Python – Übung 5

1 Dateiinhalt verarbeiten

Gegeben ist die Datei `namen.txt`, in welcher die Vor- und Nachnamen von Personen verschachtelt abgespeichert sind, z.B.:

```
Hans
Wurst
Peter
Lustig
Pippi
Langstrumpf
```

☞ Diese Datei soll in eine neue Datei `teilnehmer.txt` überführt werden, in der die Vor- und Nachnamen pro Person zusammengefasst werden, z.B.:

```
Hans Wurst
Peter Lustig
Pippi Langstrumpf
```

2 Konfigurationsdateien

2.1 Datei schreiben

Gegeben sei ein Dictionary mit verschiedenen Parametern, z.B.:

```
gui_settings = {
    "language": "English",
    "size": (700, 500),
    "dpi": 96,
    "save_on_exit": True,
}
```

☞ Implementieren Sie die Funktion `save_parameters(data, fname, encoding="utf-8")`, welche die Parameter in die angegebene Datei `fname` speichert.

```
>>> save_parameters(gui_settings, "settings.txt")
```

Dabei sollen die Schlüsselwörter mit den jeweiligen Werten zeilenweise in die Datei geschrieben werden. Als Trennzeichen soll das Gleichheitszeichen dazwischen eingefügt werden.
Hinweis: Benutzen Sie die `items()`-Methode des Dictionary, um über die Key-Value-Paare zu iterieren.

```
language=English
size=(700, 500)
dpi=96
save_on_exit=True
```

2.2 Datei lesen

☞ Implementieren Sie die Funktion `load_parameters(fname, encoding="utf-8")`, welche die angegebene Konfigurationsdatei einliest und den Inhalt als Dictionary zurückgibt, z.B.:

```
>>> load_parameters("settings2.txt")
{"language": "English", "size": "(700, 500)", "dpi": "96", "save_on_exit":
   "True"}
```

Dabei sollen leere Zeilen und überflüssige Leerzeichen ignoriert werden:

```
language=English
size= (700, 500)
dpi = 96

save_on_exit=True
```

Hinweis: Benutzen Sie die `split()`-Methode, um den Parameternamen vom Parameterwert zu trennen. Überflüssige Leerzeichen und Newline-Zeichen (\n) in einer Zeile können mit der `strip()`-Methode entfernt werden.

3 Log-Datei schreiben

☞ Implementieren Sie die Funktion `log_to_file(msg, fname, encoding="utf-8")`, welche den `msg`-Text zusammen mit dem Zeitstempel in die angegebene Datei `fname` schreibt. Bei jedem Aufruf soll der bestehende Dateiinhalt beibehalten und die neue Zeile ans Ende der Datei angefügt werden. Allfällige Zeilenumbrüche und überflüssige Leerzeichen im Text sollen ignoriert werden, z.B.:

```
>>> file = "log.txt"
>>> log_to_file("Start.", file)
>>> log_to_file("Processing␣...", file)
>>> log_to_file("Read␣data.\n␣␣␣␣Process␣data.␣␣␣\n␣␣Save␣␣␣␣data.", file)
>>> log_to_file("␣␣␣Done.\n", file)
```

Entsprechender Inhalt der `log.txt`-Datei:

```
2021-03-13T19:21:20.838967 - Start.
2021-03-13T19:21:21.341152 - Processing ...
2021-03-13T19:21:22.343364 - Read data. Process data. Save data.
2021-03-13T19:21:22.643838 - Done.
```

Hinweis: Sie können den formatierten Zeitstempel mittels den Funktionen aus dem `datetime`-Modul erzeugen, z.B.:

```
>>> from datetime import datetime
>>> datetime.utcnow().isoformat()
"2021-03-13T19:19:22.118172"
```

4 Buchstaben zählen

Ein Text wurde in der Datei `zen.txt` für die weiteren Untersuchungen abgespeichert. Nun will man wissen wie oft die Buchstaben des Alphabets (a-z) im Text vorkommen. Die Gross- und Kleinschreibung soll dabei ignoriert werden.

