

**PCEP-30-02 2.1 –
Entscheidungen
treffen und den
Ablauf mit der if-**

Anweisung verzweigen

Bedingte Anweisungen: if, if-else,
if-elif, if-elif-else

Mehrfache Bedingungen &
Verschachtelung

Lernziele (PCEP 2.1)

- Die Lernenden sollen in der Lage sein, die folgenden Aufgaben zu bewältigen:
 - Die Lernenden sollen in der Lage sein, die folgenden Aufgaben zu bewältigen:
 - Die Lernenden sollen in der Lage sein, die folgenden Aufgaben zu bewältigen:

- Verstehen der grundlegenden Konzepte von **Bedingungen und Verzweigungen** in Python
- Nutzung der `if`-Anweisung zur Steuerung des Programmablaufs
- Erweiterung der Kontrolle mit `if-else`, `if-elif` und `if-elif-else`
- Anwendung mehrerer Bedingungen mit logischen Operatoren
- Erkennen und Anwenden von **verschachtelten**

Übersicht der Themen

1. Einführung in die
Themen

2. Grundlagen der
Themen

3. Anwendungsbeispiele
der Themen

4. Zusammenfassung
der Themen

5. Vertiefende
Themen

6. Aktuelle
Themen

7. Zukünftige
Themen

8. Literatur
Themen

1. Einführung in Bedingte Anweisungen
2. Die `if`-Anweisung
3. `if-else`: Eine Alternative festlegen
4. `if-elif-else`: Mehrere Alternativen behandeln
5. Mehrere Bedingungen mit `and` und `or`
6. Verschachtelte `if`-Anweisungen
7. Best Practices und häufige Fehler

1. Einführung in Bedingte Anweisungen

Bedingte Anweisungen ermöglichen es, Code **nur unter bestimmten Bedingungen** auszuführen.

- Python verwendet das Schlüsselwort **if**, um eine Bedingung zu überprüfen.
- Falls die Bedingung **wahr (True)** ist, wird der zugehörige Codeblock ausgeführt.
- Falls sie **falsch (False)** ist, wird der Block übersprungen.

2. Die if-Anweisung

Die einfachste Form einer Bedingung ist die `if`-Anweisung:

```
age = 18
if age >= 18:
    print("Du bist volljährig.")
```

Erklärung:

- Falls `age` größer oder gleich 18 ist, gibt das Programm „*Du bist volljährig.*“ aus.
- Falls `age` kleiner als 18 ist, passiert nichts.

Blitzfrage

Was passiert, wenn die Bedingung in einer `if`-Anweisung **False** ist?

- A) Der Code läuft trotzdem weiter.
- B) Der Code innerhalb des `if`-Blocks wird übersprungen.
- C) Das Programm gibt einen Fehler aus.
- D) Python verlangt eine `else`-Anweisung.

Blitzantwort

Richtige Antwort: B) Der Code innerhalb des `if`-Blocks wird übersprungen.

- Falls die Bedingung **False** ist, wird der gesamte `if`-Block ignoriert.
- Es gibt keinen Fehler, solange die Syntax korrekt ist.

3. if-else: Eine Alternative festlegen

Manchmal möchten wir eine **Alternative** ausführen, falls die Bedingung nicht erfüllt ist:

```
age = 16
if age >= 18:
    print("Du bist volljährig.")
else:
    print("Du bist minderjährig.")
```

Erklärung:

- Falls `age >= 18` ist, gibt das Programm „*Du bist volljährig.*“ aus.
- Andernfalls wird „*Du bist minderjährig.*“ ausgegeben.

4. if-elif-else: Mehrere Alternativen behandeln

Manchmal gibt es **mehr als zwei mögliche Fälle**.
Dann nutzen wir **if-elif-else**:

```
age = 12
if age >= 18:
    print("Du bist erwachsen.")
elif age >= 13:
    print("Du bist ein Teenager.")
else:
    print("Du bist ein Kind.")
```

Erklärung:

- Wenn $\text{age} \geq 18$, wird „*Du bist erwachsen.*“ ausgegeben.
- Falls $\text{age} < 18$ aber ≥ 13 , wird „*Du bist ein Teenager.*“ ausgegeben.
- Falls beides nicht zutrifft, wird „*Du bist ein Kind.*“ ausgegeben.

