Werteanalyse

1 Aufgabenstellung

Ziel dieser Aufgabe ist es, ein Python-Programm zu entwickeln, das numerische Daten aus einer Datei einliest und statistisch auswertet. Die Zahlenwerte sind in der Eingabedatei zeilenweise gespeichert, wobei jede Zeile einen einzelnen Datenwert enthält.

Das Programm soll die Datei öffnen, die darin enthaltenen Zahlen nacheinander einlesen und daraus mindestens die folgenden statistischen Kennwerte bestimmen:

- das **Minimum**, also den kleinsten vorkommenden Wert,
- den **Durchschnitt (arithmetisches Mittel)** aller Werte,
- das **Maximum**, also den grössten vorkommenden Wert.

Die berechneten Ergebnisse sollen anschliessend übersichtlich ausgegeben werden.

Die Aufgabenstellung lässt sich damit schematisch wie folgt zusammenfassen:

Datei öffnen $\longrightarrow$ Einlesen der Daten $\longrightarrow$ statistische Auswertung

$\longrightarrow$ Datei schliessen $\longrightarrow$ Ausgabe Minimum, Durchschnitt, Maximum

Beim arithmetischen Mittel werden alle eingelesenen Werte addiert und anschliessend durch die Anzahl der Werte geteilt. Für n Werte $x_1, \ldots, x_n$ gilt:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

2 Zweck und Aufbau einer CSV-Datei

Eine CSV-Datei (Comma-Separated Values) ist ein einfach aufgebautes Textformat zur Speicherung tabellarischer Daten. Die einzelnen Datensätze werden zeilenweise gespeichert. Innerhalb einer Zeile werden mehrere Werte normalerweise durch ein Trennzeichen voneinander getrennt, häufig durch ein Komma oder ein Semikolon.

Eine einfache CSV-Datei könnte beispielsweise so aussehen:

```
12.5
8.7
15.2
4.1
10.3
```

In diesem Fall enthält jede Zeile genau einen Zahlenwert. Für die vorliegende Aufgabe ist eine solche Struktur besonders einfach zu verarbeiten.

Eine CSV-Datei kann aber auch mehrere Spalten enthalten, beispielsweise:

```
Messung;Temperatur;Druck
1;21.4;1012
2;22.1;1015
3;20.8;1011
```

Die erste Zeile kann dabei eine Kopfzeile mit den Bezeichnungen der Spalten enthalten. Beim Einlesen muss in diesem Fall berücksichtigt werden, dass die Kopfzeile keine numerischen Daten enthält.

Der Vorteil von CSV-Dateien besteht darin, dass sie ein einfaches, weit verbreitetes und von vielen Programmen unterstütztes Format darstellen. Sie können beispielsweise mit Tabellenkalkulationsprogrammen wie Excel geöffnet und bearbeitet werden, ohne dass ein spezielles proprietäres Dateiformat erforderlich ist.

3 Hinweise zur praktischen Umsetzung in Python

Für die Umsetzung der Aufgabe sind verschiedene grundlegende Python-Konzepte hilfreich. Ziel ist es, die Lösung zunächst selbst zu entwickeln. Die folgenden Hinweise geben dabei eine mögliche Vorgehensweise vor, ohne die vollständige Lösung vorwegzunehmen.

3.1 Datei öffnen und schliessen

Zum Lesen einer Datei kann in Python die Funktion `open()` verwendet werden. Sie erhält unter anderem den Namen der Datei und den gewünschten Modus.

Zum Beispiel:

```
datei = open("daten.csv", "r")
```

Der Modus `"r"` steht dabei für das Lesen (read).

