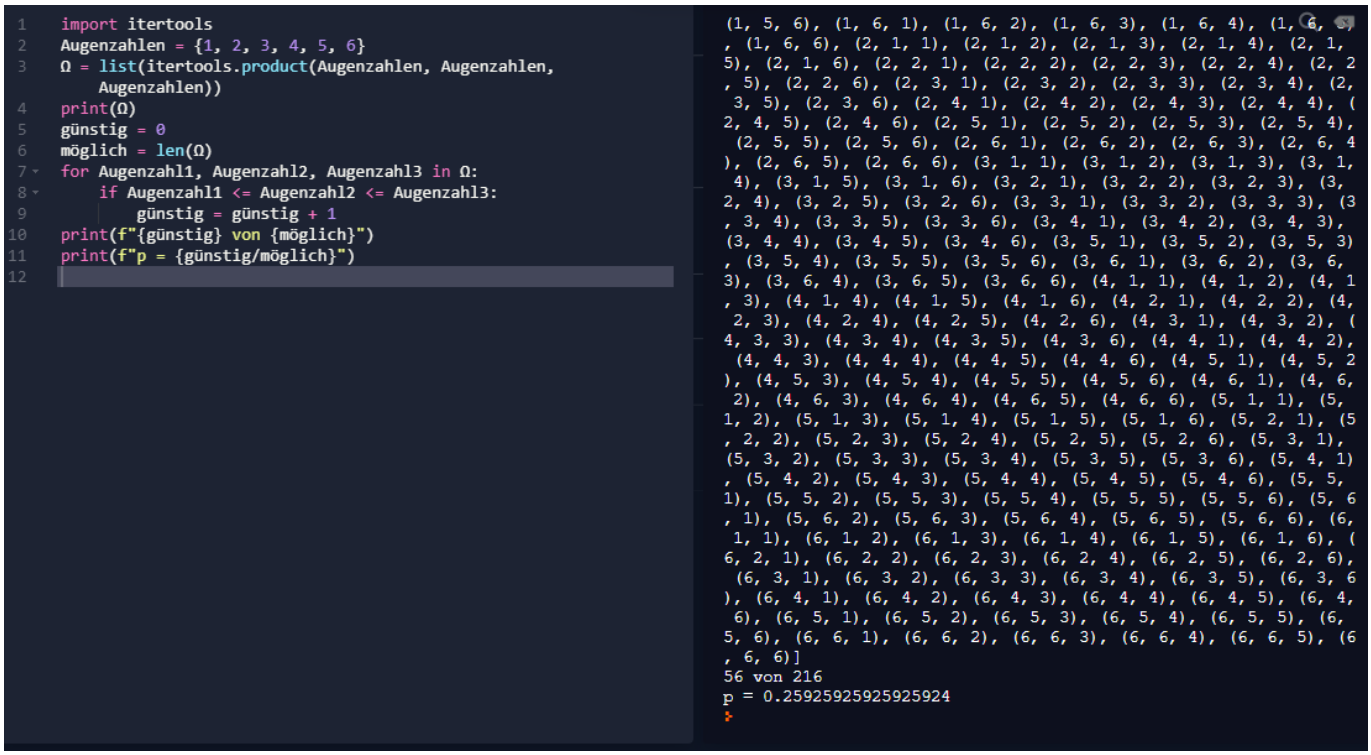
2. Geändertes Programm („list“ statt „set“):

```
import itertools
Augenzahlen = {1, 2, 3, 4, 5, 6}
Ω = list(itertools.product(Augenzahlen, Augenzahlen))
print(Ω)
günstig = 0
möglich = len(Ω)
for Augenzahl1, Augenzahl2 in Ω:
    if Augenzahl1 + Augenzahl2 ==10:
        günstig = günstig + 1
print(f"{günstig} von {möglich}")
print(f"p = {günstig/möglich}")
```

```
[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 1), (2, 2),
 (2, 3), (2, 4), (2, 5), (2, 6), (3, 1), (3, 2), (3, 3), (3, 4),
 (3, 5), (3, 6), (4, 1), (4, 2), (4, 3), (4, 4), (4, 5), (4, 6),
 (5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (5, 6), (6, 1), (6, 2),
 (6, 3), (6, 4), (6, 5), (6, 6)]
3 von 36
p = 0.08333333333333333
```

→ Nach dem Ändern des Programms sind nun die Zahlenpaare in der Ausgabe (also des Ereignisraums Ω) der „Größe“ nach geordnet!