5. Mehrere Bedingungen mit and und or

Oft müssen **mehrere Bedingungen gleichzeitig** erfüllt sein:

```
grade = 85
attendance = 90

if grade >= 80 and attendance >= 85:
    print("Du hast bestanden!")
```

- **and**: Beide Bedingungen müssen **True** sein.
- **or**: Mindestens eine der Bedingungen muss **True** sein.

Beispiel mit **or**:

```
weather = "rainy"  
temperature = 25  
  
if weather == "sunny" or temperature > 20:  
    print("Gutes Wetter!")
```

6. Verschachtelte if-Anweisungen

Wir können `if`-Anweisungen **ineinander verschachteln**:

```
age = 20
gender = "male"

if age >= 18:
    if gender == "male":
        print("Du bist ein erwachsener Mann.")
    else:
        print("Du bist eine erwachsene Frau.")
else:
    print("Du bist minderjährig.")
```

7. Häufige Fehler und Best Practices

Häufige Fehler:

1. Vergessen der Einrückung

```
if x > 10:  
print("X ist größer als 10")  # Fehler: Einrückung fehlt
```

Python benötigt eine korrekte Einrückung, sonst gibt es einen `IndentationError`.

2. Vergleich mit = statt ==

```
if x = 10:  # Fehler!  
    print("X ist 10")
```

= ist eine Zuweisung, kein

Vergleichsoperator Verwende == für

3. Falsche Nutzung von and und or

```
if x > 10 or x < 5 and y == 20: # Fehlerhafte Logik
```

Achte auf Klammern: if (x > 10 or x < 5) and y == 20: ist eindeutiger.

4. Zu viele elif-Zweige, die unnötig kompliziert sind

```
if x == 1:  
    print("Eins")  
elif x == 2:  
    print("Zwei")  
elif x == 3:
```

```
Bes: print("Drei")
      else:
        print("Andere Zahl")
```

- ✓ Nutze `elif` anstelle von mehreren `if`, um Code effizienter zu gestalten.
Falls nur eine bestimmte Anzahl von Werten geprüft wird, könnte eine Liste oder ein Dictionary übersichtlicher sein.
- ✓ Setze logische Operatoren bewusst ein (`and`, `or`), um die Lesbarkeit zu verbessern.
- ✓ Verwende Klammern, wenn du mehrere Bedingungen kombinierst, um Missverständnisse zu vermeiden.
- ✓ Halte `if-elif-else`-Blöcke so einfach und übersichtlich wie möglich.

8. Übungsaufgaben

Aufgabe 1: Einfache if-Anweisung

Schreibe ein Programm, das prüft, ob eine eingegebene Zahl gerade oder ungerade ist.

```
num = int(input("Gib eine Zahl ein: "))

if num % 2 == 0:
    print("Die Zahl ist gerade.")
else:
    print("Die Zahl ist ungerade.")
```

Aufgabe 2: Mehrere Bedingungen mit elif

Erstelle ein Programm, das basierend auf einer Eingabe eine Bewertung in Schulnoten (1-6) ausgibt.

```
score = int(input("Gib deine Punktzahl ein: "))

if score >= 90:
    print("Note: 1")
elif score >= 80:
    print("Note: 2")
elif score >= 70:
    print("Note: 3")
elif score >= 60:
    print("Note: 4")
elif score >= 50:
    print("Note: 5")
else:
    print("Note: 6")
```

Aufgabe 3: Verschachtelte if-Anweisungen

Schreibe ein Programm, das überprüft, ob eine Person erwachsen ist (≥ 18 Jahre) und ob sie Autofahren darf (z. B. Führerschein vorhanden).

```
age = int(input("Gib dein Alter ein: "))
license = input("Hast du einen Führerschein? (ja/nein) ")

if age >= 18:
    if license.lower() == "ja":
        print("Du darfst Autofahren.")
    else:
        print("Du bist alt genug, aber hast keinen Führerschein.")
else:
    print("Du bist noch zu jung zum Autofahren.")
```

Diese Übungsaufgaben helfen dabei, verschiedene Szenarien für `if`, `if-else`, `if-elif-else` und verschachtelte Bedingungen zu trainieren.

