Übungsblatt 6

1. Die folgenden Python3 Programme berechnen dieselben Resultate:
   Programm 1

```
import itertools
omega = set(itertools.product({"K", "Z"}, repeat=5))
E=0
for w in omega:
    E=E+w.count("K")
E=E/len(omega)
V=0
for w in omega:
    V=V+(w.count("K")-E)**2
V=V/len(omega)
print(f"Erwartungswert={E}")
print(f"Varianz={V}")
```

```
Erwartungswert=2.5
Varianz=1.25
```

   Programm 2:

```
import itertools
omega=set(itertools.product({"K", "Z"}, repeat = 5))
E=0
V=0
for w in omega:
    x=w.count("K")
    E=E+x
    V=V+x*x
E=E/len(omega)
V=V/len(omega)-E*E
print(f"Erwartungswert={E}")
print(f"Varianz={V}")
```

```
Erwartungswert=2.5
Varianz=1.25
```

(a) Überprüfen Sie, dass die Resultate übereinstimmen, auch für 6 oder 7 Würfe.
   Für 6 Würfe:
   Programm 1:

```
import itertools
omega = set(itertools.product({"K", "Z"}, repeat=6))
E=0
for w in omega:
    E=E+w.count("K")
E=E/len(omega)
V=0
for w in omega:
    V=V+(w.count("K")-E)**2
V=V/len(omega)
print(f"Erwartungswert={E}")
print(f"Varianz={V}")
```

```
Erwartungswert=3.0
Varianz=1.5
```

Programm 2:

```
import itertools
omega=set(itertools.product({"K", "Z"}, repeat = 6))
E=0
V=0
for w in omega:
    x=w.count("K")
    E=E+x
    V=V+x*x
E=E/len(omega)
V=V/len(omega)-E*E
print(f"Erwartungswert={E}")
print(f"Varianz={V}")
```

```
Erwartungswert=3.0
Varianz=1.5
```

Für 7 Würfe:

Programm 1:

```
import itertools
omega = set(itertools.product({"K", "Z"}, repeat=7))
E=0
for w in omega:
    E=E+w.count("K")
E=E/len(omega)
V=0
for w in omega:
    V=V+(w.count("K")-E)**2
V=V/len(omega)
print(f"Erwartungswert={E}")
print(f"Varianz={V}")
```

```
Erwartungswert=3.5
Varianz=1.75
```

Programm 2:

```
import itertools
omega=set(itertools.product({"K", "Z"}, repeat = 7))
E=0
V=0
for w in omega:
    x=w.count("K")
    E=E+x
    V=V+x*x
E=E/len(omega)
V=V/len(omega)-E*E
print(f"Erwartungswert={E}")
print(f"Varianz={V}")
```

```
Erwartungswert=3.5
Varianz=1.75
```

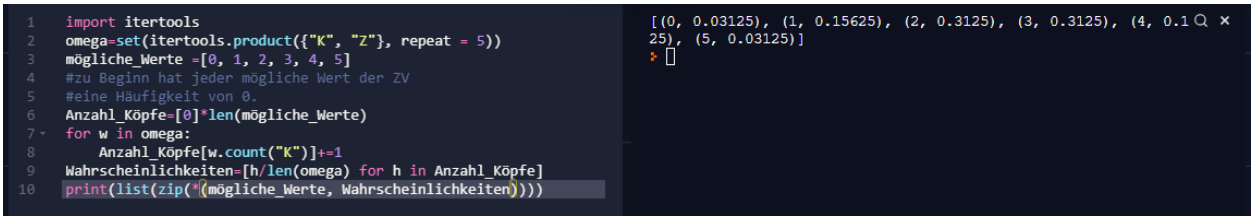
(b) Erklären Sie, wie die Programme funktionieren. Welche Unterschiede finden Sie zwischen denen?

Programm 1: Hier wird der Erwartungswert mit Null definiert. Anschließend wird die Funktion des Erwartungswertes aufgestellt. Man berechnet hier E+Anzahl Kopf dieses Ergebnis wird dann durch alle Möglichkeiten in Omega dividiert.
Für die Varianz wird auch eine Funktion aufgestellt. Diese lautet: die Varianz + die Anzahl an Kopf – Erwartungswert^2. Anschließend wird das Ergebnis durch die gesamte Anzahl (len) von omega dividiert.

