TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Professur für Didaktik der Informatik
Dr. Thiemo Leonhardt

Programmierparadigmen
Funktionale Modellierung

Professur für **Didaktik der Informatik**

Funktionen in der Programmierung

Wiederholung zu Unterprogrammen:

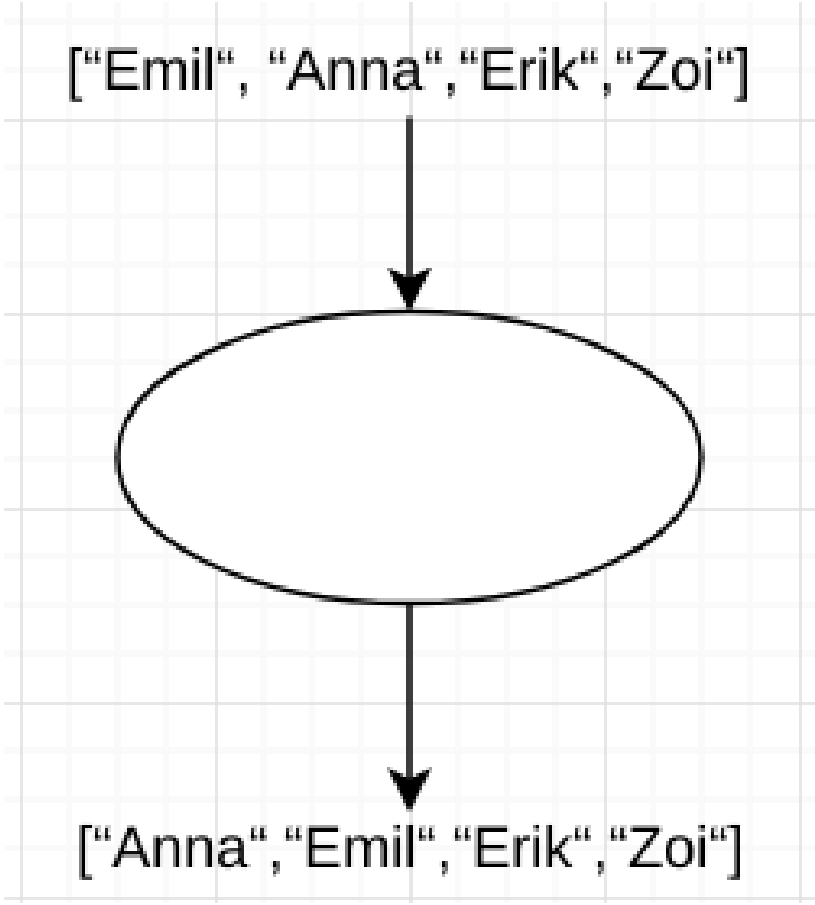
- lösen Teilaufgabe
- fassen Anweisungen in einem Block zusammen
- werden nicht unmittelbar ausgeführt, sondern erst beim Aufruf

Besonderheit von Funktionen gegenüber Prozeduren:

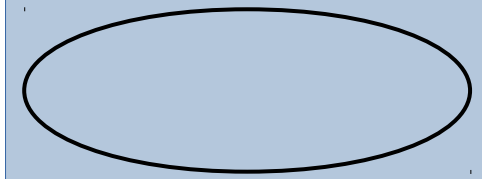
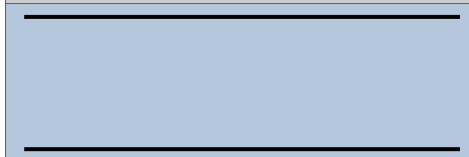
- Funktionen liefern eine Rückgabe und
- können damit in Ausdrücken verwendet werden (Verkettung).

