Werkzeuge für das wissenschaftliche Arbeiten

Python for Machine Learning and Data Science

Abgabe: 15.12.2023

Contents

1 Projektaufgabe

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

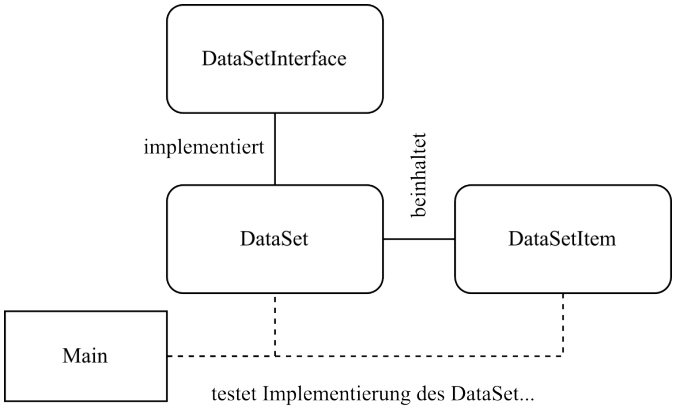

Abbildung 1: Darstellung der Klassenbeziehungen

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, Objekte vom Typ `DataSetItem`, in einem Datensatz zusammengefasst werden. Es gibt eine Klasse `DataSetInterface`, die die Schnittstelle definiert und die Operationen eines Datensatzes angibt. Bisher fehlt jedoch noch die Implementierung eines Datensatzes mit allen Operationen.

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

1.2 Aufbau

Es gibt drei Dateien, `dataset.py`, `main.py` und `implementation.py`. In der `dataset.py` befinden sich die Klassen `DataSetInterface` und `DataSetItem`, in der Datei `implementation.py` muss die Klasse `DataSet` implementiert werden. Die Datei `main.py` nutzt die Klassen `DataSet` und `DataSetItem` aus den jeweiligen Dateien und testet die Schnittstelle und Operationen von `DataSetInterface`.

1.3 Methoden

Bei der Klasse `DataSet` sind insbesondere folgende Methoden zu implementieren, die genaue Spezifikation finden Sie in der `dataset.py`:

1. `__setitem__`(self, name, id_content) - Hinzufügen eines Datums, mit Name, ID und Inhalt.
2. `__iadd__`(self, item) - Hinzufügen eines `DataSetItem`.
3. `__delitem__`(self, name) - Löschen eines Datums basierend auf dem Namen.
4. `__contains__`(self, name) - Prüfung, ob ein Datum mit diesem Namen im Datensatz vorhanden ist.
5. `__getitem__`(self, name) - Abrufen eines Datums über seinen Namen.
6. `__and__`(self, dataset) - Schnittmenge zweier Datensätze bestimmen und als neuen Datensatz zurückgeben.
7. `__or__`(self, dataset) - Vereinigungen zweier Datensätze bestimmen und als neuen Datensatz zurückgeben.

8. `__iter__`(self) - Iteration über alle Daten des Datensatzes.
9. `filtered_iterate`(self, filter) - Gefilterte Iteration über den Datensatz, wobei eine Lambda-Funktion als Filter dient.
10. `__len__`(self) - Anzahl der Daten im Datensatz abrufen.

2 Abgabe

Programmieren Sie die Klasse `DataSet` in der Datei `implementation.py` zur Lösung der oben beschriebenen Aufgabe im VPL. Sie können auch direkt auf Ihrem Computer programmieren, dazu finden Sie alle drei benötigten Dateien zum Download im Moodle.

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

** Dateien befinden sich im Ordner "/code/" dieses Git-Repositories.*