TECHNISCHE UNIVERSITÄT DRESDEN

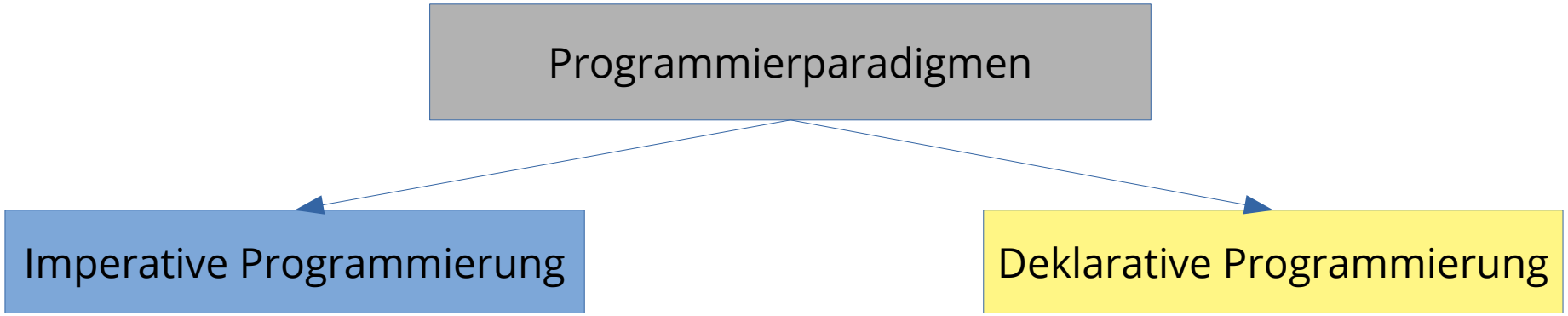
DRESDEN concept

Professur für Didaktik der Informatik
Dr. Thiemo Leonhardt

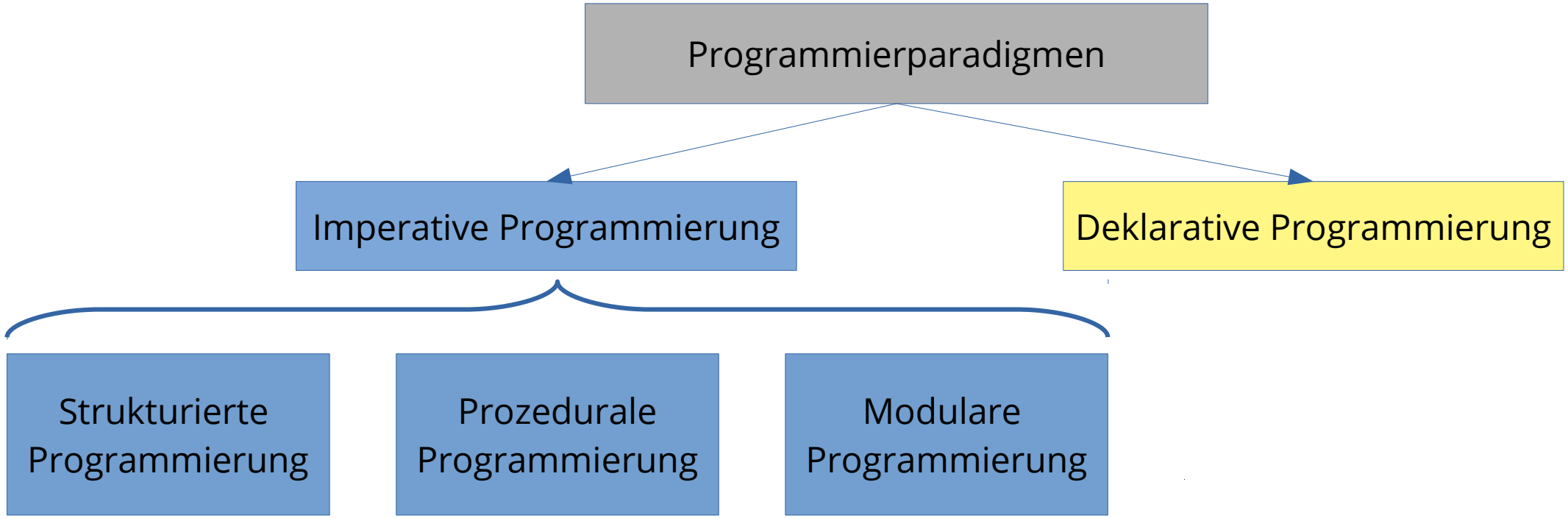
Programmierparadigmen
Imperative Programmierung

