DS 2 – Weitere Übungen

Operatoren: rechnen, vergleichen, verknüpfen

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: Teilen mit Rest ●○○

Bestimme das Ergebnis jedes Ausdrucks zuerst im Kopf und trage es in die Tabelle ein. Überprüfe dann mit `print(...)` am Computer.

Ausdruck	Ergebnis	Ausdruck	Ergebnis
`29 // 5`		`2 ** 4`	
`29 % 5`		`17 % 2`	
`40 % 8`		`10 - 4 // 2`	
`3 % 8`		`(10 - 4) // 2`	

Lösung

✅ Lösung

Lösung

```
print(29 // 5, 29 % 5, 40 % 8, 3 % 8)
print(2 ** 4, 17 % 2, 10 - 4 // 2, (10 - 4) // 2)
```

```
5 4 0 3
16 1 8 3
```

Lösung

Ausgabe:

```
5 4 0 3
16 1 8 3
```

Lösung

- Die 5 passt 5-mal ganz in die 29, es bleibt der Rest 4 (denn 5 · 5 + 4 = 29).
- `40 % 8` ist `0`: 40 ist durch 8 teilbar.
- `3 % 8` ist `3`: Die 8 passt keinmal in die 3, also ist die ganze 3 der Rest.
- `17 % 2` ist `1`: 17 ist ungerade.
- In `10 - 4 // 2` wird zuerst `4 // 2 = 2` gerechnet, denn `//` ist Punktrechnung. Mit Klammern wird zuerst `10 - 4 = 6` gerechnet, dann `6 // 2 = 3`.

Aufgabe 2: True oder False? ●○○

Es gilt `a = 4` und `b = 9`. Bestimme für jeden Vergleich, ob er `True` oder `False` ergibt. Überprüfe deine Antworten danach am Computer.

Ausdruck	Ergebnis	Ausdruck	Ergebnis
`a < b`		`a * 2 > b`	
`b != 9`		`b % a == 1`	
`a >= 5`		`a + 5 == b`	

Lösung

✅ **Lösung**

Lösung

```
1 a = 4
2 b = 9
3 print(a < b, b != 9, a >= 5)
4 print(a * 2 > b, b % a == 1, a + 5 == b)
```

```
True False False
False True True
```

Lösung

Ausgabe:

```
True False False
False True True
```

Lösung

- `b != 9`: `b` ist gleich 9, die Frage „ist ungleich 9?“ wird also mit `False` beantwortet.
- `a >= 5`: 4 ist weder größer als 5 noch gleich 5.
- `a * 2 > b`: Zuerst wird gerechnet (`8`), dann verglichen: `8 > 9` ist falsch.
- `b % a == 1`: 9 geteilt durch 4 ergibt 2 Rest 1. Der Rest ist also 1.

Aufgabe 3: Wahrheitstabelle ausfüllen ●○○

Fülle die Tabelle aus. Bestimme zuerst die Spalte `not B`. Mit ihrer Hilfe kannst du die beiden letzten Spalten ausfüllen.

A	B	not B	A and not B	A or not B
`False`	`False`			
`False`	`True`			
`True`	`False`			
`True`	`True`			

Lösung

✅ **Lösung**

Lösung

A	B	not B	A and not B	A or not B
`False`	`False`	`True`	`False`	`True`
`False`	`True`	`False`	`False`	`False`
`True`	`False`	`True`	`True`	`True`
`True`	`True`	`False`	`False`	`True`

`not` wird vor `and` und `or` ausgewertet. `A and not B` ist nur in der dritten Zeile wahr: Dort ist `A` wahr **und** `not B` wahr. `A or not B` ist nur in der zweiten Zeile falsch: Dort sind `A` und `not B` beide falsch.

Zum Nachprüfen kannst du für `A` und `B` Werte festlegen, zum Beispiel:

Lösung

```
1 A = True
2 B = False
3 print(not B, A and not B, A or not B)
```

```
True True True
```

Lösung

Ausgabe:

```
True True True
```

Aufgabe 4: Eier verpacken ●○○

Eier werden in Kartons zu je 6 Stück verpackt. Das Programm soll eine Anzahl von Eiern einlesen und ausgeben, wie viele Kartons voll werden und wie viele Eier übrig bleiben. Ergänze die beiden Lücken `___` mit `//` und `%`. Teste mit `20`, `18` und `5`.

