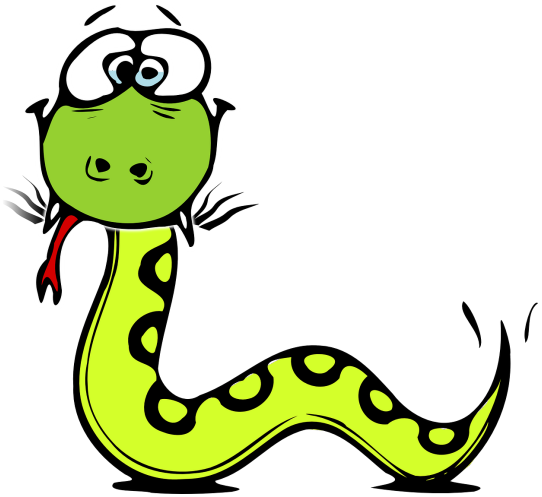
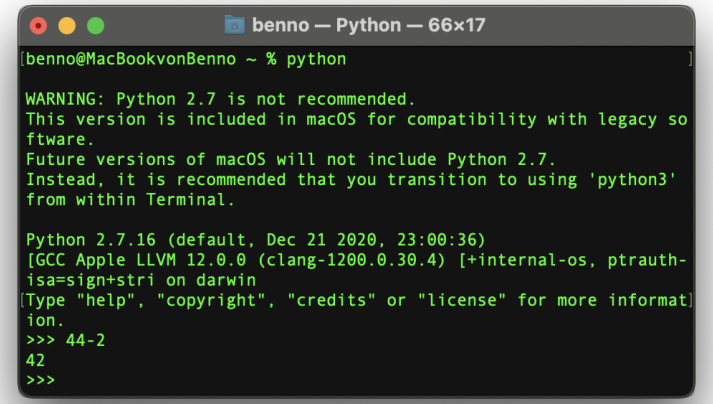
Einführung in Python



