Übungsblatt 6

1. Programm in Python eingeben:

Programm1

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

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

Programm1

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

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

a) *Überprüfen ob Ergebnisse übereinstimmen:*

- *6 Würfe: Bei beiden Programmen gelangt man auf folgendes Ergebnis:*

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

- *7 Würfe: Bei beiden Programmen gelangt man auf folgendes Ergebnis:*

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

→ Man sieht also, dass die Ergebnisse für den Erwartungswert und die Varianz bei beiden Programmen immer identisch sind, unabhängig von der Anzahl der Würfe.

b) *Erklärung der Programme:*

- ***Funktionsweisen der Programme:***
  i. *Programm 1: Als erstes werden wieder die Sammlung der nützlichen Befehle, also die itertools, eingefügt, um diese auch verwenden zu*

können. Anschließend wird der Ereignisraum Ω definiert. Dieser soll der Menge aus den kartesischen Produkten von K und Z gebildet werden. Gleichzeitig wird auch festgelegt, dass insgesamt fünf Mal gewürfelt werden soll.
Nun wird die Variable E auf den Wert 0 gesetzt und es wird für alle ω im Ereignisraum festgelegt, dass sich die Variable E so zusammensetzt werden soll, dass zum aktuellen Wert des Erwartungswertes E die Anzahl der Kopfwürfe im Ereignisraum addiert wird. Anschließend wird noch festgelegt, dass der aktuelle, mögliche, Erwartungswert durch die Menge aller möglichen Ausgänge dividiert werden soll.
Nun wird auch noch die Variable V auf den Wert 0 gesetzt und es wird für alle ω im Ereignisraum festgelegt, dass die Variable V so zusammengesetzt werden soll, dass die Summe der Kopfwürfe minus dem Erwartungswert gerechnet wird und dieser Wert anschließend quadriert wird. Anschließend wird noch festgelegt, dass der aktuelle Wert für die Varianz V noch durch alle möglichen Ausgänge dividiert werden soll.
Zum Abschluss wird noch festgelegt, dass jeweils der Erwartungswert und die Varianz im Ausgabefeld ausgegeben werden sollen.

ii. *<u>Programm 1:</u> Als erstes werden wieder die Sammlung der nützlichen Befehle, also die itertools, eingefügt, um diese auch verwenden zu können. Anschließend wird der Ereignisraum Ω definiert. Dieser soll der Menge aus den kartesischen Produkten von K und Z gebildet werden. Gleichzeitig wird auch festgelegt, dass insgesamt fünf Mal gewürfelt werden soll.*
Nun werden die Variablen E und V auf den Wert 0 gesetzt.
Anschließend wird definiert, dass x dafür steht, dass die Kopfwürfe der Ausgänge gezählt werden.
Nun wird definiert, dass sich die Variable E so zusammensetzt werden soll, dass zum aktuellen Wert des Erwartungswertes E die Anzahl der Kopfwürfe im Ereignisraum addiert wird und dass die Variable V so zusammengesetzt werden soll, dass die Summe der Kopfwürfe minus dem Erwartungswert gerechnet wird und dieser Wert anschließend quadriert wird.
Danach wird noch festgelegt, dass der Erwartungswert durch die Anzahl aller möglichen Werte dividiert werden sollen und dass die Varianz auch durch die Anzahl der möglichen Würfe dividiert werden sollte und von diesem Wert noch der quadrierte Erwartungswert abgezogen werden soll.
Zum Abschluss wird noch festgelegt, dass jeweils der Erwartungswert und die Varianz im Ausgabefeld ausgegeben werden sollen.

- ***<u>Unterschied der beiden Programme:</u>***
  Im Großen und Ganzen machen beiden Programme eigentlich meistens dasselbe, nur in unterschiedlicher Reihenfolge.
  Zum Beispiel wird im ersten Programm zuerst der Erwartungswert ganz

berechnet und dann erst die Varianz. Im zweiten Programm hingegen wird werden alle Schritte immer für E und V gleich hintereinander durchgeführt. Zum Beispiel werden E und V gleich hintereinander auf 0 gesetzt, dann wird der nächste Schritt für wieder beide Variablen durchgeführt.
Ein zweiter Unterschied der beiden Programme ist, dass im zweiten Programm der Befehl ω.count("K") als x definiert wird, um diesen Befehl in weiteren Schritten nicht immer ausschreiben zu müssen, sondern einfach x schreiben zu können.
Einen dritten kleinen Unterschied gibt es noch, und zwar, dass im zweiten Programm der Schritt, dass der quadrierte Erwartungswert von der Varianz in einem anderen Schritt durchgeführt wird, nämlich erst beim dividieren der Varianz durch die Anzahl an möglichen Ausgänge.

