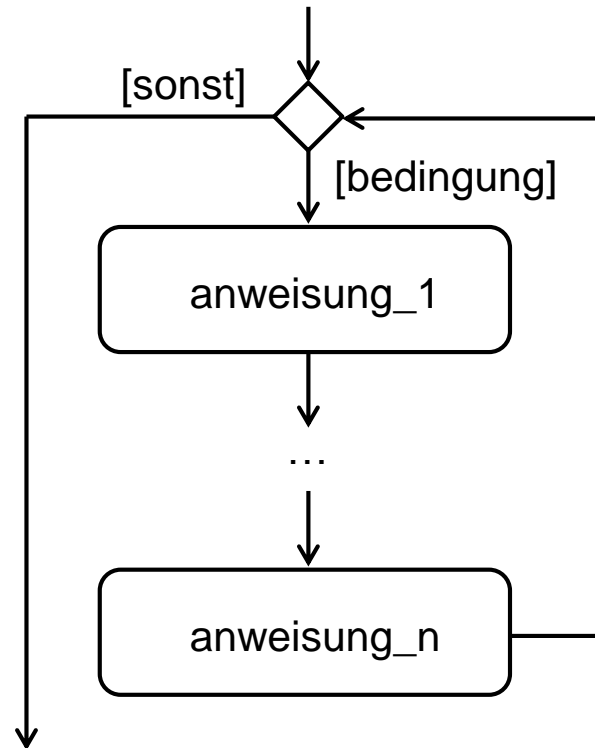


- Ein weiterer Datentyp in Python sind **boolesche Werte**. Es gibt nur die Werte True (wahr) und False (falsch).
- Die oben genannten Vergleichsoperatoren liefern als Ergebnis einen booleschen Wert.
- Die logischen Operatoren and, or, not lassen sich auf boolesche Ausdrücke anwenden.
- Beispiel:

```
position = -1.5  
print(not (position >= -1 and position >= 1))    # Ausgabe: True
```

■ while-Schleife:

Als Diagramm:

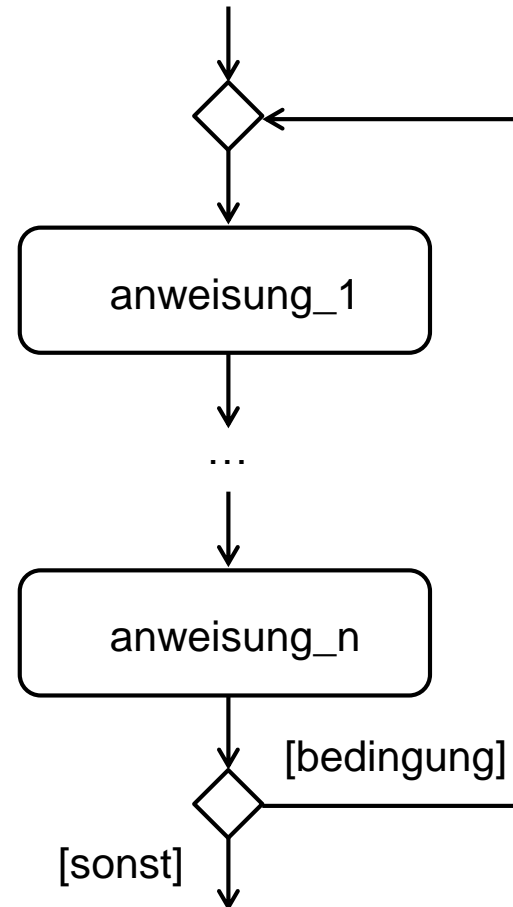


In Python:

```
while bedingung:  
    anweisung_1  
    ...  
    anweisung_n
```

■ Die Einrückung von Blöcken ist in Python unbedingt erforderlich!

■ do-while-Schleife:



```
while True:
    anweisung_1
    ...
    anweisung_n
    if not bedingung:
        break
```

- Die break-Anweisung kann auch an einer beliebigen Stelle im Schleifenkörper stehen. Diese Anweisung bricht die Schleife sofort ab.

- for-Schleifen:

- In Pseudocode:

