Repetition and Built-Ins

Claas de Boer, Tilman Hinnerichs

07.01.2021

Python-Grundlagen

Was zuletzt geschah...

- Wiederholung von Klassen. dicts, functionen, sets, listen
- Begin inbuilts

Built-In Functions

Built-In Functions

Bekannte Built-In Funktionen

- `int, float, str`
- `len, range, ...`

Built-In Functions

Bekannte Built-In Funktionen

- `int, float, str`
- `len, range, ...`

Neue und spannende Built-In Funktionen:

- `enumerate`
- `map`
- `lambda`
- `zip`
- `filter`

enumerate

`enumerate` nimmt iterable und gibt tuple von Index und Element zurück

```
for i in range(len(my_list)-1):
    ele = my_list[i]
    f(ele, i%dim1)

for i, ele in enumerate(L):
    next_ele = my_list[i+1]
    f(ele, i%dim1)

```

map und lambda

```
1          def f(a,b):
2              return a%b + 42
3
4          f = lambda a,b: a%b + 42
5
```

map und lambda

```
1         def f(a,b):
2             return a%b + 42
3
4         f = lambda a,b: a%b + 42
5
```

```
1         # map(func, iterable) -> Funktion auf alle Elemente
2
3         #use the list() function to display a readable version
      of the result:
4         list(map(lambda a: a+5, list_of_numbers))
5
6
```

filter

```
ages = [5, 12, 17, 18, 24, 32]

def myFunc(x):
    if x < 18:
        return False
    else:
        return True

adults = filter(myFunc, ages)

for x in adults:
    print(x)

```

zip

```
>>> a = ("John", "Charles", "Mike")
>>> b = ("Jenny", "Christy", "Monica", "Vicky")

>>> x = zip(a, b)

#use the list() function to display a readable version of the
    result:

>>> list(x)
[('John', 'Jenny'), ('Charles', 'Christy'), ('Mike', 'Monica')]

>>> x = [(a_ele, b_ele) for a_ele in a for b_ele in b]


```

- `enumerate`: enumerate(iter) -> [(i, iter[i]) for i in range(len(iter))]
- `map`: map(func, iter) -> [func(i for i in iter)]
- `zip`: zip(iter1, iter2) -> [(ele1, ele2) for (ele1, ele2) in zip(iter1,iter2)]
- `filter`: filter(func, iter) -> [i for i in iter if func(i)]

Einige Aufgaben 8

Schreibe ein Python Programm, welches

1. eine jedes Element einer gegebenen Liste verdoppelt
2. alle geraden Elemente eine gegebenen Liste von Zahlen zurückgibt
3. 1-dimensionale Listen von Koordinaten nimmt und diese zu 3D-Koordinaten zusammenfügt
4. 3 gleichlange Listen von Zahlen Elementweise addiert
5. die Unterschiede zweier gegebener, gleichlanger Listen ausgibt (elementweiser Vergleich)

Aufgaben

1. Erweitere die Spielerklasse, sodass ein Spieler mehrere Würfel besitzen kann. Wenn er würfelt, so soll er mit allen Würfeln nacheinander würfeln und eine Ergebnisliste zurückbekommen.
2. Modelliert das Würfelspiel als eigene Klasse.

↪ Lösung in Snippets

Module einbinden

```
1  from random import randint
2  print(randint(1, 10))
3
4  from wuerfel import *
5  d20 = Wuerfel(20)
6  print(d20.wuerfeln())
7
8  import spieler
9  spieler = spieler.Spieler("Arndt", d20)
10 print(spieler.name)
11
```

Nach Informationen zu einer Bibliothek suchen

1. Öffnet die Suchmaschine eures Vertrauens
2. Sucht nach `python random`
3. Klickt auf den Link zur offiziellen Dokumentation unter `https://docs.python.org/3/library/random.html`
4. Scrollt zu `random.randint`
5. Scrollt zu `random.choice` und `random.choices`

Einige Aufgaben 9.0

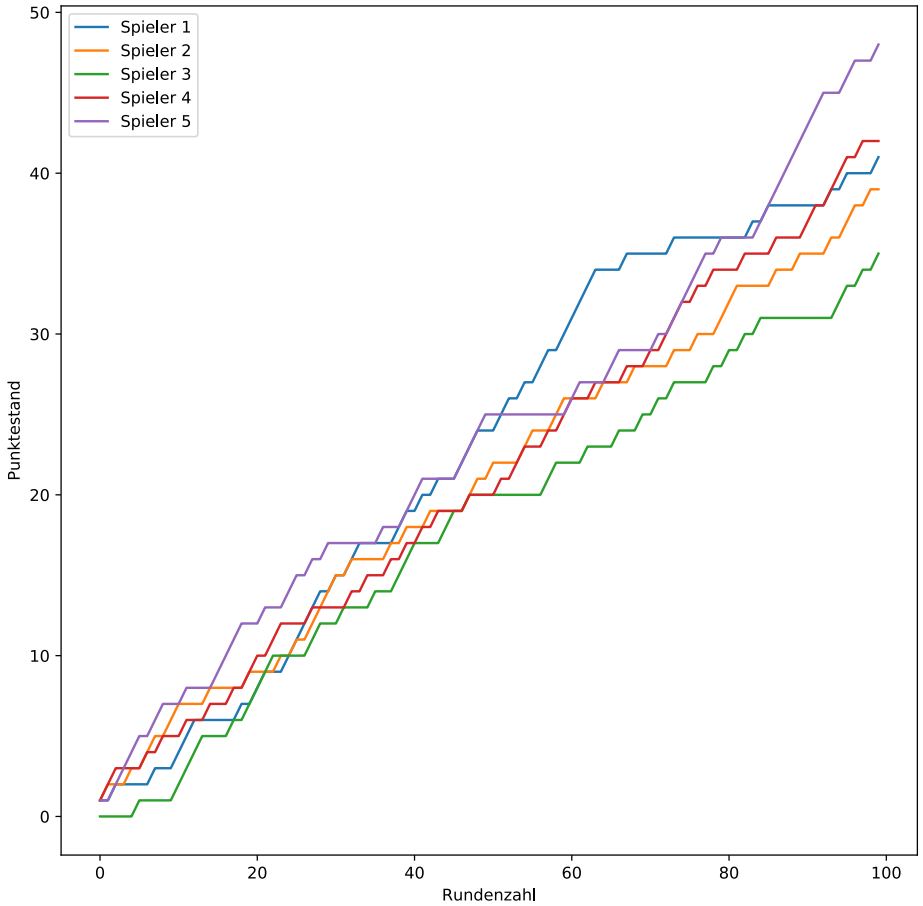
Schreibe ein Python Programm, welches für eine Gruppe von Spielern, repräsentiert durch `[1,2,..,num_spieler]`, das Spiel ”Flaschendrehen“ simuliert, und dabei

