

# Risiko

Wir wollen hier eine Form des Glücksspiels mit Würfeln untersuchen. (Ganz aktuell gibt es sogar eine 'Herr der Ringe'-Version.)

## Die Regel:

Es gibt zwei Spieler, den **Angreifer** und den **Verteidiger**. Der Angreifer wirft mit 3 Würfeln, wobei die größten beiden geworfenen Augenzahlen gelten. Der Verteidiger wirft 2 Würfel.

Nun werden die höchsten Augenzahlen des Angreifers und des Verteidigers verglichen. Ist die des Angreifers höher, so bekommt er einen Punkt, andernfalls erhält ihn der Verteidiger. Dann verfährt man mit den beiden anderen Augenzahlen analog.

Bsp:

Angreifer: 5,1,3; Verteidiger 3,4

Erster Vergleich A5 gegen V4, der Angreifer erhält einen Punkt.

Zweiter Vergleich A3 gegen V3, der Verteidiger bekommt den Punkt.

Angreifer: 2,4,6; Verteidiger 5,2

Erster Vergleich A6 gegen V5, der Angreifer erhält einen Punkt.

Zweiter Vergleich A4 gegen V2, der Angreifer bekommt noch einen Punkt.

Angreifer: 4,4,3; Verteidiger 4,4

Erster Vergleich A4 gegen V4, der Verteidiger erhält einen Punkt.

Zweiter Vergleich A4 gegen V4, der Verteidiger bekommt noch einen Punkt.

## Überlegung:

Hätten beide Spieler je 2 Würfel und die höhere Zahl gewinnt, wäre das Spiel offensichtlich fair, jeder der beiden hätte die gleichen Gewinnchancen. Allerdings gibt es Gleichstände wie 3 zu 3, wo keiner gewinnt.

Bei 'Risiko' ist die Gewinnbestimmung etwas verändert – bei Gleichstand gewinnt der Verteidiger. Damit das aber kein einseitiger Vorteil wird, bekommt auch der Angreifer einen Bonus in Form des dritten Würfels. Eine einzelne geworfene kleine Zahl ist damit für ihn ohne Auswirkung, da sie nicht gewertet wird.

Nun stellt sich die Frage: halten sich die beiden Vorteile die Waage, oder zieht einer der beiden Kontrahenten größeren Nutzen aus der Spielregel?

## Lösung:

Ein kurzer Überschlag: 5 Würfel sind im Spiel, das macht  $6^5 = 7776$  mögliche Ausgänge. Die Zahl ist nicht zu groß – wir simulieren einfach alle Möglichkeiten mit Python!

Erste Lösung **risiko1()**:

mit geschachtelten Schleifen, für jeden Würfel eine. Sortierung der Augenzahlen manuell mithilfe von if-Abfragen (In jede Programmiersprache übertragbar).

Zweite Lösung **risiko2()**:

wir erstellen, da es ja nicht allzu viele Möglichkeiten sind, praktischerweise eine Liste aller möglichen Würfe im Voraus. Und kombinieren dann jeden Angreiferwurf mit jedem Verteidigerwurf. Die Sortierung lassen wir von Python selbst durchführen. (Das wird in anderen Sprachen komplizierter). Vergleich die Quelltexte – welche Version ist verständlicher?

```
#####
#
# Simulation des Würfelspiels 'Risiko':
# Ergebnis 8391 7161 53.9544753086
# Der Angreifer hat 8391 Punkte, der Verteidiger 7161 Punkte. Der Angreifer gewinnt also zu fast 54%
#
#####
```

```
def risikol():
    ap, vp = 0, 0 # Punkte für Angreifer und Verteidiger
    augen = [1, 2, 3, 4, 5, 6]
    for a1 in augen:
        for a2 in augen:
            for a3 in augen:
                aa1, aa2, aa3 = a1, a2, a3 # Angreifer-Augenzahlen sortieren
                if aa1 < aa2: aa1, aa2 = aa2, aa1
                if aa2 < aa3: aa2, aa3 = aa3, aa2
                if aa1 < aa2: aa1, aa2 = aa2, aa1

                for v1 in augen:
                    for v2 in augen:
                        vv1, vv2 = v1, v2 # Verteidiger-Augenzahlen sortieren
                        if vv1 < vv2: vv1, vv2 = vv2, vv1

                # Auswertung
                if aa1 > vv1: ap = ap + 1
                else: vp = vp + 1
                if aa2 > vv2: ap = ap + 1
                else: vp = vp + 1

    print ap, vp, float(ap)/(ap+vp)*100
```

```
#####
```

```
def risiko2():
    ap, vp = 0, 0
    augen = [1, 2, 3, 4, 5, 6]

    # bilde Listen aller möglichen Würfe für Angreifer und Verteidiger
    angriff = [[x, y, z] for x in augen for y in augen for z in augen]
    verteidigung = [[x, y] for x in augen for y in augen]

    for a in angriff:
        a.sort() # Angreifer-Augenzahlen steigend sortieren
        a.reverse() # umdrehen, damit fallend
        for v in verteidigung:
            v.sort() # genauso für den Verteidiger
            v.reverse()

            # Auswertung
            if a[0] > v[0]: ap = ap + 1
            else: vp = vp + 1
            if a[1] > v[1]: ap = ap + 1
            else: vp = vp + 1

    print ap, vp, float(ap)/(ap+vp)*100
```

```
#####
```