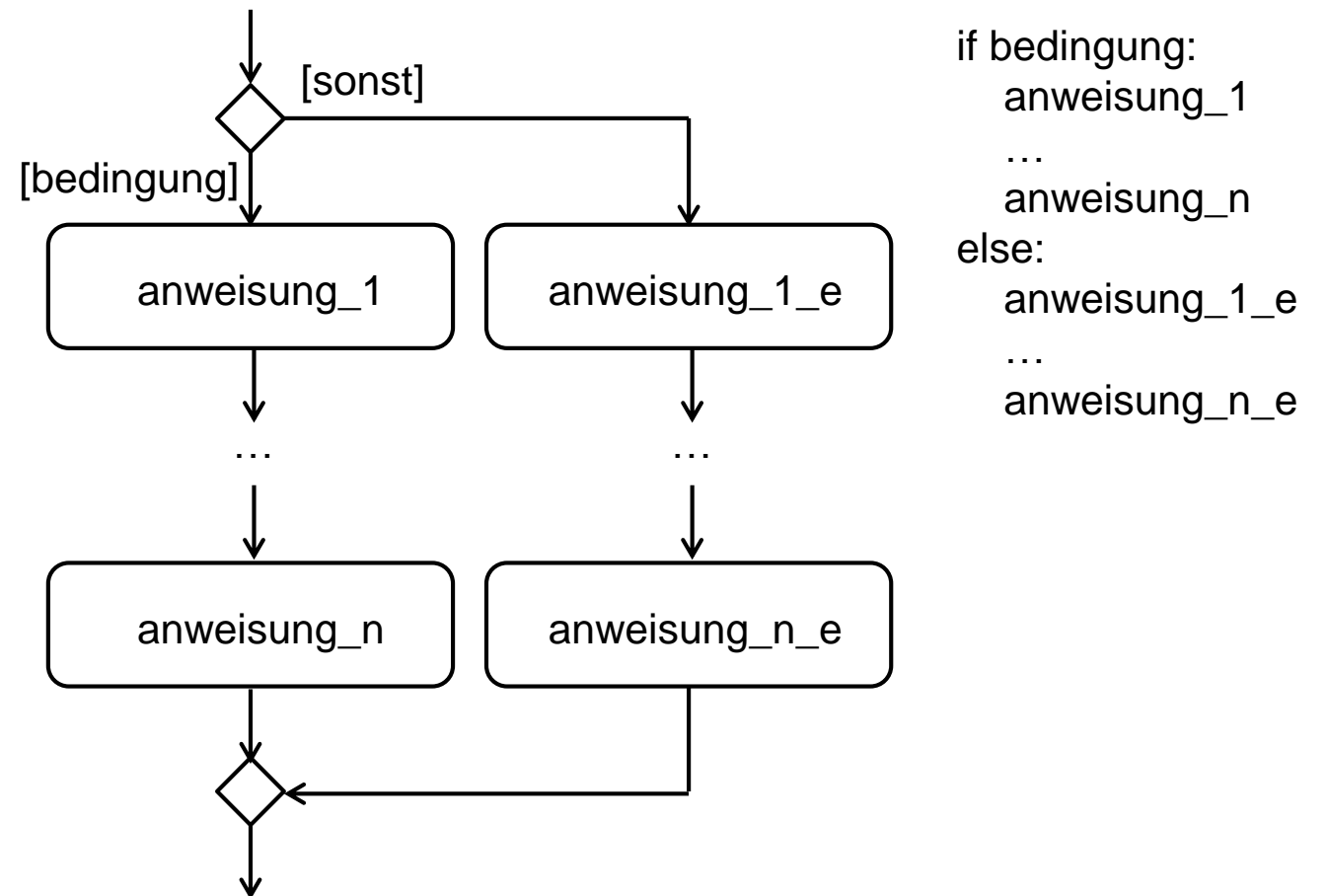
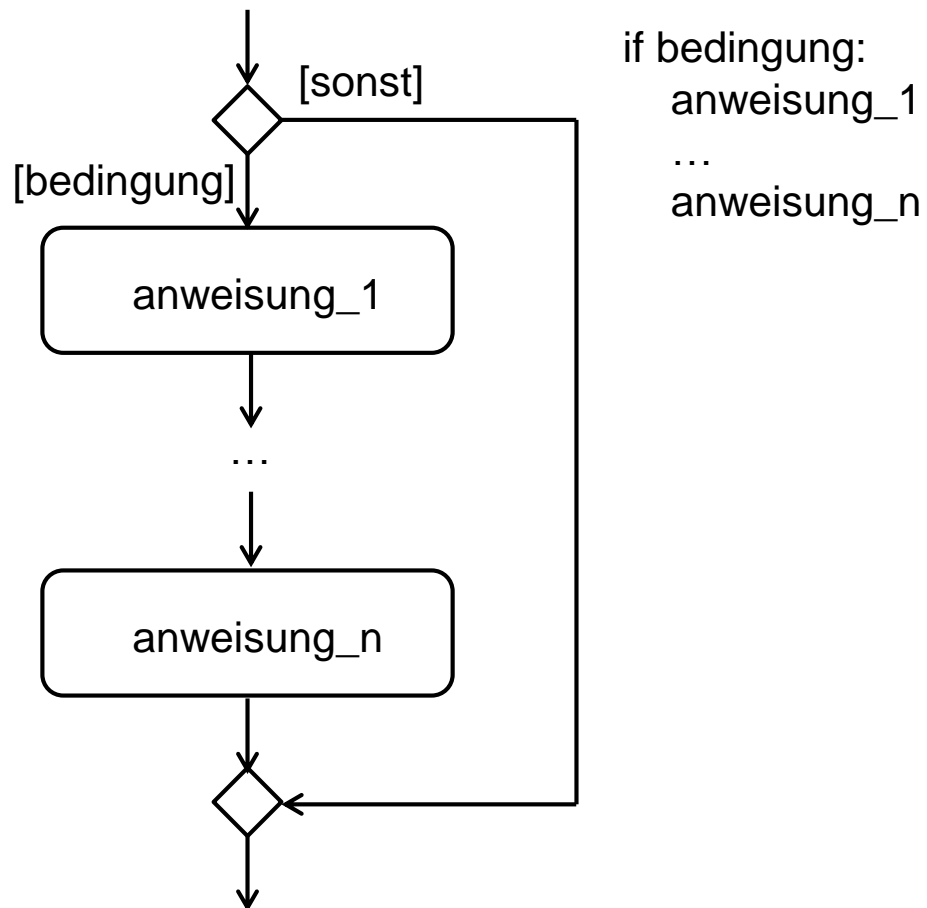
```
for i = Startwert to Endwert with Schrittweite do
    anweisung_1
    ...
    anweisung_n
```