Professur für **Didaktik der Informatik**

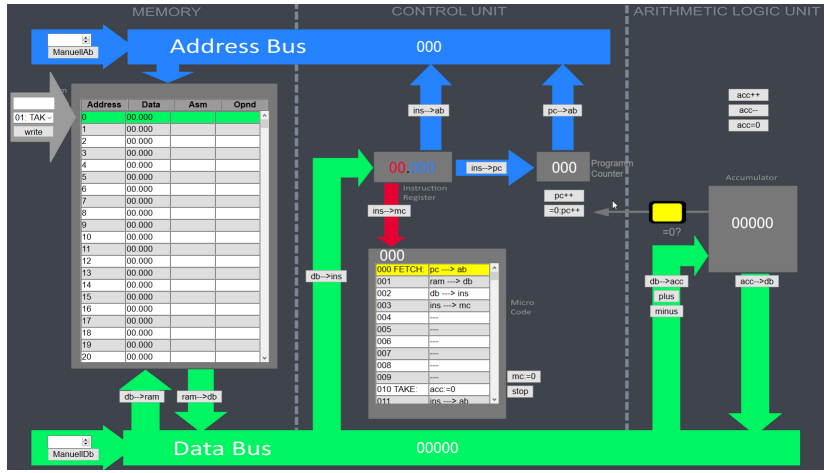
Klassifikation von Programmiersprachen

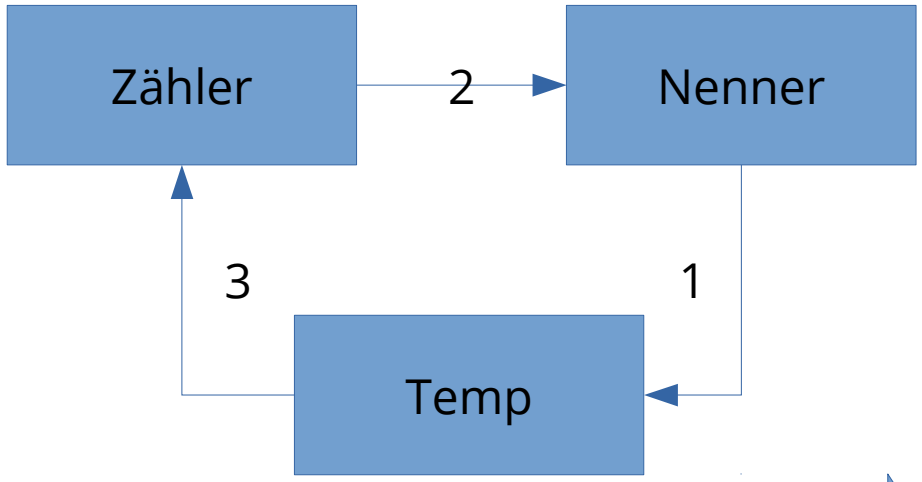


Klassifikation von Programmiersprachen

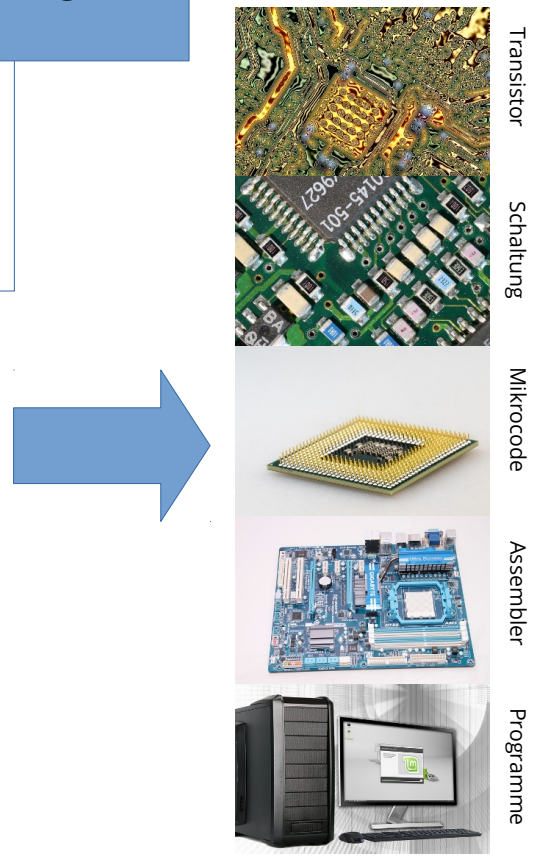


Imperative Programmierung



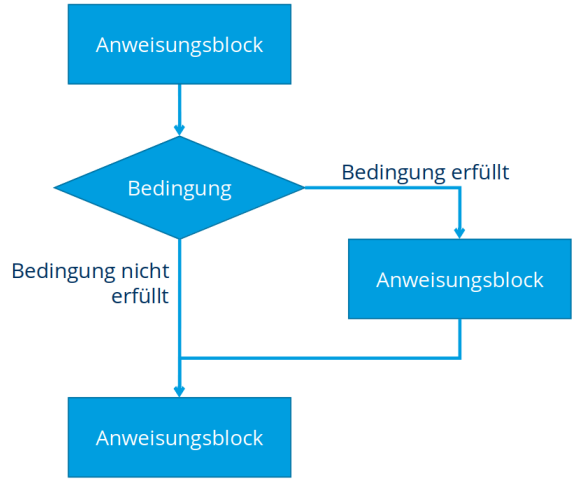



```
1
2  zaehler = 5
3  nenner = 6
4
5  # Kehrwert
6  tmp = zaehler
7  zaehler = nenner
8  nenner = tmp
```



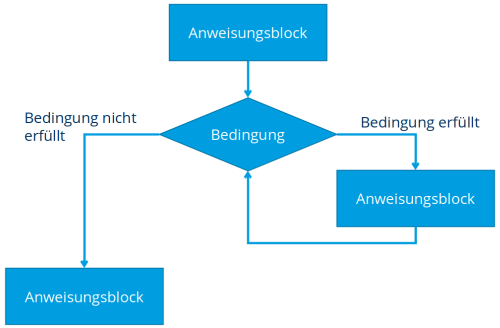
In der imperativen Sicht ist das zentrale Konzept die Zuweisung und der Zustand von Variablen.