Generell sind die Ergebnisse jedoch bei beiden Programmen identisch, nur im ursprünglichen Programm ungeordnet, im neuen hingegen geordnet.

3. Geändertes Programm (Augenzahl3 einfügen und mit <= arbeiten):



```
import itertools
Augenzahlen = {1, 2, 3, 4, 5, 6}
Ω = list(itertools.product(Augenzahlen, Augenzahlen,
    Augenzahlen))
print(Ω)
günstig = 0
möglich = len(Ω)
for Augenzahl1, Augenzahl2, Augenzahl3 in Ω:
    if Augenzahl1 <= Augenzahl2 <= Augenzahl3:
        günstig = günstig + 1
print(f"{günstig} von {möglich}")
print(f"p = {günstig/möglich}")
```

```
(1, 5, 6), (1, 6, 1), (1, 6, 2), (1, 6, 3), (1, 6, 4), (1, 6, 5)
, (1, 6, 6), (2, 1, 1), (2, 1, 2), (2, 1, 3), (2, 1, 4), (2, 1,
5), (2, 1, 6), (2, 2, 1), (2, 2, 2), (2, 2, 3), (2, 2, 4), (2, 2
, 5), (2, 2, 6), (2, 3, 1), (2, 3, 2), (2, 3, 3), (2, 3, 4), (2,
 3, 5), (2, 3, 6), (2, 4, 1), (2, 4, 2), (2, 4, 3), (2, 4, 4), (
2, 4, 5), (2, 4, 6), (2, 5, 1), (2, 5, 2), (2, 5, 3), (2, 5, 4),
 (2, 5, 5), (2, 5, 6), (2, 6, 1), (2, 6, 2), (2, 6, 3), (2, 6, 4
), (2, 6, 5), (2, 6, 6), (3, 1, 1), (3, 1, 2), (3, 1, 3), (3, 1,
 4), (3, 1, 5), (3, 1, 6), (3, 2, 1), (3, 2, 2), (3, 2, 3), (3,
2, 4), (3, 2, 5), (3, 2, 6), (3, 3, 1), (3, 3, 2), (3, 3, 3), (3
, 3, 4), (3, 3, 5), (3, 3, 6), (3, 4, 1), (3, 4, 2), (3, 4, 3),
(3, 4, 4), (3, 4, 5), (3, 4, 6), (3, 5, 1), (3, 5, 2), (3, 5, 3)
, (3, 5, 4), (3, 5, 5), (3, 5, 6), (3, 6, 1), (3, 6, 2), (3, 6,
3), (3, 6, 4), (3, 6, 5), (3, 6, 6), (4, 1, 1), (4, 1, 2), (4, 1
, 3), (4, 1, 4), (4, 1, 5), (4, 1, 6), (4, 2, 1), (4, 2, 2), (4,
 2, 3), (4, 2, 4), (4, 2, 5), (4, 2, 6), (4, 3, 1), (4, 3, 2), (
4, 3, 3), (4, 3, 4), (4, 3, 5), (4, 3, 6), (4, 4, 1), (4, 4, 2),
 (4, 4, 3), (4, 4, 4), (4, 4, 5), (4, 4, 6), (4, 5, 1), (4, 5, 2
), (4, 5, 3), (4, 5, 4), (4, 5, 5), (4, 5, 6), (4, 6, 1), (4, 6,
 2), (4, 6, 3), (4, 6, 4), (4, 6, 5), (4, 6, 6), (5, 1, 1), (5,
1, 2), (5, 1, 3), (5, 1, 4), (5, 1, 5), (5, 1, 6), (5, 2, 1), (5
, 2, 2), (5, 2, 3), (5, 2, 4), (5, 2, 5), (5, 2, 6), (5, 3, 1),