- In Python:

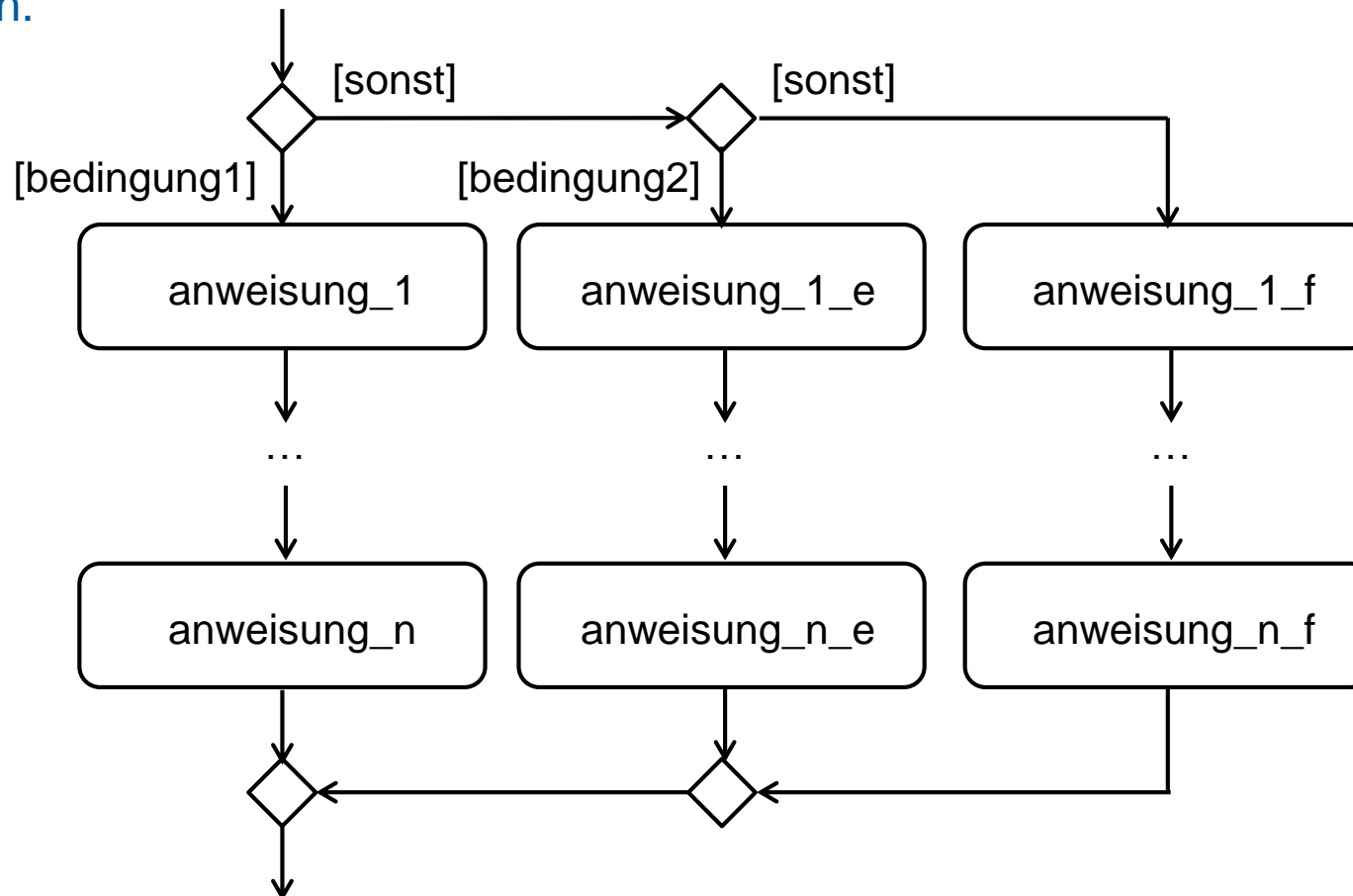
```
for i in range(Startwert, Endwert + 1, Schrittweite):
    anweisung_1
    ...
    anweisung_n
```

Die Schrittweite muss nicht angegeben werden. Sie beträgt dann 1.