Beispiel – Algorithmus irrelevant

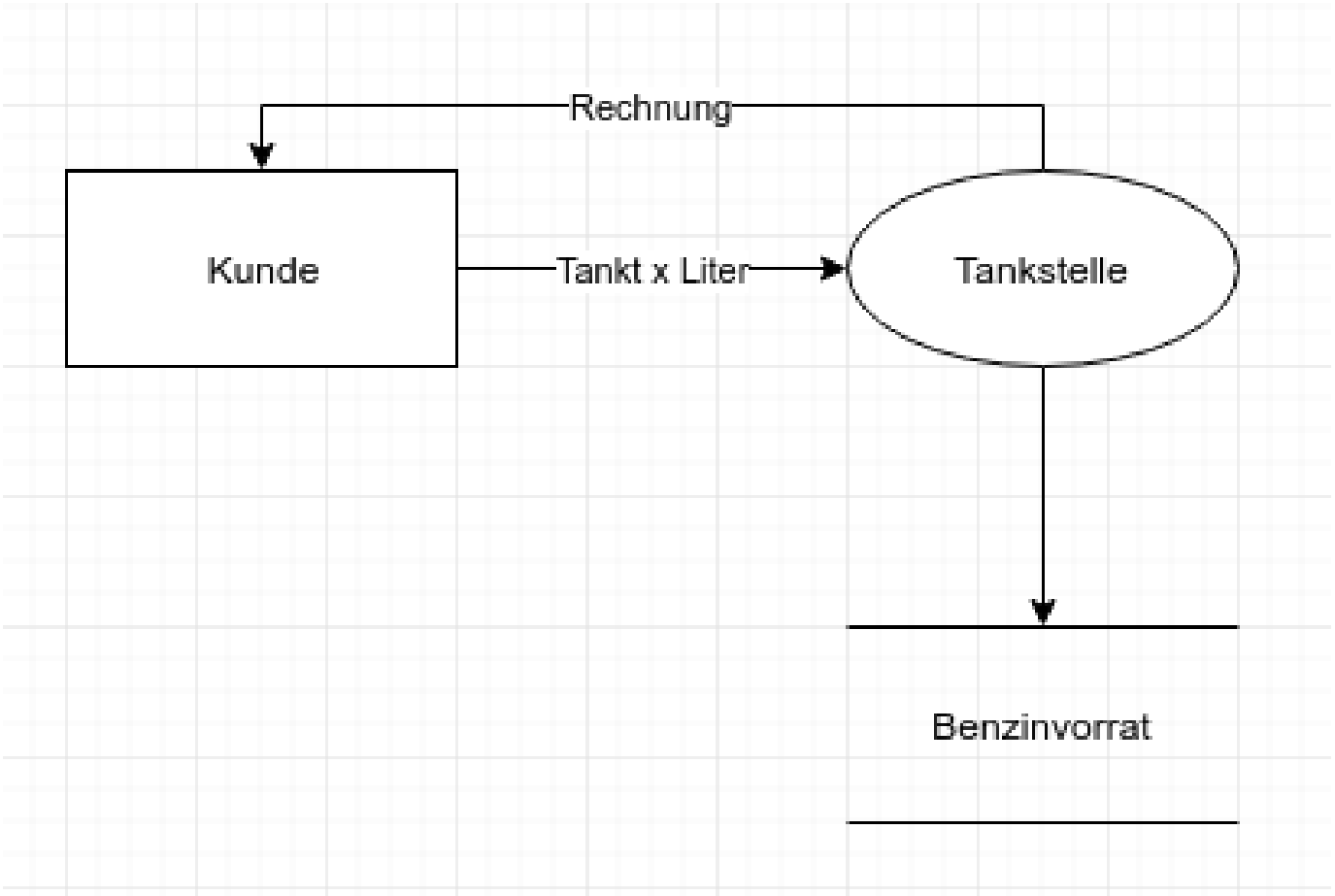
- Aufgabe
  - Sortiere
    - [“Emil“, “Anna“,“Erik“,“Zoi“]
  - Ergebnis
    - [“Anna“,“Emil“,“Erik“,“Zoi“]
  - In der funktionalen Sicht steht die **Funktion** im Mittelpunkt nicht die konkrete Umsetzung des Algorithmus.
    - Bubblesort, Quicksort, …



Funktionale Modellierung - Datenflussdiagramme

	Datenverarbeitende Prozesse	Nehmen Daten über Eingangskanäle entgegen, verarbeiten diese und geben die Daten über Ausgangskanäle weiter.
	Datenquellen oder -senken	Kommunikation nach außen und von außen in das System
	Speicherkomponenten	Hier werden Daten gespeichert und abgeholt.
	Datenflüsse	Leitungen über die Daten transportiert werden.