schülerforschungszentrum
südwürttemberg

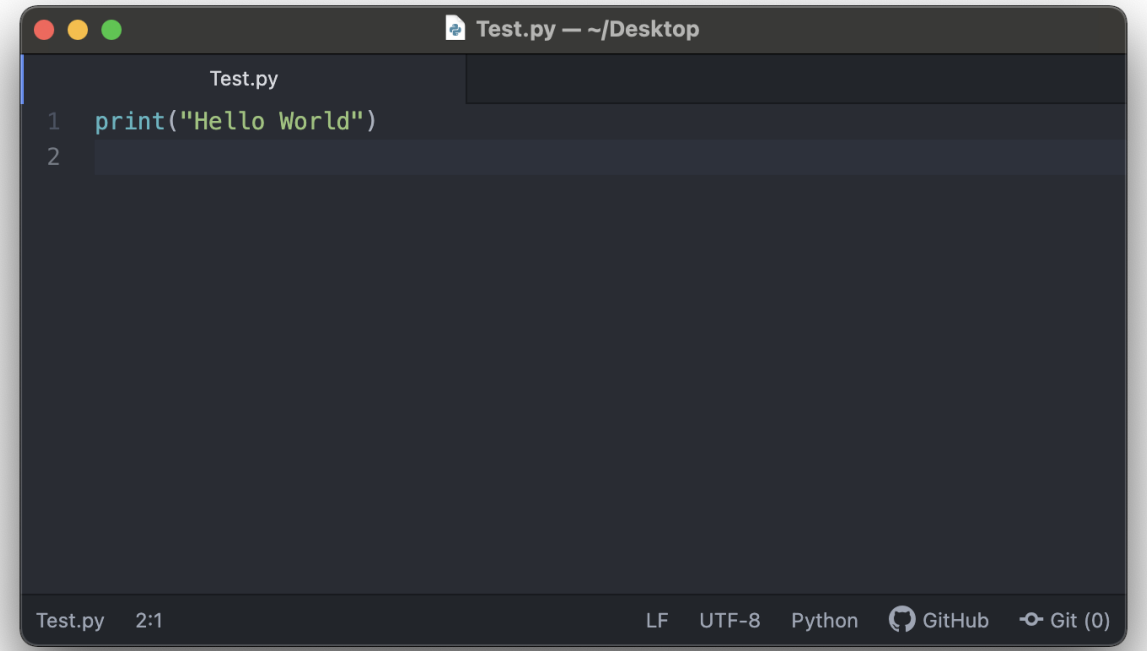
Erste Schritte

- **Möglichkeit 1:** direkt auf der Kommandozeile
  → Aufruf des Compilers über `python` bzw. `python3`



Erste Schritte

- **Möglichkeit 1:** direkt auf der Kommandozeile
  → Aufruf des Compilers über `python` bzw. `python3`
- **Möglichkeit 2:** Programm in Datei mit Endung `.py` speichern
  → Ausführen mit `python3 <Dateiname>.py`



Das erste Programm

```
1 print("Hallo Welt!")
```

- `print()`-Anweisung für Ausgabe von Text auf der Kommandozeile
- Text in Anführungszeichen → *String*

Was wird hier angezeigt? → `Hallo Welt!`

Kommentare

- Dokumentation von Code
- alles was hinter # steht wird nicht ausgeführt

```
1 print("Hallo") #Hallo ausgeben
2 #Auch das hier ist ein Kommentar!
```

Aufgabe 1

Schreibe ein Python-Programm, das beim Starten automatisch deinen Namen anzeigt.

Aufgabe 2

Füge deinem Programm aus Aufgabe 1 einen Kommentar hinzu, der erklärt, was das Programm macht. Außerdem soll er deinen Namen und das heutige Datum enthalten.

Variablen

- Speichern von Zahlen (**int**):

```
1 i = 41
2 print(i+1)
```

Ausgabe? → 42

- Kommazahlen (**float**):

```
1 d = 5.3
```

- Text (**str**):

```
1 s = "Hallo"
2
3 # oder:
4 s= 'Hallo'
```

Aufgabe 3

Versuche durch Ausprobieren üblicher Zeichen eine Addition, Subtraktion, Multiplikation und Division durchzuführen.

Aufgabe 4

Kannst du herausfinden, was das %-Zeichen macht?

Aufgabe 5

Für besonders Schnelle:
Kannst du herausfinden, was der Unterschied zwischen / und // ist?

Operationen

Zahlen:

- Grundrechenarten: `+`, `-`, `*`, `/`, `//`, `%`

```
1 i = 42
2
3 print(i+3) # 45
4 print(i-5) # 37
5 print(i*2) # 84
6 print(i/5) # 8.4
7
8 print(i//5) # 8
9 print(i%5) # 2
```

⚠ **Beachte:** Ganzzahlige Division mit //, Rest mit % („modulo“)

Operationen

Zahlen:

- Wert direkt ändern mit +=, -=, *=, /=

```
1 i = 5
2 i += 10 # macht das Gleiche wie i=i+10
3 print(i) # 15
```

Strings:

- Zusammenfügen mit +

```
1 s = "Hallo"
2 s = s + " Welt!"
3 print(s) # Hallo Welt!
```

Datentyp umwandeln

- Typ ändern mit `<neuerTyp>(<Variable>)`
- also z. B.:

```
1 x = 3.82
2 y = int(x)
```

Welchen Wert hat y? → 3

Aufgabe 6

In dieser Aufgabe wollen wir einen Würfel programmieren, also einen Zufallsgenerator, der Ganzzahlen zwischen 1 und 6 ausgibt. Nutze dafür dein Wissen von heute über Datentypen, Operationen und Typ-Umwandlungen.

Tipp: Mit dem Befehl `random.random()` kannst du eine Zufallszahl (float) zwischen 0 und 1 erzeugen:

```
1 import random
2 x = random.random()
```

⚠ Beachte, dass die Zufallszahl Nachkommastellen besitzt!

