1 a)

```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
```

```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}")

#import itertools
#Ω = set(itertools.product({"K", "Z"}, repeat=7))
```

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

```python
import itertools
Ω = set(itertools.product({"K", "Z"}, repeat=6))
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 = 3.0
Varianz = 1.5
```

```python
import itertools
Ω = set(itertools.product({"K", "Z"}, repeat=6))
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}")

#import itertools
```

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

```python
import itertools
Ω = set(itertools.product({"K", "Z"}, repeat=7))
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 = 3.5
Varianz = 1.75
```

```python
import itertools
Ω = set(itertools.product({"K", "Z"}, repeat=7))
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 = 3.5
Varianz = 1.75
```

1 b)

```
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}")
```

```
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}")
```

Bis zur Zeile 4 sind die Befehle gleich. In der Zeile 4 hingegen berechnet das erste Programm den Erwartungswert Auch die Berechnung des Erwartungswertes ist bis auf den „Zeitpunkt“ der Berechnung identisch. Die Varianz wird hingegen auf zwei verschiedene Arten berechnet, da beim ersten Programm V=V+(count(K)-E)^2 und beim anderen zuerst x definiert wurde und dann der Erwartungswert und die Varianz gebildet wird.

Meiner Meinung nach ist das zweite Programm effizienter und für auch zu weniger Schwierigkeiten in der Berechnung und beim Eintippen der Befehle, da eine weitere Variable „x“ definiert wurde.

Zwischenfrage:

Da ich mir nicht sicher war, weshalb in der Zeile 9 zwei Sterne/(Multiplikationszeichen??) verwendet wurden, versuchte ich eins zu entfernen. Dadurch änderte sich die Varianz auf 0.

```
1   import itertools
2   Ω = set(itertools.product({"K", "Z"}, repeat=7))
3   E = 0
4   for ω in Ω:
5       E = E + ω.count("K")
6   E = E / len(Ω)
7   V = 0
8   for ω in Ω:
9       V = V + (ω.count("K")-E)*2
10  V = V / len(Ω)
11  print(f"Erwartungswert = {E}")
12  print(f"Varianz = {V}")
13
14  b=0
15  b=b+2
16  b=b**3
17  print(b)
```

```
Erwartungswert = 3.5
Varianz = 0.0
8
```

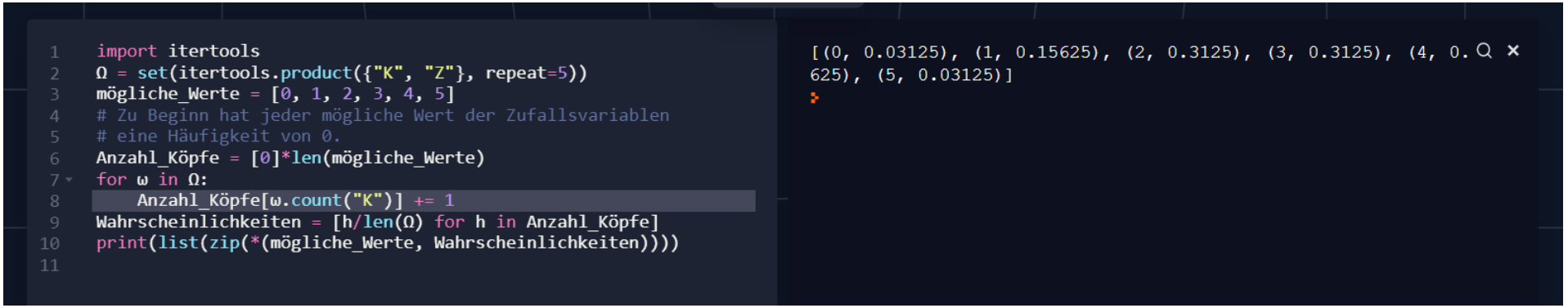
Antwort:

Dabei handelte es sich um das Exponenten Zeichen.

```python
import itertools
Ω = set(itertools.product({"K", "Z"}, repeat=7))
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 = 3.5
Varianz = 0.0
```

2 a)