Programm 2: Hier führt man neben w noch eine Variable x ein. Dieses x zählt uns nun die Anzahl an Kopf würfen, dadurch werden die Funktion übersichtlicher. Der Erwartungswert ist dadurch einfach E+x (im Grunde dasselbe wie vorher) und die Varianz bleibt auch die gleiche Formel, da wir unten in der letzten Varianzberechnung wieder dividieren und E*E abziehen.

Die beiden Programme unterscheiden sich also nur darin, dass wir die Funktionen zur Berechnung anders definieren. Einmal nützen wir eine Hilfsvariable einmal nicht.

2. Betrachten wir folgendes Programm:



```
import itertools
omega=set(itertools.product({"K", "Z"}, repeat = 5))
mögliche_Werte =[0, 1, 2, 3, 4, 5]
#zu Beginn hat jeder mögliche Wert der ZV
#eine Häufigkeit von 0.
Anzahl_Köpfe=[0]*len(mögliche_Werte)
for w in omega:
    Anzahl_Köpfe[w.count("K")]+=1
Wahrscheinlichkeiten=[h/len(omega) for h in Anzahl_Köpfe]
print(list(zip(*(mögliche_Werte, Wahrscheinlichkeiten))))
```

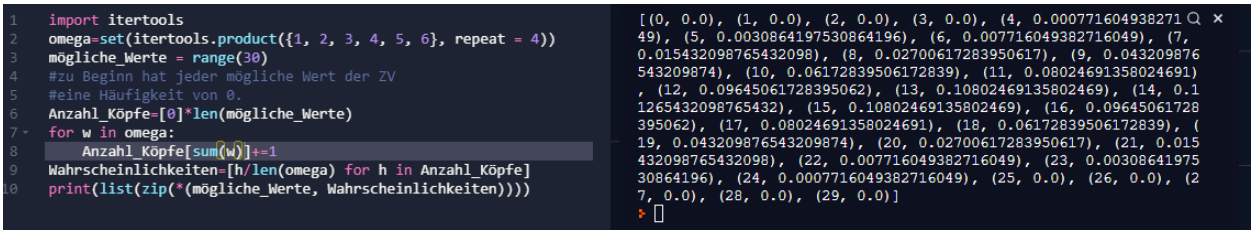
```
[(0, 0.03125), (1, 0.15625), (2, 0.3125), (3, 0.3125), (4, 0.1
25), (5, 0.03125)]
```

(a) Welche Berechnung wird vom Programm durchgeführt? Verteilungsfunktion und Wahrscheinlichkeitsfunktion
Die Wahrscheinlichkeit nie Kopf zu werden ist 0.03125 usw.

(b) Betrachten wir folgende Änderung:

```
2  Ω = set(itertools.product({1, 2, 3, 4, 5, 6}, repeat=4))
3  mögliche_Werte = range(30)

8      Anzahl_Köpfe[sum(ω)] += 1
```



```
import itertools
omega=set(itertools.product({1, 2, 3, 4, 5, 6}, repeat = 4))
mögliche_Werte = range(30)
#zu Beginn hat jeder mögliche Wert der ZV
#eine Häufigkeit von 0.
Anzahl_Köpfe=[0]*len(mögliche_Werte)
for w in omega:
    Anzahl_Köpfe[sum(w)]+=1
Wahrscheinlichkeiten=[h/len(omega) for h in Anzahl_Köpfe]
print(list(zip(*(mögliche_Werte, Wahrscheinlichkeiten))))
```

```
[(0, 0.0), (1, 0.0), (2, 0.0), (3, 0.0), (4, 0.000771604938271
49), (5, 0.0030864197530864196), (6, 0.007716049382716049), (7,
0.015432098765432098), (8, 0.02700617283950617), (9, 0.043209876
543209874), (10, 0.06172839506172839), (11, 0.08024691358024691)
, (12, 0.09645061728395062), (13, 0.10802469135802469), (14, 0.1
1265432098765432), (15, 0.10802469135802469), (16, 0.09645061728
395062), (17, 0.08024691358024691), (18, 0.06172839506172839), (
19, 0.043209876543209874), (20, 0.02700617283950617), (21, 0.015
432098765432098), (22, 0.007716049382716049), (23, 0.00308641975
30864196), (24, 0.000771604938271 6049), (25, 0.0), (26, 0.0), (2
7, 0.0), (28, 0.0), (29, 0.0)]
```

Erklären Sie das Resultat von diesem geänderten Programm!
Es zeigt uns wie wahrscheinlich es ist eine 0 zu haben, nämlich gar nicht – klar weil 0 nicht vorkommt. Es wird uns also in einer Spanne (range) von 0 bis 30 gezeigt, wie wahrscheinlich es das ein Fall davon eintritt.