Challenge Slides

Challenge 1: Bedingte Anweisungen

Schreibe ein Programm, das prüft, ob eine Zahl **positiv, negativ oder null** ist.

Multiple-Choice Fragen

Frage 1

Welche Aussage trifft auf eine `if`-Anweisung zu?

- A) Sie wird immer ausgeführt.
- B) Sie wird nur ausgeführt, wenn die Bedingung `True` ist.
- C) Sie muss mit `else` enden.
- D) Sie darf keine logischen Operatoren verwenden.

Frage 1 – Antwort

Richtige Antwort: B) Sie wird nur ausgeführt, wenn die Bedingung True ist.

A) Falsch – Eine if-Anweisung wird nur dann ausgeführt, wenn die Bedingung True ist.

B) Richtig – Die Bedingung muss True sein, damit der Codeblock ausgeführt wird.

C) Falsch – Eine if-Anweisung benötigt kein else.

D) *Falsch – if kann mit logischen Operatoren wie and und or verwendet werden.*

Frage 2

Welche der folgenden Bedingungen ist korrekt in Python geschrieben?

A) `if x = 5:`

B) `if x == 5:`

C) `if (x => 5):`

D) `if x === 5:`

Frage 2 – Antwort

Richtige Antwort: B) `if x == 5:`

- A) *Falsch – `=` ist eine Zuweisung, kein Vergleich.*
- B) *Richtig – `==` ist der korrekte Vergleichsoperator in Python.*
- C) *Falsch – `=>` ist kein gültiger Operator in Python, `>=` wäre korrekt.*
- D) *Falsch – `===` existiert nicht in Python, sondern in JavaScript.*

Frage 3

Welche der folgenden Aussagen trifft auf `if-else` zu?

- A) Die `else`-Anweisung wird nur ausgeführt, wenn die `if`-Bedingung `True` ist.
- B) `else` ist in Python optional.
- C) `else` benötigt immer eine eigene Bedingung.
- D) Eine `if`-Anweisung darf nicht ohne `else` existieren.

Frage 3 – Antwort

Richtige Antwort: B) `else` ist in Python optional.

- A) *Falsch – `else` wird ausgeführt, wenn die `if`-Bedingung `False` ist.*
- B) *Richtig – Eine `if`-Anweisung kann auch ohne `else` existieren.*
- C) *Falsch – `else` benötigt keine eigene Bedingung.*
- D) *Falsch – `if` kann alleine stehen.*

Frage 4

Was passiert, wenn mehrere `elif`-Bedingungen zutreffen?

- A) Alle zutreffenden `elif`-Blöcke werden ausgeführt.
- B) Nur der erste zutreffende `elif`-Block wird ausgeführt.
- C) `elif` wird ignoriert, wenn `if` bereits `True` war.
- D) `elif`-Bedingungen dürfen keine Vergleiche enthalten.

Frage 4 – Antwort

Richtige Antwort: B) Nur der erste zutreffende `elif`-Block wird ausgeführt.

A) Falsch – Python prüft von oben nach unten und führt nur den ersten zutreffenden Block aus.

B) Richtig – Sobald eine `elif`-Bedingung `True` ist, werden alle folgenden ignoriert.

C) Falsch – `elif` wird nur ignoriert, wenn `if` bereits `True` war.

D) Falsch – `elif` kann Vergleiche enthalten.

Frage 5

Welche logische Bedingung ergibt True?

- A) $5 > 10$ and $10 > 5$
- B) $5 < 10$ or $10 < 5$
- C) not ($5 < 10$)
- D) $5 == 10$

Frage 5 – Antwort

Richtige Antwort: B) $5 < 10$ or $10 < 5$

A) *Falsch – $5 > 10$ ist False, also ergibt and insgesamt False.*

B) *Richtig – $5 < 10$ ist True, also gibt or insgesamt True zurück.*

C) *Falsch – not ($5 < 10$) ist False, weil $5 < 10$ True ist.*

D) *Falsch – $5 == 10$ ist False.*

Frage 6

Welche der folgenden Aussagen ist korrekt für verschachtelte `if`-Anweisungen?

