LU02.L02 - Codeabdeckung

Musterlösung zu LU02.A02 - Codeabdeckung. Die Resultate und die Abdeckung wurden mit `coverage.py` nachgeprüft.

Testfälle

Jeder Testfall ist ein vollständiger Programmlauf. Die Notenwerte werden zuerst eingegeben, danach die Beträge.

Nr	Bezeichnung	Notenwerte	Beträge	Erwartetes Resultat (Ausgabe)
T1	Normalfall	100, 50, 20, 10	80, 0	50 K$, 20 K$, 10 K$
T2	Notenwert ist keine Zahl	100, zehn, 50, 20, 10	30, 0	Gib eine Ganzzahl ein, 20 K$, 10 K$
T3	Negativer Betrag	100, 50, 20, 10	-20, 0	Negative Beträge sind unmöglich
T4	Betrag ist keine Zahl	100, 50, 20, 10	viel, 0	Gib eine Ganzzahl ein
T5	Sofort beenden	100, 50, 20, 10	0	(keine Ausgabe)
T6	Kleinster Betrag	100, 50, 20, 10	1, 0	1 K$
T7	Unsortierte Noten, Rest mit 1 K$ | 10, 100, 20, 50 | 186, 0 | 100 K$, 50 K$, 20 K$, 10 K$, 6 × 1 K$			
T8	Betrag kleiner als alle Noten	200, 100, 50, 20	3, 0	1 K$, 1 K$, 1 K$
T9	Mehrere Bezüge nacheinander	100, 50, 20, 10	30, 70, 0	20 K$, 10 K$, 50 K$, 20 K$
T10	Grosser Betrag, grosse Noten	1000, 500, 100, 5	1606, 0	1000 K$, 500 K$, 100 K$, 5 K$, 1 K$

Zähltabelle Anweisungen

Nr	Anweisung	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	abgedeckt
A1	`bills = read_bills()`	x	x	x	x	x	x	x	x	x	x	ja
A2	`bills.sort(reverse=True)`	x	x	x	x	x	x	x	x	x	x	ja
A3	`withdrawl(bills)`	x	x	x	x	x	x	x	x	x	x	ja
A4	`values = [1]`	x	x	x	x	x	x	x	x	x	x	ja
A5	`print('Gib die Notenwerte ein')`	x	x	x	x	x	x	x	x	x	x	ja
A6	`index = 1`	x	x	x	x	x	x	x	x	x	x	ja
A7	`value = int(input(…))`	x	x	x	x	x	x	x	x	x	x	ja
A8	`values.append(value)`	x	x	x	x	x	x	x	x	x	x	ja
A9	`index += 1` (in `read_bills`)	x	x	x	x	x	x	x	x	x	x	ja
A10	`print('Gib eine Ganzzahl ein')` (in `read_bills`)		x									ja
A11	`return values`	x	x	x	x	x	x	x	x	x	x	ja
A12	`amount = -1`	x	x	x	x	x	x	x	x	x	x	ja
A13	`amount = int(input(…))`	x	x	x	x	x	x	x	x	x	x	ja

Nr	Anweisung	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	abgedeckt
A14	`output_bills(amount, bills)`	x	x				x	x	x	x	x	ja
A15	`print('Negative Beträge sind unmöglich')`			x								ja
A16	`print('Gib eine Ganzzahl ein')` (in `withdrawl`)				x							ja
A17	`index = 0`	x	x				x	x	x	x	x	ja
A18	`print(f'{bills[index]} K$')`	x	x				x	x	x	x	x	ja
A19	`amount -= bills[index]`	x	x				x	x	x	x	x	ja
A20	`index += 1` (in `output_bills`)	x	x				x	x	x	x	x	ja

Zähltabelle Bedingungen

Nr	Bedingung	Ergebnis	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	abgedeckt
B1	`while index < 5`	true	x	x	x	x	x	x	x	x	x	x	ja
		false	x	x	x	x	x	x	x	x	x	x	ja
B2	`while amount != 0`	true	x	x	x	x	x	x	x	x	x	x	ja
		false	x	x	x	x	x	x	x	x	x	x	ja
B3	`if amount > 0`	true	x	x				x	x	x	x	x	ja
		false	x	x	x	x	x	x	x	x	x	x	ja
B4	`elif amount < 0`	true			x								ja
		false	x	x	x	x	x	x	x	x	x	x	ja
B5	`while amount > 0`	true	x	x				x	x	x	x	x	ja
		false	x	x				x	x	x	x	x	ja
B6	`if amount >= bills[index]`	true	x	x				x	x	x	x	x	ja
		false	x	x				x	x	x	x	x	ja

Abdeckung

- **Anweisungsabdeckung** = 20 ÷ 20 × 100 % = **100 %**
- **Bedingungsabdeckung** = 12 ÷ 12 × 100 % = **100 %**

Erkenntnisse

- Schon die vier Testfälle **T1 bis T4** erreichen zusammen 100 % Anweisungs- und Bedingungsabdeckung. Die übrigen Testfälle erhöhen die Abdeckung nicht mehr, prüfen aber wichtige Fälle der **Funktion**: Grenzwerte (T6), unsortierte Eingaben (T7), Beträge kleiner als alle Noten (T8), mehrere Bezüge (T9).
- Die Bedingung `amount > 0` ergibt in **jedem** Testfall auch `false`, weil jeder Lauf mit der Eingabe `0` endet. Ebenso wird `while index < 5` immer zuerst `true` und am Ende `false`.
- 100 % Abdeckung beweist nicht, dass das Programm fehlerfrei ist (siehe LU02c). Ein Notenwert `0` oder ein negativer Notenwert wird zum Beispiel ohne Meldung akzeptiert. Weil die Note 1 immer vorhanden ist, funktioniert die Auszahlung trotzdem. Ob das gewünscht ist, sagt die Abdeckung nicht.

M450-LU02, M450-C1B, M450-B1I

Marcel Suter

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

Permanent link:
https://wiki.bzz.ch/modul/m450/learningunits/lu02/loesungen/coverage

Last update: **2026/11/17 17:48**