■ Bedingte Anweisung:

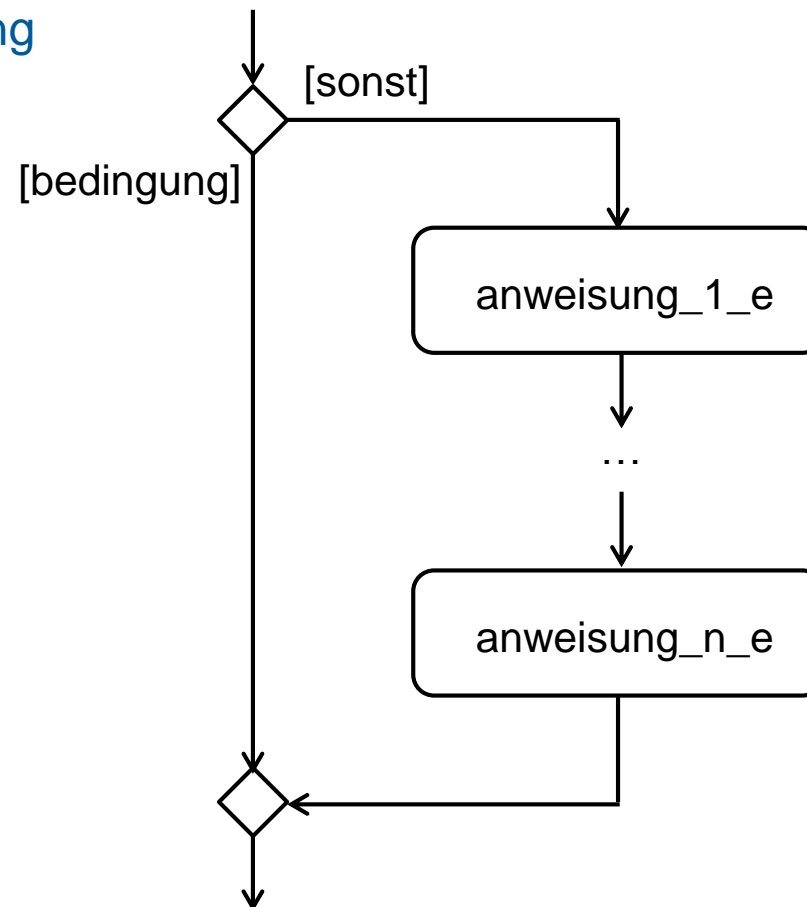


- Geschachtelte bedingte Anweisungen: Die Anzahl der elif-Zweige ist beliebig. Der else-Zweig kann auch wegfallen.



```
if bedingung1:
    anweisung_1
    ...
    anweisung_n
elif bedingung2:
    anweisung_1_e
    ...
    anweisung_n_e
else:
    anweisung_1_f
    ...
    anweisung_n_f
```

- Ein leerer erster Zweig ist zwar ungewöhnlich aber machbar. Da es in Python keine leeren Blöcke gibt, muss die `pass`-Anweisung verwendet werden:



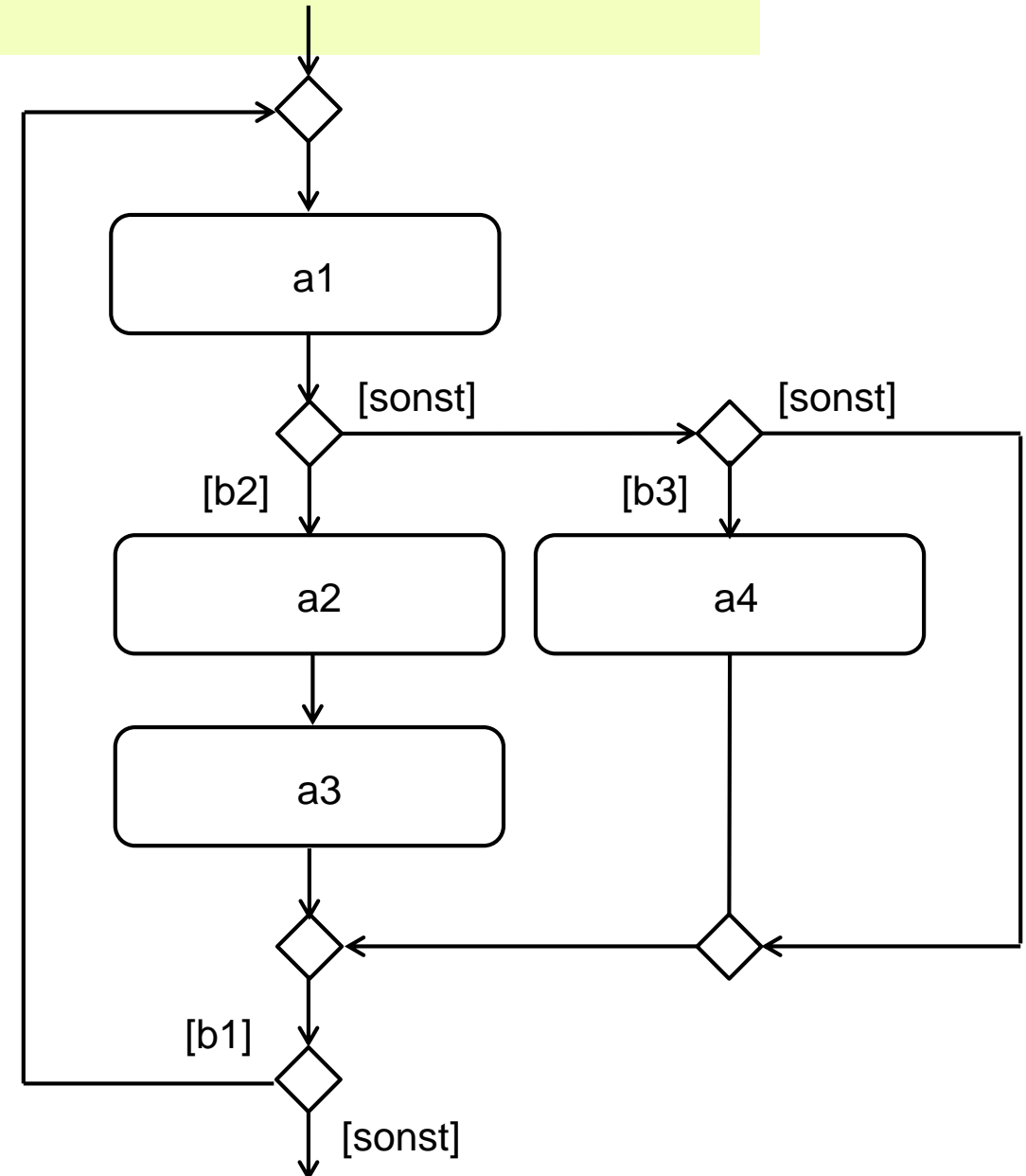
```
if bedingung:  
    pass  
else:  
    anweisung_1_e  
    ...  
    anweisung_n_e
```

