

LU01.A11 - Spaghetticode entwirren

Bringen Sie eine unübersichtliche Funktion auf die drei Grundstrukturen der strukturierten Programmierung zurück – Sequenz, Selektion und Iteration – ohne ihr Verhalten zu verändern.

Ausgangslage

Die folgende Funktion berechnet den Umsatz aus einer Liste von Bestellungen. Sie ist korrekt, aber niemand liest sie gern: eine while-Schleife mit Abbruchflagge, ein von Hand hochgezählter Index, continue-Sprünge und drei Verschachtelungsebenen.

Im Repository steht sie in `referenz.py` und ist der Massstab: Ihre Fassung muss für jede Eingabe exakt dasselbe liefern. **Diese Datei dürfen Sie nicht verändern.**

```
def umsatz_alt(bestellungen):
    i = 0
    fertig = False
    total = 0
    while not fertig:
        if i >= len(bestellungen):
            fertig = True
        else:
            b = bestellungen[i]
            if b['status'] == 'storniert':
                i = i + 1
                continue
            else:
                if b['menge'] <= 0:
                    i = i + 1
                    continue
                else:
                    total = total + b['menge'] * b['preis']
                    i = i + 1
    return total
```

Testdaten

```
BESTELLUNGEN = [
    {'artikel': 'Maus',      'status': 'offen',      'menge': 3, 'preis':
24.90},
    {'artikel': 'Tastatur', 'status': 'storniert', 'menge': 2, 'preis':
79.50},
    {'artikel': 'Monitor',  'status': 'offen',      'menge': 0, 'preis':
```

```
249.00},
    {'artikel': 'Kabel',      'status': 'geliefert', 'menge': 10, 'preis':
9.90},
    {'artikel': 'Dock',      'status': 'offen',      'menge': 1, 'preis':
189.00},
]
```

Anforderungen

1. **Analysieren:** Markieren Sie im Code von `referenz.py`, wo Sequenz, Selektion und Iteration vorkommen. Welche Stellen entsprechen keiner der drei Grundstrukturen, sondern imitieren einen Sprung?
2. **Umbauen:** Implementieren Sie in `main.py` die beiden vorgegebenen Funktionen. Vorgaben:
 - Die Schleife läuft direkt über die Elemente (`for bestellung in bestellungen`), ohne Index und ohne Abbruchflagge.
 - Kein `continue`, kein `break`, kein `while`.
 - Höchstens **eine** Verschachtelungsebene innerhalb der Schleife.
 - Die Bedingung, ob eine Bestellung überhaupt zählt, steckt in `ist_verrechenbar` und gibt `True` oder `False` zurück.
3. **Begründen:** Halten Sie in zwei bis drei Sätzen fest, welche Fehlerquellen im Ausgangscode durch den Umbau verschwunden sind.

Vorgegebene Funktionen

Funktion	Rückgabe
<code>ist_verrechenbar(bestellung)</code>	<code>True</code> , wenn die Bestellung nicht storniert ist und die Menge grösser als 0 ist
<code>umsatz(bestellungen)</code>	Summe aus Menge mal Preis über alle verrechenbaren Bestellungen, bei leerer Liste 0

Beispieloutput

```
Umsatz neu: 362.70
```

Zusatzfragen

1. Der Ausgangscode zählt `i` an drei verschiedenen Stellen hoch. Was passiert, wenn man eine dieser Stellen vergisst? Probieren Sie es in einer Kopie aus.
2. Das Theorem von Böhm und Jacopini besagt, dass jeder Algorithmus mit den drei Grundstrukturen auskommt. Wie stützt Ihr Umbau diese Aussage?
3. **Für Schnelle:** Formulieren Sie die Anforderung an `umsatz` deklarativ – als Satz über das Ergebnis, ohne Schleife und Zwischensumme.

Wichtig

Das Verhalten der Funktion darf sich nicht ändern – auch nicht in Randfällen.
Geprüft werden: leere Liste, alle Bestellungen storniert, Menge 0, Menge negativ.
Zwei weitere Tests lesen `main.py` als Syntaxbaum ein und prüfen, dass eine `for`-Schleife vorhanden ist und weder `while` noch `continue` oder `break` vorkommen.

Bewertung

Teil	Punkte
Unittests	11
Linting	5
Total	16

Abgabe

Der Code wird über das Classroom-Repository abgegeben und automatisch bewertet. Die **Analyse**, die **Begründung** und die **Zusatzfragen** geben Sie zusätzlich in Moodle ab.

 © Kevin Maurizi

From:
<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:
<https://wiki.bzz.ch/modul/m323/learningunits/lu01/aufgaben/spaghetticode?rev=1787043621>

Last update: **2026/08/18 11:00**