2. Programm in Python eingeben:

```
1  import itertools
2  Ω = set(itertools.product({"K", "Z"}, repeat=5))
3  mögliche_Werte = [0, 1, 2, 3, 4, 5]
4  Anzahl_Köpfe = [0]*len(mögliche_Werte)
5  for ω in Ω:
6      Anzahl_Köpfe[ω.count("K")] += 1
7  Wahrscheinlichkeiten = [h/len(Ω) for h in Anzahl_Köpfe]
8  print(list(zip(*(mögliche_Werte, Wahrscheinlichkeiten))))
9
```

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

a) ***Durchgeführte Berechnung:***

Das Programm rechnet jeweils die Wahrscheinlichkeit dafür aus, wie oft eine gewisse Anzahl an Kopf geworfen wird. Also die Wahrscheinlichkeit, dass bei fünf Würfen 0-mal, 1-mal, 2-mal, 3-mal, 4-mal, 5-mal Kopf geworfen wird.
Hier kommt heraus, dass 2 und 3 Kopfwürfe mit einer Wahrscheinlichkeit von 31,25% am wahrscheinlichsten sind und 0 und 5 Kopfwürfe mit einer Wahrscheinlichkeit von 3,125% am unwahrscheinlichsten sind.

b) ***Programm mit Änderung in Zeile 2,3,6:***

```
1  import itertools
2  Ω = set(itertools.product({1, 2, 3, 4, 5, 6}, repeat=4))
3  mögliche_Werte = range(30)
4  Anzahl_Köpfe = [0]*len(mögliche_Werte)
5  for ω in Ω:
6      Anzahl_Köpfe[sum(ω)] += 1
7  Wahrscheinlichkeiten = [h/len(Ω) for h in Anzahl_Köpfe]
8  print(list(zip(*(mögliche_Werte, Wahrscheinlichkeiten))))
9
```

```
[(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, 0.0007716049382716049), (25, 0.0), (26, 0.0), (2
7, 0.0), (28, 0.0), (29, 0.0)]
```

→ *Dieses Programm berechnet nun die Wahrscheinlichkeiten für die Summe der Augenzahlen eines Würfels bei viermaligem Würfel.*
Es wird also ein Würfel 4-mal geworfen und die Augensummen der 4 Würfe werden addiert. Anschließend wird die Wahrscheinlichkeit für jeden möglichen Augensummen-Wert berechnet.

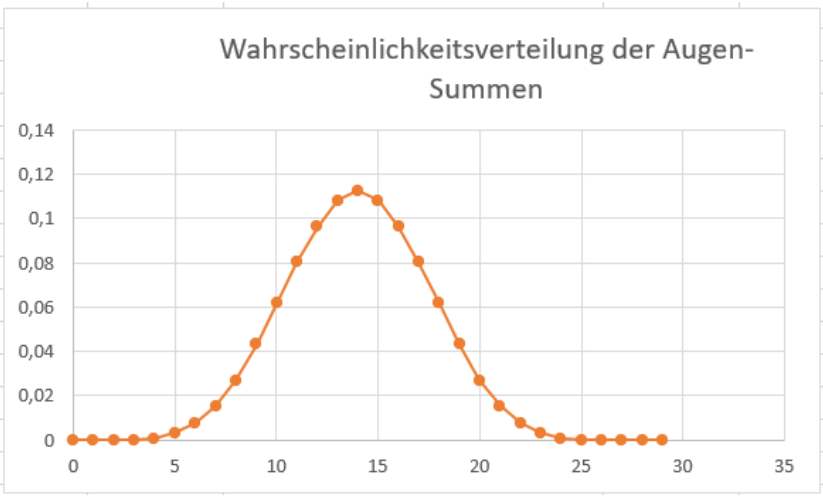
Hier kommt heraus, dass die Wahrscheinlichkeit, dass die 4 Augensummen addiert den Wert 0, 1, 2, 3, 25, 26, 27, 28, 29 ergeben bei 0% liegt, also dass diese Summen bei 4 Würfen nicht vorkommen kann.
Dass die Summe der 4 Augenzahlen bei den 4 Würfen 14 ergibt, ist mit einer Wahrscheinlichkeit von 11,27% am höchsten. Das bedeutet, dass es bei 4 Würfen am wahrscheinlichsten ist, eine Gesamt-Augensumme von 14 zu erhalten.