Schleifen und bedingte Anweisungen in Python (8)



Frage

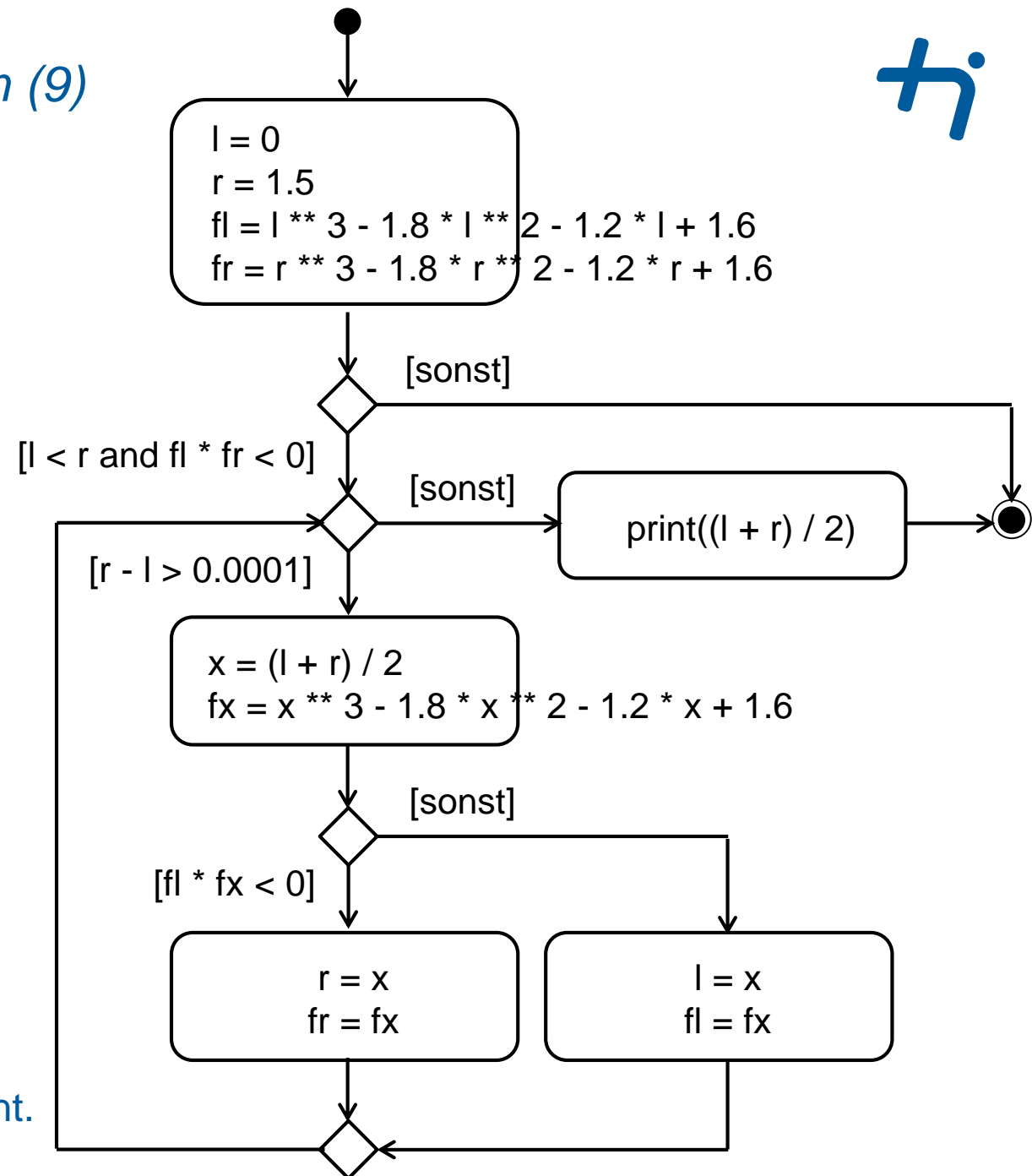
- Welchen Python-Anweisungen entspricht dieses Ablaufdiagramm?



