Kontrollfluss, Funktionen und Strings

Bedingte Anweisungen: if/elif/else

- Keine geschweiften Klammern
- Einrückung definiert Blöcke
- Doppelpunkt leitet Block ein

```
# Einfacher und lesbarer als in C
if note >= 90:
    print("Sehr gut")
elif note >= 80:
    print("Gut")
else:
    print("Befriedigend")
```

Boolesche Ausdrücke in Python

```
# In Python: Viele "falsche" Werte
x = 0        # False
x = ""       # False (leerer String)
x = []       # False (leere Liste)
x = None     # False (Pythons NULL)


# Alles andere ist True
name = "Max"  # True (nicht-leerer String)
zahl = 42     # True (nicht-null)
```

Vergleichsoperatoren

```
# Logische Operatoren in Klartext
if x < 7 and y > 8:     # statt && in C
    print("Beide wahr")

if x < 3 or y < 20:     # statt || in C
    print("Mind. eines wahr")

if not x == y:          # statt ! in C
    print("Ungleich")


# Verkettete Vergleiche
if 13 <= alter <= 19:   # Teenager
    print("Teenager")   # Eleganter als in C!
```

Besonderheiten bei Vergleichen

```
# Gleichheit vs. Identität
x = [1, 2, 3]
y = [1, 2, 3]
print(x == y)      # True  - gleicher Inhalt
print(x is y)      # False - verschiedene Objekte


# None vergleichen
x = None
if x is None:          # Richtig!
    print("Ist None")
if x == None:          # Funktioniert, aber nicht idiomatisch
    print("Ist None")
```

Kompakte Syntax

```
benutzer = "Admin"
passwort = "geheim"


if benutzer == "Admin" and passwort == "geheim":
    print("Zugriff gewährt")


# Bedingter Ausdruck (Ternärer Operator)
alter = 20
status = "erwachsen" if alter >= 18 else "minderjährig"
```

Schleifen

```
# While-Schleife (vertraut aus C)
counter = 0
while counter < 5:
    print(counter)
    counter += 1    # es gibt kein counter++


# For-Schleife (anders als in C)
for x in range(5):  # läuft von 0 bis 4
    print(x)


# Vorschau: Später wird es noch eleganter
# for zeichen in text:            # Direkte String-Iteration
# for element in liste:           # Direkte Listen-Iteration
# for schluessel in woerterbuch:  # Dictionary-Iteration
```

Abbruch von Schleifen

```
# Break und Continue wie in C
for i in range(10):
    if i == 3:
        continue    # Springt zur nächsten Iteration
    if i == 8:
        break       # Beendet die Schleife
    print(i)


# Aber: Kein do-while!
while True:
    # Code
    if not bedingung:
        break   # Pythons Weg für do-while
```

Fortgeschrittene Schleifenbenutzung

```
# Strings aufbauen (noch umständlich)
result = ""
for i in range(5):
    result += str(i)   # Neuer String bei jeder Operation!
print(result)          # "01234"

# Zählen von Vorkommen (noch mühsam)
text = "hallo"
count = 0
for i in range(len(text)):
    if text[i] == "l":
        count += 1
print(count)  # 2

# Vorschau: Diese Operationen sind eleganter möglich:
# text.count('l')            # Direkte Zählung
# ''.join(str(i) for i ...)  # Effiziente String-Verkettung
```

Weitere Beispiele

```python
# 1. Wörter in einem Text zählen (C-Stil)
def count_words(text):
    count = 0
    in_word = False
    for i in range(len(text)):
        if text[i] != ' ' and not in_word:
            count += 1
            in_word = True
        elif text[i] == ' ':
            in_word = False
    return count

# 2. Matrix durchlaufen (noch umständlich)
matrix = [[1, 2, 3], [4, 5, 6]]
for i in range(len(matrix)):
    for j in range(len(matrix[i])):
        print(matrix[i][j])  # Einzelne Elemente
```

Weitere Beispiele

```python
# 1. Wörter in einem Text zählen (C-Stil)
def count_words(text):
    count = 0
    in_word = False
    for i in range(len(text)):
        if text[i] != ' ' and not in_word:
            count += 1
            in_word = True
        elif text[i] == ' ':
            in_word = False
    return count


# Beispiel für text = "Hallo du da"
```

```
i = 0: 'H'
  - kein Leerzeichen und nicht in_word
  - count wird 1, in_word wird True

i = 1: 'a'
  - kein Leerzeichen und in_word ist True
  - nichts ändert sich
...

i = 4: 'o'
  - immer noch im ersten Wort

i = 5: ' '
  - Leerzeichen gefunden: in_word wird False

i = 6: 'd'
  - kein Leerzeichen und nicht in_word
  - count wird 2, in_word wird True

...
```

