

Informatik 2 für Regenerative Energien

Klausur vom 11. Juli 2025

Jörn Loviscach

Versionsstand: 11. Juli 2025, 09:03



This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Germany License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/de/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

15 Punkte für die erste Aufgabe; 3 Punkte für alle weiteren Aufgaben. Mindestpunktzahl zum Bestehen: 20 Punkte. Hilfsmittel: maximal vier einseitig oder zwei beidseitig beschriftete DIN-A4-Spickzettel beliebigen Inhalts, möglichst selbst verfasst oder zusammengestellt; Wörterbuch (z. B. Deutsch–Portugiesisch); kein Skript, keine anderen Texte, kein Taschenrechner, kein Computer (auch nicht wearable), kein Handy.

1. Im Python-Programmlisting im Anhang sind 15 Fehler, darunter keine Tippfehler und höchstens ein Fehler pro Zeile. Erstellen Sie eine Liste mit 15 Zeilen aus den Fehlern und ihren jeweiligen Korrekturen nach dem folgenden Muster:

Zeile	korrekter Programmtext
321	bla: int = 23
543	if self.blubb < 7:

2. Der (korrigierte) Code aus dem Programmlisting im Anhang wird ausgeführt. Was sind die drei ersten Zeilen an Ausgaben des Programms? Optional: Beschreiben Sie, wie Sie zu Ihrem Ergebnis kommen.
3. Die Methode `regle_raum()` der Klasse `Automation` soll eine Exception werfen, wenn noch gar kein Modus festgelegt ist. Was ändern Sie dazu wie an dem (korrigierten) Code aus dem Programmlisting?
4. Im korrigierten Code aus dem Programmlisting soll sich jeder `Sensor` alle seine bisherigen Messergebnisse merken. Geben Sie an, welche Änderungen dazu nötig sind.

5. Leiten Sie von der Klasse `Aktor` des korrigierten Code aus dem Programm-Listing eine Klasse `Aktorgruppe` ab und implementieren Sie diese sinnvoll. Der Konstruktor der `Aktorgruppe` nimmt eine Liste von `Aktor`-Instanzen entgegen. Alle diese sollen immer gleich eingestellt werden.
6. Klasse `Beleuchtung` des korrigierten Code soll so erweitert werden, dass man als eine Gesamtzahl abrufen kann, wie oft Instanzen davon auf einen Wert von über 90 gestellt worden sind.
7. Geben Sie so präzise wie möglich den Typ dieser Variablen an:

```
akademie = {
    'Alrik': {
        'Ränge': [('Lehrling', 2020), ('Novize', 2022)],
        'Zauber': {'Feuerball', 'Unsichtbarkeit'},
        'Besitztümer': {'Stab', 'Amulett'}
    },
    'Miriel': {
        'Ränge': [('Lehrling', 2021)],
        'Zauber': {'Heilung', 'Unsichtbarkeit'},
        'Besitztümer': {'Trank'}
    }
}
```

8. Was steht nach Ausführung dieses Python-Programmfragments in den Variablen `x`, `y` und `z`? Geben Sie möglichst auch Zwischenschritte an, damit Ihr Gedankengang nachvollziehbar ist.

```
städte = ['Berlin', 'Paris']
tage = [3, 4]
plan_a = {
    'Route': städte,
    'Dauer': tage
}
plan_b = {
    'Route': ['Rom', 'Madrid'],
    'Dauer': tage
}
übersicht = [plan_a, plan_b]
städte.append('Prag')
plan_a['Dauer'][1] += 2
plan_b['Dauer'][0] -= 1
x = übersicht[0]['Route'][-1]
y = sum(übersicht[1]['Dauer'])
z = 'Prag' in plan_b['Route']
```

Dieses Listing enthält 15 Fehler!

Dies ist eine Skizze für eine Gebäudeautomation, die zwei Größen (Helligkeit und Blendstärke) nach Vorgaben und Bedarf regelt. Der Code am Ende macht die Benutzung der Klassen vor.

