DS 4 – Übung: Verzweigungen lesen, testen und erweitern

Informatik 10 – MINT-Zweig | Textuelle Programmierung und Datenstrukturen

Lernziele

Ich kann

- Fehler in Programmen mit Verzweigungen finden, mit Hilfe der Fehlermeldung erklären und beheben,
- ein Programm mit Verzweigungen lesen und seine Ausgabe vorhersagen, ohne es auszuführen,
- zu einem Programm sinnvolle Testfälle auswählen, auch für die Grenzfälle,
- eine Mehrfachauswahl als `CASE OF` beschreiben und in Python mit `match`/`case` umsetzen,
- ein Programm mit einer Mehrfachauswahl und einer verschachtelten Verzweigung schreiben.

🗣️ Sprachhilfe

- *„Da ist ein Fehler in Zeile 2."* → „In Zeile 2 fehlt der Doppelpunkt. Python meldet `SyntaxError: expected ':'`."
- *„Ich hab's mit 5 probiert."* → „Im Testfall mit der Eingabe 5 erwarte ich die Ausgabe ‚positiv'. Das Programm liefert ‚positiv', der Test ist bestanden."
- *„Match guckt, was es ist."* → „Die Mehrfachauswahl vergleicht den Wert von `op` nacheinander mit den festen Werten der Fälle."

Worum es geht

Programmiererinnen und Programmierer **lesen** fremden Code, **suchen Fehler** und **erweitern** Programme. Die Fehlersuche heißt *Debugging*, „Entwanzen", denn einen Programmfehler nennt man *Bug* („Käfer"). Heute übst du das mit Verzweigungen, lernst gut ausgewählte **Testfälle** und eine neue Verzweigung: die **Mehrfachauswahl**.

1 Das brauchst du

```
if Bedingung_1:
    Anweisungen, wenn Bedingung_1 wahr ist
elif Bedingung_2:
    Anweisungen, wenn Bedingung_1 falsch und Bedingung_2 wahr ist
else:
    Anweisungen, wenn keine Bedingung wahr ist
```

> 📌 **Merksatz:** Nur der Block der **ersten zutreffenden** Bedingung wird ausgeführt. Achte auf **Doppelpunkt** und **Einrückung**. `==` vergleicht, `=` weist zu.

Fehler im Code	Meldung von Python	Abhilfe
Doppelpunkt vergessen: `if zahl > 0`	`SyntaxError: expected ':'`	`:` ans Zeilenende
Block nicht eingerückt	`IndentationError: expected an indented block …`	Block 4 Leerzeichen einrücken
Zeile ohne Grund eingerückt	`IndentationError: unexpected indent`	Einrückung entfernen
`=` statt `==` in der Bedingung	`SyntaxError: invalid syntax. Maybe you meant '==' …`	`==` schreiben
Bedingung hinter `else`: `else zahl < 0:`	`SyntaxError: expected ':'`	`elif` statt `else`
`elif`-Bedingungen in falscher Reihenfolge	keine, aber falsches Ergebnis	Reihenfolge prüfen

2 Programme testen

Ein Programm ohne Fehlermeldung ist noch lange nicht richtig. Man **testet** es: mit verschiedenen Eingaben ausführen und die Ausgabe mit dem vergleichen, was herauskommen **soll**. Eine Eingabe mit erwarteter Ausgabe heißt **Testfall**. Die erwartete Ausgabe überlegst du **vorher**.

1. **Jeder Zweig mindestens einmal.**
2. **Die Grenzfälle**, also die Werte direkt an einer Grenze. Bei `alter < 18` sind das 17 und 18.

Beispiel: Ein Programm soll ausgeben, ob eine Zahl positiv, negativ oder null ist.

Eingabe	erwartete Ausgabe	Warum dieser Test?
`5`	`Die Zahl ist positiv`	Zweig „positiv“
`-3`	`Die Zahl ist negativ`	Zweig „negativ“
`0`	`Die Zahl ist null`	Zweig „null“, zugleich die Grenze
`1` und `-1`	positiv bzw. negativ	direkt neben der Grenze

> 📌 **Merksatz:** Ein **Testfall** besteht aus einer Eingabe und der vorher überlegten erwarteten Ausgabe. Teste jeden Zweig und jeden Grenzfall.

📝 Übungen: Schritt für Schritt selbstständig

Das **Musterbeispiel** zeigt dir das Vorgehen beim Debugging. Danach werden die Übungen schrittweise schwieriger, die Hilfen werden weniger.

🧩 Musterbeispiel (vorgemacht) – Bestanden?

