

Die Funktion "permutation" im Paket "random" wird mit folgendem Hilfetext beschrieben:

```
random.shuffle(x[, random])
```

Shuffle the sequence x in place. The optional argument random is a 0-argument function returning a random float in [0.0, 1.0); by default, this is the function random(). To shuffle an immutable sequence and return a new shuffled list, use sample(x, k=len(x)) instead. Note that even for small len(x), the total number of permutations of x can quickly grow larger than the period of most random number generators. This implies that most permutations of a long sequence can never be generated. For example, a sequence of length 2080 is the largest that can fit within the period of the Mersenne Twister random number generator.

Wieso wirft der folgende Code einen Fehler? _____

```
import random

karten = tuple(range(10))
random.shuffle(karten)
```

Antwort: Da der Datentyp „tuple“ immutable ist, kann „karten“ nicht „in place“ verändert (geschuffled) werden.

Gegeben eine Liste mit Strings wie z.B.

```
liste = ['hallo', 'hi', 'naaa', 'ola', 'servus', 'huhu', 'guten tag']
```

Nutzen sie die Befehle filter und map um nur diejenigen Worte zu behalten, die mit dem Buchstaben 'h' beginnen und ersetzen sie für diese Worte das 'h' durch ein 'H'. Füllen sie dazu folgende Lücken

```
map(--- LÜCKE 1 ----, filter(--- LÜCKE 2 ----, liste))
```

unter Benutzung der lambda Notation.

Lücke 1: _____ lambda x: 'H'+x[1:]

Lücke 2: _____ lambda x: x[0]=='h'

Was passiert, wenn folgender Code ausgeführt wird, der Benutzer eine 3.3 eingibt und Enter drückt?

```
x = input('Geben sie eine Zahl ein: ')\nprint('Das doppelte der Zahl ist', 2*x)
```

Man bekommt die Ausgabe 3.33.3, da inputs vom Datentyp string sind und das Multiplizieren eines Strings mit 2 diesen lediglich verdoppelt.

Erklären sie die Spielregeln des folgenden funktional implementierten Spiels. Insbesondere erklären sie

- a) Wie viele Spieler gibt es?
- b) Wann gibt es wie viele Punkte?
- c) Was wird am Ende ausgegeben?

Hinweis: Der für sie evt. unbekannte Befehl "tuple(random.sample(range(1,runden),runden-1))" erzeugt ein Tupel zufällig angeordneter integer von 1 bis einschließlich "runden".

```
import random

runden = 10
states = ({ "karten": tuple(random.sample(range(1,runden),runden-1)), "punkte":0, "runde":1},)

def naechsteRunde(state):
    return { "karten": state["karten"][:-1],
            "punkte": state["punkte"] +
                (state["runde"]*(state["karten"][-1]>(state["runde"])),
            "runde":state["runde"]+1}

def rungame(states):
    if states[-1]["karten"]:
        return rungame(states + (naechsteRunde(states[-1]),))
    else:
        return states

from functools import reduce
print(reduce(lambda x,y: x+y,
            map(lambda x: "In Runde {} haben wir {} Punkte \n".format(x["runde"], x["punkte"]),
                rungame(states))))
```

Es gibt einen Spieler. Er besitzt 10 gemischte Karten der Wertigkeit von 1 bis 10. Jede Runde zieht er eine zufällige Karte und schaut ob der Wert der Karte größer ist als die Runde, die das Spiel schon läuft. Falls ja, bekommt er/sie so viele Punkte wie die aktuelle Rundenummer. Falls nein, bekommt er keine Punkte. Am Ende wird für jede Runde die aktuelle Anzahl der Punkte ausgegeben.

Vervollständigen sie den nachfolgenden Code um eine rekursive Funktion zu schreiben, die ein Tupel entgegennimmt und das Tupel zurückgibt in dem nur die Einträge des Eingabetupels vom Typ 'float' enthalten sind.

```
tup = (12, 'hello', 53.9, True, range(12), 3.1415)

def tuple_with_floats_only(tup):
    if tup:
        if ## Lücke 1 ##
            return (tup[0],) + ### Lücke 2 ###
        else:
            return ### Lücke 3 ###
    else:
        return ### Lücke 4 ###

print(tuple_with_floats_only(tup))
```

Lücke 1: _____ (type(tup[0])==float):

Lücke 2: _____ tuple_with_floats_only(tup[1:])

Lücke 3: _____ tuple_with_floats_only(tup[1:])

Lücke 4: _____ ()