Python – Übung 4

1 Funktionsparameter

1.1 Beliebige Positionsparameter

👉 Implementieren Sie die Funktion `rms(*values)`, welche den quadratischen Mittelwert aus einer beliebigen Anzahl von Argumenten berechnet. Es sollen nur numerische Datentypen (`int` und `float`) berücksichtigt werden, d.h. andere Datentypen werden ignoriert. Falls kein einziges Argument gültig ist, soll ein `TypeError` mit der Meldung "too few numeric arguments" geworfen werden.

```
>>> rms(2, 3)
2.5495097567963922
>>> rms(1, 5, "x", 4)
3.7416573867739413
>>> rms("hallo")
Traceback (most recent call last):
...
TypeError: too few numeric arguments
```

1.2 Beliebige Schlüsselwortparameter

👉 Implementieren Sie die Funktion `vorstellung(**angaben)`, welche die Informationen über eine Person aufnimmt und wie folgt darstellt:

```
>>> vorstellung(Name="Susi", Alter=25, Wohnort="Rapperswil")
Ihr Name ist Susi.
Ihr Alter ist 25.
Ihr Wohnort ist Rapperswil.
```

1.3 Optionale Parameter

👉 Implementieren Sie die Funktion `append_to_book(name, phone_number, book)`, welche folgendes Verhalten zeigt:

- Falls ein bestehendes Dictionary als `book`-Argument übergeben wird, dann wird darin ein neues Schlüssel/Wert-Paar mit den Argumenten `name` und `phone_number` erstellt, z.B.:

  ```
  >>> my_book = {"Max": "+41582574111"}
  >>> append_to_book("Moritz", "+41582574112", book=my_book)
  >>> print(my_book)
  {"Max": "+41582574111", "Moritz": "+41582574112"}
  ```

- Falls das `phone_number`-Argument weggelassen wird, dann wird als Platzhalter ein String mit drei Bindestriche `"---"` eingefügt, z.B.:

```
>>> append_to_book("Susi", book=my_book)
>>> print(my_book)
{"Max": "+41582574111", "Moritz": "+41582574112", "Susi": "---"}
```

- Falls das `book`-Argument weggelassen wird, dann erstellt die Funktion selber ein neues Dictionary-Objekt, welches zurückgegeben wird, z.B.:

```
>>> new_book = append_to_book("Julia")
>>> print(new_book)
{"Julia": "---"}
```

```
>>> new_book2 = append_to_book("Romeo")
>>> print(new_book2)
{"Romeo": "---"}
```

2 Comprehensions

👉 Implementieren Sie den folgenden Codeausschnitt mittels der List-Comprehension.

```
values_as_string = ["3", "20.3", "-7.14", "-55.5", "0.1"]
values = []
for v in values_as_string:
    values.append(float(v))
```

👉 Implementieren Sie den folgenden Codeausschnitt mittels der List-Comprehension.

```
values = [3, 20.3, -7.14, -55.5, 0.1]
pos_values = []
for v in values:
    if v >= 0:
        pos_values.append(v)
```

👉 Implementieren Sie den folgenden Codeausschnitt mittels der List-Comprehension.
Hinweis: Benutzen Sie die eingebaute `any()`-Funktion[1].

```
words = ["Schule", "Bücher", "Laptop", "Fläche"]
umlaut = []
for w in words:
    for u in "äöü":
        if u in w.lower():
            umlaut.append(w)
            break
```

👉 Welchen Output generieren die nachfolgenden Comprehensions?

```
list1 = ["A", "B", "C"]
list2 = ["D", "E", "F"]
print([(a, b) for a in list1 for b in list2])
```

[1] https://docs.python.org/3/library/functions.html#any

```
numbers = [1, 2, 3, 4, 5, 6]
print({n**3 for n in numbers if n % 2})
```

```
chars = ["A", "B", "C"]
quantity = [2, 3, 4]
print({c: [c*n for n in quantity] for c in chars})
```

```
print([n for n in range(2, 20)
      if all(n % m for m in range(2, int(n**0.5) + 1))])
```

Gegeben seien folgende Superhelden:

```
superheros = ["Superman", "Batman", "Spiderman", "Wolverine", "Superwoman"]
```

- Implementieren Sie eine Dictionary-Comprehension, welche als Key den entsprechenden Namen aus `superheros` hält und als Wert die Länge des jeweiligen Namen speichert.
- Ergänzen Sie Ihre Dictionary-Comprehension, so dass nur diejenigen Namen berücksichtigt werden, welche mit `"man"` enden.

3 Aufgaben aus dem Buch

- Lösen Sie folgende Aufgaben aus dem Buch.

Kapitel	Seiten	Aufgaben
Listen-Abstraktion	320	1, 2

4 Zweidimensionale Liste

Gegeben sei eine zweidimensionale Liste in Form einer $(m \times n)$-Matrix, z.B.:

```
matrix = [
    [1, 2, 3, 4, 5, 6],
    [10, 20, 30, 40, 50, 60],
    [100, 200, 300, 400, 500, 600],
]
```

- Konvertieren Sie die 2D-Matrix zu einem semikolon-separierten String `matrix_str`, wie im untenstehenden Beispiel gezeigt wird. Benutzen Sie dafür die `.join()`-Methode des String-Datentypen und die List-Comprehension.

```
matrix_str = """1;2;3;4;5;6
10;20;30;40;50;60
100;200;300;400;500;600"""
```

- Erstellen Sie mittels der List-Comprehension und der `zip()`-Funktion die Liste `matrix_csum`, welche die Spaltensummen der Matrix beinhaltet, z.B.:

```
matrix_csum = [111, 222, 333, 444, 555, 666]
```