1. einen zufälligen Spieler auswählt und damit eine Spielrunde simuliert.
2. 5 Runden simuliert.
3. 5 Runden simuliert, sodass keiner zwei mal ausgewählt werden kann.
4. die Reihenfolge der Spieler zufällig mischt
5. für einen Spieler den Anteil von Kirschsaft an seinem KiBa ausgibt.

matplotlib

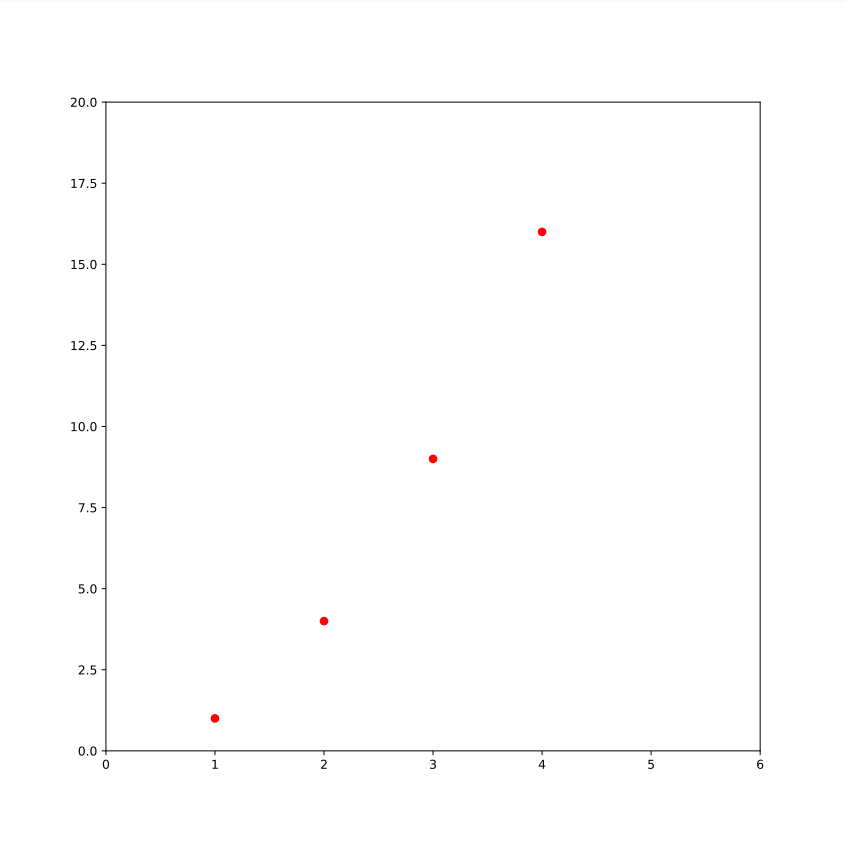
Motivation

- Mit Python berechnete Daten plotten:



Einfaches Beispiel

```
1 from matplotlib import pyplot
2 pyplot.plot([1,2,3,4], [1,4,9,16], 'ro')
3 pyplot.axis([0, 6, 0, 20])
4 pyplot.show()
5
```



Matplotlib instalieren

Linux (Ubuntu):

```
pip3 install --user matplotlib

```

Windows:

```
python.exe -m pip install matplotlib

```

Matplotlib Tutorial:
`https://matplotlib.org/users/pyplot_tutorial.html`

Einige Aufgaben 10.0

1. Erstelle ein Diagramm, in dem der Entstand eines Spiels nach der Simulation (`.simuliere()`) für jeden Spieler dargestellt ist.
2. Erstelle ein Diagramm, in dem der Punktestand aller Spieler im zeitlichen Verlauf dargestellt ist (s. Beispiel in dieser Präsentation)
3. Füge einen schummelnden Spieler zu deinem Spiel hinzu:
   3.1 Mehr Würfel
   3.2 Andere Seitenzahl
   3.3 seid kreativ . . .

   Wie wirkt sich das Schummeln bei den verschiedenen Wertungsmodi aus?
   (`"wuerfelsumme"`, `"wuerfelmaximum"`, `"paschzahl"`)
   Dies wird bei höherer Rundenzahl möglicherweise deutlicher.