Schleifen und bedingte Anweisungen in Python (9)



- Wir können jetzt ein Ablaufdiagramm und das entsprechende Python-Skript zur Nullstellenbestimmung mit einer Schleife und bedingten Anweisungen erstellen.
- Zuerst werden die Variablen für die Intervallgrenzen und deren Polynomwerte initialisiert.
- Die folgende bedingte Anweisung prüft, ob eine Nullstelle bestimmt werden kann.
- Danach folgt eine while-Schleife, die abbricht, wenn der Fehler maximal 0.0001 beträgt. In der Schleife werden die Intervallmitte und dessen Polynomwert berechnet. Abhängig davon wird mit der linken oder rechten Intervallhälfte weitergemacht.



Schleifen und bedingte Anweisungen in Python (10)

Aufgaben

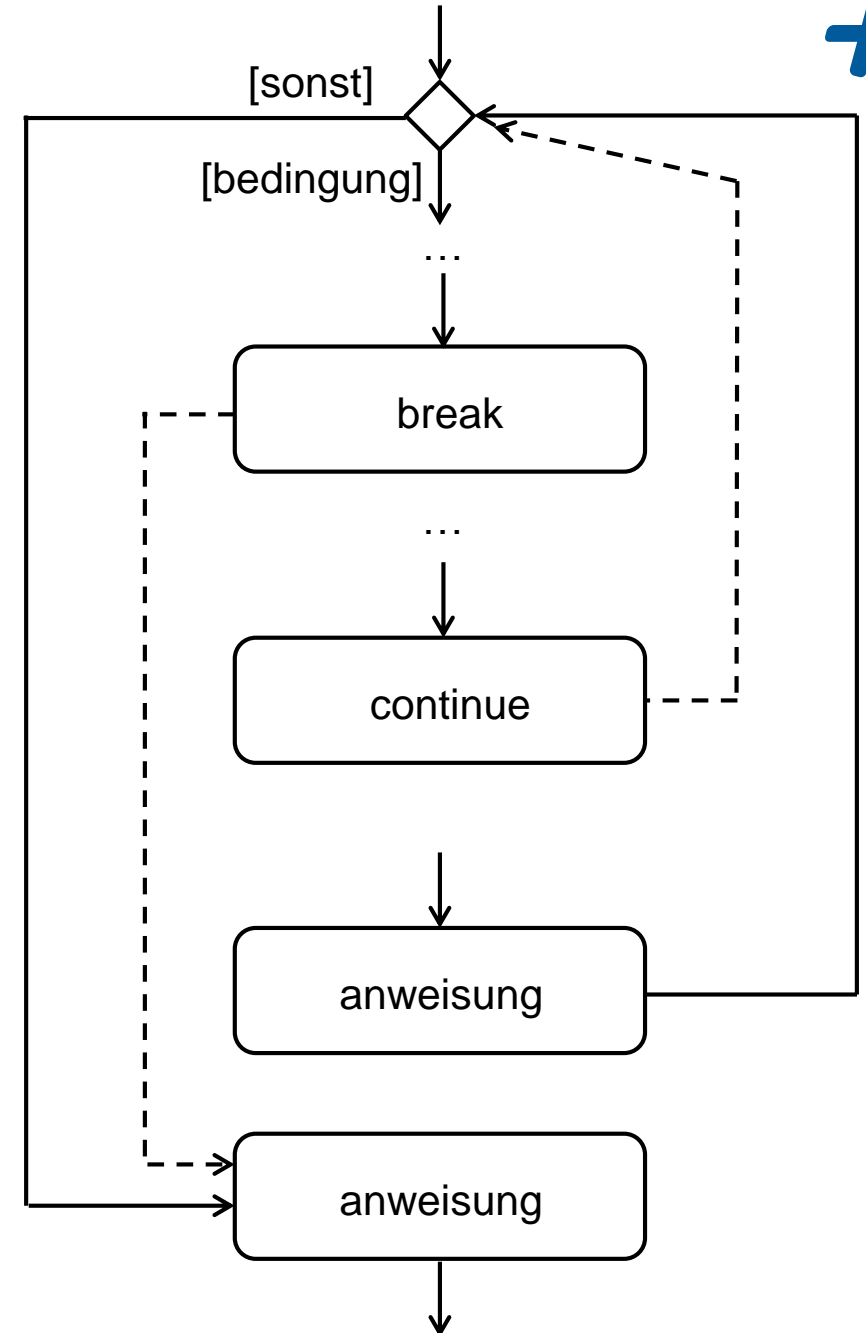
1. Erstellen Sie zu den Ablaufdiagrammen der
 - a) Aufgaben 1 – 4 und 6 im Kapitel „Schleifen“ und der
 - b) Aufgaben 1 – 3, 5 und 6 im Kapitel „Bedingte Anweisungen“gleichwertige für Python-Skripte mit while-Schleifen.
2. Erstellen Sie ein Python-Skript zur Berechnung der Summe der Zahlen von 1 bis n als fußgesteuerte while-Schleife.
3. Erstellen Sie Python-Skripte mit for-Schleifen zur Berechnung
 - a) der Summe der Zahlen von 1 bis n
 - b) der Fakultät von n (Aufgabe 1 im Kapitel „Schleifen“)
 - c) des n-ten Glied der Zahlenfolge 1, 2, 4, 7, 11, 16, 22, ... (Aufgabe 4 im Kapitel „Schleifen“)

Die Eingaben soll jeweils mit input und die Ausgaben mit print erfolgen.