Strukturierte Programmierung



Notation in Python:

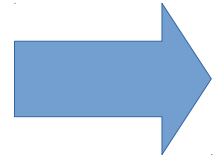
```
1 Anweisungen
2 if Bedingung:
3     Anweisungen
4 Anweisungen
5 …
```

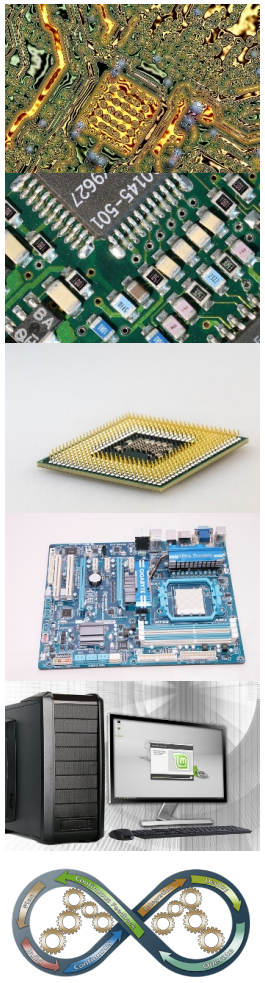


Notation in Python:

```
1 Anweisungen
2 while Bedingung:
3     Anweisungen
4 Anweisungen
5 …
```

```
1  print()
2
3  x = 10
4
5  if x = 10:
6      print('True')
7  else:
8      print('False')
9
10 while x == 10:
11     print('x hat den Wert 10')
12     x = x - 1
```