Aufgabe 6 - Lösung

```python
import random

a = random.random() # Zufallszahl zwischen 0 und 1 (float)
b = a*6 # Zufallszahl in Zahl zwischen 0 und 5 umwandeln (float)
b += 1 # Zahl zwischen 0 und 6 (float)
c = int(b) # Nachkommastellen abschneiden (int)

print(c)
```

Aufgabe 6 - Lösung (alternativ)

```python
import random

d = random.randint(1,6)
print(d)
```

Text einlesen

Mithilfe des `input()`-Befehls lassen sich Strings von der Kommandozeile einlesen:

```
1 s = input()
2 print(s) # Hier wird ausgegeben, was eingegeben wurde.
```

Ab der entsprechenden Stelle wartet das Programm so lange, bis der Nutzer etwas eingegeben und mit `Enter` bestätigt hat.

Aufgabe 7

Schreibe ein Programm, das deinen Namen einliest und dich anschließend folgendermaßen begrüßt:

```
Hallo <Name>! Wie alt bist du?
```

Daraufhin soll dein Alter abgefragt und angezeigt werden:

```
Alter: <Alter> Jahre
```

Vergleichen

- gleich/ungleich: ==, !=
- größer/kleiner: >, >=, <, <=

→ Wozu brauchen wir das?

If...Else

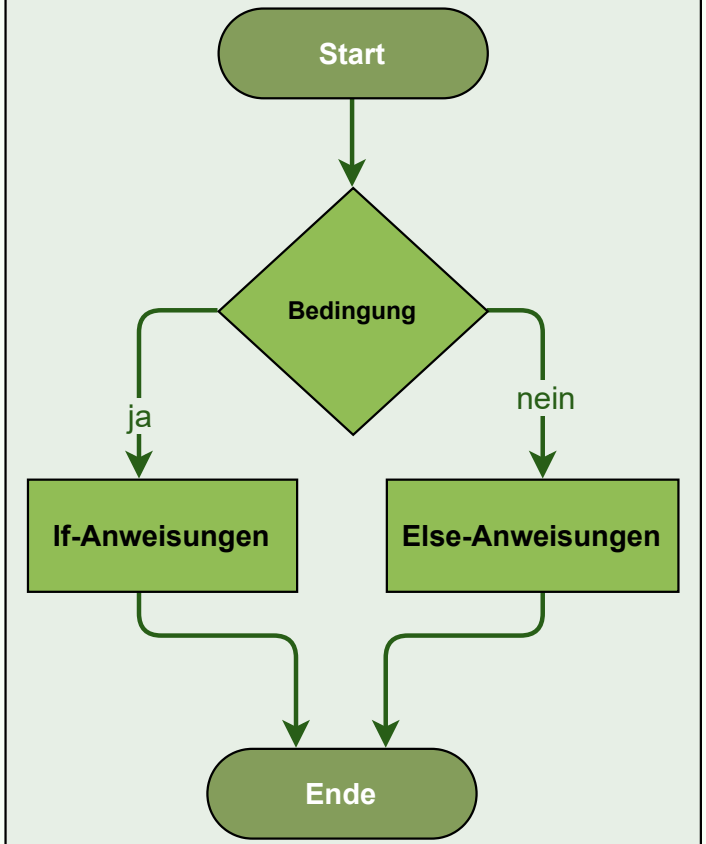
- Bisher: Ausführung einzelner Anweisungen, feste Reihenfolge
- Wir wollen aber auch entscheiden können wann etwas ausgeführt wird

```
1 a = 77
2 b = 42
3
4 if a > b:
5     print("a ist groesser als b")
6 elif a < b:
7     print("a ist kleiner als b")
8 else:
9     print("a und b sind gleich")
```

