

Programmierkonzepte in den Naturwissenschaften

11. Programmierparadigmen

PD Dr. Till Biskup

Physikalische Chemie und Didaktik der Chemie
Universität des Saarlandes
Sommersemester 2020





- ❏ Paradigmen beeinflussen die Sicht des Programmierers auf mögliche Lösungswege für eine Problemstellung.
- ❏ Bestimmte Paradigmen haben die Entwicklung zentraler Aspekte der Programmierung erleichtert bzw. ermöglicht.
- ❏ (Grobe) Kenntnis der Paradigmen ist wichtig, um sich innerhalb der Disziplin verständigen zu können.
- ❏ Programmierparadigmen schließen einander in der Regel nicht gegenseitig aus.
- ❏ Alle drei hier vorgestellten Paradigmen zielen auf Modularität und Wiederverwendbarkeit von Code.

Begriffsklärung: Was ist ein (Programmier-)Paradigma?

Warum sich mit Programmierparadigmen befassen?

Programmierparadigmen: eine Übersicht

Was ist ein (Programmier-)Paradigma?

Eine Begriffsklärung und Herkunftsbestimmung



Paradigmata

allgemein anerkannte wissenschaftliche Leistungen,
die für eine gewisse Zeit einer Gemeinschaft von Fachleuten
maßgebende Probleme und Lösungen liefern

(Thomas S. Kuhn)

- ▶ Von Robert W. Floyd 1979 auf Programmierung angewandt
 - unter direkter Bezugnahme auf Thomas Kuhn
- 👉 Programmierparadigmen beeinflussen die Wahl der Lösungswege für eine gegebene Fragestellung.

Thomas S. Kuhn: Die Struktur wissenschaftlicher Revolutionen, Suhrkamp, Frankfurt 1976, S. 10
Robert W. Floyd, *Commun. ACM* 22:455, 1979

Warum sich mit Paradigmen befassen?

Paradigmen sind die vorherrschenden „Erzählungen“ einer Disziplin



vorherrschende „Erzählungen“ einer Disziplin

- ▶ beeinflussen die Sicht auf mögliche Lösungswege
 - schränken mitunter die Möglichkeiten ein
 - fundamental unterschiedliche Herangehensweisen
 - ▶ Einfluss auf die Qualität der Software
 - mitunter entscheidend für intellektuelle Beherrschbarkeit
 - erleichtern die Abbildung der Realität auf Code
 - ▶ erleichtern die Kommunikation
 - Namen transportieren relativ klar definierte Konzepte.
 - verwendetes Paradigma oft am Code klar erkennbar
- ☞ Paradigmen wirken strukturierend und sind wesentlich für die intellektuelle Beherrschbarkeit.

Bedeutung für die Wissenschaftlichkeit

- ▶ „Sauberer Code“
 - lesbar, modular, testbar
 - Voraussetzung für Kriterien der Wissenschaftlichkeit
 - ▶ objektorientierte Programmierung
 - hilft bei Modularität und Testbarkeit
 - sorgt für Wiederverwendbarkeit
 - ▶ intellektuelle Beherrschbarkeit
 - wissenschaftliche Datenauswertung ist inhärent komplex
 - Paradigmen helfen bei der Bewältigung der Komplexität
- ☞ Paradigmen unterscheiden sich *grundsätzlich*,
Grundkenntnisse sind deshalb auch in der Wissenschaft wichtig.

Drei grundlegende Programmierparadigmen

- ▶ strukturiert
 - nur zwei Kontrollstrukturen: Selektion und Iteration
 - Ein Codeblock hat nur einen Ein- und Ausgang.
- ▶ objektorientiert
 - Daten und zugehörige Methoden bilden eine Einheit.
 - Wiederverwendbarkeit durch Vererbung/Polymorphismus
- ▶ funktional
 - basiert auf der Anwendung von Funktionen auf Argumente
 - Verzicht auf Zuweisungen
- ☛ Strukturierte und objektorientierte Programmierung greifen ineinander (OO-Methoden meist strukturiert programmiert).

Ziel: Modularität und Wiederverwendbarkeit von Code

- ▶ strukturiert
 - Verzicht auf `goto`-Befehle für beliebige Sprünge
 - Modularisierung: einzelne, unabhängige Programmteile
 - ▶ objektorientiert
 - Verzicht auf Zeiger auf Funktionen
 - Polymorphismus durch Strukturen der Programmiersprache
 - ▶ funktional
 - Verzicht auf Zuweisungen
 - keine (ungewollten) Seiteneffekte, ideal für Parallelisierung
- ☞ Disziplinierung durch Verzicht ermöglicht die intellektuelle Beherrschbarkeit und sorgt so für höhere Qualität.

Beispiel: Fibonacci-Folge

$$f_n = f_{n-1} + f_{n-2} \quad \text{für } n > 2 \quad \text{mit } f_0 = 0, \quad f_1 = f_2 = 1$$

Listing 1: Fibonacci-Zahlen, strukturiert

```
def fibonacci(n, first=0, second=1):  
    for _ in range(n):  
        print(first) # Seiteneffekt  
        first, second = second, first + second # Zuweisung  
    fibonacci(10)
```

Listing 2: Fibonacci-Zahlen, funktional

```
fibonacci = (lambda n, first=0, second=1: # Lambda-Ausdruck  
    [] if n == 0 else  
    [first] + fibonacci(n-1, second, first+second)) # Rekursion  
print(fibonacci(10))
```



- ❏ Paradigmen beeinflussen die Sicht des Programmierers auf mögliche Lösungswege für eine Problemstellung.
- ❏ Bestimmte Paradigmen haben die Entwicklung zentraler Aspekte der Programmierung erleichtert bzw. ermöglicht.
- ❏ (Grobe) Kenntnis der Paradigmen ist wichtig, um sich innerhalb der Disziplin verständigen zu können.
- ❏ Programmierparadigmen schließen einander in der Regel nicht gegenseitig aus.
- ❏ Alle drei hier vorgestellten Paradigmen zielen auf Modularität und Wiederverwendbarkeit von Code.