Umweltmessdaten

Allgemeines

- Verwenden Sie zur Programmentwicklung als Modulnamen (Dateinamen) für dieses Praktikum den Namen `p03NameVorname.py` und halten sich **strikt** an die in der Aufgabe **vorgegebenen Funktionsnamen und Parameterliste** (Grund: Unit-Test s. u.)

  `Name` ist durch Ihren Nachnamen und `Vorname` durch Ihren Vornamen zu ersetzen. Der Dateiname sollte weiterhin keine Leerzeichen enthalten.

- Bevor Sie Ihre Lösung den Betreuern vorstellen, können Sie mit dem in moodle bereitgestellten Unit-Test `p03unit.py` überprüfen, ob Ihre Lösung die richtigen Ergebnisse liefert.
  - Passen Sie die erste Import-Anweisung im Modul `p03unit.py` gemäß Ihrem Namen an.
  - Im Anaconda-Prompt können Sie den Unit-Test in Ihrem Projektverzeichnis über folgende Anweisung ausführen:
    `python p03unit.py`
- Schreiben Sie Ihre Funktionstests so, dass diese nicht bei Aufruf des Unit-Tests aufgerufen werden.

- Sollten in der Aufgabenstellung Fragen zu beantworten sein, dann beantworten Sie diese bitte in einem Dokumentationskommentar am Anfang Ihre Python-Datei, z. B.:

```
"""
      Antworten zu den in der Aufgabenstellung gestellten
      Fragen:
      Frage 1: Ihre Antwort
      Frage 2: Ihre Antwort
      ...........
"""
```

Das Bayerische Landesamt für Umwelt stellt die aktuellen Messdaten der durch die in Bayern aufgestellten Umweltmessstationen zum Herunterladen zur Verfügung. Auf der Internetseite
https://www.lfu.bayern.de/luft/immissionsmessungen/messwerte/stationen/detail/803/172
finden Sie die Messdaten für die Lothstraße der letzten Tage als **1-Stundenmittelwerte**.
Laden Sie sich – zur Interpretation der Dateien - die aktuellen NO_2 (Stickstoffdioxid - Dateiname `csv_803_172.csv`) und über den Reiter "Feinstaub" die PM_{10} (Feinstaub)-Werte (Dateiname `csv_803_132G.csv`) jeweils als CSV-Datei von der Seite (siehe `"Download"` auf der Seite ganz unten).
Dateien im *CSV*[1]*-Format* sind Textdateien, in denen die einzelnen Einträge durch Kommas, die einzelnen Datensätze durch Zeilenvorschübe voneinander getrennt sind. Eine Zeile entspricht also einem Datensatz.
Im deutschsprachigen Raum wird ein Semikolon anstelle eines Kommas als Trennzeichen verwendet. Überprüfen und interpretieren Sie den Inhalt der Dateien mit einem **Texteditor**. Beachten Sie, dass jede Datei drei Spalten zur Verfügung stellt. Die ersten beiden Spalten enthalten zeitliche Angaben, die dritte den zugehörigen Messwert.
Hinweis: Verwenden Sie für die Versuchsbearbeitung nur die in **moodle** mitgelieferten CSV-Dateien.

[1] comma separated values

1 Das Einlesen der Messwerte

Definieren Sie eine Funktion `read_data(fn, val_name='value')`, die die Daten einer Datei mit Pfadname `fn` einliest und geeignet als Tabelle zur Verfügung stellt. Es gibt verschiedene Möglichkeiten der Repräsentation. So könnte man die Daten als Liste von Dictionaries oder als Dictionary von Listen oder Tupel darstellen. Auch eine Liste von Tupel oder ein Tupel von Tupel wäre möglich.
Wir entscheiden uns hier für ein Dictionary von Listen.
Ein Dictionary-Eintrag ermöglicht dabei den Zugriff auf **alle Zeilen einer Spalte** der CSV-Datei. Eine CSV-Datei erzeugt also ein Dictionary mit drei Einträgen. Das Dictionary soll in diesem Teil der Aufgabe die folgenden drei Keys aufweisen: `'date'`, `'time'` und `'value'`. Wobei der dritte Schlüsselname (Key) beliebig gewählt werden kann (s. optionaler Parameter der Funktion). Dieses Dictionary wird im Folgenden "Tabelle" genannt.

