IM-ADV1: Automatisierung – Datenverarbeitung I

Übung: Textdateien lesen und schreiben mit *Python*

Version: HS22

Aufgabe 1: Koordinatenliste lesen und verarbeiten

a) Bestimme die Anzahl Datensätze in der Datei *Koordinatenliste.dat* mit Hilfe eines *Python*-Codes. Eine Zeile entspricht einem Datensatz. Überleg Dir dazu zwei unterschiedliche Lösungsansätze.

b) Lies die Datensätze aus der Datei *Koordinatenliste.dat* mit Hilfe eines *Python*-Codes zeilenweise und gib die Datensätze auf den Bildschirm aus.

c) Die Datensätze aus der Datei *Koordinatenliste.dat* entsprechen einer Weglinie, d.h. die Koordinaten als Wegpunkte bilden in der aufgeführten Reihenfolge die Eckpunkte einer Liniengeometrie. Lies die Datensätze mit Hilfe eines *Python*-Codes **zeilenweise** und berechne die Weglänge. Gib das Resultat auf Meter gerundet auf den Bildschirm aus. (NB: Die Koordinatenwerte beziehen sich auf das Schweizerische Landeskoordinatensystem LV95.)
Tipp: Wenn Daten aus einer Datei gelesen werden, sind diese vom Typ `str` (Zeichenkette). Um mit den Daten rechnen zu können, müssen diese zuerst mit `int()` oder `float()` in numerische Datentypen konvertiert werden.

d) Überprüfe das Resultat der Weglängenberechnung aus Aufgabenteil c) mit Hilfe eines anderen Werkzeugs.

e) Löse Aufgabenteil c) nicht durch zeilenweises Lesen der Datensätze, sondern lies den gesamten Dateiinhalt mit **einem einzigen `read`-Methodenaufruf** und berechne dann wieder die Weglänge.

Institut Digitales Bauen | Hofackerstrasse 30, 4132 Muttenz | T +41 77 512 76 98 | fredy.spring@fhnw.ch | www.fhnw.ch

Aufgabe 2: *CSV*-Dateien lesen und schreiben

Mit dem *Python*-Modul `csv` können Textdateien mit Trennzeichen orientierten Datenspalten direkt gelesen werden. Das *Python*-Modul `csv` enthält zu diesem Zweck eine sogenannte Reader- und eine Writer-Klasse. Ein *Python*-Code, der diese Klassen verwendet, folgt folgendem Code-Schema:

```
import csv

# CSV-Datei öffnen und CSV-Reader initialisieren
myfile = open('Beispieldatei.csv', 'rt', encoding='utf-8')
mycsvreader = csv.reader(myfile, delimiter=",")

# nach Bedarf: Überspringt eine Zeile (z.B. Kopfzeile)
next(mycsvreader)

# CSV-Datei zeilenweise lesen
for line in mycsvreader:
    print(line)            # Hinweis: Beachte den Datentyp von line!

# CSV-Datei schliessen
myfile.close()

# ---------------------------------------------------------------------

# Neue CSV-Datei anlegen, öffnen und CSV-Writer initialisieren
newfile = open('PLZ_ORT.csv', 'wt')
csvwriter = csv.writer(newfile, delimiter=';', lineterminator='\n')

# nach Bedarf: Kopfzeile schreiben
csvwriter.writerow(['PLZ', 'Ort'])

# CSV-Datei zeilenweise schreiben
csvwriter.writerow(['1000', 'Lausanne'])
csvwriter.writerow(['1003', 'Lausanne'])
csvwriter.writerow(['1004', 'Lausanne'])
csvwriter.writerow(['1005', 'Lausanne'])
...

# Neue CSV-Datei schliessen und freigeben
newfile.close()
```

a) Lies die *CSV*-Datei *CH_PLZ_GDE.csv* mit Hilfe eines *CSV*-Reader-Objekts in *Python*. Als Trennzeichen gilt das Semikolon `';'`. Gib die Ortsnamen ohne die Postleitzahlen als Resultat auf den Bildschirm aus.

b) Löse Aufgabenteil a) analog mit der *dat*-Datei und dem Leerschlag `' '` als Trennzeichen.

c) Schreibe die Daten aus der Datei *CH_PLZ_GDE.csv* aus Aufgabenteil a) in eine neue Datei, in der die Spalte Ortsnamen und die Spalte Postleitzahl vertauscht sind. Ein Datensatz besteht dann also aus dem Ortsnamen gefolgt von der Postleitzahl.

d) Doppelklicke die neue *CSV*-Dateien im Dateisystem (unter *Windows* im *Datei-Explorer*).

e) Löse den Aufgabenteil c) nochmals allerdings nun mit der zusätzlichen Anforderung, dass die Datensätze alphabetisch nach Ortsnamen sortiert in die neue Datei geschrieben werden.