- A) Verschachtelte `if`-Anweisungen sind in Python nicht erlaubt.
- B) Ein `if` kann innerhalb eines anderen `if`-Blocks stehen.
- C) `if`-Blöcke dürfen keine weiteren Bedingungen enthalten.

D) `else` kann nicht in einem verschachtelten `if`-Block stehen.

Frage 6 – Antwort

Richtige Antwort: B) Ein `if` kann innerhalb eines anderen `if`-Blocks stehen.

A) *Falsch – Verschachtelte `if`-Anweisungen sind erlaubt und häufig genutzt.*

B) *Richtig – Ein `if` kann innerhalb eines anderen `if`-Blocks stehen.*

C) *Falsch – `if`-Blöcke dürfen Bedingungen enthalten.*

D) *Falsch – else kann sehr wohl in einem verschachtelten if-Block stehen.*

Frage 7

Welche Operatoren werden in `if`-Bedingungen häufig verwendet?

A) `+`, `-`, `*`, `/`

B) `==`, `!=`, `>`, `<`, `>=`, `<=`

C) `&&`, `||`, `!`

D) `:=`, `??`, `::`

Frage 7 – Antwort

Richtige Antwort: B) ==, !=, >, <, >=, <=

A) *Falsch – +, -, *, / sind mathematische Operatoren, keine Vergleichsoperatoren.*

B) *Richtig – Vergleichsoperatoren werden in if-Bedingungen verwendet.*

C) *Falsch – &&, ||, ! sind aus anderen Programmiersprachen, nicht aus Python.*

D) *Falsch – :=, ??, :: sind in Python nicht gültig.*

Frage 8

Welche der folgenden Bedingungen prüft auf Gleichheit?

A) `if x = 5:`

B) `if x == 5:`

C) `if x => 5:`

D) `if x equals 5:`

Frage 8 – Antwort

Richtige Antwort: B) `if x == 5:`

A) *Falsch – `=` ist eine Zuweisung, nicht ein Vergleich.*

B) *Richtig – `==` ist der korrekte Vergleichsoperator in Python.*

C) *Falsch – `=>` ist kein gültiger Operator, `>=` wäre korrekt.*

D) *Falsch – `equals` gibt es in Python nicht als Vergleichsoperator.*

Frage 9

Welche der folgenden Bedingungen nutzt `or` korrekt?

A) `if x > 10 and x < 5:`

B) `if x > 10 or x < 5:`

C) `if x > 10 not x < 5:`

D) `if x > 10 & x < 5:`

Frage 9 – Antwort

Richtige Antwort: B) `if x > 10 or x < 5`:

A) *Falsch – `and` würde hier immer `False` ergeben, da x nicht gleichzeitig größer als 10 und kleiner als 5 sein kann.*

B) *Richtig – `or` sorgt dafür, dass die Bedingung erfüllt ist, wenn eine der beiden Aussagen `True` ist.*

C) *Falsch – `not` ist hier falsch platziert.*

D) *Falsch – `&` ist ein bitweiser Operator und nicht für logische Vergleiche gedacht.*

Frage 10

Welche Aussage über verschachtelte `if`-Anweisungen ist korrekt?

- A) Sie sind in Python nicht erlaubt.
- B) Sie ermöglichen komplexere Entscheidungsstrukturen.
- C) Sie müssen immer mit `else` enden.
- D) Sie dürfen keine `elif`-Bedingungen enthalten.

Frage 10 – Antwort

Richtige Antwort: B) Sie ermöglichen komplexere Entscheidungsstrukturen.

A) *Falsch – Verschachtelte if-Anweisungen sind erlaubt und weit verbreitet.*

B) *Richtig – Sie ermöglichen es, mehrere Bedingungen hierarchisch zu prüfen.*

C) *Falsch – else ist optional.*

D) *Falsch – elif kann in verschachtelten Strukturen verwendet werden.*

