Werkzeuge fuer das wissenschaftliche Arbeiten

Phyton for Machine Learning and Data Science

Abgabe: 15.12.2023

Contents

---

1 Projektaufgabe

In dieser Aufgabe beschäftigen wir uns mit der Objektorientierung in Python. Der Fokus liegt auf der Implementierung einer Klasse, dabei nutzen wir insbesondere auch Magic Methods.

1.1 Einleitung

Ein Datensatz besteht aus mehreren Daten; ein einzelnes Datum wird durch ein Objekt der Klasse `DataSetItem` repräsentiert. Jedes Datum hat einen Namen (Zeichenkette), eine ID (Zahl) und beliebigen Inhalt.

Nun sollen mehrere Daten, also Objekte vom Typ `DataSetItem`, in einem Datensatz zusammengefasst werden. Sie haben sich schon auf eine Schnittstelle und die benötigten Operationen geeinigt, die ein Datensatz unterstützen muss. Es gibt eine Klasse `DataSetInterface`, die die Schnittstelle definiert und die Operationen jedes Datensatzes angibt. Bisher fehlt jedoch noch die Implementierung eines Datensatzes mit allen Operationen.

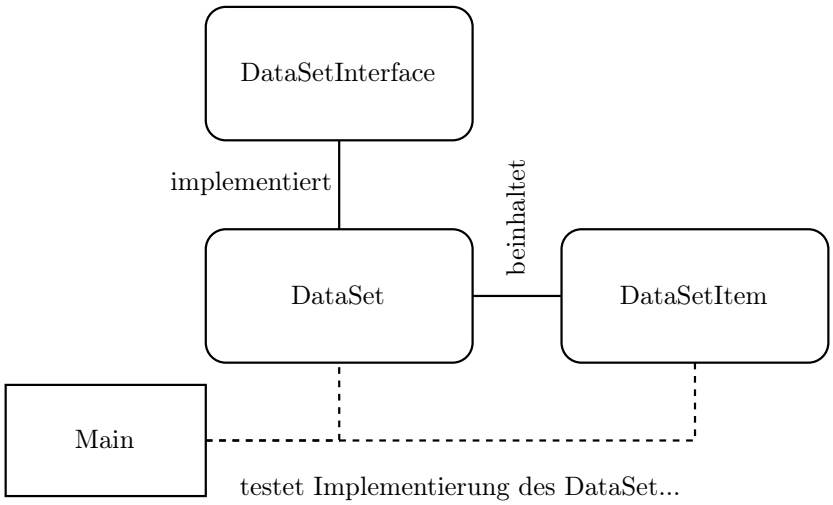

Figure 1: Darstellung der Klassenbeziehungen.

Implementieren Sie eine Klasse `DataSet` als Unterklasse von `DataSetInterface`.

1.2 Aufbau

Es gibt drei Dateien:

- `dataset.py`: Enthält die Klassen `DataSetInterface` und `DataSetItem`.
- `implementation.py`: Hier muss die Klasse `DataSet` implementiert werden.
- `main.py`: Nutzt die Klassen `DataSet` und `DataSetItem` aus den jeweiligen Dateien und testet die Schnittstelle und die Operationen von `DataSetInterface`.

1.3 Methoden

Die Klasse `DataSet` muss insbesondere die folgenden Methoden implementieren. Die genaue Spezifikation finden Sie in der Datei `dataset.py`:

- `__setitem__(self, name, id_content)`: Hinzufügen eines Datums mit Name, ID und Inhalt.
- `__iadd__(self, item)`: Hinzufügen eines `DataSetItem`.
- `__delitem__(self, name)`: Löschen eines Datums anhand des Namens. Der Name eines Datums ist ein eindeutiger Schlüssel und darf nur einmal pro Datensatz vorkommen.

- `__contains__(self, name)`: Prüfung, ob ein Datum mit diesem Namen im Datensatz vorhanden ist.
- `__getitem__(self, name)`: Abrufen des Datums über seinen Namen.
- `__and__(self, dataset)`: Bestimmung der Schnittmenge zweier Datensätze und Rückgabe als neuen Datensatz.
- `__or__(self, dataset)`: Bestimmung der Vereinigung zweier Datensätze und Rückgabe als neuen Datensatz.
- `__iter__(self)`: Iteration über alle Daten des Datensatzes (optional mit einer Sortierung).
- `filtered_iterate(self, filter)`: Gefilterte Iteration über einen Datensatz. Eine Lambda-Funktion mit den Parametern Name und ID dient als Filter.
- `__len__(self)`: Abrufen der Anzahl der Daten in einem Datensatz.

2 Abgabe

Programmieren Sie die Klasse `DataSet` in der Datei `implementation.py`, um die oben beschriebene Aufgabe im VPL zu lösen. Alternativ können Sie die Aufgabe auch direkt auf Ihrem Computer bearbeiten. Dazu finden Sie alle drei benötigten Dateien im Moodle zum Download.

Das VPL nutzt den gleichen Code, wobei die Datei `main.py` um weitere Testfälle und Überprüfungen erweitert wurde. Diese Überprüfungen dienen dazu, sicherzustellen, dass Sie die richtigen Klassen verwenden.

Hinweis: Die Dateien befinden sich im Ordner `/code/` dieses Git-Repositories.