Schleifen und bedingte Anweisungen in Python (11)



- Zur Programmierung der do-while-Schleife in Python haben wir die break-Anweisung verwendet. Ganz allgemein bricht die break-Anweisung die Ausführung einer Schleife ab.
- Es gibt auch eine continue-Anweisung, die den Durchlauf durch die Schleife beendet und ggf. den nächsten Durchlauf beginnt.



- Schleifen können in Python im Gegensatz zu anderen Programmiersprachen einen else-Zweig besitzen. Der else-Zweig wird ausgeführt, nachdem die Schleifenbedingung nicht mehr zutrifft und die Schleife beendet wird. Sie wird nicht ausgeführt, wenn die Schleife durch break verlassen wird. Beispiel:

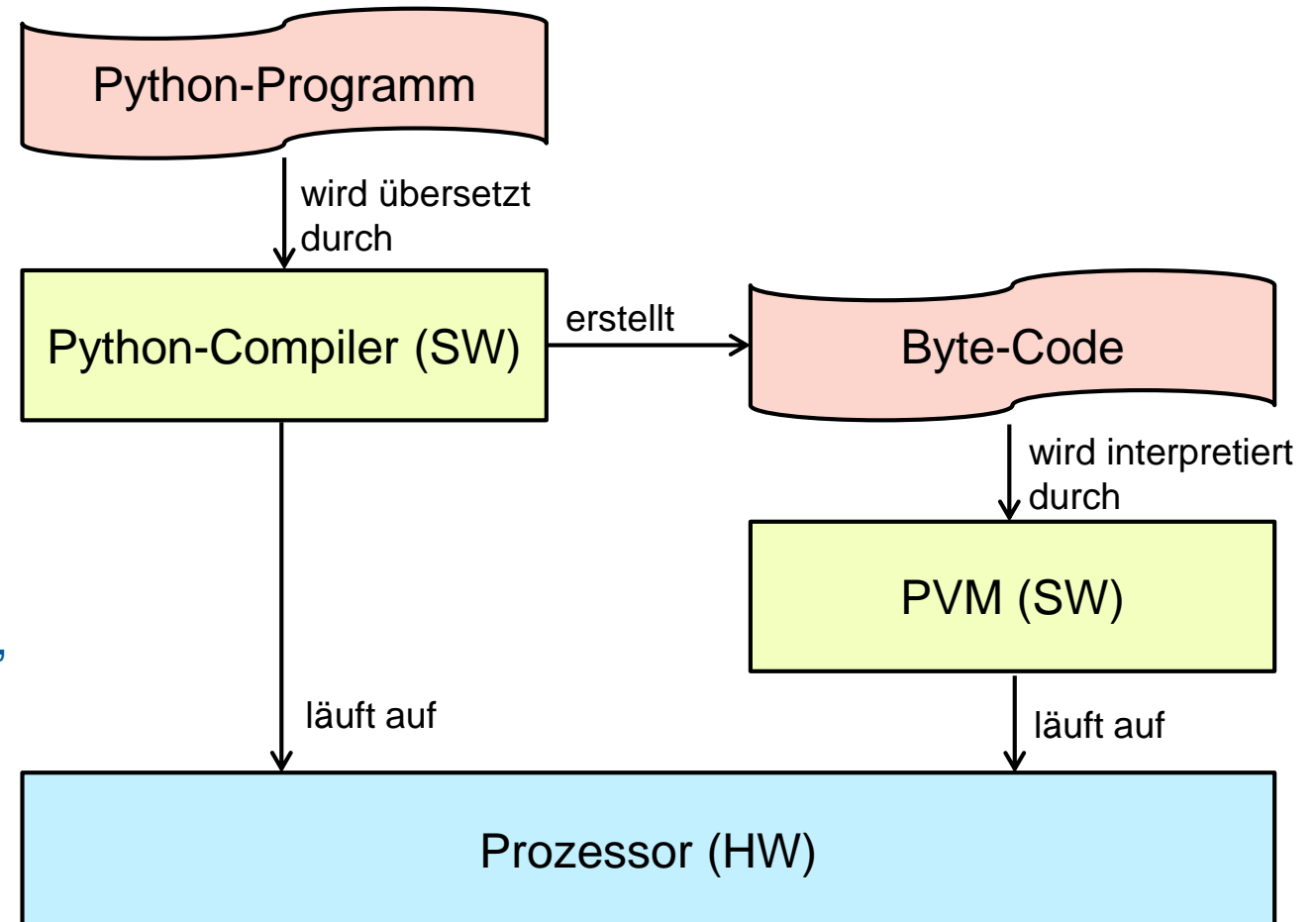
```
n = 2000
i = 1

while i < 1000:
    i = i + 1
    if not (i <= n):
        break
else:
    print("Maximale Anzahl von Schleifendurchgängen erreicht.")
```