Beispiel - Tankstelle

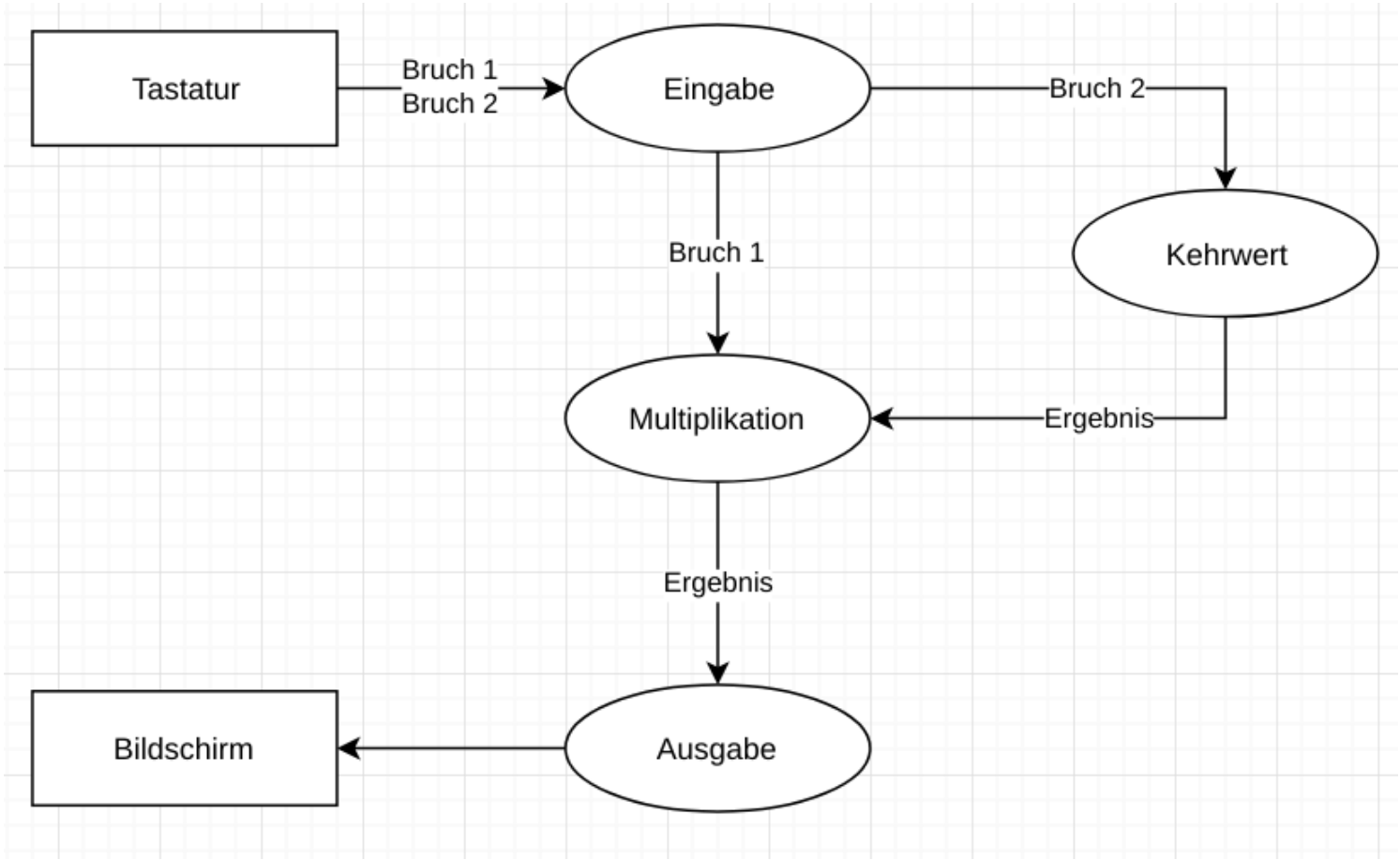


Funktionale Modellierung – innere Struktur eines Programms

- Beispiel
  - Aufteilung eines Programms zur Bruchrechnung mit 2 Elementen
    - Eingabe
    - Verarbeitung
    - Ausgabe
- Aufgabe
  - Aufteilen des Beispiels in sinnvolle Unterprogramme
  - Es geht **nicht** um den Algorithmus!