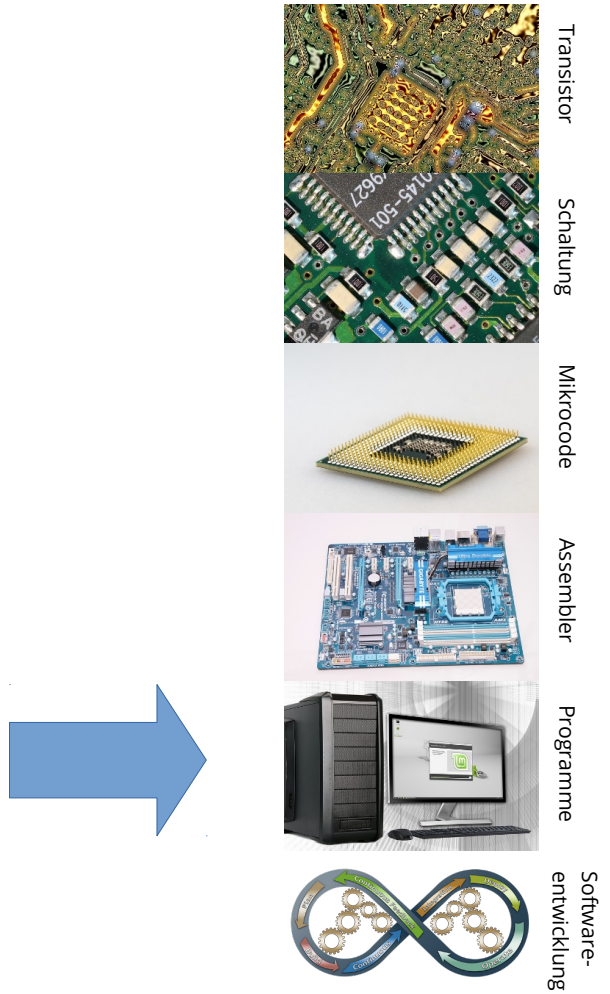


Erweiterung um Sequenz (Blöcke), Selektion (if) und Iteration (while, for)

Prozedurale Programmierung

Beispiel in Python:

```
1  def drawBigSquare(color):
2      Rect(50, 50, 300, 300, fill=color)
3
4  drawBigSquare('orange')
```



Erweiterung um Prozeduren als Unterprogramme und Einführung lokaler und globaler Variablen.
Unterprogramme sind wieder verwertbar und können beliebig oft im Programm aufgerufen werden.

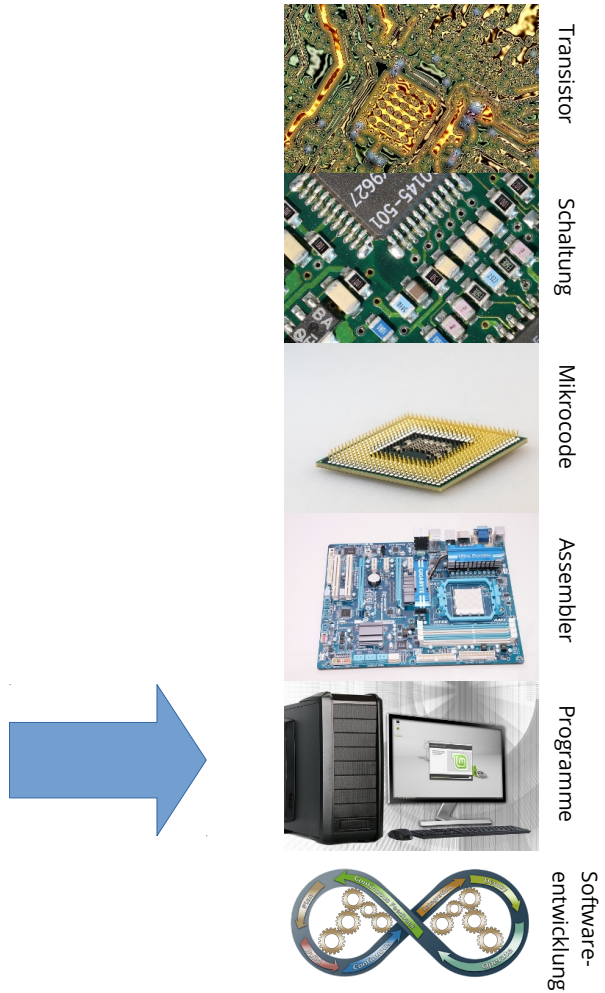
Modulare Programmierung

```
1 from random import randint
2
3 zufallszahl = randint(0,5)
```

