1. **Einlesen des Daten**

Das Programm wird in Python implementiert.

Z. 4: Hier wird die Daten eingelesen und in list1 hinzugefügt.

2. **Deklaration wichtiger Variablen**

Z. 5: In der Liste players_starke werden dann Spielstärken im int-Format eingetragen.

Z.10: In dieser Methode habe ich überwiegend das Format dictionary verwendet. plays_dict ist darauffolgend eine dictionary der Spieler, da die Spieler von 0 nummeriert sind und deren values sind jeweils die Spielstärken.
z.B. {0: 10, 1: 10, 2: 10, 3: 10, 4: 80, 5: 80, 6: 80, 7: 100}

Z. 13: der stärkste Spieler.

Z. 15: players ist dann Liste der Spieler, z.B. [0, 1, 2, 3, 4, 5, 6, 7]

Z. 16: wir mischen die players Liste.

3. **Methoden**

- showdown: diese Funktion stellt den eigentlichen Spielverlauf dar. Zwar nimmt sie eine dictionary von zwei Spieler {spieler1:starke1, spieler2:starke2}, vergleicht deren Spielstärken nach Tobis Regeln und gibt den Sieger wieder. Und zwar wie folgt:
  r ist eine zufällige Zahl zwischen 0 und 1, steht für die Wahrscheinlichkeit.
  list(dic.values())[0] gibt starke1 wieder.
  sum(dic.values()) gibt starke1+starke2.
  r < list(dic.values())[0]/sum(dic.values()) heißt, dass die Wahrscheinlichkeit, einen Kugel des Spielers 1 herauszuziehen größer als r, die zufällig generierte Zahl, ist. Daher gewinnt Spieler 1 und umgekehrt.

- liga: nimmt plays_dict als Übergabe, z.B. `{0: 10, 1: 10, 2: 10, 3: 10, 4: 80, 5: 80, 6: 80, 7: 100}`
  Z. 30: dieser Generator schafft dictionaries von 2 Spielern. Da in diesem Turnier soll jeder Spieler einmal gegen jeden anderen Spieler, habe ich die Funktion itertools.combinations() verwendet. Sie erzeugt eine Liste von N x

(N-1)/2 Elemente, mit N der Anzahl der Spieler. Allerdings wenn die combinations() Funktion die Dictionaries als Übergabe bekommt, werden nur keys wiedergegeben. Um wieder dictionaries zu erhalten, habe ich eine schöne dictionary comprehension angewendet:

```
[{j: dict_of_plays[j] for j in i} for i in …
>> [{0: 10, 1: 10}, {0: 10, 2: 10}, {0: 10, 3: 10}, {0: 10, 4: 80},...
```

Bei jedem Element in dieser Liste wird dann die Methode showdown aufgerufen und der Spieler wird in die Liste winners hinzugefügt. Danach wird in Zeile 36 gezählt, welcher Spieler wie viel Mal gewonnen hat, und zwar im dictionary Format, z.B. {1:3, 2:5} deutet darauf hin, der Spieler1 hat 3 Mal gewonnen und Spieler2 5 Mal insgesamt. In Zeile 39 wird gesucht, welchen Spieler hat die größte value, (max(win_time.values())) und gibt den zurück. In einigen Spielen kann es zu einem Unentschieden kommen, von daher die min() Methode, um den Spieler mit der niedrigen Spielnummer wiederzugeben.

- KO: nimmt vermischte players als Übergabe, z.B. [2, 6, 7, 5, 4, 1, 3, 0]
  Z. 51: die oben genannte Liste wird in zweier Gruppen eingeteilt, jede Gruppe enthält wiederum 2 dictionaries.
  Z. 55: diese wird iteriert, durch die Methode showdown() gefiltert und der Gewinner werden in eine Liste hinzugefügt.
  Diese Methode ko() wiederholt sich rekursive, da die Liste als Parameter immer eine gerade Anzahl von Elemente hat, bis nur ein Gewinner übrig bleibt.

- KO_5: nimmt vermischte players als Übergabe, z.B. [2, 6, 7, 5, 4, 1, 3, 0]
  Z. 71: die oben genannte Liste wird in zweier Gruppen eingeteilt, die jeweils 2 dictionaries enthält.
  Z. 75: hier spielen die Spieler 5 Mal gegeneinander. Der mit der meisten Siegen, zwar `max(set(one_time_wins), key = one_time_wins.count)` kommt dann in die Gewinner-Liste.
  Z. 83: ähnlich wie die obere Funktion wiederholt sich diese auch rekursiv weiter.

- count_win_of_starkster: nimmt die Spielvarianten, also entweder liga() oder ko(), oder ko_5() und num_of_plays, wie oft das Turnier wiederholt wird. Diese Funktion zählt nun, wie viel Mal der stärkste Spieler über eine beliebige Anzahl der Wiederholungen gewinnt. Hier gibt es ein if-else statement

aufgrund der verschiedenen Parametertypen zwischen liga() und den anderen Methoden.
Beispiel:

```
x = count_wins_of_starkster(ko, 100)
>> 36
```

Diese deutet darauf hin, dass der stärkste Spieler die Wahrscheinlichkeit von ungefähr 36% hat, das KO-Turniervariant zu gewinnen.

Anmerkung:
Die Methode an sich finde ich ziemlich verständnisvoll. Ein Problem gibt es trotzdem: dictionaries. Die Entscheidung, dictionaries anzuwenden ist die falsche Entscheidung, zumindest bei dieser Aufgabe. Es ist mir später bei Aufgabe 5 aufgefallen, dass einfach Listen verwendet worden sein sollten, da hätte ich sowas wie `[dic.items()][0]` nicht im Spiel haben müssen, stattdessen könnte ich mich auf die Position des Elements in der Liste beziehen, z.B. `liste[0]`.