- Beispiellösung
  - Eingabe von 2 Brüchen
    - Jeweils Zähler und Nenner
  - Ausgabe eines Bruchs
  - Kehrwert des 2ten Bruches
  - Multiplikation von 2 Brüchen

Funktionale Modellierung



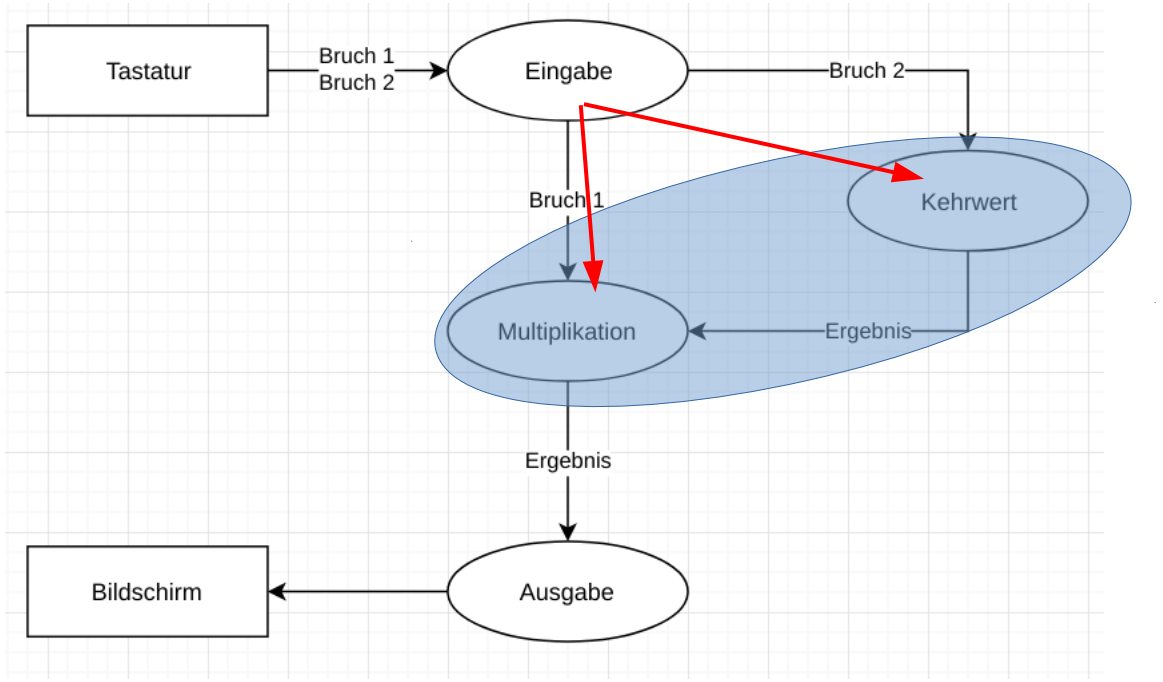
Funktionale Programmierung

```
def eingabe():
    pass

def kehrwert(bruch):
    pass

def multiplikation(bruch1, bruch2).
    pass

def ausgabe(bruch):
    pass
```

- Datenverarbeitende Prozesse werden in Funktionen übersetzt.
- Eingänge dieser in Parameter der Funktionen.