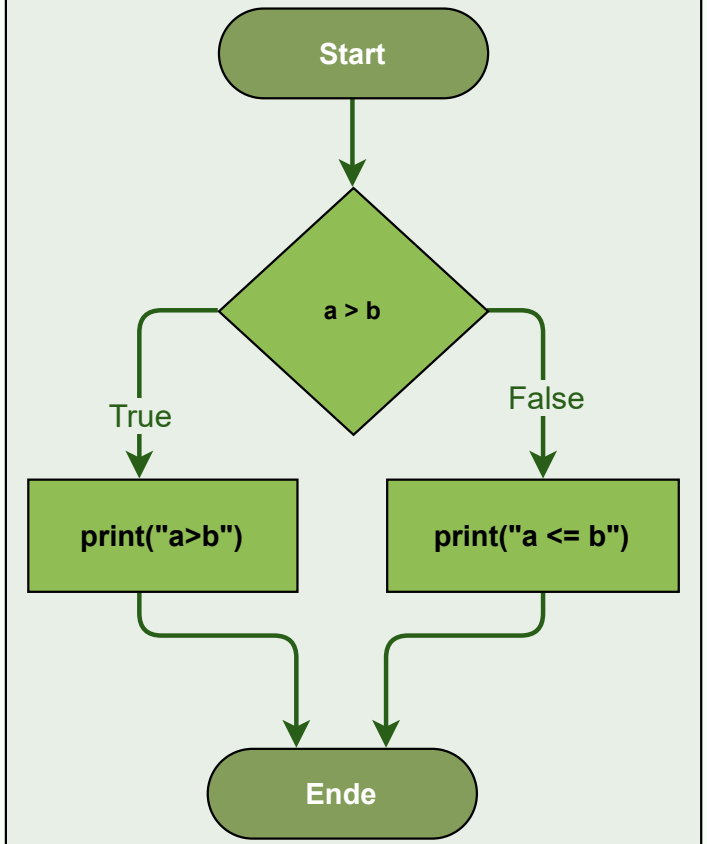
⚠ **Einrückung!** (Tab)

If-Abfrage

Allgemein



Beispiel



Bei allen Diagrammen wurde das Deklarieren einiger Variablen aus Gründen der Übersicht weggelassen.

Aufgabe 8

Schreibe ein Programm, das eine Zahl einliest und anzeigt, ob diese gerade oder ungerade ist.

Tipp: Der Rest einer Division kann mit % („modulo“) bestimmt werden:

```
1 div = 5 // 3 # div = 1
2 rest = 5 % 3 # rest = 2
```

Tipp: Mit `input()` kann nur Text eingelesen werden, hier muss der Datentyp umgewandelt werden!

While-Schleife

- Wiederholung solange die Bedingung gilt

```
1 i = 0
2 while i < 4:
3     print(i)
4     i+=1
```

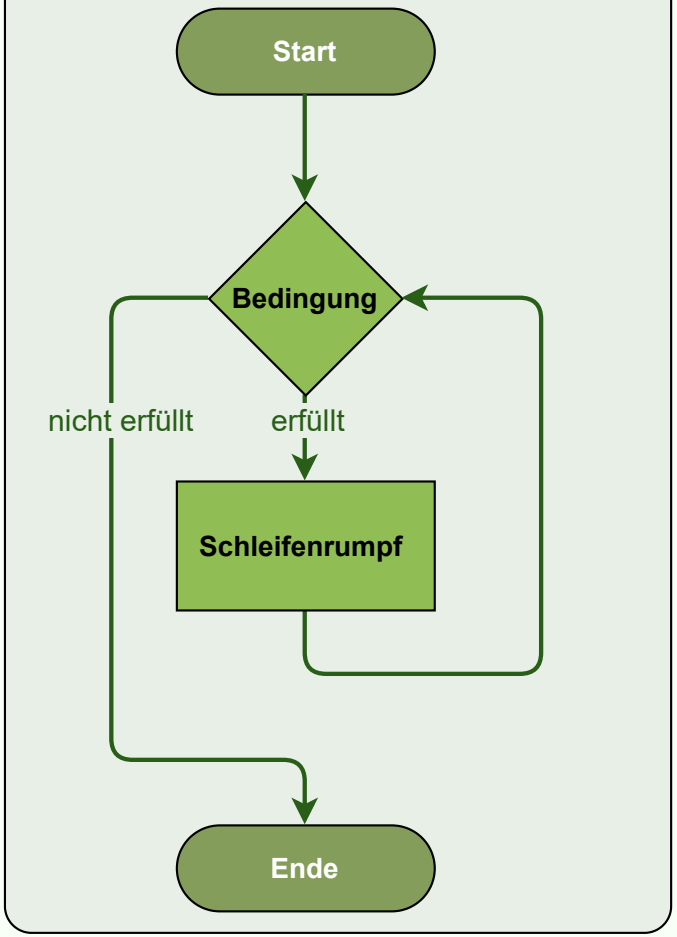
⚠ **Einrückung!** (Tab)