Hinweise: Ein String kann mit den Methoden `.upper()` und `.lower()` in Gross- bzw. Kleinschreibung umgewandelt werden. Im Modul `string` existieren vorgefertigte Zeichenketten, die verschiedene Zeichen gruppieren, z.B.:

```
>>> import string
>>> string.ascii_lowercase
"abcdefghijklmnopqrstuvwxyz"
```

☞ Lesen Sie den Text aus der Datei `zen.txt` ein und erstellen Sie mittels der Dictionary-Comprehension das Dictionary `chars`, welches jeden Buchstaben (von `a` bis `z` in Kleinschreibung) als Key und dessen Anzahl als Value enthält, z.B.:

```
chars = {"a": 53, "b": 21, "c": 17, ...}
```

☞ Erstellen Sie zudem mittels der List-Comprehension die Liste `chars_sorted`, welche die Buchstaben des Alphabets in Kleinschreibung und in absteigender Häufigkeit beinhaltet. Benutzen Sie die eingebaute `sorted()`-Funktion[1], um die Daten zu sortieren.

```
chars_sorted = ["e", "t", "a", "i", ...]
```

5 Aufgaben aus dem Buch

☞ Lösen Sie folgende Aufgaben aus dem Buch.

Kapitel	Seiten	Aufgaben
Listen und Tupel im Detail	155-156	2, 3, 5

6 GPS Log Viewer

Mittels der HTML-Datei `gps_log_viewer.html` kann eine Strecke auf OpenStreetMap dargestellt werden, wie in Abb. 1 gezeigt wird. Die Streckendaten werden in einer separaten Javascript-Datei `data.js` gespeichert, wie in Listing 1 gezeigt wird. Die Daten müssen im sog. GeoJSON-Format[2] abgespeichert werden, wobei die Koordinaten der einzelnen Wegpunkte mit den entsprechenden Längen- und Breitengrade angegeben werden.

[1] `https://docs.python.org/3/library/functions.html#sorted`
[2] `https://tools.ietf.org/html/rfc7946`

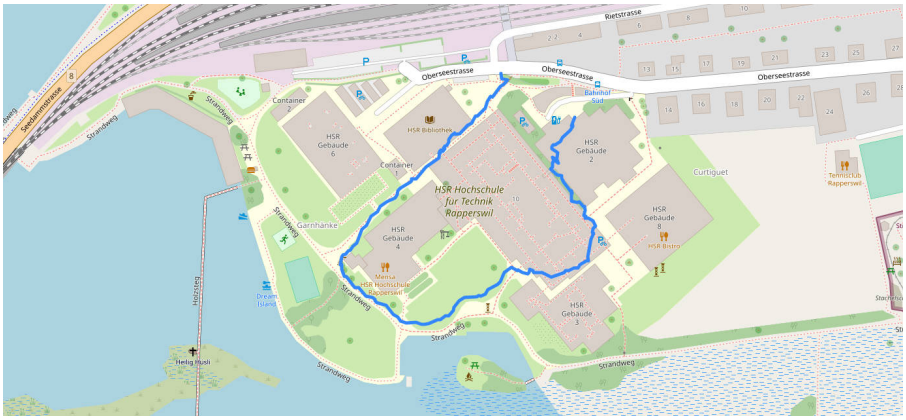


Abbildung 1: Ausgabe des GPS Log Viewers.

```
var track = {
    "type": "Feature",
    "geometry": {
        "type": "LineString",
        "coordinates": [
            [8.818028,47.223778],
            [8.817993,47.223808],
        ]
    }
};
```

Listing 1: Inhalt der Javascript-Datei data.js

Ein GPS-Empfänger speichert die ASCII-basierten Datensätze zeilenweise im sog. NMEA-Format ab, wie in Listing 2 dargestellt.

```
$GPGGA,165929.000,4713.4306,N,00849.0783,E,1,16,0.6,410.5,M,48.0,M,,*58
$GPGLL,4713.4306,N,00849.0783,E,165929.000,A,A*50
$GPVTG,277.1,T,,M,0.0,N,0.0,K,A*0E
$GPGSA,A,3,24,12,15,19,17,13,10,18,25,,,,1.2,0.6,1.0*34
$GNGSA,A,3,65,66,72,74,75,83,84,,,,,,1.2,0.6,1.0*28
$GPRMC,165929.000,A,4713.4306,N,00849.0783,E,0.0,277.1,180118,,,A*65
```

Listing 2: NMEA-Nachrichten.