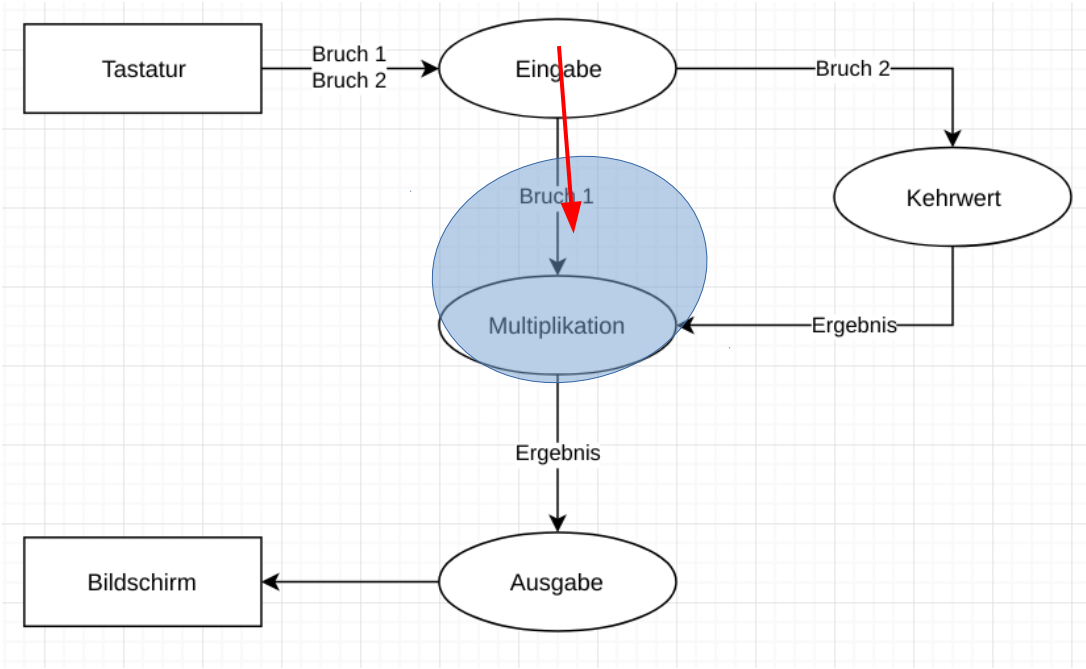
Funktionale Programmierung

- Die Eingabe gibt die Daten an 2 Funktionen weiter
  - Multiplikation
  - Kehrwert
- Datenweitergabe kann durch Aufruf der Funktion erfolgen.



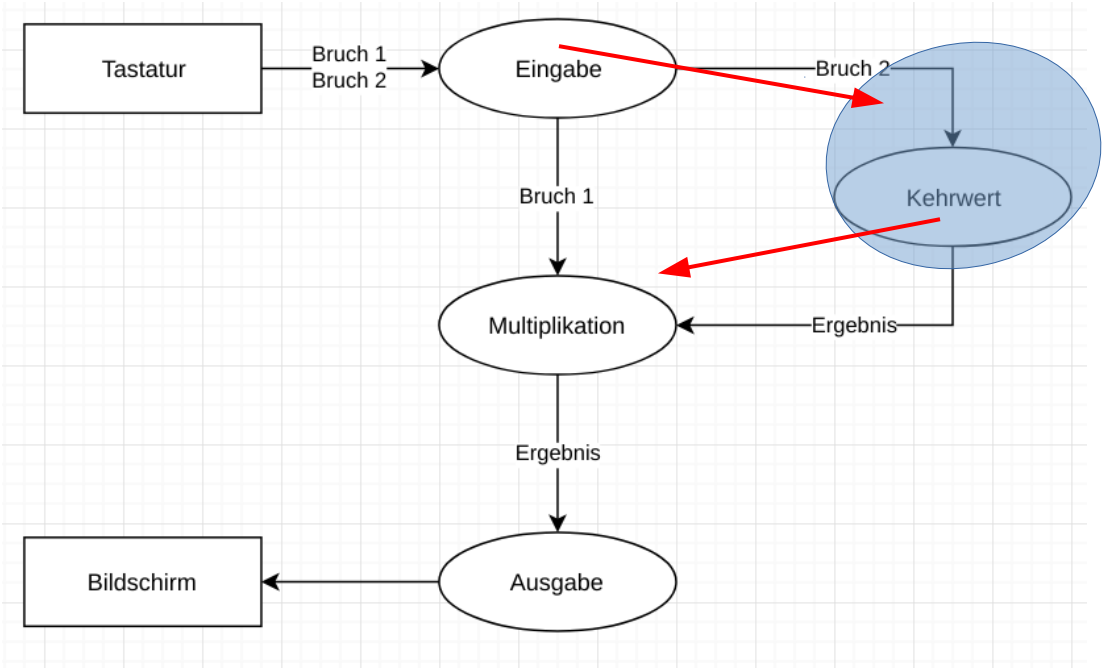
Funktionale Programmierung

```
def eingabe():
    #bruch1
    multiplikation(bruch1, ???)
    pass

def kehrwert(bruch):
    pass

def multiplikation(bruch1, bruch2):
    pass

def ausgabe(bruch):
    pass
```

