

Werkzeuge für das wissenschaftliche Arbeiten

Python for Machine Learning and Data Science

Abgabe: 15.12.2023

Inhaltsverzeichnis

1	Projektaufgabe	1
1.1	Einleitung	1
1.2	Aufbau	2
1.3	Methoden	2
2	Abgabe	3

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.

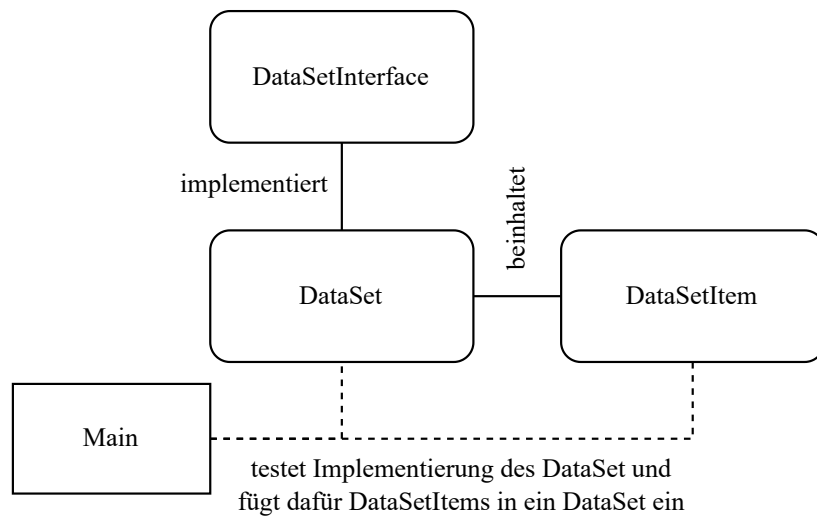


Abbildung 1: Darstellung der Klassenbeziehungen.

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 bel. Inhalt.

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

Die folgenden Methoden sollen in der Klasse `DataSet` implementiert werden:

- `__setitem__(self, name, id_content)`
Hinzufügen eines Datums.
- `__iadd__(self, item)`
Hinzufügen eines `DataSetItem`.
- `__delitem__(self, name)`
Löschen eines Datums anhand des Namens.
- `__contains__(self, name)`
Prüfung, ob ein Datum im Datensatz vorhanden ist.
- `__getitem__(self, name)`
Abrufen eines Datums über den Namen.
- `__and__(self, dataset)`
Schnittmenge zweier Datensätze als neuen Datensatz zurückgeben.
- `__or__(self, dataset)`
Vereinigung zweier Datensätze als neuen Datensatz zurückgeben.
- `__iter__(self)`
Iteration über alle Daten.

- `filtered_iterate(self, filter)`
Gefilterte Iteration über einen Datensatz.
- `__len__(self)`
Anzahl der Daten 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 die `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.