Didaktische Anmerkung: Einige Stellen sind gegenüber einem pythonischen Stil vereinfacht.

```
1  from abc import ABC, abstractmethod
2  import time
3
4
5  class Sensor(ABC): 2 usages
6      def __init__(self, name):
7          self.name = name
8
9      @abstractmethod 1 usage (1 dynamic)
10     def messen(self):
11
12
13     class Helligkeitssensor(Sensor): 1 usage
14         def messen(self): 1 usage (1 dynamic)
15             return 30 # Simuliert einen Helligkeitwert zwischen 0 und 100.
16
17
18     # Misst, wie stark man durch einfallendes Sonnenlicht geblendet wird.
19     class Blendsensor(Sensor): 1 usage
20         def messen(self): 1 usage (1 dynamic)
21             return 90 # Simuliert einen Blendwert zwischen 0 und 100.
22
23
24     class Aktor: 2 usages
25         def __init__(self, name):
26             self.name = name
27             self.zustand = 23 # Simuliert einen Zustand zwischen 0 und 100.
28
29         def setze_zustand(self, wert):
30             self.zustand = self.wert
31             print(f'{self.name}: {self.zustand}')
32             # Im echten Programm Stellbefehl an Gerät senden.
33
34         def ändere_zustand(self, änderung): 3 usages (3 dynamic)
35             self.zustand + änderung = wert
36             if wert < 0:
37                 wert = 0
38             elif wert > 100:
39                 wert = 100
40             setze_zustand(wert)
41
42
```

```
43 class Beleuchtung(Aktor): 1 usage
44     pass
45
46
47 class Jalousie(Aktor): 1 usage
48     pass
49 # Beachte: Mehr Jalousie => weniger Blendstärke
50
51
52 class Raum(ABC): 1 usage
53     def __init__(self):
54         self.name = name
55         self.größen = None
56         self.modus = None
57
58     # invers==True bedeutet, dass ein höherer Wert für den Aktor
59     # einen niedrigeren Messwert des Sensors bewirkt.
60     def füge_größe_hinzu(self, größe, sensor, aktor, invers=False): 2 usages
61         self.größen.append( (größe, sensor, aktor, invers) )
62
63     def setze_modus(self, modus): 1 usage (1 dynamic)
64         self.modus = modus
65
66
67 class Automation: 1 usage
68     def setze_modus(self, raum, modus): 2 usages (1 dynamic)
69         raum.setze_modus(modus)
70
71     def regle_raum(self, raum): 1 usage
72         for element in raum.größen:
73             # Die folgende Zeile ist korrekt.
74             größe, sensor, aktor, invers = element
75
76             # Größe überspringen, wenn keine Regel dafür
77             if größe not in raum.modus:
78                 break
79
80             regel = raum.modus(größe)
81             strategie = regel[0]
82             aktueller_wert = sensor.messen
83
84             if strategie == 'proportional':
85                 diff = regel[1] - aktueller_wert
86                 if invers:
87                     diff = 1 - diff
88                 aktor.ändere_zustand(0.1 * diff)
89
90             elif strategie == 'tolerant':
91                 inkr = -1 if invers else 1
92                 if aktueller_wert > regel[2]:
93                     aktor.ändere_zustand(-inkr)
94                 elif aktueller_wert < regel[1]:
95                     aktor.ändere_zustand(-inkr)
```

```
96
97
98 mein_hörsaal = Raum(D2)
99 mein_hörsaal.füge_größe_hinzu('Helligkeit',
100                               Helligkeitssensor('HelligkeitD2'),
101                               Beleuchtung('LichtD2'))
102 mein_hörsaal.füge_größe_hinzu('Blendstärke',
103                               Blendsensor('BlendstärkeD2'),
104                               Jalousie('JalousieD2'),
105                               True)
106
107 klausurmodus = {
108     'Helligkeit': ('proportional', 70),
109     # Fahre die Jalousien nicht, wenn die Blendstärke zwischen 10 und 30 ist.
110     'Blendstärke': ('tolerant', 10, 30)
111 }
112
113 automation = Automation
114 automation.setze_modus(klausurmodus, mein_hörsaal)
115
116 while True:
117     automation.regle_raum(mein_hörsaal)
118     time.sleep(1) # Diese Zeile ist korrekt.
```