Codeklasse

Der smarte Kaffeeautomat

Thema: Rückgabewerte & Gültigkeitsbereiche (Scope)

Name: _____

Klasse: _____ Datum: _____

Grundlagen & Merkkasten

WISSEN

Bisher haben Funktionen Aufgaben erledigt und Ergebnisse oft direkt mit `print()` auf dem Bildschirm angezeigt.
In professionellen Programmen müssen Funktionen jedoch Ergebnisse berechnen und diesen Wert an das Hauptprogramm zurückgeben (`return`), damit mit dem Ergebnis weitergerechnet werden kann.
Zudem gilt das Prinzip des Gültigkeitsbereichs (Scope): Variablen, die innerhalb einer Funktion deklariert werden, sind lokal und existieren außerhalb der Funktion nicht.

`def berechne_preis(menge, einzelpreis):`
`  gesamt = menge * einzelpreis`
`  return gesamt`
	Das Schlüsselwort `return` beendet die Funktion sofort und gibt das berechnete Ergebnis an den Aufrufer zurück.

`endbetrag = berechne_preis(2, 3)`	Der Rückgabewert der Funktion wird in der Variablen `endbetrag` zur Weiterverarbeitung gespeichert.

`def test():`
`  lokal = 42`
`# ausserhalb: lokal existiert nicht!`
	Lokale Variablen existieren nur während der Funktionsausführung und sind im Hauptprogramm unbekannt.

1. Code-Puzzle (Reihenfolge & Einrückung)

AUFGABE 1

Bringe die Code-Blöcke in die richtige Reihenfolge, sodass die Funktion `kalkuliere_kaffee` den Gesamtpreis berechnet und zurückgibt.
Anschließend soll der Rückgabewert im Hauptprogramm gespeichert und ausgegeben werden.

[ ]	`def kalkuliere_kaffee(grundpreis, sirup):`
[ ]	`summe = grundpreis + sirup`
[ ]	`return summe`
[ ]	`endpreis = kalkuliere_kaffee(3, 1)`
[ ]	`print('Bestellung: Cappuccino')`
[ ]	`print('Preis:', endpreis, 'Euro')`
[ ]	`print(summe)`
[ ]	`return print(summe)`

Hinweis: Trage in die eckigen Klammern die korrekte Reihenfolge (1, 2, 3...) ein. Streiche überflüssige oder fehlerhafte Zeilen (Distraktoren) durch!

2. Fehlersuche & Code-Reparatur

AUFGABE 2

Im folgenden Skript für eine Rabatt-Berechnung gibt es zwei Probleme:
1. Die Funktion `berechne_rabatt` gibt das Ergebnis nicht an das Hauptprogramm zurück.
2. Das Hauptprogramm versucht auf die lokale Variable `rabatt_betrag` zuzugreifen, was zu einem `NameError` führt.

Korrigiere den Code, sodass die Funktion den rabattierten Betrag mit `return` zurückliefert und das Hauptprogramm diesen in `preis_nach_rabatt` speichert.

Fehlerhafter Ausgangs-Code:

```
1  def berechne_rabatt(basispreis, rabatt_prozent):
2      ersparnis = basispreis * (rabatt_prozent / 100)
3      rabatt_betrag = basispreis - ersparnis
4      # FEHLER: Hier fehlt die Rückgabe!
5
6  # FEHLER: Die Zuweisung und der Variablenzugriff sind fehlerhaft!
7  berechne_rabatt(10, 20)
8  print("Endpreis:", rabatt_betrag, "Euro")
```

Deine Korrektur / Notizen:

3. Programmier-Challenge

AUFGABE 3

Programmiere das Abrechnungsmodul für den Automaten:

1. Schreibe eine Funktion `kassieren(anzahl, einzelpreis)`:
- Berechne darin den Gesamtbetrag (`anzahl * einzelpreis`).
- Gib diesen Betrag mit `return` zurück.
2. Rufe die Funktion mit `3` als Einzelpreis und `2` als Anzahl auf.
3. Speichere das Ergebnis in der globalen Variablen `gesamtbetrag`.
4. Gib auf der Konsole genau folgende Zeilen aus:
`Kunde: Sam`
`Summe: <gesamtbetrag> Euro`

Dein Python-Code:

Transfer & Code-Analyse: print() vs. return und Scope

AUFGABE 4

Betrachte folgendes Codebeispiel:

```
def verdoppeln(x):
    ergebnis = x * 2
    print(ergebnis)

wert = verdoppeln(5)
summe = wert + 10
```

1. Erkläre, warum in der letzten Zeile ein Fehler (`TypeError`) auftritt.
2. Was unterscheidet den Befehl `print()` grundlegend von der Anweisung `return` hinsichtlich des Datenflusses?
3. Ist die Variable `ergebnis` im Hauptprogramm aufrufbar? Begründe mit dem Begriff des Gültigkeitsbereichs (Scope).