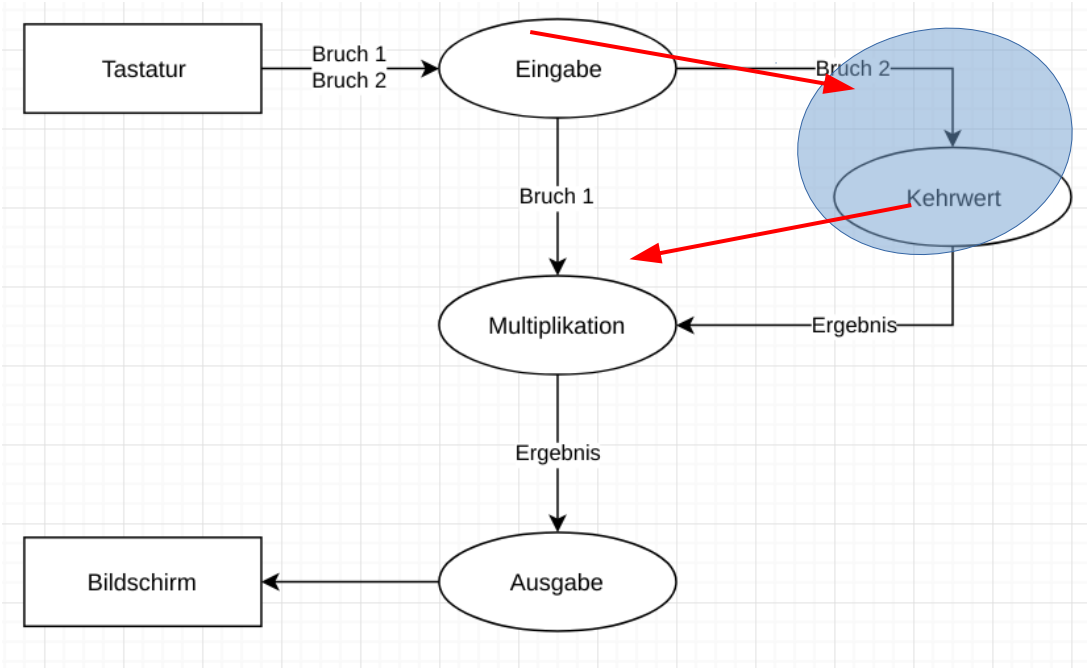


Funktionale Programmierung

```
def eingabe():
    #bruch1
    multiplikation(bruch1, ???)
    pass

def kehrwert(bruch):
    pass

def multiplikation(bruch1, bruch2):
    pass

def ausgabe(bruch):
    pass
```