→ Was wird ausgegeben?

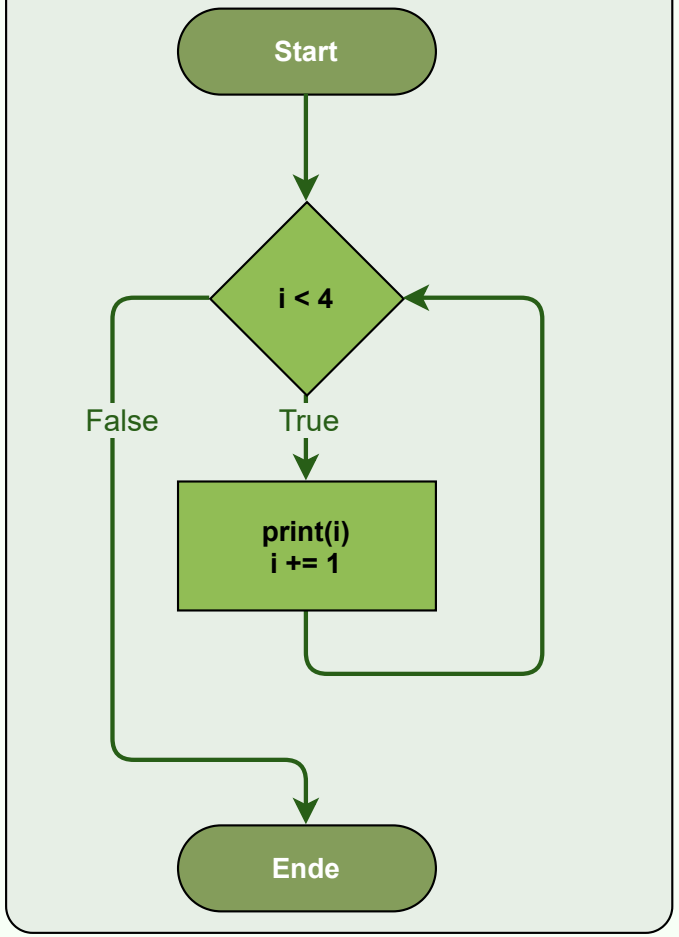
0
1
2
3

While Schleife

Allgemein



Beispiel



Bei allen Diagrammen wurde das Deklarieren einiger Variablen aus Gründen der Übersicht weggelassen.

Aufgabe 9

Schreibe ein Programm, das alle geraden Zahlen zwischen 0 und 10 ausgibt.

Challenge: Es gibt drei verschiedene Herangehensweisen, findest du alle?

Pause machen

- Anhalten des Programms mit `sleep`-Befehl (Zeit in Sekunden)
  → Muss zuerst importiert werden!

```
1 import time
2
3 print("Erste Ausgabe")
4 time.sleep(5) # Warte 5 Sekunden
5 print("Zweite Ausgabe nach 5 Sekunden")
```

Aufgabe 10

Schreibe einen Timer: Dieser soll zunächst eine Zeit in Sekunden einlesen. Anschließend sollen die Sekunden bis auf 0 heruntergezählt und angezeigt werden.

