2025/11/29 12:33 1/4 LU02.A11 - Immutable Dataclass

LU02.A11 - Immutable Dataclasses



Erstellen Sie mehrere pure functions, die zusammenarbeiten, um eine komplexe Datenstruktur auf Basis von immutable Dataclasses zu manipulieren.

Aufgabenstellung

Im Kontext der funktionalen Programmierung spielt die Unveränderlichkeit von Datenstrukturen eine zentrale Rolle. Durch die Verwendung von immutable Dataclasses stellen wir sicher, dass einmal erstellte Objekte nicht unbeabsichtigt verändert werden können. Dies führt zu vorhersehbarem Verhalten, erleichtert das Debugging und reduziert die Wahrscheinlichkeit von Seiteneffekten in Ihrem Code. In dieser Aufgabe lernen Sie, wie man immutable Dataclasses verwendet, um komplexe Datenstrukturen zu manipulieren, ohne den ursprünglichen Zustand zu verändern. Das fördert die Entwicklung von stabilen und wartbaren Programmen.

- 1. Definieren Sie eine @dataclass(frozen=True) namens Student mit den Attributen name (str), grades (List[int]), und graduated (bool).
- 2. Implementieren Sie eine Funktion add_grade(student: Student, grade: int) → Student, die eine neue Instanz der Dataclass Student zurückgibt, wobei die neue Note der Liste hinzugefügt wird.
- 3. Implementieren Sie eine Funktion calculate_average(student: Student) → float, die den Durchschnitt der Noten berechnet.
- 4. Implementieren Sie eine Funktion graduate_student(student: Student) → Student, die eine neue Instanz der Dataclass zurückgibt, bei der das Attribut graduated auf True gesetzt wird, wenn der Notendurchschnitt 70 oder mehr beträgt.
- 5. Schreiben Sie ein Programm, das mehrere Noten hinzufügt, den Durchschnitt berechnet und den Studenten "absolviert", falls die Kriterien erfüllt sind. Geben Sie die Ergebnisse aus.

Code Vorlage

```
def add_grade(student, grade):
    """

    Returns a new Student instance with the added grade.
    """
    new_grades = student.grades + [grade]
    return Student(name=student.name, grades=new_grades,
graduated=student.graduated)

def calculate_average(student):
    """
    Returns the average of the student's grades.
    """
```

```
if not student.grades:
        return 0.0
    return sum(student.grades) / len(student.grades)
def graduate_student(student):
   Graduates the student if the average grade is 70 or above.
   average = calculate average(student)
    if average >= 70:
        return Student(name=student.name, grades=student.grades,
graduated=True)
    return student
if name == ' main ':
    student = Student(name='John Doe')
    student = add grade(student, 85)
    student = add grade(student, 75)
    student = add grade(student, 60)
   print(f'Noten: {student.grades}')
   average = calculate average(student)
   print(f'Durchschnitt: {average}')
    student = graduate student(student)
    print(f'Absolviert: {student.graduated}')
```

Schritt für Schritt

- 1. Definieren Sie die immutable Dataclass Student in einer separaten Datei student.py.
- 2. Implementieren Sie die Funktion add_grade, die eine neue Instanz zurückgibt und die Note hinzufügt.
- 3. Implementieren Sie die Funktion calculate_average, die den Notendurchschnitt berechnet.
- 4. Implementieren Sie die Funktion graduate_student, die eine neue Instanz zurückgibt, wenn der Student die Anforderungen erfüllt.
- 5. Führen Sie mehrere Funktionsaufrufe durch, um die Noten zu aktualisieren, den Durchschnitt zu berechnen und den Studenten bei Erfüllung der Kriterien zu absolvieren.

Lösung

```
# student.py
from dataclasses import dataclass, field
from typing import List

@dataclass(frozen=True)
class Student:
    name: str
```

https://wiki.bzz.ch/ Printed on 2025/11/29 12:33

2025/11/29 12:33 3/4 LU02.A11 - Immutable Dataclass

```
grades: List[int] = field(default_factory=list)
graduated: bool = False
```

```
# main.py
from student import Student
def add_grade(student, grade):
   Returns a new Student instance with the added grade.
   new_grades = student.grades + [grade]
    return Student(name=student.name, grades=new grades,
graduated=student.graduated)
def calculate average(student):
   Returns the average of the student's grades.
   if not student.grades:
        return 0.0
    return sum(student.grades) / len(student.grades)
def graduate student(student):
   Graduates the student if the average grade is 70 or above.
   average = calculate average(student)
    if average >= 70:
        return Student(name=student.name, grades=student.grades,
graduated=True)
    return student
if __name__ == '__main ':
    student = Student(name='John Doe')
    student = add_grade(student, 85)
    student = add grade(student, 75)
    student = add grade(student, 60)
   print(f'Noten: {student.grades}')
   average = calculate average(student)
    print(f'Durchschnitt: {average}')
    student = graduate student(student)
    print(f'Absolviert: {student.graduated}')
```



From:

https://wiki.bzz.ch/ - BZZ - Modulwiki

Permanent link:

https://wiki.bzz.ch/modul/m323/learningunits/lu02/aufgaben/dataclass2?rev=1724835459

Last update: 2024/08/28 10:57



https://wiki.bzz.ch/ Printed on 2025/11/29 12:33