Das Programm soll ab 50 Punkten „bestanden“ ausgeben, sonst „nicht bestanden“. Es enthält **zwei Fehler**. Führe die Zelle aus.

```python
punkte = int(input("Punkte: "))
if punkte >= 50
    print("bestanden")
else punkte < 50:
    print("nicht bestanden")
```

```
  File "<Zelle 1>", line 2
    if punkte >= 50
                  ^
SyntaxError: expected ':'
```

So gehst du vor:

1. **Letzte Zeile der Meldung lesen:** `SyntaxError: expected ':'` in Zeile 2. Am Ende von `if punkte >= 50` fehlt der Doppelpunkt. Ergänzen, neu starten.
2. **Nächste Meldung:** Wieder `expected ':'`, jetzt in Zeile 4. Hinter `else` darf keine Bedingung stehen. `else` heißt ohnehin „in allen anderen Fällen“, also einfach `else:`.
3. **Testen** mit vorher überlegten Testfällen:

Eingabe	erwartete Ausgabe	Warum?
`80`	`bestanden`	Zweig „bestanden“
`20`	`nicht bestanden`	Zweig „nicht bestanden“
`50`	`bestanden`	Grenzfall, wegen `>=`
`49`	`nicht bestanden`	direkt neben der Grenze

Die korrigierte Fassung:

```python
punkte = int(input("Punkte: "))
if punkte >= 50:
    print("bestanden")
else:
    print("nicht bestanden")
```

```
Punkte: 65
bestanden
```

🐞 Übung 1 – Fehler finden (Debugging)

Das Programm soll ausgeben, ob eine Zahl positiv, negativ oder null ist. Es enthält **drei Fehler**.

1. Finde die Fehler **bevor** du es ausführst. Notiere jeweils Zeile und Fehler.
2. Führe es aus. Python meldet immer nur den **ersten** Fehler. Behebe ihn und starte neu, wie im Musterbeispiel.
3. Teste dein korrigiertes Programm mit den Testfällen aus Teil 2.

> 💡 **Tipp:** Zwei der drei Fehler stehen in derselben Zeile.

```python
zahl = int(input("Zahl eingeben: "))
if zahl = 0
    print("Die Zahl ist null")
elif zahl > 0:
print("Die Zahl ist positiv")
else:
    print("Die Zahl ist negativ")
```

```
  File "<Zelle 3>", line 2
    if zahl = 0
       ^^^^^^^^
SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?
```

Die Fehler:

…

🔮 Übung 2 – FizzBuzz lesen

Lies das Programm, **ohne** es auszuführen.

1. Was gibt es für `zahl = 17` aus? Begründe mit den Bedingungen.
2. Sage die Ausgabe für `zahl = 9`, `10` und `30` voraus.
3. Warum wird `zahl % 15 == 0` **zuerst** geprüft?

Ändere danach den Wert in Zeile 1 und prüfe deine Vorhersagen.

```python
zahl = 17
if zahl % 15 == 0:
    print("FizzBuzz")
elif zahl % 3 == 0:
    print("Fizz")
elif zahl % 5 == 0:
    print("Buzz")
else:
    print(zahl)
```

```
17
```

Meine Vorhersagen und Begründungen:

…

3 Mehrfachauswahl: `case of`

Oft wird **ein und derselbe Wert** mit mehreren festen Möglichkeiten verglichen, zum Beispiel eine Note in Worten. Mit `if` und `elif` steht in jeder Zeile wieder `note == …`:

```python
note = int(input("Note (1-6): "))
if note == 1:
    print("sehr gut")
elif note == 2:
    print("gut")
elif note == 3:
    print("befriedigend")
elif note == 4:
    print("ausreichend")
elif note == 5:
    print("mangelhaft")
elif note == 6:
    print("ungenügend")
else:
    print("Das ist keine gültige Note.")
```

```
Note (1-6): 2
gut
```

Unabhängig von einer Programmiersprache beschreibt man eine **Mehrfachauswahl** so:

```
CASE note OF
    1: Ausgabe "sehr gut"
    2: Ausgabe "gut"
    ...
    6: Ausgabe "ungenügend"
    OTHERWISE: Ausgabe "Das ist keine gültige Note."
END CASE
```

`OTHERWISE` („andernfalls“) ist der **Auffangfall**, wenn keiner der Werte passt.

> 📌 **Definition:** Bei einer **Mehrfachauswahl** (`CASE OF`) wird **ein Ausdruck** mit mehreren **festen Werten** verglichen. Ausgeführt wird der Fall, dessen Wert passt. Passt keiner, wird der Auffangfall ausgeführt.

In Python: `match` und `case`

```python
note = int(input("Note (1-6): "))
match note:
    case 1:
        print("sehr gut")
    case 2:
        print("gut")
    case 3:
        print("befriedigend")
    case 4:
        print("ausreichend")
    case 5:
        print("mangelhaft")
    case 6:
        print("ungenügend")
    case _:
        print("Das ist keine gültige Note.")
```

```
Note (1-6): 2
gut
```

- **Zeile 2:** `match note:` legt fest, **welcher Wert** verglichen wird. Doppelpunkt, dann einrücken.
- **Zeilen 3–14:** Jedes `case` nennt einen festen Wert. Beim ersten passenden Fall wird der Block darunter ausgeführt, die übrigen werden übersprungen.

