Codeklasse

Der Rennwagen-Konfigurator

Thema: Klassen & Objekte

Name: _____

Klasse: _____ Datum: _____

Grundlagen & Merkkasten

WISSEN

In der Programmierung strukturieren wir komplexe Daten über Klassen und Objekte.
Eine Klasse fungiert dabei wie eine Keks-Ausstechform oder ein architektonischer Bauplan:
Sie beschreibt die Eigenschaften, die jedes spätere Exemplar besitzt.
Aus einem einzigen Bauplan können beliebig viele konkrete Objekte (Instanzen) erzeugt werden,
die über die Punktnotation ihre eigenen, individuellen Attributwerte erhalten.

`class Auto:` `  pass`	Definiert eine neue Klasse (Bauplan). 'pass' dient als Platzhalter für leere Blöcke.
`wagen1 = Auto()`	Erzeugt eine neue, konkrete Instanz (Objekt) aus der Klasse Auto.
`wagen1.farbe = 'Rot'`	Punktnotation: Weist dem Objekt wagen1 das Attribut 'farbe' zu.
`print(wagen1.tempo)`	Liest das Attribut 'tempo' des Objekts wagen1 aus.

1. Code-Puzzle (Reihenfolge & Einrückung)

AUFGABE 1

Bringe die Code-Blöcke in die richtige Reihenfolge, um:
1. Die Klasse `Auto` zu definieren,
2. Ein Rennwagen-Objekt `racer` zu instanziieren,
3. Die Attribute `marke` und `tempo` zuzuweisen,
4. Die Daten auf der Konsole auszugeben.

[ ]	`class Auto:`
[ ]	`pass`
[ ]	`racer = Auto()`
[ ]	`racer.marke = 'TurboRacer'`
[ ]	`racer.tempo = 280`
[ ]	`print('Marke:', racer.marke)`
[ ]	`print('Tempo:', racer.tempo, 'km/h')`
[ ]	`Auto.marke = 'TurboRacer'`
[ ]	`racer = Auto`

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 Code haben sich zwei typische Anfängerfehler eingeschlichen:
1. Bei der Objekterzeugung fehlen die Funktions-Klammern `()`.
2. Ein Attributzugriff verwendet fälschlicherweise den Klassennamen statt des konkreten Objektnamens.

Korrigiere den Code, sodass die Ausgabe fehlerfrei gelingt!

Fehlerhafter Ausgangs-Code:

```
1  class Rennwagen:
2      pass
3
4  # 1. Objekt instanziieren (Achtung: Syntax prüfen!)
5  auto1 = Rennwagen
6  auto1.modell = "ApexGT"
7  auto1.farbe = "Neongelb"
8
9  # 2. Ausgabe korrigieren
10 print("Modell:", auto1.modell)
11 print("Farbe:", Rennwagen.farbe)
```

Deine Korrektur / Notizen:

3. Programmier-Challenge

AUFGABE 3

Baue deine eigene Rennwagen-Verwaltung mit Klassen auf:

1. Definiere eine Klasse `Auto` mit dem Platzhalter `pass`.
2. Erzeuge eine Instanz namens `car1` und weise folgende Attribute zu:
- `marke` = `'TurboRacer'`
- `farbe` = `'Rot'`
- `top_speed` = `280`
3. Erzeuge eine zweite Instanz namens `car2` und weise folgende Attribute zu:
- `marke` = `'PhantomEV'`
- `farbe` = `'Silber'`
- `top_speed` = `240`
4. Gib für beide Autos jeweils eine Zeile im Format aus:
`Auto 1: <marke> | <farbe> | <top_speed> km/h`
`Auto 2: <marke> | <farbe> | <top_speed> km/h`

Dein Python-Code:

Transfer-Aufgabe: Bauplan vs. Instanz

AUFGABE 4

Erkläre anhand der Keks-Ausstechform- oder Bauplan-Analogie den präzisen Unterschied zwischen einer Klasse und einem Objekt (Instanz).
Was geschieht im Speicher, wenn du den Befehl `auto1 = Auto()` und danach `auto2 = Auto()` ausführst?