DS 4 – Weitere Übungen

Übung: Verzweigungen lesen, testen und erweitern

Hier kannst du das Gelernte weiter üben. Die Punkte im Titel zeigen, wie schwer eine Aufgabe ist: ●○○ leicht, ●●○ mittel, ●●● knifflig. Fang mit den leichten an.

Die Aufgaben stehen auch im Skript im Abschnitt „Weitere Übungen“; die Lösungen im Lösungsheft.

Aufgabe 1: Welcher Tag? ●○○

Lies das Programm, **ohne** es auszuführen. Gib an, was es für `tag = 3` ausgibt. Gib ebenso die Ausgabe für `tag = 7` und für `tag = 1` an. Prüfe danach am Computer, indem du den Wert in Zeile 1 änderst.

```
1  tag = 3
2  match tag:
3      case 1:
4          print("Montag")
5      case 2:
6          print("Dienstag")
7      case 3:
8          print("Mittwoch")
9      case _:
10         print("anderer Tag")
```

```
Mittwoch
```

Deine Antwort:

…

Lösung

✅ **Lösung**

Lösung

tag	3	7	1
Ausgabe	`Mittwoch`	`anderer Tag`	`Montag`

Python vergleicht `tag` der Reihe nach mit den festen Werten 1, 2 und 3. Beim ersten passenden Fall wird der Block darunter ausgeführt. Für `7` passt keiner der Werte, deshalb greift der Auffangfall `case _`.

Aufgabe 2: Ampel mit match ●○○

Schreibe das folgende Programm so um, dass es statt `if/elif/else` eine Mehrfachauswahl mit `match` und `case` verwendet. Das Programm soll danach genau dasselbe ausgeben. Teste mit `rot`, `grün` und `blau`.

```python
farbe = input("Ampelfarbe: ")
if farbe == "rot":
    print("Stehen bleiben!")
elif farbe == "gelb":
    print("Warten!")
elif farbe == "grün":
    print("Gehen!")
else:
    print("Diese Farbe kennt die Ampel nicht.")
```

```
Ampelfarbe: rot
Stehen bleiben!
```

Lösung

✅ **Lösung**

Lösung

```python
farbe = input("Ampelfarbe: ")
match farbe:
    case "rot":
        print("Stehen bleiben!")
    case "gelb":
        print("Warten!")
    case "grün":
        print("Gehen!")
    case _:
        print("Diese Farbe kennt die Ampel nicht.")
```

```
Ampelfarbe: rot
Stehen bleiben!
```

Lösung

Getestet mit den Eingaben: `blau`

Lösung

Ausgabe:

```
Ampelfarbe: «blau»
Diese Farbe kennt die Ampel nicht.
```

Lösung

Aus `farbe == "rot"` wird `case "rot":`. Der Name `farbe` steht nur noch **einmal** oben hinter `match`. Aus `else` wird der Auffangfall `case _`. Die `print`-Zeilen sind jetzt zwei Stufen eingerückt (acht Leerzeichen), weil sie im Block von `case` stehen und dieser im Block von `match`.

Aufgabe 3: Wochentage ●○○

Das Programm soll zu einem Kürzel von `Mo` bis `Fr` den ganzen Wochentag ausgeben. Bei allen anderen Eingaben soll es „Das ist kein Schultag.“ ausgeben. Ergänze die Lücken `___` und die fehlenden Fälle für `Mi`, `Do` und `Fr`.

Lückencode

```python
kuerzel = input("Kürzel (Mo bis Fr): ")
match ___:
    case "Mo":
        print("Montag")
    case ___:
        print("Dienstag")
    # Hier fehlen die Fälle für Mi, Do und Fr.
    ___:
        print("Das ist kein Schultag.")
```

Lösung

✅ **Lösung**

Lösung

```
kuerzel = input("Kürzel (Mo bis Fr): ")
match kuerzel:
    case "Mo":
        print("Montag")
    case "Di":
        print("Dienstag")
    case "Mi":
        print("Mittwoch")
    case "Do":
        print("Donnerstag")
    case "Fr":
        print("Freitag")
    case _:
        print("Das ist kein Schultag.")
```

```
Kürzel (Mo bis Fr): Mo
Montag
```

Lösung

Getestet mit den Eingaben: `Do`

Lösung

Ausgabe:

```
Kürzel (Mo bis Fr): «Do»
Donnerstag
```

Lösung

Die Kürzel sind Texte, deshalb stehen sie in Anführungszeichen. Die Eingabe wird **nicht** umgewandelt. Der Auffangfall `case _` muss als letzter Fall stehen. Teste auch `So` und `mo`: Beide landen im Auffangfall, denn `"mo"` ist wegen der Kleinschreibung ein anderer Text als `"Mo"`.