- **Zeilen 15–16:** `case _` ist der **Auffangfall**. Der Unterstrich passt zu jedem Wert. Er entspricht `else` und steht als **letzter** Fall.

Das funktioniert genauso mit Texten. 🔮 **Vorhersage:** Was gibt die Zelle aus?

```
operator = "-"
match operator:
    case "+":
        print("Addition")
    case "-":
        print("Subtraktion")
    case _:
        print("Unbekannter Operator")
```

```
Subtraktion
```

📌 **Merksatz:** `CASE OF` eignet sich, wenn **ein Ausdruck mit mehreren festen Werten** verglichen wird. In Python heißt das `match`/`case`, `case _` ist der Auffangfall. Für **Bereiche** („unter 6 Jahren“) und Bedingungen mit `and`/`or` bleibt `if`/`elif`/`else` die richtige Wahl.

⚠️ **Stolperstein:** `match` gibt es erst seit Python 3.10. Steht `case _` nicht am Ende, meldet Python einen `SyntaxError`, weil die Fälle darunter nie erreicht würden.

📝 Übung 3 – Taschenrechner mit CASE OF (Lücken füllen)

Der Taschenrechner fragt zwei Zahlen und einen Operator (`+`, `-`, `*`, `/`) ab. Bei einer Division durch 0 und bei einem unbekannten Operator gibt er eine verständliche Meldung aus. Ersetze jedes `___`. Die Zelle läuft erst, wenn alle Lücken gefüllt sind.

```
Erste Zahl: 7
Operator (+ - * /): /
Zweite Zahl: 2
Ergebnis: 3.5
```

Teste mit allen vier Operatoren, mit `7 / 0` und mit einem unbekannten Operator wie `^`.

Lückencode

```
a = float(input("Erste Zahl: "))
op = input("Operator (+ - * /): ")      # Text, wird nicht umgewandelt
b = float(input("Zweite Zahl: "))
match ___:
    case "+":
        print("Ergebnis:", a + b)
    case "-":
        print("Ergebnis:", a - b)
    case ___:
        print("Ergebnis:", a * b)
    case "/":
        if ___:
            print("Ergebnis:", a / b)
        else:
            print("Division durch 0 ist nicht erlaubt!")
    case ___:
        print("Unbekannter Operator")
```

📝 Übung 4 – Eintrittspreis (mit Gerüst)

Im Freibad gelten diese Preise:

- Kinder unter 6 Jahren: Eintritt frei

- 6 bis 17 Jahre: 2,50 Euro
- ab 18 Jahren: 4,00 Euro, mit Schüler- oder Studierendenausweis 3,00 Euro

Das Programm fragt zuerst das Alter. **Nur** ab 18 fragt es zusätzlich nach dem Ausweis (`ja`/`nein`).

1. Lege **zuerst** eine Tabelle mit Testfällen an, die alle Zweige und Grenzfälle abdeckt.
2. Ersetze dann die Kommentare durch Programmzeilen und teste mit deiner Tabelle.

Eingaben	erwartete Ausgabe	Warum dieser Test?

```
# Alter einlesen
# if: unter 6 → Eintritt frei
# elif: unter 18 → 2,50 Euro
# else:
#     Ausweis abfragen (nur hier!)
#     innere Verzweigung: ja → 3,00 Euro, sonst → 4,00 Euro
```

🚀 Zusatzaufgabe – Taschenrechner erweitern (selbstständig)

Kopiere deinen Taschenrechner aus Übung 3 und erweitere ihn um die Operatoren `//`, `%` und `**`. Auch bei `//` und `%` darf nicht durch 0 geteilt werden. Lege zuerst die Testfälle fest. Kontrolle: `7 // 2` → 3.0, `7 % 2` → 1.0, `7 ** 2` → 49.0.

📌 Das Wichtigste in Kürze

- Fehlermeldungen von unten lesen, ersten Fehler beheben, neu starten.
- **Testfall** = Eingabe + vorher überlegte erwartete Ausgabe. Jeden Zweig und jeden Grenzfall testen.
- Bei `elif` den **speziellsten** Fall zuerst prüfen (FizzBuzz).
- **Mehrfachauswahl** `CASE OF`: ein Wert, mehrere feste Möglichkeiten. In Python `match`/`case`, Auffangfall `case _` am Ende.
- Vor einer Division prüfen, ob durch 0 geteilt wird.

Zum Weiterüben: `DS04_Weitere_Uebungen.ipynb`.