Für Schnelle: Erweitere deinen Timer so, dass er Sekunden, Minuten und Stunden getrennt einlesen kann und die verbleibende Zeit im Format `HH:MM:SS` anzeigt.

Liste / list

List:

- Datenstruktur
- Ansammlung an Werten
- Zugriff über Index (beginnend bei 0)
- Schreib und Leserechte

Erstellen einer Liste:

```
1 list = ['SFZ BW' , 42 , "Hi I bims" , 'a', 23.02]
```

Auslesen einer Liste über Index:

```
1 list[1] # 42
```

Tupel / tuple

Tupel:

- Datenstruktur
- Ansammlung an Werten
- Zugriff über Index (beginnend bei 0)
- ⚠**nur** Leserecht

Erstellen eines Tupel:

```
1 tuple = ('Python' , 'Java' , "C" , "C++" )
```

Auslesen eines Tupel über Index:

```
1 tuple[0] # 'Python'
```

For-Schleife

- Durchlaufen einer Datenstruktur
  → Liste, Tupel, ...

```
1 ls = ['SFZ BW' , 42 , "Hi I bims" , 'a', 23.02] # Liste
2
3 for x in ls:
4     print(x)
```

Ausgabe?

```
SFZ BW
42
Hi I bims
a
23.02
```

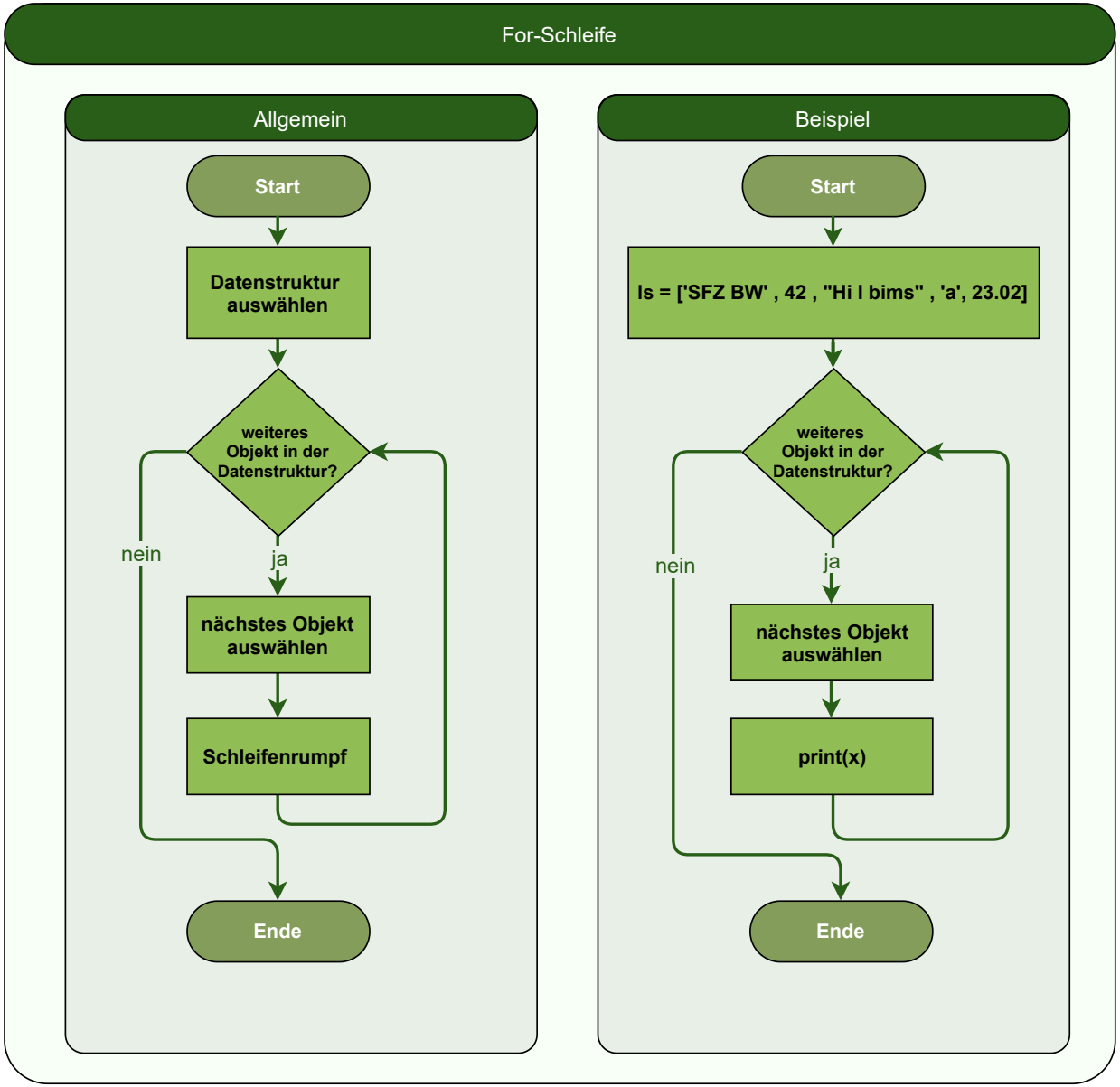
For-Schleife

- for-Schleife als Ersatz für while-Schleife
- Nutzung der `range`-Funktion

```
1 for x in range(1,5):
2     print(x)
```

Ausgabe?

1
2
3
4



Bei allen Diagrammen wurde das Deklarieren einiger Variablen aus Gründen der Übersicht weggelassen.

Aufgabe 11

Verändere deinen Timer aus Aufgabe 10 so, dass er eine For- anstelle einer While-Schleife verwendet.

Schnelle dürfen natürlich wieder den Timer erweitern, sodass er Sekunden, Minuten und Stunden getrennt einlesen kann und die verbleibende Zeit im Format `HH:MM:SS` anzeigt.