Aufgabe 4: Testfälle für ein Spiel ●○○

Ein Spiel teilt Spielerinnen und Spieler nach ihrem Level ein:

```
level = int(input("Level: "))
if level < 1:
    print("ungültiges Level")
elif level <= 10:
    print("Anfänger")
elif level <= 30:
    print("Fortgeschritten")
else:
    print("Profi")
```

```
Level: 3
Anfänger
```

Lege eine Tabelle mit Testfällen an: Jeder Zweig soll mindestens einmal vorkommen, außerdem alle Grenzfälle. Trage zu jedem Testfall die **erwartete** Ausgabe ein, **bevor** du das Programm ausführst. Prüfe dann am Computer.

Eingabe	erwartete Ausgabe	Warum dieser Test?

Lösung

✅ Lösung

Lösung

Eingabe	erwartete Ausgabe	Warum dieser Test?
`0`	`ungültiges Level`	Zweig „ungültig“, Grenzfall
`1`	`Anfänger`	Grenzfall, kleinstes gültiges Level
`10`	`Anfänger`	Grenzfall, wegen `<=` noch Anfänger
`11`	`Fortgeschritten`	Grenzfall, erstes Level danach
`30`	`Fortgeschritten`	Grenzfall
`31`	`Profi`	Grenzfall, Zweig „Profi“

Die Grenzen liegen bei 1, 10 und 30. An jeder Grenze testet man den Wert **auf** der Grenze und den Wert direkt daneben. Weitere sinnvolle Tests sind ein „normaler“ Wert in jedem Bereich, zum Beispiel `5`, `20` und `50`, und eine negative Zahl wie `-3`.

Aufgabe 5: Fehler finden: Pizzagröße ●●○

Das Programm soll zu einer Pizzagröße (`S`, `M` oder `L`) den Preis ausgeben. Es enthält **drei Fehler**.

```
1 groesse = input("Größe (S, M, L): ")
2 match groesse
3     case "S":
4         print("Preis: 6,50 Euro")
5     case "M":
6     print("Preis: 8,00 Euro")
7     case "L":
8         print("Preis: 9,50 Euro")
9     else:
10        print("Diese Größe gibt es nicht.")
```

```
  File "<Zelle 7>", line 2
    match groesse
                ^
SyntaxError: expected ':'
```

1. Finde die drei Fehler. Notiere jeweils die Zeile und was falsch ist.
2. Schreibe das Programm richtig auf und teste es mit `M` und mit `XL`.

Deine Antwort:

…

Lösung

✅ Lösung

Lösung

1. **Zeile 2:** Hinter `match groesse` fehlt der Doppelpunkt (`SyntaxError: expected ':'`).
2. **Zeile 6:** Die `print`-Anweisung ist nicht eingerückt. Sie muss im Block von `case "M":` stehen (`IndentationError`).
3. **Zeile 9:** In einer Mehrfachauswahl gibt es kein `else`. Der Auffangfall heißt `case _:`.

Lösung

```python
groesse = input("Größe (S, M, L): ")
match groesse:
    case "S":
        print("Preis: 6,50 Euro")
    case "M":
        print("Preis: 8,00 Euro")
    case "L":
        print("Preis: 9,50 Euro")
    case _:
        print("Diese Größe gibt es nicht.")
```

```
Größe (S, M, L): M
Preis: 8,00 Euro
```

Lösung

Getestet mit den Eingaben: `XL`

Lösung

Ausgabe:

```
Größe (S, M, L): «XL»
Diese Größe gibt es nicht.
```

Lösung

Mit der Eingabe `M` gibt das Programm `Preis: 8,00 Euro` aus. Die Preise stehen als Text in der Ausgabe, deshalb darf dort ein Komma stehen.

Aufgabe 6: match oder if? ●●○

Entscheide für jede Situation, ob eine Mehrfachauswahl mit `match`/`case` oder eine Verzweigung mit `if`/`elif`/`else` besser passt. Begründe kurz.

Situation	`match` oder `if`?
a) Eine Schulnote von 1 bis 6 in Worten ausgeben	
b) Aus einer Temperatur eine Warnstufe bestimmen (unter 0, 0 bis 30, über 30)	
c) Ein Menü mit den Punkten `a`, `b` und `c`	
d) Prüfen, ob jemand mindestens 16 ist **und** eine Einverständniserklärung hat	
e) Zu einer Monatsnummer von 1 bis 12 den Namen des Monats ausgeben	

Deine Antwort:

…

Lösung

✅ Lösung

Lösung

Situation	Auswahl	Begründung
a) Schulnote in Worten	`match`	Ein Wert wird mit festen Werten 1 bis 6 verglichen.
b) Temperatur-Warnstufe	`if`	Es geht um Bereiche, nicht um feste Werte.
c) Menü mit `a`, `b`, `c`	`match`	Ein Wert wird mit festen Texten verglichen.
d) Alter und Erklärung	`if`	Zwei Bedingungen werden mit `and` verknüpft.
e) Monatsname	`match`	Ein Wert wird mit festen Werten 1 bis 12 verglichen.