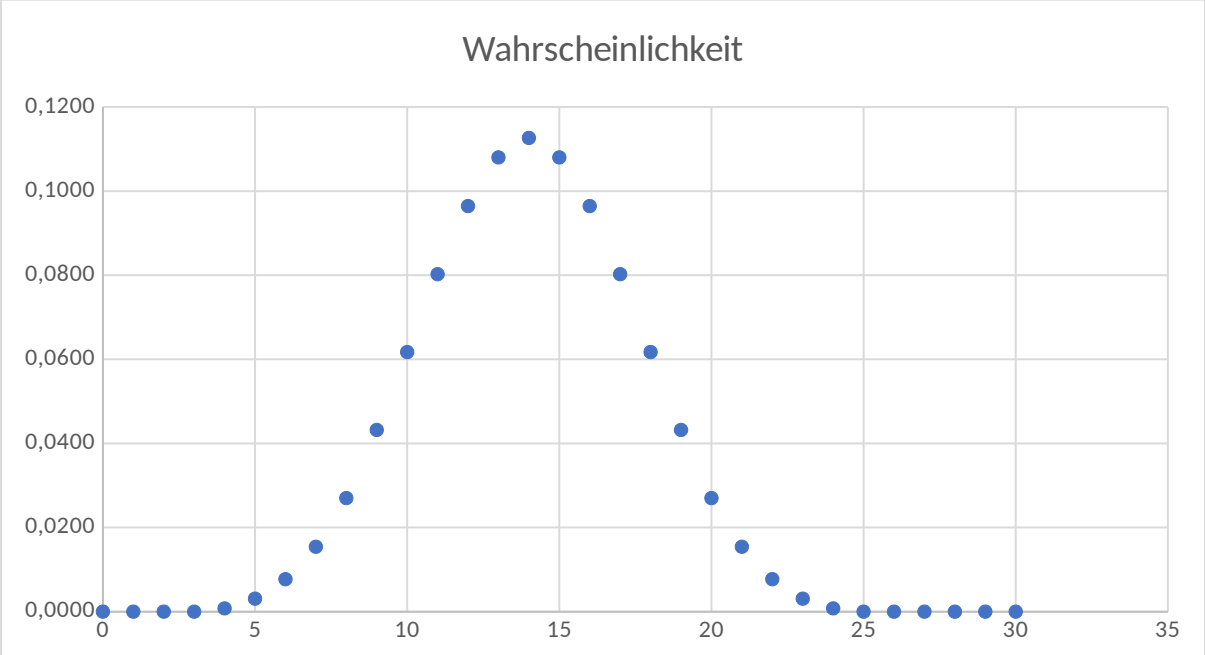
(c) Wir versuchen nun ein graphisches Diagramm zu erstellen. Deswegen betrachten wir folgende Änderung (statt der Zeile 10)

```
10  for w in zip(*(mögliche_Werte, Wahrscheinlichkeiten)):
11      print(f"{w[0]},{w[1]}")
```