Hinweis: Da Dateien im CSV-Format Textdateien sind, können Sie diese Dateien genau so einlesen, wie Sie es im ersten Praktikum kennengelernt haben. Mit `s.split(";")` wird ein String `s` in einzelne Teil-Strings zerlegt, deren Trennzeichen das Semikolon war.
Frage: Warum spricht man in diesem Fall von einer Methode und nicht von einer Funktion?

Hinweis für Profis: Es gibt eine nicht ganz einfach verständliche einzeilige Lösung zum Erstellen der Listen eines Directory-Eintrags. Beispiel:

```
m = [[1, 10, 100], [2, 20, 200], [3, 30, 300], [4, 40, 400]]
mt = list(zip(*m))    # Unpacking und zip
```

2 Typische Kenndaten der Messungen

Definieren Sie eine Funktion `stats(table, val_name)`, die ein Tupel zurückliefert in dem sich Anzahl, Minimum, Maximum und Durchschnitt aller Werte des angegebenen Keys `val_name` (für die Tabelle aus Teil 1 würde man hier `'value'` übergeben) der Tabelle `table` zurückliefert.
Sie können davon ausgehen, dass zumindest ein Messwert vorhanden ist. Die Messwerte liegen als Strings vor, die sich fehlerfrei in Fließkommazahlen umwandeln lassen.
Testen Sie diese Funktion mit den Werten aus den zur Verfügung gestellten Dateien. Geben Sie dabei die Ergebnisse geeignet formatiert aus, verwenden Sie dazu die formatierte Ausgabe mit der Methode `format()`.

3 Vorbereitung der Zusammenführung zweier Messwerttabellen

Definieren Sie die Hilfsfunktion `add_entry(table, d, t, val_name0, val_name1, val0, val1)`, die die Tabelle `table` um einen Eintrag für den durch `d` und `t` angegebenen Zeitpunkt (Datum/Zeit) und für die Messwerte `val0`/`val1` mit den Messwertnamen `val_name0`/`val_name1` ergänzt.
Die Funktion dient zum Belegen einer Tabelle mit zwei Messwert-Datensätze (z. B. `val_name0: 'NO2'`, `val_name1: 'FS'`) für einen angegebenen Zeitpunkt.

In der Funktion wird **nicht** überprüft, ob es bereits einen Eintrag für den angegebenen Zeitpunkt gibt, sondern die übergebenen Daten werden einfach an die vorgegebenen Listen (im Dictionary) angehängt.
Testen Sie Ihre Funktion indem folgendes Dictionary

```
wetter={'date': [], 'time': [], 'temp': [], 'hygro': []}
```

um zwei Datensätze ergänzt wird. Das Dictionary soll danach wie folgt aussehen:

```
wetter={'date': ['22.11.2019', '23.11.2019'], 'time': ['8:15', '8:15'], 'temp':
                    ['5.5', '6.0'], 'hygro': ['54','65']}
```

4 Hilfsfunktion `check_time` kennenlernen

Die Funktion `check_time(table0, table1, i0, i1)` ist bereits in der Datei `v3_util.py` vorhanden. Sie ermittelt, ob der zum Zeilenindex `i0` von `table0` gehörige Zeitpunkt vor/gleichzeitig/nach dem zum Zeilenindex `i1` von `table1` gehörige Zeitpunkt ist. Die Funktion gibt entsprechend die Werte -1/0/1 zurück.
Ist der Index einer Tabelle ungültig (zu groß), wird der Eintrag der anderen Tabelle als zeitlich davor eingestuft.
Importieren Sie diese Funktion aus `v3_util.py` in Ihr Programm. Testen Sie die Funktion, indem Sie die beiden heruntergeladenen Tabellen einlesen (nennen Sie die Schlüssel der Messwerte am besten `'NO2'` bzw. `'FS'`) und rufen `check_time` für verschiedene Zeilenkombinationen (verschiedene Werte für die Indizes `i0` und `i1`, die auch außerhalb des gültigen Bereiches liegen können) auf.
Die zum Testen gewählten Zeilenkombinationen sollten natürlich so gewählt werden, dass man auch alle möglichen Ergebnisse (-1, 0 oder 1) erhält.