Die Faustregel: Wird **ein** Wert mit mehreren **festen** Werten verglichen, passt `match`. Bei Bereichen wie „unter 0“ oder bei verknüpften Bedingungen ist `if` die richtige Wahl. Mit `if` könnte man zwar auch a), c) und e) lösen, `match` ist dort aber übersichtlicher.

Aufgabe 7: Längen umrechnen ●●○

Schreibe ein Programm, das eine Länge (Kommazahl) und eine Einheit (`km`, `m` oder `cm`) einliest und die Länge in Metern ausgibt. Verwende eine Mehrfachauswahl. Bei einer unbekannten Einheit soll eine verständliche Meldung erscheinen. Teste mit `2.5` und `km`, mit `150` und `cm` und mit `3` und `mm`.

So könnte es aussehen:

Ausgabe:

```
Länge: «2.5»
Einheit (km, m, cm): «km»
Das sind 2500.0 Meter.
```

Lösung

✅ **Lösung**

Lösung

```python
laenge = float(input("Länge: "))
einheit = input("Einheit (km, m, cm): ")
match einheit:
    case "km":
        print("Das sind", laenge * 1000, "Meter.")
    case "m":
        print("Das sind", laenge, "Meter.")
    case "cm":
        print("Das sind", laenge / 100, "Meter.")
    case _:
        print("Die Einheit", einheit, "kenne ich nicht.")
```

```
Länge: 5
Einheit (km, m, cm): km
Das sind 5000.0 Meter.
```

Lösung

Getestet mit den Eingaben: `150`, `cm`

Lösung

Ausgabe:

```
Länge: «150»
Einheit (km, m, cm): «cm»
Das sind 1.5 Meter.
```

Lösung

Ein Kilometer hat 1000 Meter, ein Meter hat 100 Zentimeter. Deshalb wird bei `km` mit 1000 multipliziert und bei `cm` durch 100 geteilt. Bei `m` muss nichts gerechnet werden. Die Länge wird mit `float` eingelesen, die Einheit bleibt ein Text. Mit `3` und `mm` gibt das Programm `Die Einheit mm kenne ich nicht.` aus.

Aufgabe 8: Taschenrechner erweitern ●●●

Erweitere deinen Taschenrechner aus dem Auftrag *Taschenrechner mit CASE OF* um drei weitere Operatoren: `//` (Ganzzahldivision), `%` (Rest) und `**` (Potenz). Auch bei `//` und `%` darf die zweite Zahl nicht 0 sein. Lege eine

Tabelle mit Testfällen an, in der jeder neue Operator und jeder Sonderfall mindestens einmal vorkommt, und teste dein Programm damit.

Deine Antwort:

…

Lösung

✅ Lösung

Lösung

```python
a = float(input("Erste Zahl: "))
op = input("Operator (+ - * / // % **): ")
b = float(input("Zweite Zahl: "))

match op:
    case "+":
        print("Ergebnis:", a + b)
    case "-":
        print("Ergebnis:", a - b)
    case "*":
        print("Ergebnis:", a * b)
    case "/":
        if b != 0:
            print("Ergebnis:", a / b)
        else:
            print("Division durch 0 ist nicht erlaubt!")
    case "//":
        if b != 0:
            print("Ergebnis:", a // b)
        else:
            print("Division durch 0 ist nicht erlaubt!")
    case "%":
        if b != 0:
            print("Ergebnis:", a % b)
        else:
            print("Division durch 0 ist nicht erlaubt!")
    case "**":
        print("Ergebnis:", a ** b)
    case _:
        print("Unbekannter Operator")
```

```
Erste Zahl: 12
Operator (+ - * / // % **): +
Zweite Zahl: 5
Ergebnis: 17.0
```

Lösung

Getestet mit den Eingaben: `7`, `%`, `0`

Lösung

Ausgabe:

```
Erste Zahl: «7»
Operator (+ - * / // % **): «%»
Zweite Zahl: «0»
Division durch 0 ist nicht erlaubt!
```

Lösung

Eingaben	erwartete Ausgabe
`7`, `//`, `2`	`Ergebnis: 3.0`
`7`, `%`, `2`	`Ergebnis: 1.0`
`2`, `**`, `10`	`Ergebnis: 1024.0`
`7`, `//`, `0`	`Division durch 0 ist nicht erlaubt!`
`7`, `%`, `0`	`Division durch 0 ist nicht erlaubt!`
`7`, `^`, `2`	`Unbekannter Operator`

Für jeden neuen Operator gibt es einen eigenen Fall. `//` und `%` teilen ebenfalls, deshalb brauchen sie wie `/` eine verschachtelte Verzweigung. Ohne sie bricht das Programm mit einem `ZeroDivisionError` ab. Die Ergebnisse haben alle `.0`, weil die Zahlen mit `float` eingelesen werden. Die alten Testfälle aus dem Auftrag *Taschenrechner mit CASE OF* solltest du auch noch einmal ausprobieren.