(5, 3, 2), (5, 3, 3), (5, 3, 4), (5, 3, 5), (5, 3, 6), (5, 4, 1)
, (5, 4, 2), (5, 4, 3), (5, 4, 4), (5, 4, 5), (5, 4, 6), (5, 5,
1), (5, 5, 2), (5, 5, 3), (5, 5, 4), (5, 5, 5), (5, 5, 6), (5, 6
, 1), (5, 6, 2), (5, 6, 3), (5, 6, 4), (5, 6, 5), (5, 6, 6), (6,
 1, 1), (6, 1, 2), (6, 1, 3), (6, 1, 4), (6, 1, 5), (6, 1, 6), (
6, 2, 1), (6, 2, 2), (6, 2, 3), (6, 2, 4), (6, 2, 5), (6, 2, 6),
 (6, 3, 1), (6, 3, 2), (6, 3, 3), (6, 3, 4), (6, 3, 5), (6, 3, 6
), (6, 4, 1), (6, 4, 2), (6, 4, 3), (6, 4, 4), (6, 4, 5), (6, 4,
 6), (6, 5, 1), (6, 5, 2), (6, 5, 3), (6, 5, 4), (6, 5, 5), (6,
5, 6), (6, 6, 1), (6, 6, 2), (6, 6, 3), (6, 6, 4), (6, 6, 5), (6
, 6, 6)]
56 von 216
p = 0.2592592592592924
```

→ *Hier werden nun aus Zahlenpaaren Zahlentripel.*
Ebenfalls werden nun die Tripel nicht mehr auf die Summe der Augenzahlen geprüft, sondern es wird jeweils geprüft, ob die Augenzahlen von Augenzahl1 bis hin zu Augenzahl 3 im jeweiligen Tripel der Größe nach aufsteigend sind.
Als günstige werden nun jene Tripel gezählt, auf die diese Eigenschaft, also dass die Augenzahlen darin aufsteigend sind, zutrifft.

4. Geändertes Programm (mit repeat arbeiten):

```python
import itertools
Augenzahlen = {1, 2, 3, 4, 5, 6}
Ω = list(itertools.product(Augenzahlen, repeat=3))
print(Ω)
günstig = 0
möglich = len(Ω)
for Augenzahl1, Augenzahl2, Augenzahl3 in Ω:
    if Augenzahl1 <= Augenzahl2 <= Augenzahl3:
        günstig = günstig + 1
print(f"{günstig} von {möglich}")
print(f"p = {günstig/möglich}")
```

```
(1, 5, 6), (1, 6, 1), (1, 6, 2), (1, 6, 3), (1, 6, 4), (1, 6, 5)
, (1, 6, 6), (2, 1, 1), (2, 1, 2), (2, 1, 3), (2, 1, 4), (2, 1,
5), (2, 1, 6), (2, 2, 1), (2, 2, 2), (2, 2, 3), (2, 2, 4), (2, 2
, 5), (2, 2, 6), (2, 3, 1), (2, 3, 2), (2, 3, 3), (2, 3, 4), (2,
 3, 5), (2, 3, 6), (2, 4, 1), (2, 4, 2), (2, 4, 3), (2, 4, 4), (
2, 4, 5), (2, 4, 6), (2, 5, 1), (2, 5, 2), (2, 5, 3), (2, 5, 4),
 (2, 5, 5), (2, 5, 6), (2, 6, 1), (2, 6, 2), (2, 6, 3), (2, 6, 4