5 Zusammenführen beider Messungen zu einer Tabelle

Definieren Sie eine Funktion `merge(data0, data1)`, die zwei eingelesene Tabellen (Dictionaries) akzeptiert, in eine gemeinsame Tabelle (neues Dictionary) kombiniert und als Funktionswerte an den Aufrufer zurückgibt. Es kann davon ausgegangen werden, dass die Tabellen nach der Messungszeit sortiert sind und pro Tabelle (neben Zeitpunkt der Messung) nur ein Messwert enthalten ist. In der kombinierten Tabelle sollen Messwerte, die zur selben Zeit gehören, in derselben Zeile erscheinen. Liegt für einen Zeitpunkt ein Messwert nicht vor, soll in die Tabelle stattdessen `None` eingetragen werden. Die neue Tabelle soll dabei zeitlich sortiert bleiben.

Hinweise:

- Nutzen Sie die Funktionen `check_time` und `add_entry`.
- Schaffen Sie es, die frei gewählten Messwertnamen (Schlüssel im Dictionary) aus der Tabelle zu extrahieren und nicht vorgeben zu müssen?
  Tipp: Verwenden Sie dazu die Mengenoperationen, die Sie für `set`-Objekte kennengelernt haben. Die Schlüsselnamen `'date'` und `'time'` wurden, wie Sie bereits wissen, als unveränderlich festgelegt.
- Der klassische Merge-Algorithmus zweier Tabellen nutzt zwei Indizes `curr_ndx0` und `curr_ndx1` mit deren Hilfe man die Tabellen durchforstet. Er läuft wie folgt ab:
  - Es werden zwei Indizes `curr_ndx0` und `curr_ndx1` verwendet. Damit merkt man sich die Position des nächsten einzufügenden Eintrags aus Tabelle `data0` bzw. `data1`.
  - Solange (`while`-Schleife) in beiden Tabellen noch Einträge vorhanden sind, die noch eingefügt werden müssen, wählt man immer den zeitlich kleineren Eintrag, fügt diesen in die neue Tabelle ein und erhöht `curr_ndx0` oder `curr_ndx1` entsprechend.
  - Sind die Einträge aus den beiden Tabellen zeitgleich, fügt man die Messwerte aus beiden Tabellen zu diesem gleichen Zeitpunkt in die neue Tabelle ein und erhöht beide Indizes.
  - Nach Beendigung der `while`-Schleife sind noch die restlichen Elemente der Tabelle, die noch Datensätze besitzt, anzufügen.
    Für Profis: Sie können die noch fehlenden Einträge der anderen Tabelle auf einem Schlag ohne Schleife anfügen.
- Testen Sie diese Funktion indem Sie die zur Verfügung gestellten Dateien `TESTcsv_803_172.csv` (NO_2-Werte) und `TESTcsv_803_132G.csv` (Feinstaubwerte) einlesen und daraus eine neue Tabelle erzeugen, die beide Messwertspalten für die NO_2- und Feinstaubwerte enthält.

6 Abschluss des Projekts durch Ausgabe der relevanten Daten

Ergänzen Sie Ihr Programm, so dass alle Messwerte der kombinierten Tabelle geeignet tabellarisch formatiert ausgegeben werden und abschließend unter Verwendung der obigen Funktionen die Durchschnittswerte der NO_2- und der Feinstaubbelastung auf dem Bildschirm ausgibt.
Ändern Sie dazu die Funktion `stats` so ab, dass `None`-Einträge ignoriert werden.

Optionale Änderung (als Übung für zuhause):
Ändern Sie die interne Struktur Ihrer Datenbanktabelle in ein Dictionary von Dictionaries ab. Als ersten Key legt man den Zeitpunkt der Messung fest (➔ Tuple aus Datum und Uhrzeit) und erhält ein Dictionary mit den einzelnen Messwerten.
Klassifizieren Sie die Vorteile bzw. Nachteile dieser Lösung.