Zur Erinnerung hier nochmal Aufgabe 10

Schreibe einen Timer: Dieser soll zunächst eine Zeit in Sekunden einlesen. Anschließend sollen die Sekunden bis auf 0 runtergezählt und angezeigt werden.

Für Schnelle: Erweitere deinen Timer so, dass er Sekunden, Minuten und Stunden getrennt einlesen kann und die verbleibende Zeit im Format `HH:MM:SS` anzeigt.

Aufgabe 12

Schreibe ein Programm, das eine bestimmte Anzahl an Zahlen einliest und daraus den Durchschnitt berechnet. Dazu sollen zunächst die Anzahl der Zahlen und anschließend die einzelnen Zahlen nacheinander eingelesen werden. Der Durchschnitt soll daraufhin auf der Kommandozeile ausgegeben werden.

Beispiel:

Eingabe:		Ausgabe:
`4`	← Anzahl der Zahlen	`18.9`
`21.0`	↓ Zahlen	
`5.6`		
`7`		
`42`		

Tipp: Mit `<listenname>.append(<element>)` kannst du einer Liste ein Element hinzufügen.

Funktionen

- lange Programm werden schnell unübersichtlich
- Manche Programmteile sollen mehrmals ausgeführt werden

→ **Funktionen** (Unterprogramme)

```
1 def funktion():
2     print("Hallo")
```

Aufruf:

```
1 funktion()
```

Funktionen - Übergabe

```
1 def printName(name):
2     print("Mein Name ist " + name)
```

Aufruf:

```
1 printName("Cordula")
```

Ausgabe? → `Mein Name ist Cordula`

Funktionen - Rückgabe

```
1 def add(a, b):
2     ergebnis = a + b
3     return ergebnis
```

Aufruf:

```
1 x = 17
2 y = 25
3 summe = add(x, y)
4 print(summe)
```

Ausgabe? → 42