Weitere Beispiele

```python
# 1. Wörter in einem Text zählen (C-Stil)
def count_words(text):
    count = 0
    in_word = False
    for i in range(len(text)):
        if text[i] != ' ' and not in_word:
            count += 1
            in_word = True
        elif text[i] == ' ':
            in_word = False
    return count

# 2. Matrix durchlaufen (noch umständlich)
matrix = [[1, 2, 3], [4, 5, 6]]
for i in range(len(matrix)):
    for j in range(len(matrix[i])):
        print(matrix[i][j])  # Einzelne Elemente
```

Funktionen

```python
# Definition mit 'def'
def begrüsse(name):    # Kein Rückgabetyp nötig!
    print(f"Hallo {name}")

def begrüsse(name="Welt"): # Standardwerte für Parameter
    print(f"Hallo {name}")

def get_koordinaten():
    x = 3
    y = 4
    return x, y    # Mehrfache Rückgabe!

# Dokumentation mit Docstrings
def quadrat(x):
    """Berechnet das Quadrat einer Zahl.

    Args:
        x: Die zu quadrierende Zahl
    Returns:
        Das Quadrat von x
    """
    return x * x
```

Verhalten von Funktionen

```python
# Parameter sind Referenzen
def modify_number(x):
    x = x + 1        # Ändert nur lokale Kopie!
    print(f"Innen: {x}")

def modify_string(s):
    s = s + "!"      # Ändert nur lokale Kopie!
    print(f"Innen: {s}")

# Demonstration
num = 42
modify_number(num)
print(f"Außen: {num}")    # Immer noch 42!

text = "Hallo"
modify_string(text)
print(f"Außen: {text}")   # Immer noch "Hallo"!
```

Mehrere Rückgabewerte

```python
# Eine Funktion mit mehreren Rückgabewerten
def get_kreis_info(radius):
    umfang = 2 * 3.14 * radius
    flaeche = 3.14 * radius * radius
    return umfang, flaeche


# Aufruf und Verwendung
radius = 5
u, f = get_kreis_info(radius)
print(f"Umfang: {u}")     # Umfang: 31.4
print(f"Fläche: {f}")     # Fläche: 78.5


# Alternative Schreibweise
ergebnis = get_kreis_info(radius)
print(f"Umfang: {ergebnis[0]}")
print(f"Fläche: {ergebnis[1]}")
```

Strings: flexibler als in C

```
# Strings erstellen
text = "Hallo"        # Doppelte Anführungszeichen
name = 'Welt'         # Einfache gehen auch

# Praktisch bei Apostrophen und Zitaten
message = "It's a nice day"   # Apostroph im String
quote = 'Er sagte "Hallo"'    # Zitat im String

# Grundlegende Operationen
print(len(text))           # Länge: 5
print(text + " " + name)   # Verkettung
print("Hi " * 3)           # Wiederholen: "Hi Hi Hi"

# Auf einzelne Zeichen zugreifen
first = text[0]    # Erstes Zeichen
last = text[-1]    # Letztes Zeichen
```

String-Methoden

```
# Groß-/Kleinschreibung
text = "Hallo, Welt"
print(text.upper())         # HALLO, WELT
print(text.lower())         # hallo, welt

# Leerzeichen entfernen
eingabe = "   Hi dort   "
print(eingabe.strip())      # "Hi dort"

# Überprüfungen
print("123".isdigit())      # True
print("abc".isalpha())      # True
print("Hi".startswith("H")) # True
print("Hi".endswith("i"))   # True

# Text finden und ersetzen
message = "Hallo Welt"
pos = message.find("Welt")  # pos: 6
neu = message.replace("Welt", "Python") # "Hallo Python"
```

Beispiel: Namensformatierer

```
# Eingabe verarbeiten und formatieren
name = input("Name: ")      # Eingebaute input() Funktion

# String verarbeiten
if name.strip() == "":
    print("Kein Name eingegeben!")
else:
    teile = name.split()    # Bei Leerzeichen trennen

    for teil in teile:
        gross = teil.upper()
        klein = teil.lower()
        gemischt = teil.title()  # Jedes Wort großgeschrieben
        print(f"{gross} -> {klein} -> {gemischt}")
```

