Bedingungen in Python

Eine **Bedingung** ist ein Ausdruck, der `True` (wahr) oder `False` (falsch) ergibt. Mit `if` entscheidest du, **ob** ein Block ausgeführt wird. Mit `while` entscheidest du, **wie lange** er wiederholt wird.

1. Werte vergleichen

Zeichen	Bedeutung	Beispiel
`==`	gleich	`preis == 0`
`!=`	ungleich	`preis != 0`
`<`	kleiner als	`preis < 0`
`<=`	kleiner als oder gleich	`alter <= 16`
`>`	größer als	`preis > 0`
`>=`	größer als oder gleich	`bezahlt >= 6`

Merke: `preis = 0` speichert den Wert 0. `preis == 0` prüft, ob der Wert 0 ist.

2. Bedingungen verknüpfen

Wort	Wann ist das Ergebnis wahr?	Beispiel
`and`	Beide Bedingungen sind wahr.	`zahl >= 1 and zahl <= 10`
`or`	Mindestens eine Bedingung ist wahr.	`muenze == 1 or muenze == 2`
`not`	Die Bedingung ist falsch.	`not (preis == 0)`

Tipp: Setze Klammern, wenn du `and` und `or` kombinierst: `(muenze == 1 or muenze == 2) and bezahlt < 6`.

3. Mit if, elif und else entscheiden

Python prüft von oben nach unten. **Nur der erste passende Block** läuft. Trifft keine Bedingung zu, läuft der **else**-Block, falls es einen gibt.

```
geheim = 7
tipp = int(input("Dein Tipp: "))
if tipp < geheim:
    print("Zu klein.")
elif tipp > geheim:
    print("Zu groß.")
else:
    print("Richtig geraten!")
```

`elif` bedeutet „sonst, falls …“. `else` gilt für alle übrigen Fälle und hat keine eigene Bedingung.

Sprechende Variablennamen. Wähle Namen, die verraten, was ein Wert bedeutet. Bei `if tipp < geheime_zahl:` erkennst du sofort, dass der Tipp mit der gesuchten Zahl verglichen wird. Bei `if x < y:` musst du erst nachschauen, wofür `x` und `y` stehen. Sprechende Namen helfen dir und anderen, den Code schneller zu verstehen, Fehler zu finden und ihn später zu ändern. Wenn die Einheit wichtig ist, schreib sie in den Namen, zum Beispiel `preis_in_cent`.

4. Mit while wiederholen

Python prüft die Bedingung **vor jedem Durchlauf**. Ist sie wahr, läuft der eingerückte Block. Ist sie falsch, geht es nach der Schleife weiter. Ist sie schon beim ersten Prüfen falsch, läuft der Block kein einziges Mal.

Beispiel: Eingaben an der Pausenkiosk-Kasse

```
preis = int(input("Preis in Cent (0 = Ende): "))

while preis != 0:
    if preis < 0:
        print("Ungültiger Preis.")
    else:
        print("Preis erfasst.")

    preis = int(input("Nächster Preis in Cent (0 = Ende): "))

print("Einkauf beendet.")
```

Die neue Eingabe steht **in der Schleife**, aber **außerhalb der Verzweigung**. So wird nach einem gültigen und nach einem ungültigen Preis erneut gefragt. Bei 0 endet die Schleife.

5. Typische Fehler

Fehler	So machst du es richtig
`if preis = 0:`	Vergleiche mit ==: `if preis == 0:`.
`if muenze == 1 or 2:`	Schreibe beide Vergleiche aus: `if muenze == 1 or muenze == 2:`. Die falsche Version ist auch bei anderen Münzwerten wahr.
`preis = input(...)` und danach `preis > 0`	`input()` liefert Text. Für ganze Zahlen nutze `preis = int(input(...))`.
`if preis < 0` ohne Doppelpunkt	Setze nach `if`, `elif`, `else` und `while` einen Doppelpunkt. Rücke den jeweiligen Block um vier Leerzeichen ein.
`while preis != 0:` ohne neue Eingabe im Block	Aktualisiere `preis` in der Schleife. Sonst kann die Bedingung dauerhaft wahr bleiben.
Der Fahrkartenautomat stoppt nur bei `bezahlt == 6`.	Lass ihn mit `while bezahlt < 6:` weiterlaufen. Dann endet die Schleife auch bei 7 Euro.

Kurzcheck: Teste bei einem Grenzwert einen Wert **darunter**, **genau darauf** und **darüber**. Beim Fahrkartenpreis von 6 Euro: Was passiert bei 5, 6 und 7 Euro?