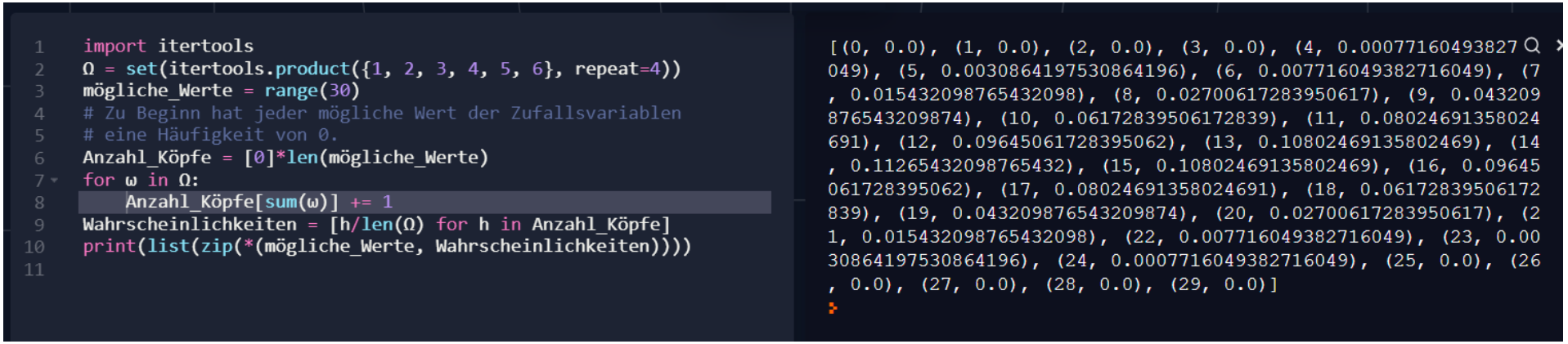
```python
import itertools
Ω = set(itertools.product({"K", "Z"}, repeat=5))
mögliche_Werte = [0, 1, 2, 3, 4, 5]
# Zu Beginn hat jeder mögliche Wert der Zufallsvariablen
# eine Häufigkeit von 0.
Anzahl_Köpfe = [0]*len(mögliche_Werte)
for ω in Ω:
    Anzahl_Köpfe[ω.count("K")] += 1
Wahrscheinlichkeiten = [h/len(Ω) 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.
625), (5, 0.03125)]
```

Das Programm berechnet die Wahrscheinlichkeiten, wie oft Kopf geworden wird. Dabei werden zu der Anzahl die jeweilige Wahrscheinlichkeit abgebildet. So hat die Wsk 0-mal Kopf zu würfeln die Wahrscheinlichkeit von 0.03125.

2 b)