Nach der Verarbeitung sollte die Datei wieder geschlossen werden:

```
datei.close()
```

Noch besser ist die Verwendung einer `with`-Anweisung. Dadurch wird sichergestellt, dass die Datei nach dem Verlassen des entsprechenden Blocks automatisch geschlossen wird:

```
with open("daten.csv", "r") as datei:
    # Datei hier verarbeiten
```

3.2 Dateihandle mit einer Schleife durchlaufen

Das von `open()` zurückgegebene Objekt kann direkt mit einer `for`-Schleife durchlaufen werden. Dabei erhält man nacheinander jeweils eine Zeile der Datei:

```
with open("daten.csv", "r") as datei:
    for zeile in datei:
        # zeile verarbeiten
```

Da jede Zeile typischerweise mit einem Zeilenumbruch endet, kann es sinnvoll sein, diesen zunächst zu entfernen. Dafür kann beispielsweise die Methode `strip()` verwendet werden:

```
wert = zeile.strip()
```

3.3 Umwandlung von Text in Zahlen (Casting)

Die eingelesenen Zeilen sind zunächst Zeichenketten (`str`). Für mathematische Berechnungen müssen sie in einen geeigneten numerischen Datentyp umgewandelt werden.

Für Gleitkommazahlen kann beispielsweise `float()` verwendet werden:

```
wert = float(zeile.strip())
```

Danach kann die Variable `wert` wie eine normale Zahl verwendet werden.

3.4 Idee zur Bestimmung von Minimum und Maximum

Minimum und Maximum können während des Einlesens der Datei laufend bestimmt werden. Es ist dafür nicht notwendig, sämtliche Werte zu speichern.

Die Grundidee besteht darin, jeweils den bisher kleinsten und den bisher grössten Wert zu speichern.

Eine elegante Methode, um schon mit dem ersten Wert einen Vergleich durchführen zu können, besteht darin, für die Variablen, die das aktuelle Maximum bzw. Minimum enthalten, besonders kleine bzw. besonders grosse Startwerte zu wählen, welche garantiert kleiner bzw. grösser als der erste Wert sind. Eine gute Wahl sind:

```
minimum = float('inf')  # ist grösser als die grösste darstellbare Zahl
maximum = float('-inf') # ist kleiner als die kleinste darstellbare Zahl
```

Bei jedem weiteren Wert wird dann geprüft:

- Ist der neue Wert kleiner als das bisherige Minimum? Falls ja, wird das Minimum aktualisiert.
- Ist der neue Wert grösser als das bisherige Maximum? Falls ja, wird das Maximum aktualisiert.

Dazu werden Vergleichsoperatoren wie < und > verwendet.

3.5 Idee zur Berechnung des Durchschnitts

Für den Durchschnitt muss zusätzlich die Summe aller eingelesenen Werte sowie die Anzahl der Werte bekannt sein. Während des Einlesens kann daher laufend

- die Summe um den jeweils neuen Wert erhöht und
- ein Zähler für die Anzahl der eingelesenen Werte erhöht

werden.

Nach dem Einlesen der gesamten Datei lässt sich der Durchschnitt aus diesen beiden Grössen berechnen:

$$\bar{x} = \frac{\text{Summe aller Werte}}{\text{Anzahl der Werte}}.$$

Auf diese Weise müssen die einzelnen Werte nicht dauerhaft im Speicher abgelegt werden. Das Programm kann die Datei somit auch dann verarbeiten, wenn sie sehr viele Zahlen enthält.

4 Loslegen

1. Erstelle in deinem Informatikorder ein Unterverzeichnis `werteanalyse`.
2. Speichere dieses PDF und die Datei `zahlen.csv` in diesem Verzeichnis.
3. Starte IDLE, öffne eine neue Datei und speichere sie unter `werteanalyse.py`
4. Versuche mit den oben angegebenen Hinweisen, das in der Aufgabenstellung beschriebene Python-Programm zu schreiben.
5. Versuche zuerst, so viel wie möglich selbständig zu programmieren. Selbstverständlich darfst du das Python 3 Cheat Sheet verwenden. Wenn dir eine Fehlermeldungen nicht weiterhilft, kannst die KI deines Vertrauens fragen. Bestehe aber darauf, dass dir die KI erklärt, *was* du falsch gemacht hast.
6. Erst wenn du gar nicht mehr weiter weisst, kannst du Teile oder zur Not auch das ganze Programm von einer KI schreiben lassen. *Dann aber sollte sie dir jeden Abschnitt erklären und du musst diese Erklärungen so lange studieren, bis du alle verstanden hast.*
7. Selbstkontrolle:
   - Minimum: 12.0
   - Durchschnitt: 499313.3234
   - Maximum: 999996.0