

# Modularisierung 2.0

---

Claas de Boer, Tilman Hinnerichs

21. Januar 2021

Python-Grundlagen

## Rückblick: Einbinden von Externen Bibliotheken

---

Auslagern von Code in nach Verantwortlichkeiten sortierte Module und Packages ermöglicht es eine übersichtliche Code Base aufzubauen.

- Modul := Python Datei **.py**
- Package := „Ordner“ von Modulen

# Importieren eines Moduls

Zum Importieren eines Moduls/Package kann das `import` statement verwendet werden.

```
1 # Import des gesamten Inhalts eines Moduls/PKG
2 # **mit** dem Namespace des importierten Moduls
3 import random                # -> aufruf: random.randint()
4 import random as rnd         # -> aufruf: rnd.randint()
5
6 # Import des gesamten Inhalts eines Moduls/PKG
7 # **in** den Namespace des importierenden Moduls
8 from random import *         # -> aufruf: randint()
9
10 # Import einer einzelnen Funktion
11 # **in** den Namespace des importierenden Moduls
12 from random import randint   # -> aufruf: randint()
13
```

# Import Hilfen

```
1 import random
2
3 # Der Klassiker: help()
4 help(random)
5
6 # Liste alle importierten Variablen, Funktionen, und Klassen auf
7 dir(random)
8
9 # Gib den Speicherort des importierten Moduls aus
10 print(random.__file__)
```

# Import Pitfalls

Module `example.py`

```
1 CONSTANT = 1
2
3 # Code der ausgeführt wird, wenn example.py ausgeführt wird
4 print("Hallo Mama, ich bin in einer Präsentation!")
```

```
1 >>> import example
2 >>> Hallo Mama!
3 >>>
```

Wie können wir es vermeiden Code beim Import auszuführen?

# Boilerplate Code

Module `example.py`

```
1 CONSTANT = 1
2
3 # Code der nur ausgeführt wird,
4 # wenn example.py direkt ausgeführt wird
5 if __name__ == "__main__":
6     print("Hallo Mama, ich bin in einer Präsentation!")
```

```
1 >>> import example
2 >>>
```

Das in `example.py` enthaltene `print`-Statement wird nur ausgeführt, wenn `example.py` direkt ausgeführt wird.

# Ein eigenes Package

```
[claas ~/ex]€ tree .  
.  
├── example.py  
└── package  
    ├── a.py  
    └── b.py  
  
1 directory, 3 files
```

In `example.py`:

```
1 # Import modul a aus package 'package'  
2 import package.a  
3  
4 # Import modul b aus package 'package'  
5 import package.b
```










- Importiere nur so viel wie nötig (vermeidet Überdeckungen im Namespace)
- Importiere nur von tieferen Hierarchie Leveln
- Vermeide zyklische Imports (A importiert B, B importiert A)

# Aufgaben

Kategorie	Beschreibung	Wertung	Beispiel
<b>Einser</b>	Jede Kombination	Die Summe der Augenzahlen der Einser	 zählt 3
<b>Zweier</b>	Jede Kombination	Die Summe der Augenzahlen der Zweier	 zählt 6
<b>Dreier</b>	Jede Kombination	Die Summe der Augenzahlen der Dreier	 zählt 12
<b>Vierer</b>	Jede Kombination	Die Summe der Augenzahlen der Vierer	 zählt 8
<b>Fünfer</b>	Jede Kombination	Die Summe der Augenzahlen der Fünfer	 zählt 0
<b>Sechser</b>	Jede Kombination	Die Summe der Augenzahlen der Sechser	 zählt 18

1. Erweitere die Klasse des Würfelspielers, sodass dieser 5 Würfel besitzt (siehe vorherige Sessions).
2. Implementiere eine Scoring Funktion, die, ausgehend von 5 Resultaten (Liste) und einer Kategorie, den Wert der Resultate berechnet. Werden die Bedingungen der Kategorie nicht erfüllt, ist der Wurf 0 Punkte wert.

# Aufgaben

Kategorie	Beschreibung	Wertung	Beispiel
Dreierpasch	Mindestens drei gleiche Zahlen	Summe aller Augenzahlen	 zählt 17
Viererpasch	Mindestens vier gleiche Zahlen	Summe aller Augenzahlen	 zählt 24
Full House	Drei gleiche und zwei gleiche, andere Zahlen	25	 zählt 25
Kleine Straße	Vier aufeinanderfolgende Augenzahlen (1-2-3-4, 2-3-4-5 oder 3-4-5-6)	30	 zählt 30
Große Straße	Fünf aufeinanderfolgende Augenzahlen (1-2-3-4-5 oder 2-3-4-5-6)	40	 zählt 40
Kniffel / Yahtzee	5 gleiche Zahlen	50	 zählt 50
Chance	Jede Kombination	Summe aller Augenzahlen	 zählt 13

1. Ergänze die Kategorien um den Dreierpasch, den Viererpasch, das Full House, die Kleine Straße, die Große Straße, Kniffel sowie Chance.
2. **Bonus:** Implementiere eine Funktion, die für einen Wurf den größtmöglichen Score sowie den Namen der dazugehörigen Kategorie ausgibt.

Tipp: Vielleicht hilft dir `collections.Counter`