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

```

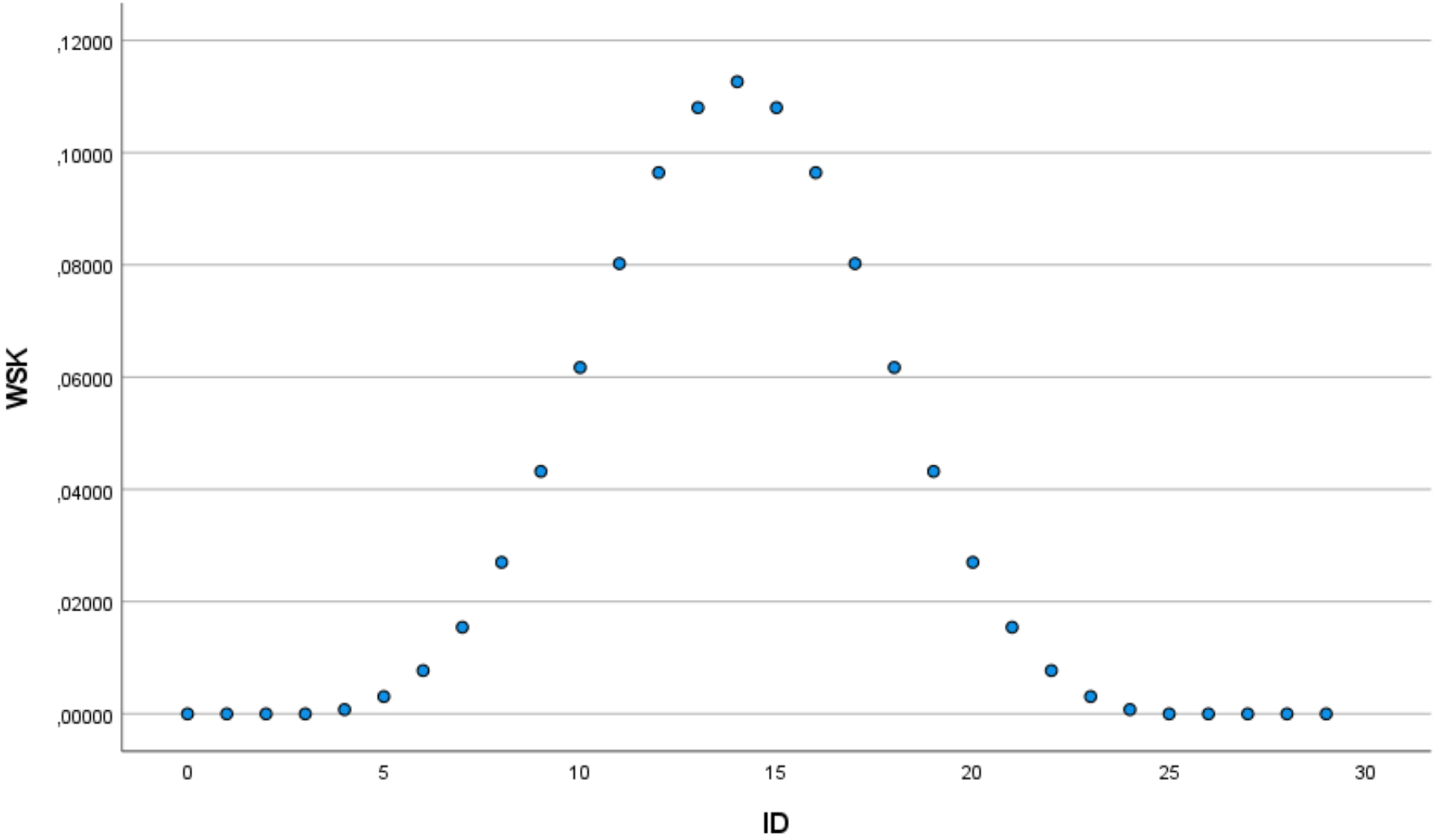
```
[(0, 0.0), (1, 0.0), (2, 0.0), (3, 0.0), (4, 0.00077160493827
049), (5, 0.0030864197530864196), (6, 0.007716049382716049), (7
, 0.015432098765432098), (8, 0.02700617283950617), (9, 0.043209
876543209874), (10, 0.06172839506172839), (11, 0.08024691358024
691), (12, 0.09645061728395062), (13, 0.10802469135802469), (14
, 0.11265432098765432), (15, 0.10802469135802469), (16, 0.09645
061728395062), (17, 0.08024691358024691), (18, 0.06172839506172
839), (19, 0.043209876543209874), (20, 0.02700617283950617), (2
1, 0.015432098765432098), (22, 0.007716049382716049), (23, 0.00
30864197530864196), (24, 0.0007716049382716049), (25, 0.0), (26
, 0.0), (27, 0.0), (28, 0.0), (29, 0.0)]
```

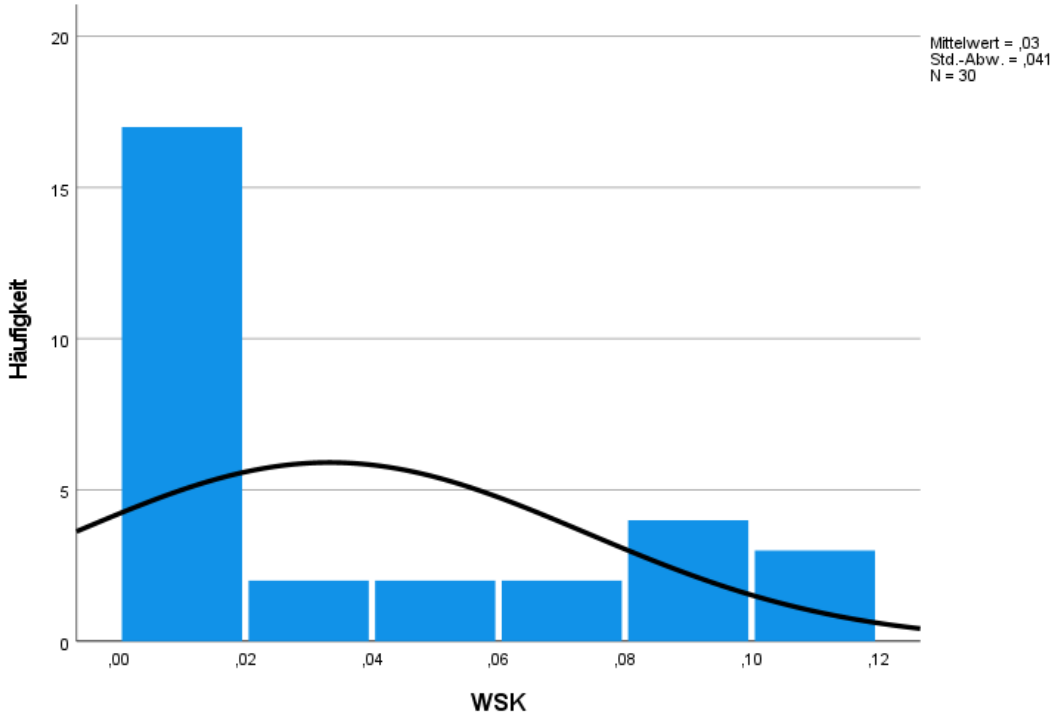
Nun handelt es sich nicht mehr um eine Münze sondern um einen Würfel, der 4-mal geworfen wird und man die Wahrscheinlichkeiten der Augensummen berechnet. So sind z.B. die Werte 0 bis 3 und 25 bis Unendlich bei 4-mal würfeln nicht möglich.

2 c) Normalverteilung

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