Lückencode

```
1 eier = int(input("Wie viele Eier? "))
2 kartons = ___    # volle Kartons
3 rest = ___       # übrige Eier
4 print(eier, "Eier ergeben", kartons, "volle Kartons und",
5       rest, "übrige Eier.")
```

Lösung

✅ **Lösung**

Lösung

```
1 eier = int(input("Wie viele Eier? "))
2 kartons = eier // 6
3 rest = eier % 6
4 print(eier, "Eier ergeben", kartons, "volle Kartons und",
5       rest, "übrige Eier.")
```

```
Wie viele Eier? 23
23 Eier ergeben 3 volle Kartons und 5 übrige Eier.
```

Lösung

Getestet mit den Eingaben: `20`

Lösung

Ausgabe:

```
Wie viele Eier? «20»
20 Eier ergeben 3 volle Kartons und 2 übrige Eier.
```

Lösung

Eingabe	kartons	rest
20	3	2
18	3	0
5	0	5

`eier // 6` gibt an, wie oft die 6 ganz in die Anzahl passt. `eier % 6` ist der Rest. Wer die beiden Operatoren vertauscht, bekommt bei 20 Eiern „2 Kartons und 3 Eier“ – das ist falsch.

Aufgabe 5: Bedingungen aufschreiben ●●○

Schreibe zu jeder Aussage einen Python-Ausdruck, der `True` ergibt, wenn die Aussage stimmt, und sonst `False`. Teste jeden Ausdruck am Computer mit zwei passenden Werten.

1. Die Zahl `x` ist durch 3 teilbar.
2. Das Alter `alter` liegt zwischen 10 und 15 (beide Grenzen gehören dazu).
3. Die Punktzahl `punkte` ist kleiner als 0 oder größer als 100.
4. Die Zahl `zahl` ist ungerade.
5. Die Anzahl `stunden` ist nicht 0.

Deine Antwort:

…

Lösung

✅ **Lösung**

Lösung

1. `x % 3 == 0`
2. `alter >= 10 and alter <= 15`
3. `punkte < 0 or punkte > 100`
4. `zahl % 2 == 1` (oder `zahl % 2 != 0`)
5. `stunden != 0` (oder `not stunden == 0`)

Kontrolle mit Beispielwerten:

Lösung

```
x = 12
alter = 15
punkte = 105
zahl = 8
stunden = 0
print(x % 3 == 0, alter >= 10 and alter <= 15)
print(punkte < 0 or punkte > 100, zahl % 2 == 1, stunden != 0)
```

```
True True
True False False
```

Lösung

Ausgabe:

```
True True
True False False
```

Lösung

Bei 2. braucht man auf **beiden** Seiten von `and` einen vollständigen Vergleich. `alter >= 10 and <= 15` ist ein Syntaxfehler. Weil die Grenzen dazugehören, steht `>=` und `<=`, nicht `>` und `<`. Bei 3. muss es `or` heißen: Eine Zahl kann nicht gleichzeitig kleiner als 0 **und** größer als 100 sein.

Aufgabe 6: Fehler finden: Teilbar durch 5? ●●○

Das Programm soll eine ganze Zahl einlesen und einen Wahrheitswert ausgeben, der angibt, ob die Zahl durch 5 teilbar ist. Es enthält **drei Fehler**.

```
1 zahl = input("Zahl: ")
2 durch_fuenf = zahl % 5 = 0
3 print("Durch 5 teilbar:" durch_fuenf)
```

```
  File "<Zelle 7>", line 2
    durch_fuenf = zahl % 5 = 0
                  ^^^^^^^^
SyntaxError: cannot assign to expression
```

1. Finde die drei Fehler. Notiere jeweils die Zeile und was falsch ist.
2. Schreibe das Programm richtig auf und teste es mit `35` und `42`.

Deine Antwort:

…

Lösung

✅ **Lösung**

Lösung

1. **Zeile 1:** Die Eingabe wird nicht in eine Zahl umgewandelt. Mit dem Text `"35"` kann `%` nicht den Rest berechnen. Python meldet einen `TypeError`. Richtig: `int(input(...))`.
2. **Zeile 2:** Für den Vergleich braucht man `==`, nicht `=`. Python meldet einen `SyntaxError`.
3. **Zeile 3:** Zwischen dem Text und der Variablen fehlt das Komma. Python meldet `SyntaxError: invalid syntax. Perhaps you forgot a comma?`

Lösung

```
1 zahl = int(input("Zahl: "))
2 durch_fuenf = zahl % 5 == 0
3 print("Durch 5 teilbar:", durch_fuenf)
```

```
Zahl: 15
Durch 5 teilbar: True
```

Lösung

Getestet mit den Eingaben: `35`

Lösung

Ausgabe:

```
Zahl: «35»
Durch 5 teilbar: True
```

Lösung

Mit der Eingabe `42` gibt das Programm `Durch 5 teilbar: False` aus, denn 42 geteilt durch 5 ergibt 8 Rest 2.