), (2, 6, 5), (2, 6, 6), (3, 1, 1), (3, 1, 2), (3, 1, 3), (3, 1,
 4), (3, 1, 5), (3, 1, 6), (3, 2, 1), (3, 2, 2), (3, 2, 3), (3,
2, 4), (3, 2, 5), (3, 2, 6), (3, 3, 1), (3, 3, 2), (3, 3, 3), (3
, 3, 4), (3, 3, 5), (3, 3, 6), (3, 4, 1), (3, 4, 2), (3, 4, 3),
(3, 4, 4), (3, 4, 5), (3, 4, 6), (3, 5, 1), (3, 5, 2), (3, 5, 3)
, (3, 5, 4), (3, 5, 5), (3, 5, 6), (3, 6, 1), (3, 6, 2), (3, 6,
3), (3, 6, 4), (3, 6, 5), (3, 6, 6), (4, 1, 1), (4, 1, 2), (4, 1
, 3), (4, 1, 4), (4, 1, 5), (4, 1, 6), (4, 2, 1), (4, 2, 2), (4,
 2, 3), (4, 2, 4), (4, 2, 5), (4, 2, 6), (4, 3, 1), (4, 3, 2), (
4, 3, 3), (4, 3, 4), (4, 3, 5), (4, 3, 6), (4, 4, 1), (4, 4, 2),
 (4, 4, 3), (4, 4, 4), (4, 4, 5), (4, 4, 6), (4, 5, 1), (4, 5, 2
), (4, 5, 3), (4, 5, 4), (4, 5, 5), (4, 5, 6), (4, 6, 1), (4, 6,
 2), (4, 6, 3), (4, 6, 4), (4, 6, 5), (4, 6, 6), (5, 1, 1), (5,
1, 2), (5, 1, 3), (5, 1, 4), (5, 1, 5), (5, 1, 6), (5, 2, 1), (5
, 2, 2), (5, 2, 3), (5, 2, 4), (5, 2, 5), (5, 2, 6), (5, 3, 1),
(5, 3, 2), (5, 3, 3), (5, 3, 4), (5, 3, 5), (5, 3, 6), (5, 4, 1)
, (5, 4, 2), (5, 4, 3), (5, 4, 4), (5, 4, 5), (5, 4, 6), (5, 5,
1), (5, 5, 2), (5, 5, 3), (5, 5, 4), (5, 5, 5), (5, 5, 6), (5, 6
, 1), (5, 6, 2), (5, 6, 3), (5, 6, 4), (5, 6, 5), (5, 6, 6), (6,
 1, 1), (6, 1, 2), (6, 1, 3), (6, 1, 4), (6, 1, 5), (6, 1, 6), (
6, 2, 1), (6, 2, 2), (6, 2, 3), (6, 2, 4), (6, 2, 5), (6, 2, 6),
 (6, 3, 1), (6, 3, 2), (6, 3, 3), (6, 3, 4), (6, 3, 5), (6, 3, 6
), (6, 4, 1), (6, 4, 2), (6, 4, 3), (6, 4, 4), (6, 4, 5), (6, 4,
 6), (6, 5, 1), (6, 5, 2), (6, 5, 3), (6, 5, 4), (6, 5, 5), (6,
5, 6), (6, 6, 1), (6, 6, 2), (6, 6, 3), (6, 6, 4), (6, 6, 5), (6
, 6, 6)]
56 von 216
p = 0.25925925925925924
```

→ *Diese Schreibweise ist dann nützlich, wenn es eine hohe Anzahl an Variablen gibt. So muss man die Variable nur einmal eingeben und dann noch angeben wie oft man diese Variable haben will.*
Somit bekommt man automatisch die gewünschte Anzahl an Variablen, jedoch durch eine viel kürzere Eingabe, als wenn man die Variablen einzeln aufzählt.