Kopieren Sie das Output dieses Programms in ein Tabellenkalkulationsprogramm ein und erstellen Sie ein XY-Diagramm, sodass die Wahrscheinlichkeitsverteilung graphisch dargestellt wird! Tipp: Die Sprache der Tabelle muss auf Englisch geändert werden, sodass der Dezimalpunkt von Python richtig interpretiert wird. Hinweis: Google Tabellen kann die Beistriche auch als Trennungszeichen interpretieren, siehe **http://apps-experts.de/2016/03/neue-funktion-text-in-spalten-fuer-google-tabellen/**.

```python
import itertools
omega=set(itertools.product({1, 2, 3, 4, 5, 6}, repeat = 4))
mögliche_Werte = range(30)
#zu Beginn hat jeder mögliche Wert der ZV
#eine Häufigkeit von 0.
Anzahl_Köpfe=[0]*len(mögliche_Werte)
for w in omega:
    Anzahl_Köpfe[sum(w)]+=1
Wahrscheinlichkeiten=[h/len(omega) for h in Anzahl_Köpfe]
for w in zip(*(mögliche_Werte, Wahrscheinlichkeiten)):
    print(f"{w[0]}, {w[1]}")
```

```
0, 0.0
1, 0.0
2, 0.0
3, 0.0
4, 0.0007716049382716049
5, 0.0030864197530864196
6, 0.007716049382716049
7, 0.015432098765432098
8, 0.027006172839506173
9, 0.043209876543209874
10, 0.06172839506172839
11, 0.08024691358024691
12, 0.09645061728395062
13, 0.10802469135802469
14, 0.11265432098765432
15, 0.10802469135802469
16, 0.09645061728395062
17, 0.08024691358024691
18, 0.06172839506172839
19, 0.043209876543209874
20, 0.027006172839506173
21, 0.015432098765432098
22, 0.007716049382716049
23, 0.0030864197530864196
24, 0.0007716049382716049
25, 0.0
26, 0.0
27, 0.0
28, 0.0
29, 0.0
```



(d) Ergänzen Sie das Programm mit den nächsten Zeilen:

```python
Verteilung = [sum(Wahrscheinlichkeiten[:h]) for h in mögliche_Werte]
print(list(zip(*(mögliche_Werte, Verteilung))))
```

Erklären Sie, welche Berechnung hier durchgeführt wird.
Ich vermute, dass wir hier die Verteilungsfunktion aufstellen – vorher war es eine Wahrscheinlichkeitsfunktion.

```python
import itertools
omega=set(itertools.product({1, 2, 3, 4, 5, 6}, repeat = 4))
mögliche_Werte = range(30)
#zu Beginn hat jeder mögliche Wert der ZV
#eine Häufigkeit von 0.
Anzahl_Köpfe=[0]*len(mögliche_Werte)
for w in omega:
    Anzahl_Köpfe[sum(w)]+=1
Wahrscheinlichkeiten=[h/len(omega) for h in Anzahl_Köpfe]
for w in zip(*(mögliche_Werte, Wahrscheinlichkeiten)):
    print(f"{w[0]}, {w[1]}")
Verteilung =[sum(Wahrscheinlichkeiten[:h])for h in
    mögliche_Werte]
print(list(zip(*(mögliche_Werte, Verteilung))))
```

```
0, 0.0
1, 0.0
2, 0.0
3, 0.0
4, 0.0007716049382716049
5, 0.0030864197530864196
6, 0.007716049382716049
7, 0.015432098765432098
8, 0.02700617283950617
9, 0.043209876543209874
10, 0.06172839506172839
11, 0.08024691358024691
12, 0.09645061728395062
13, 0.10802469135802469
14, 0.11265432098765432
15, 0.10802469135802469
16, 0.09645061728395062
17, 0.08024691358024691
18, 0.06172839506172839
19, 0.043209876543209874
```

```
18, 0
19, 0
20, 0
21, 0
22, 0
23, 0
24, 0
25, 0
26, 0
27, 0
28, 0
29, 0
[(0,
93827
3), (
97222
1975)
 0.55
13580
0, 0.
25925
25, 1
```

```
0, 0.0
1, 0.0
2, 0.0
3, 0.0
0, 0.0
1, 0.0
2, 0.0
3, 0.0
4, 0.0007716049382716049
5, 0.0030864197530864196
6, 0.007716049382716049
7, 0.015432098765432098
8, 0.02700617283950617
9, 0.043209876543209874
10, 0.06172839506172839
11, 0.08024691358024691
12, 0.09645061728395062
13, 0.10802469135802469
14, 0.11265432098765432
15, 0.10802469135802469
16, 0.09645061728395062
17, 0.08024691358024691
```

```
0.0), (4, 0.0), (5, 0.000771604
3580245), (7, 0.011574074074074
, 0.05401234567901234), (10, 0.0
06172839506), (12, 0.239197530864
, (14, 0.4436728395061728), (15,
43518518518519), (17, 0.760802469
95), (19, 0.9027777777777779), (2
9729938271604939), (22, 0.9884259
542), (24, 0.9992283950617284), (
(28, 1.0), (29, 1.0)]
```