Aufgabe 7: Ausleihe in der Bücherei ●●○

In der Stadtbücherei gilt: Man darf ein Buch ausleihen, wenn man einen Ausweis hat **und** entweder keine offenen Gebühren hat **oder** jünger als 18 ist. Im Programm sieht das so aus:

```
1 hat_ausweis = True
2 gebuehren = 0
3 alter = 30
4 darf_ausleihen = hat_ausweis and (gebuehren == 0 or alter < 18)
5 print("Darf ausleihen:", darf_ausleihen)
```

```
Darf ausleihen: True
```

1. Bestimme `darf_ausleihen` für die vier Fälle in der Tabelle. Prüfe am Computer, indem du die Werte in den Zeilen 1 bis 3 änderst.
2. Jemand lässt die Klammern weg: `hat_ausweis and gebuehren == 0 or alter < 18`. Bestimme das Ergebnis für den vierten Fall und erkläre den Unterschied.

hat_ausweis	gebuehren	alter	darf_ausleihen
True	0	30	
True	2	15	
True	2	40	
False	0	12	

Deine Antwort:

…

Lösung

✅ **Lösung**

Lösung

hat_ausweis	gebuehren	alter	darf_ausleihen
True	0	30	True
True	2	15	True
True	2	40	False
False	0	12	False

Im zweiten Fall hat die Person zwar Gebühren, ist aber jünger als 18. Die Klammer ergibt `False or True`, also `True`. Im vierten Fall fehlt der Ausweis. Dann ist das Ganze wegen `and` sofort `False`, egal was in der Klammer steht.

2. Ohne Klammern wird zuerst `and` ausgewertet, danach `or`. Für den vierten Fall:

Lösung

```
1 hat_ausweis = False
2 gebuehren = 0
3 alter = 12
4 print(hat_ausweis and gebuehren == 0 or alter < 18)
```

```
True
```

Lösung

Ausgabe:

```
True
```

Lösung

`False and True` ergibt `False`, danach `False or True` ergibt `True`. Ohne Klammern dürfte also jedes Kind unter 18 ausleihen, auch **ohne** Ausweis. Die Klammern sorgen dafür, dass der Ausweis immer nötig ist.

Aufgabe 8: Ziffern zerlegen ●●●

Schreibe ein Programm, das eine dreistellige Zahl einliest und ihre Ziffern einzeln ausgibt: die Hunderterziffer, die Zehnerziffer und die Einerziffer. Zum Schluss soll es die **Quersumme** ausgeben, also die Summe der drei Ziffern. Verwende dafür `//` und `%`.

So könnte es aussehen:

Ausgabe:

```
Dreistellige Zahl: «472»
Hunderter: 4
Zehner: 7
Einer: 2
Quersumme: 13
```

Tipp: Die Einerziffer ist der Rest beim Teilen durch 10. Die Hunderterziffer sagt, wie oft die 100 ganz in die Zahl passt. Für die Zehnerziffer brauchst du zwei Schritte: `472 // 10` ergibt `47`. Welche Rechnung liefert daraus die `7`?

Lösung

✅ Lösung

Lösung

```python
zahl = int(input("Dreistellige Zahl: "))
hunderter = zahl // 100
zehner = zahl // 10 % 10
einer = zahl % 10
print("Hunderter:", hunderter)
print("Zehner:", zehner)
print("Einer:", einer)
print("Quersumme:", hunderter + zehner + einer)
```

```
Dreistellige Zahl: 472
Hunderter: 4
Zehner: 7
Einer: 2
Quersumme: 13
```

Lösung

Getestet mit den Eingaben: `472`

Lösung

Ausgabe:

```
Dreistellige Zahl: «472»
Hunderter: 4
Zehner: 7
Einer: 2
Quersumme: 13
```

Lösung

- **Zeile 2:** Die 100 passt 4-mal ganz in die 472.
- **Zeile 3:** `//` und `%` werden von links nach rechts ausgewertet: `472 // 10` ergibt `47`, und `47 % 10` ergibt `7`. Auch `zahl % 100 // 10` ist richtig: `472 % 100` ergibt `72`, und `72 // 10` ergibt `7`.
- **Zeile 4:** Der Rest beim Teilen durch 10 ist immer die letzte Ziffer.

Teste auch eine Zahl mit einer 0 in der Mitte, zum Beispiel `305`: Dann ist die Zehnerziffer `0` und die Quersumme `8`.