Kommando-zeilen-argumente

```python
import sys              # Zugriff auf Argumente

# Zugriff auf Argumente
program_name = sys.argv[0]    # Name des Programms
arguments = sys.argv[1:]      # Alle anderen Argumente

# Praktisches Beispiel
def main():
    # Prüfen ob Argumente vorhanden
    if len(sys.argv) < 2:
        print("Usage: python programm.py argument")
        sys.exit(1)

    # Erstes Argument verwenden
    name = sys.argv[1]
    print(f"Argument: {name}")

# Standardidiom für den Programmstart
if __name__ == "__main__":
    main()
```

EXTRAS IN 3 MINUTEN

FRAGEN – ANTWORTEN – RÄTSEL
UND KURZE ZUSAMMENFASSUNG

Aufgabe 1

Was macht dieser Code?

```
def find_matching_pairs(text):
    pairs = ""
    i = 0
    while i < len(text) - 1:
        if text[i] == text[i + 1]:
            pairs = pairs + text[i] + text[i + 1] + " "
        i += 1
    return pairs


text = "hello bookkeeper"
result = find_matching_pairs(text)
print(result)
```

Aufgabe 1: Lösung

Was macht dieser Code?

```
def find_matching_pairs(text):
    pairs = ""          # Leerer String für die Ergebnisse
    i = 0
    while i < len(text) - 1:        # Bis vorletztes Zeichen
        if text[i] == text[i + 1]:  # Aktuelles = nächstes?
            # Gefundenes Paar plus Leerzeichen anhängen
            pairs = pairs + text[i] + text[i + 1] + " "
        i += 1
    return pairs


text = "hello bookkeeper"  # Test mit Beispieltext
result = find_matching_pairs(text)
print(result)  # Ausgabe: "ll oo kk ee"
```

Aufgabe 2: Was macht dieser Code?

```
def analyze_text(sentence):
    words = sentence.split()
    counts = ""

    for i in range(len(words)):
        already_counted = False
        for j in range(i):
            if words[i] == words[j]:
                already_counted = True
                break

        if not already_counted:
            count = 1
            for j in range(i + 1, len(words)):
                if words[i] == words[j]:
                    count += 1

            counts = counts + words[i] + ": " + str(count) + "\n"

    return counts
```

```
text = "the quick brown fox jumps over the lazy dog"
result = analyze_text(text)
print("Analyse:")
print(result)
```

Aufgabe 2: Lösung

```python
def analyze_text(sentence):
    words = sentence.split() # in Wörter zerlegen
    counts = ""  # Ergebnisse sammeln

    for i in range(len(words)):
        # Prüfen ob Wort schon gezählt wurde
        already_counted = False
        for j in range(i):  # Alle vorherigen Wörter prüfen
            if words[i] == words[j]:
                already_counted = True
                break  # Wort wurde schon gefunden

        if not already_counted:
            # Neues Wort gefunden - wie oft kommt es vor?
            count = 1  # Mindestens einmal
            for j in range(i + 1, len(words)):  # Alle weiteren Wörter prüfen
                if words[i] == words[j]:
                    count += 1

            counts = counts + words[i] + ": " + str(count) + "\n"

    return counts
```

```python
text = "the quick brown fox jumps over the lazy dog"
result = analyze_text(text)
print("Analyse:")
print(result)
```

```
Analyse:
the: 2
quick: 1
brown: 1
fox: 1
jumps: 1
over: 1
lazy: 1
dog: 1
```

Kontrollstrukturen:
if/elif/else, while, for

Einrückung und Doppelpunkt

range() für Zählschleifen

Strings

Einfache/doppelte Anführungszeichen

Eingebaute Methoden (upper, lower, strip, ...)

Direkte Vergleiche möglich

Boolesche Ausdrücke

and, or, not statt &&, ||, !

Von C abweichende Wahrheitswerte:
0, "", [], None = False

Kommandozeilenargumente

sys.argv[0] = Programmname

sys.argv[1:] = Argumente

Funktionen mit def:

Keine Typen nötig

Standardwerte für Parameter

Mehrfache Rückgabewerte

Problemlösungsansätze aus C funktionieren weiterhin: verschachtelte Schleifen und Bedingungen

Beispiele: Namensformatierer, Buchstabenpaare, Worthäufigkeiten