Codeklasse

Das Text-Adventure

Thema: Synthese-Projekt: Kontrollstrukturen & Listen

Name: _____

Klasse: _____ Datum: _____

Grundlagen & Merkkasten

WISSEN

Ein vollständiges interaktives Text-Adventure führt alle bisherigen Programmierkonzepte zusammen:
Variablen speichern den aktuellen Zustand (wie Lebenspunkte oder Raumname), Listen verwalten
das Inventar dynamisch, und Verzweigungen (`if` / `elif` / `else`) reagieren flexibel auf Spielaktionen.
Eine `while` -Schleife hält den Spielzyklus am Laufen, bis eine Endbedingung erfüllt ist.

`spiel_aktiv = True`
`while spiel_aktiv:`
`    ...`
	Hauptspielschleife (Game-Loop), die bis zum Eintreten einer Abbruchbedingung läuft.

`inventar.append('Item')`	Fügt ein neues Element am Ende der Liste hinzu.
`if 'Item' in inventar:`	Prüft mit dem in-Operator, ob ein bestimmtes Element in der Liste existiert.
`hp -= schaden`	Verringert den Wert einer nummerischen Variablen um den angegebenen Betrag.

1. Code-Puzzle (Reihenfolge & Einrückung)

AUFGABE 1

Setze das Skript in folgender chronologischer Reihenfolge zusammen:
1. Den Raumnamen `ort` und eine leere Liste `inventar` anlegen.
2. Den Raum betreten und die Begrüßung ausgeben.
3. Den gefundenen Gegenstand `'Rostiger Schluessel'` ins Inventar aufnehmen und den Fund melden.
4. Prüfen, ob `'Rostiger Schluessel'` im Inventar liegt, und nur in diesem Fall die Erfolgsmeldung eingerückt ausgeben.

Achtung: Behalte den Gegenstand im Inventar (nicht ablegen) und ignoriere überflüssige Blöcke!

[ ]	`ort = 'Kerker des Schattens'`
[ ]	`inventar = []`
[ ]	`print('Betrete:', ort)`
[ ]	`inventar.append('Rostiger Schluessel')`
[ ]	`print('Gefunden:', 'Rostiger Schluessel')`
[ ]	`if 'Rostiger Schluessel' in inventar:`
[ ]	`print('Tuer erfolgreich aufgeschlossen!')`
[ ]	`else:`
[ ]	`inventar.remove('Rostiger Schluessel')`

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 Spielzyklus haben sich zwei typische Logikfehler eingeschlichen:
1. In Zeile 7 wird `hp` überschrieben, statt den Schaden abzuziehen.
2. In Zeile 12 wird die Schleife bei Erreichen von 0 HP nicht gestoppt (Endlosschleife droht).

Repariere beide Stellen, sodass der Schaden korrekt abgezogen und die Schleife bei Game Over beendet wird.

Fehlerhafter Ausgangs-Code:

```python
hp = 30
schaden = 10
aktiv = True

while aktiv:
    # Fehler 1: Wert wird überschrieben statt abgezogen
    hp = schaden
    print("Treffer! Aktuelle HP:", hp)

    # Fehler 2: Beendigung der Schleife fehlt
    if hp <= 0:
        print("Game Over!")
        # Setze aktiv hier auf den passenden Wahrheitswert!
```

Deine Korrektur / Notizen:

3. Programmier-Challenge

AUFGABE 3

Programmiere das Dungeon-Entscheidungsszenario nach diesen Vorgaben:

1. Lege die Variable `hp` mit dem Startwert `30` an.
2. Erstelle eine Liste `inventar`, die als einziges Element `'Rostiger Schluessel'` enthält.
3. Ziehe von `hp` den Fallenschaden `10` ab.
4. Prüfe mit einer `if` / `else`-Verzweigung:
   - Wenn `'Rostiger Schluessel'` in `inventar` liegt, gib aus:
     `Status: Kerker des Schattens gemeistert mit HP: <hp>` (wobei `<hp>` für den verbleibenden Wert steht).
   - Andernfalls gib aus:
     `Status: Verloren im Kerker des Schattens`

Dein Python-Code:

Transfer-Aufgabe: Game-Loop & Zustandslogik analysieren

AUFGABE 4

Betrachte die Hauptschleife eines textbasierten Spiels. Warum ist die exakte Reihenfolge von Zustandsänderung (z. B. Abzug von `hp`) und Zustandsprüfung (`if hp <= 0:`) entscheidend für den fehlerfreien Ablauf? Was passiert, wenn eine `while`-Schleife keine erreichbare Abbruchbedingung besitzt?