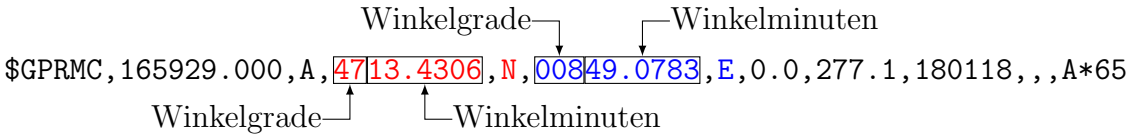
Jede Zeile beginnt mit:

- dem $-Zeichen
- gefolgt von der Geräte-ID (z.B. `GP`=GPS, `GL`=GLONASS oder `GN`=Multi-GNSS)
- und der Datensatz-ID (z.B. `GGA`, `GLL`, `RMC`, ...).

Danach folgen die Datenfelder, die durch Kommas getrennt werden. Nach dem letzten Datenwert steht das *-Zeichen, welches von der hexadezimalen Prüfzahl gefolgt wird. Die Prüfzahl wird durch die XOR-Verknüpfung der ASCII-Werte aller Zeichen zwischen dem $- und dem *-Zeichen errechnet.

Hinweis: Wandeln Sie den Stringteil mittels der `encode()`-Methode in ein `bytes`-Typ um, um die binären ASCII-Werte des Strings zu erhalten, z.B.: `line[1:-3].encode("ascii")`.

Die Informationen über den Längen- und Breitengrad befinden sich im RMC-Datensatz[3]:



Die Winkelangabe für den Breiten- und Längengrad besteht aus Winkelgrade und Winkelminuten. Zum Beispiel, der Breitengrad errechnet sich wie folgt: Breitengrad = 47+13.4306/60. Das Vorzeichen des Längen- und Breitengrades wird durch die Ausrichtung angegeben: `N`/`E`=positiv und `S`/`W`=negativ.

☞ Implementieren Sie ein Programm, welches die Längen- und Breitengrade aus den RMC-Datensätzen der NMEA-Datei **logfile.nmea** extrahiert und als Liste in die Javascript-Datei **data.js** im GeoJSON-Format schreibt.

Hinweis: Achten Sie auf mögliche Ausnahmen und fangen Sie diese ab, z.B. die Datenfelder zwischen den Kommas können auch leer sein (wenn kein Signal empfangen wird).

7 Verzeichnisliste

☞ Implementieren Sie die Funktion `directory_listing(directory, suffix=None)`, welche die Dateien und Unterverzeichnisse im angegebenen Verzeichnis-Pfad `directory` auflistet.

Der Rückgabewert der Funktion soll einen formatierten String sein. Die Dateien/Unterverzeichnisse sollen zeilenweise aufgelistet werden, wobei jede Zeile folgende drei Spalten beinhaltet:

- (1. Spalte) Angabe ob es eine Datei (`f`) oder Unterverzeichnis (`d`) ist,
- (2. Spalte) Grösse der Datei in Bytes,
- (3. Spalte) Name der Datei oder des Unterverzeichnisses.

Hinweis: Benutzen Sie das `pathlib.Path`-Objekt mit dessen Methoden `glob()` und `is_file()` und dessen Attribute `stat().st_size` und `name`.

Die Bytezahlen sollen mit Leerzeichen aufgefüllt werden, so dass alle rechtsbündig ausgerichtet werden, und nach dem Namen eines Unterverzeichnisses soll ein `"/"` angehängt werden, z.B.:

```
>>> directory_listing("C:/Anaconda3")
f    19208 api-ms-win-core-console-l1-1-0.dll
...
d        0 bin/
...
f 14053227 _conda.exe
```

Mit dem Parameter `suffix` (Dateiendung) soll die Liste gefiltert werden, z.B.:

```
>>> directory_listing("C:/Anaconda3", suffix=".txt")
f 12781 LICENSE_PYTHON.txt
```

Falls keine Dateien/Unterverzeichnisse resultieren, wird folgende Zeile zurückgegeben:

```
>>> directory_listing("C:/Anaconda3", suffix=".weird_suffix")
--- nothing found ---
```

[3] https://de.wikipedia.org/wiki/NMEA_0183#Recommended_Minimum_Sentence_C_(RMC)