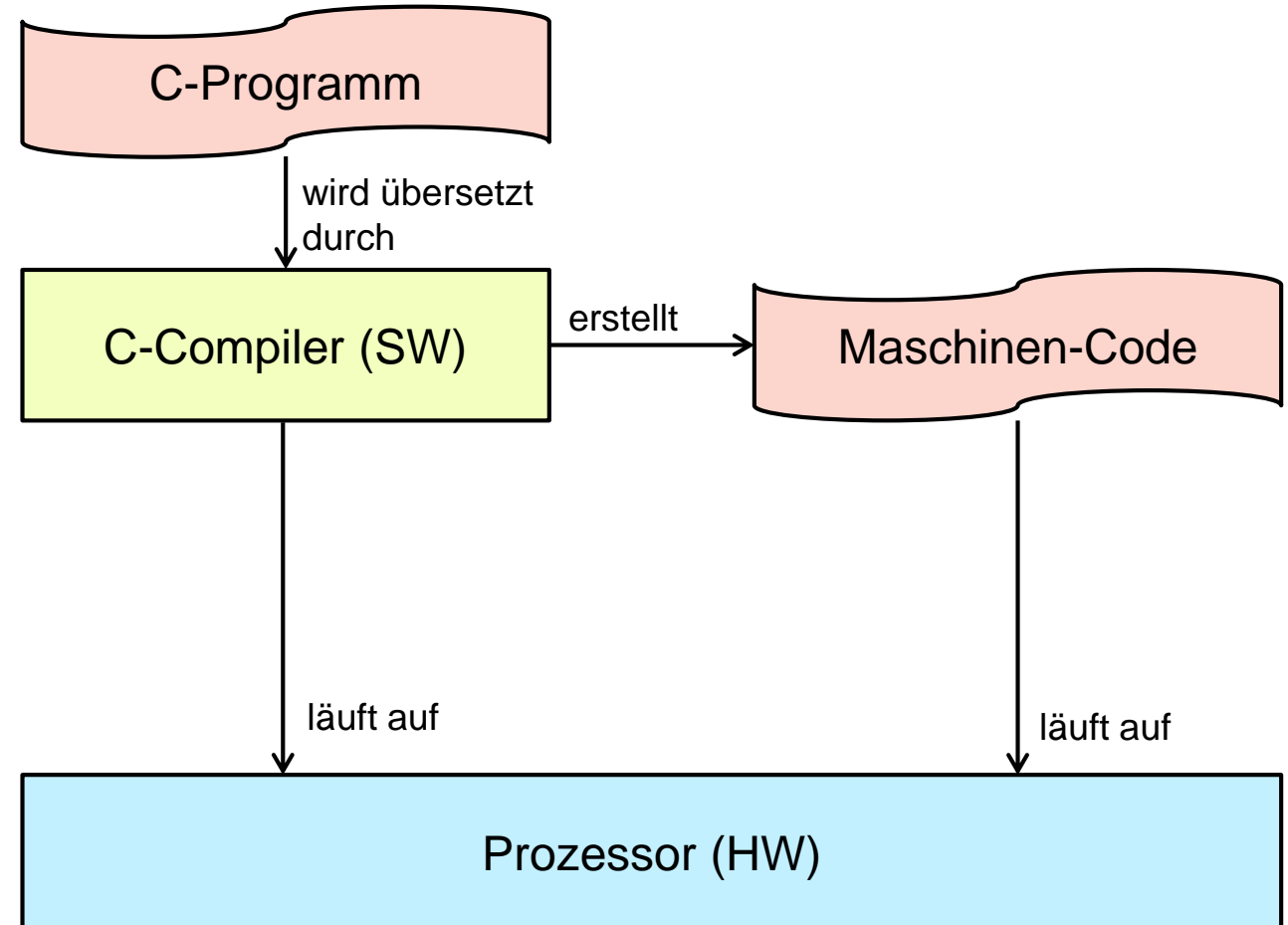
Für $n = 2000$ wird der else-Zweig ausgeführt. Falls z.B. $n = 100$, wird er nicht ausgeführt.


-  Kapitel 4.3.4, 9, 10 in (Klein 2018)

- Python-Programme werden in **Byte-Code** übersetzt, der von der **Python Virtuellen Maschine** (PVM) Schritt für Schritt interpretiert wird.
- Der Byte-Code ist unabhängig von der realen Hardware (HW). Der selbe Byte-Code kann mit einer anderen PVM auf einem anderen Prozessor laufen.
- Die PVM ist auch nur eine Software (SW), die Byte-Code als Eingabe verarbeitet.
- Diese Schritte laufen alle im Hintergrund ab. Der Programmierer muss sich nicht darum kümmern.



- Programme in Programmiersprachen wie C werden in Maschinen-Code übersetzt, der ohne virtuelle Maschine direkt ausgeführt wird.
- Der Maschinen-Code ist prozessorabhängig.



- Kurze Programme, die interpretiert werden, bezeichnet man als **Skripte**.
 - Es gibt also Python-Skripte und Python-Programme. Der Unterschied liegt nur in der Größe.
 - Es gibt aber nur C-Programme und keine C-Skripte.
-  Kapitel 3 in (Klein 2018)
- Ein kurzer Blick zurück:
 - In den Anfangszeiten der Informatik war die Programmierung ungleich aufwändiger. Programmtext wurde zeilenweise in Lochkarten gestanzt, die eingelesen wurden.