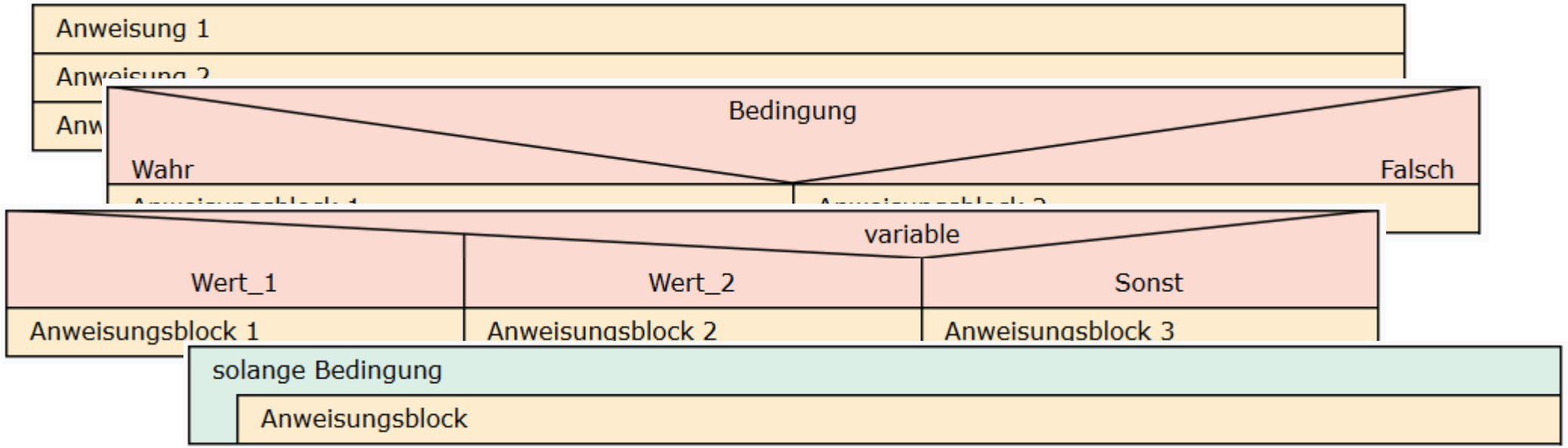
```
# random.py

def randint(min, max):
        pass
```

```
# meinProgramm.py

from random import randint

zahl = randint(0, 10)
```

Erweiterung um den Import externer Inhalte.
Dadurch können sogenannte Bibliotheken (libraries) erstellt werden, die wiederverwendbare Programmstücke enthalten.

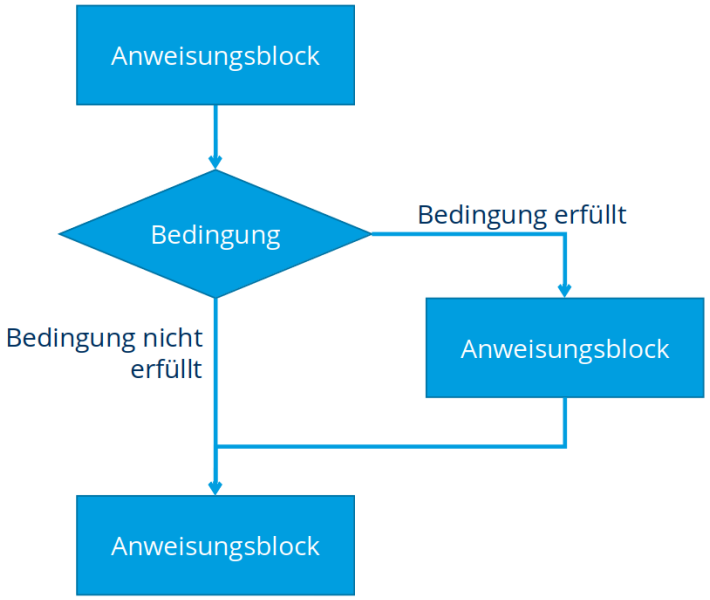


Imperative Modellierung – Nassi-Shneiderman-Diagramm 1971



Prozedur

Imperative Modellierung – PAP 1980



Notation in Python:

```
1 Anweisungen
2 if Bedingung:
3     Anweisungen
4 Anweisungen
5 …
```

```
1  print()
2
3  x = 10
4
5  if x = 10:
6      print('True')
7  else:
8      print('False')
9
10 while x == 10:
11     print('x hat den Wert 10')
12     x = x - 1
```

While-Berechenbarkeit

- While-berechenbar und Turing-berechenbar sind äquivalent
- Konsequenz:
  - jedes beliebige berechenbare Problem, lässt sich durch ein WHILE-Programm ausdrücken
  - alle weiteren Konstrukte unterstützen die Lesbarkeit, Wartbarkeit und den Entwurf von Software aber beeinflussen nicht die Ausdrucksstärke

Definition WHILE-Programm

$P ::= x_i = x_j + c$

$| \; x_i = x_j - c$

$| \; P; P$

$| \; \text{WHILE } x_i \neq 0 \text{ DO } P \text{ END}$

if-Verzweigung durch while

```
1 if Bedingung:
2     # True-Zweig
3 else:
4     # False-Zweig
```

Klassifikation von Programmiersprachen