Markdown-Tabellen (Anwendung: `**kwargs`)
Mit Markdown können wir leicht Tabellen zeichnen.

1. Schreiben Sie eine Python-Funktion `mdTable(**columns)`, die beliebig viele Schlüsselwortargumente entgegennimmt. Die Schlüssel–Wert Paare sollen Tabellenspalten wie folgt definieren: Schlüssel=Spaltenüberschrift, Wert=Python-Liste mit Spalteneinträgen. Damit ist **columns ein Dictionary dessen Werte Listen sind. Bauen Sie nun daraus einen String `mdTab` der eine Markdown–Tabelle mit diesen Spalten enthält.

2. Testen Sie die Funktion mit der Eingabe

```
mdTab = mdTable(computationTime = [0.1, 1.0, 10.0],
                precision = [1.e-2, 2.34e-3, 8.98e-5],
                algorithm = ["A", "B", "C"])
```

   und geben Sie die Ausgabe von `print(mdTab)` in eine Markdown-Zelle ein.

3. Testen Sie die Funktion mit der Eingabe

```
mdTab = mdTable(computationTime = [0.1, 1.0, 10.0],
                someValue = [1.e-2, 2.34e-3],
                algorithm = ["A", "B", "C"])
```

   und geben Sie die Ausgabe von `print(mdTab)` in eine Markdown-Zelle ein.

4. Bonus*: Erweitern Sie die Parameter–Schnittstelle um ein Positionsargument `filename` (Datentyp `string`) und speichern Sie den String `mdTab` zusätzlich in eine Textdatei `filename.md`.
   Hinweis: Sie benötigen dazu ein Dateiobjekt, welches Sie leicht mit der eingebauten Funktion `open()` erstellen. Dann können Sie die Methode `.write()` zum schreiben verwenden und die Methode `.close()`, um die Datei zu speichern und zu schließen.

Solution:

```
#!/usr/bin/env python
# coding: utf-8
# <h1>Table of Contents<span class="tocSkip"></span></h1>
# <div class="toc"><ul class="toc-item"><li><span><a href="#Markdown-Tabelle-mit-  berschrift  "
    data-toc-modified-id="Markdown-Tabelle-mit-  berschrift  -1"><span class="toc-item-num">1&
   nbsp; </span>Markdown Tabelle mit   berschrift  </a></span></li></ul></div>
# ### Markdown Tabelle mit   berschrift
def mdTable(**columns):
    """
    prints a list of dicts as markdown table. The keys are used as head
    and the content of the list as body of the table's columns.
    The lists do not need to have identical length.
    """
    mdTab = ""
    headline = "|"
    separator = "|"
    for key in columns.keys():
        headline += key  + "|"
        separator += "-|"

    mdTab += headline + "\n" + separator + "\n"
    #print(headline)
    #print(separator)
    n_rows = [len(columns[k]) for k in columns.keys()]
    for row in range(max(n_rows)):
```

```
        col_number = 0 # ColumNumber
        for key, value in columns.items():
            if row < n_rows[col_number]:
                #print("| " + str(value[row]) + " ", end="")
                mdTab += "| " + str(value[row]) + " "
            else:
                #print("| ", end="")
                mdTab += "| "
            col_number += 1
        #print("|")
        mdTab+="|\n"

    # in Datei speichern:
    # f = open("filename", "w")
    # f.write(table)
    # f.close()
    return mdTab
mdTab = mdTable(computationTime = [0.1, 1.0, 10.0],
        precision = [1.e-2, 2.34e-3, 8.98e-5],
        algorithm = ["A", "B", "C"])
print(mdTab)
# in Datei speichern:
#f=open("TESTMD", "w")
#f.write(table)
#f.close()
# |computationTime|precision|algorithm|
# |-|-|-|
# | 0.1 | 0.01 | A |
# | 1.0 | 0.00234 | B |
# | 10.0 | 8.98e-05 | C |
mdTab= mdTable(computationTime = [0.1, 1.0, 10.0],
        someValue = [1.e-2, 2.34e-3],
        algorithm = ["A", "B", "C"])
print(mdTab)
# |computationTime|someValue|algorithm|
# |-|-|-|
# | 0.1 | 0.01 | A |
# | 1.0 | 0.00234 | B |
# | 10.0 | | C |
```