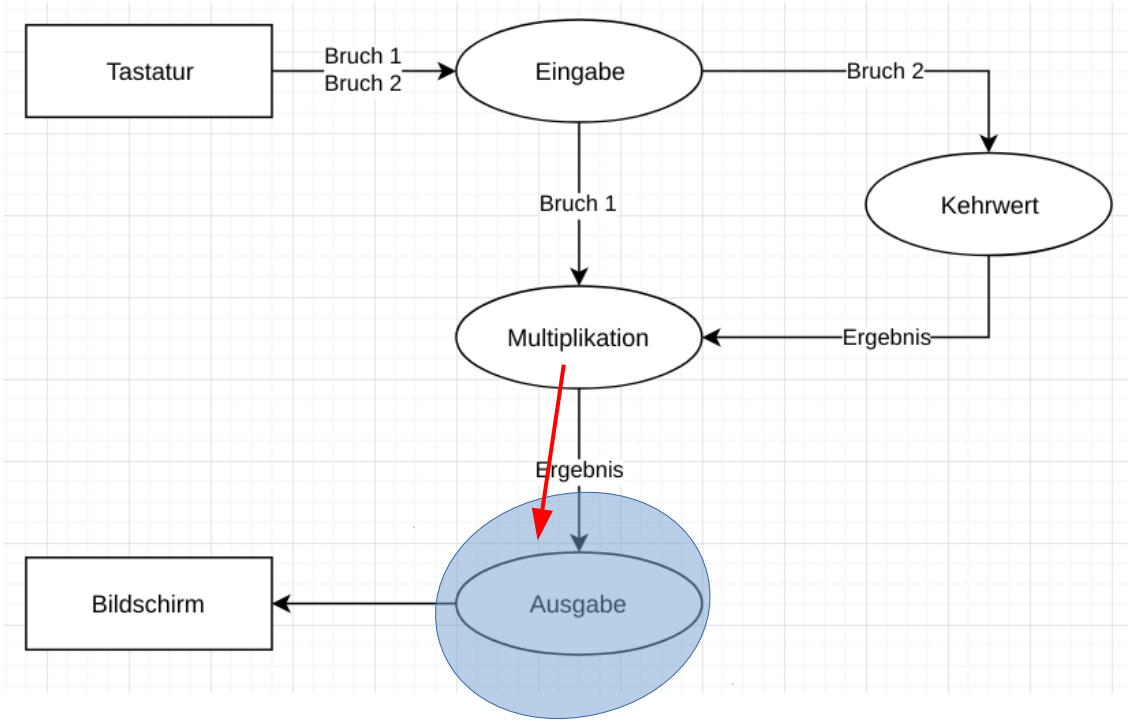


Funktionale Programmierung



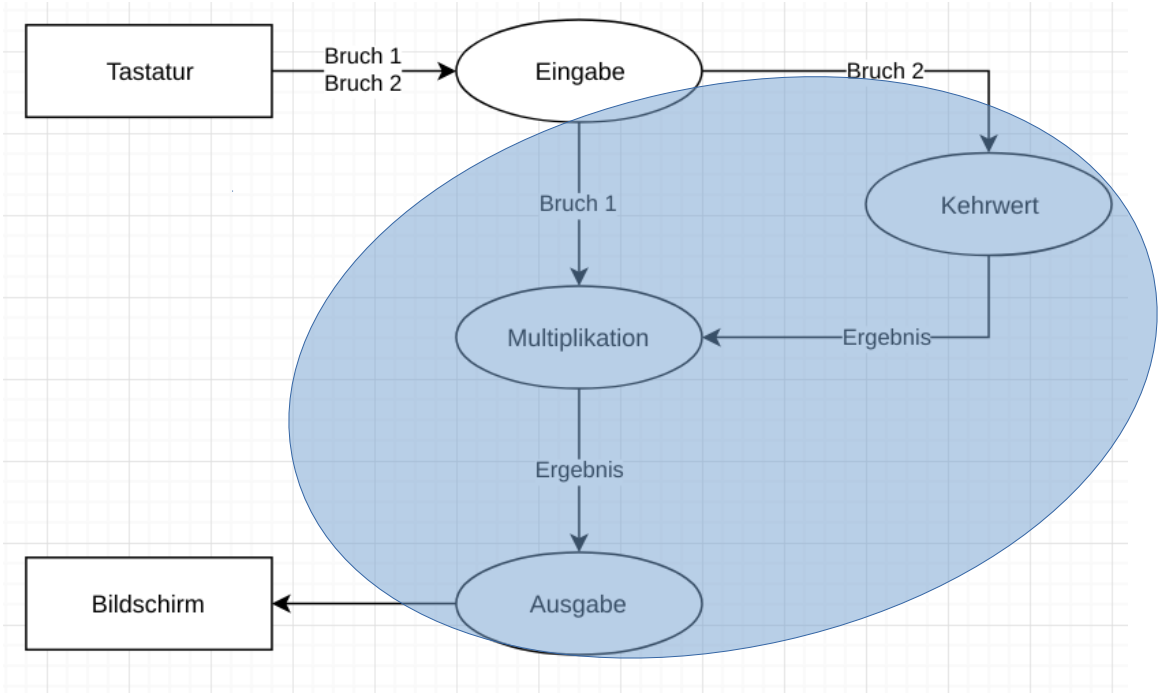

```
def eingabe():
    #bruch1 bruch2
    multiplikation(bruch1, kehrwert(bruch2))
    pass

def kehrwert(bruch):
    pass

def multiplikation(bruch1, bruch2):
    pass

def ausgabe(bruch):
    pass
```

Funktionale Programmierung




```
def eingabe():
    #bruch1 bruch2
    multiplikation(bruch1, kehrwert(bruch2))
    pass

def kehrwert(bruch):
    pass

def multiplikation(bruch1, bruch2):
    pass

def ausgabe(bruch):
    pass
```

Funktionale Programmierung




```python
def eingabe():
    # Bruch 1 und Bruch 2 einlesen
    ausgabe(multiplikation(bruch1, kehrwert(bruch2)))
    pass

def kehrwert(bruch):
    pass

def multiplikation(bruch1, bruch2):
    pass

def ausgabe(bruch):
    pass
```