c) ***Programm mit Änderung in Zeile 8,9 + grafische Darstellung:***

```
1  import itertools
2  Ω = set(itertools.product({1, 2, 3, 4, 5, 6}, repeat=4))
3  mögliche_Werte = range(30)
4  Anzahl_Köpfe = [0]*len(mögliche_Werte)
5  for ω in Ω:
6      Anzahl_Köpfe[sum(ω)] += 1
7  Wahrscheinlichkeiten = [h/len(Ω) for h in Anzahl_Köpfe]
8  for w in zip(*(mögliche_Werte, Wahrscheinlichkeiten)):
9      print(f"{w[0]},{w[1]}")
10
```

```
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
20,0.02700617283950617
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
```

Grafische Darstellung des Outputs mit Hilfe von Excel:



→ Die grafische Darstellung lässt schön erkennen, dass es sich bei der Häufigkeit der Augenzahlen-Summen um eine Normalverteilung handelt.

d) ***Programm mit Ergänzung durch Zeile 10 und 11:***

```
1  import itertools
2  Ω = set(itertools.product({1, 2, 3, 4, 5, 6}, repeat=4))
3  mögliche_Werte = range(30)
4  Anzahl_Köpfe = [0]*len(mögliche_Werte)
5  for ω in Ω:
6      Anzahl_Köpfe[sum(ω)] += 1
7  Wahrscheinlichkeiten = [h/len(Ω) for h in Anzahl_Köpfe]
8  for w in zip(*(mögliche_Werte, Wahrscheinlichkeiten)):
9      print(f"{w[0]},{w[1]}")
10 Verteilung = [sum(Wahrscheinlichkeiten[:h]) for h in
       mögliche_Werte]
11 print(list(zip(*(mögliche_Werte, Verteilung))))
12
```

```
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
20,0.02700617283950617
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
[(0, 0), (1, 0.0), (2, 0.0), (3, 0.0), (4, 0.0), (5, 0.0007716049382716049), (6, 0.0038580246913580245), (7, 0.011574074074074073), (8, 0.027006172839506172), (9, 0.05401234567901234), (10, 0.09722222222222221), (11, 0.1589506172839506), (12, 0.2391975308641975), (13, 0.33564814814814814), (14, 0.4436728395061728), (15, 0.5563271604938271), (16, 0.6643518518518519), (17, 0.7608024691358025), (18, 0.8410493827160495), (19, 0.9027777777777779), (20, 0.9459876543209877), (21, 0.9729938271604939), (22, 0.9884259259259259), (23, 0.996141975308642), (24, 0.9992283950617284), (25, 1.0), (26, 1.0), (27, 1.0), (28, 1.0), (29, 1.0)]
```

→ *Dieses Programm berechnet für jede Augenzahlen-Summe, wie hoch die Wahrscheinlichkeit dafür ist, dass die Summe der gewürfelten Augenzahlen kleiner als diese Augenzahl ist.*
Zum Beispiel: Die Augenzahlen-Summe, für die die Berechnung durchgeführt wird, ist 7. Das Programm berechnet nun die Wahrscheinlichkeit dafür, dass bei den 4 Würfen die addierten Augensummen einen Wert niedriger als 7 annehmen.

Aus diesem Grund ist das Ergebnis für die Werte bis 4 auch gleich 0, da die niedrigste Summe bei 4 Würfen 4 ist und somit die Augenzahlen-Summe 4 nicht unterschritten werden kann.

Und der Grund wieso bei allen Werten ab 25 das Ergebnis 1 lautet ist jener, dass die Augenzahlen-Summe bei 4 Würfen höchstens den Wert 24 annehmen kann. Somit ist die Wahrscheinlichkeit, dass man eine Augenzahlen-Summe kleiner als 25 erhält gleich 100%.