```
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
```


,12000
,10000
,08000
,06000
,04000
,02000
,00000
WSK
0
5
10
15
20
25
30
ID




Die Darstellung im SPSS wurde durch die Werte 0 gefälscht.

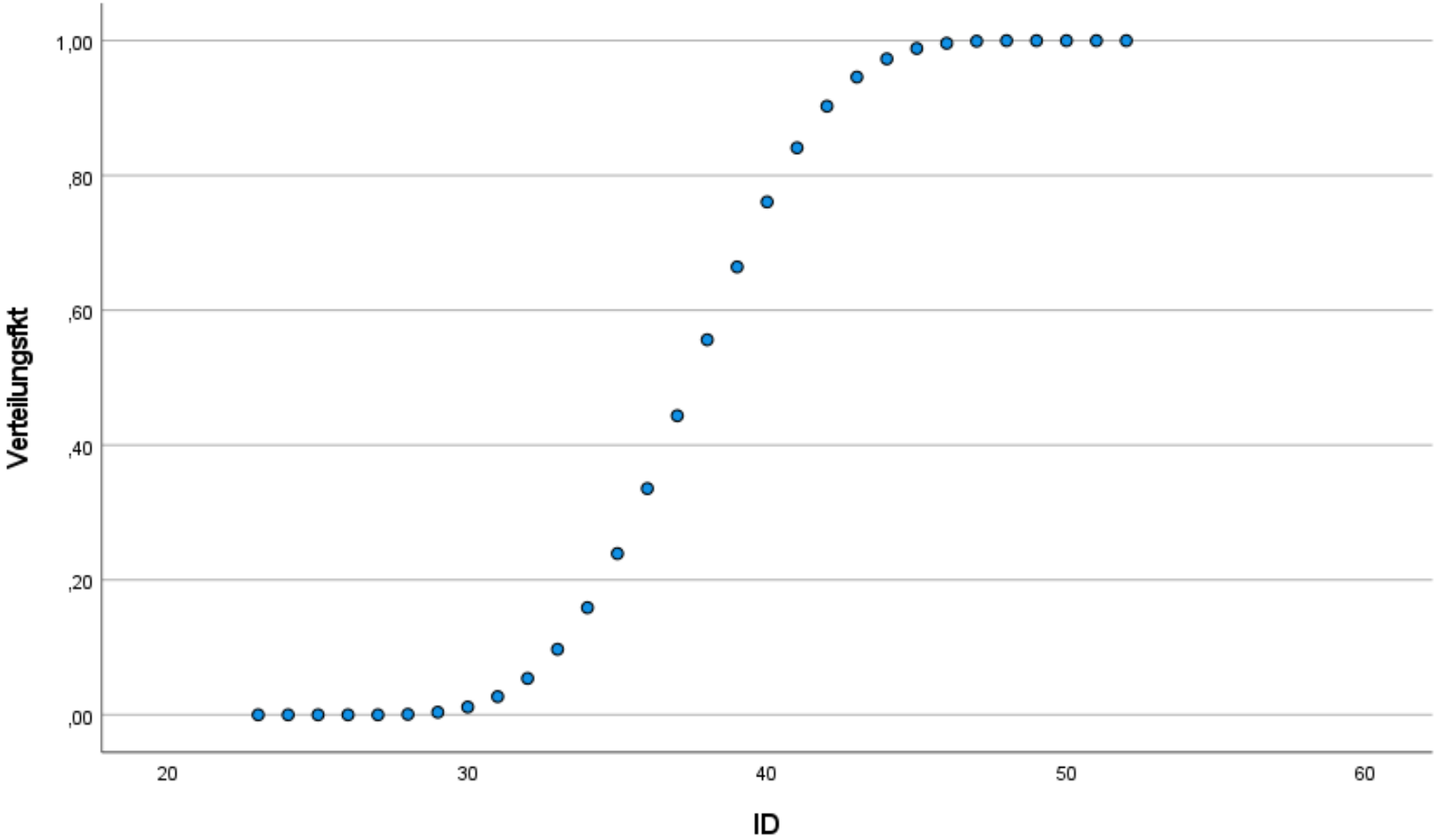
Findet keine Verwendung

2 d)

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

```
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
23,0.003086419753086419
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, 0.02700617283950617), (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)]
```

Hierbei handelt es sich um die kumulierte Häufigkeitsverteilung bzw. Verteilungsfunktion.


1,00
,80
,60
,40
,20
,00
Verteilungsfkt
20
30
40
50
60
ID