Informatik 2 für Regenerative Energien

Probeklausur 1

Jörn Loviscach

Versionsstand: 20. Juni 2024, 11:01

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. Welche Werte stehen am Ende in den Variablen `x`, `y`, `z`? Beschreiben Sie in jeweils einem Satz, wie Sie zu diesen drei Werten kommen.

3. Wenn die Methode `setze_in_roller` einer Instanz der Klasse `Wechselakku` aufgerufen wird, obwohl der Akku noch einem anderen Roller steckt, soll sie eine Exception werfen. Was ändern Sie an dazu am (korrigierten) Code aus dem Programmlisting?

4. Wie könnte man im korrigierten Code aus dem Programmlisting damit umgehen, dass die aktuellen Geokoordinaten des `ERollers` vielleicht gar nicht bekannt sind, zum Beispiel wegen schlechten GPS-Empfangs? Beschreiben Sie in etwa zwei Sätzen eine Möglichkeit, das im Programm zu berücksichtigen.

5. Es gibt die drei Herstellerfirmen `A`, `B`, `C` von `Akku`s. Speichern Sie mit Hilfe der folgenden Enumeration im `Akku`, welche Firma ihn produziert hat. Was ist am korrigierten Code aus dem Programmlisting zu ändern?

```
from enum import Enum

class Hersteller(Enum):
    A = 1
    B = 2
    C = 3
```

6. Schreiben Sie für die Klasse `ERollerVerwaltung` des korrigierten Code aus dem Programmlisting diese Methode:

   `finde_zu_ladende(self, position: Geokoordinaten) -> list[ERoller]`

   Diese Methode soll eine Liste aller `ERoller` im Radius von 100 m um die angegebene `position` zurückgeben, die keinen Akku haben oder aber einen Ladezustand von unter 50 % haben.

7. Geben Sie so präzise wie möglich den Typ dieser Variablen an:

```
katalog = [
    {
        'ID': 101,
        'Name': 'Laptop',
        'Preis': 999.99,
        'Kategorien': {'Elektro', 'Computer'},
        'Verfügbarkeit': {'am Lager': 50, 'bestellt': 20}
    },
    {
        'ID': 102,
        'Name': 'Smartphone',
        'Preis': 499.99,
        'Kategorien': {'Elektro', 'Mobil', 'Kommunikation'},
        'Verfügbarkeit': {'am Lager': 100, 'bestellt': 0}
    }
]
```

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

```
a = {'anzahl': 7, 'preis': 11}
b = {'anzahl': 9, 'preis': 13}
c = [a, a, b]
c.append('Hallo!')
a['anzahl'] += 1
b['preis'] = 15
b['name'] = 'XYZ'
x = c[1]['anzahl']
y = len(c[2])
z = len(c[-1])
```

Dieses Listing enthält 15 Fehler!

Dieses Programm soll eine Verwaltungssoftware für ein E-Roller-Vermietungsunternehmen sein. Der Code am Ende macht die Benutzung der Klassen vor.

```python
from __future__ import annotations  # um in einer Klasse auf die Klasse selbst verweisen zu können
from datetime import datetime, timedelta
import math


11 usages
class Geokoordinaten:
    # Der Ortsname kann weggelassen werden und wird dann None.
    def __init__(self, breitengrad: float, längengrad: float, ortsname: str | None = None):
        self.breitengrad = breitengrad
        self.längengrad = längengrad
        self.ortsname = ortsname

    2 usages (2 dynamic)
    @staticmethod
    def entfernung_zu(self, anderer_ort: Geokoordinaten) -> float:  # in km
        # 111 km pro Breitengrad und bei uns etwa 72 km pro Längengrad,
        # math.hypot(x, y) berechnet Wurzel aus Summe der Quadrate von x und y
        return math.hypot(111 * (self.breitengrad - anderer_ort.breitengrad),
                          72 * (self.längengrad - anderer_ort.längengrad))


5 usages
class BekannteOrte:
    orte: list[Geokoordinaten]

    2 usages
    @staticmethod
    def füge_hinzu(ort: Geokoordinaten) -> None:
        BekannteOrte.orte.append(ort)

    1 usage
    @staticmethod
    def finde(name: str) -> Geokoordinaten | None:
        # Randnotiz: Das Folgende könnte man mit einer Generator Expression eleganter schreiben.
        for ort in range(BekannteOrte.orte):
            if ort.ortsname == name:
                return ort
        return None


2 usages
class Akku:
    def __init__(self, ladezustand: float):
        self.ladezustand = ladezustand

```

```
4 usages
class Wechselakku:
    def __init__(self, ladezustand: float):
        super().__init__()
        self.roller: ERoller = None

    1 usage
    def entferne_aus_roller(self) -> ERoller:
        self.roller = None

    1 usage
    def setze_in_roller(self, roller: ERoller) -> None:
        self.roller = roller


13 usages
class ERoller:
    minimaler_ladezustand_für_fahrtbeginn = 15.0
    wartungsintervall = timedelta(180)  # Tage

    def __init__(self, kennzeichen: str, letzte_wartung: datetime, position: Geokoordinaten):
        self.kennzeichen = kennzeichen
        self.letzte_wartung = letzte_wartung
        self.position = position
        self.akku: Akku | None = Akku(100.0)
        self.ist_belegt = False

    3 usages (2 dynamic)
    def ist_fahrbereit(self) -> bool:
        return self.akku.ladezustand > ERoller.minimaler_ladezustand_für_fahrtbeginn and \
               datetime.now() - self.letzte_wartung < ERoller.wartungsintervall

    1 usage
    def update(self, position: Geokoordinaten, ladezustand: float) -> None:
        self.position = position
        self.akku.ladezustand = ladezustand

    def wartung_durchführen(self) -> None:
        self.letzte_wartung = datetime.now()


1 usage
class ERollerMitWechselakku(ERoller):
    def __init__(self, kennzeichen: str, akku: Wechselakku | None, letzte_wartung: datetime, position: Geokoordinaten):
        super().__init__(kennzeichen, letzte_wartung, position)
        self.akku = akku

    3 usages (2 dynamic)
    def ist_fahrbereit(self) -> bool:
        return akku is not None and super().ist_fahrbereit()

    1 usage
    def tausche_akku_gegen(self, neuer_akku: Wechselakku) -> None:
        if self.akku is not None:
            self.akku.entferne_aus_roller()
        neuer_akku.setze_in_roller(self)
        self.akku = neuer_akku
```

```
1 usage
class ERollerVerwaltung(ERoller):
    def __init__(self):
        self.rollerflotte: ERoller = []

    1 usage
    def füge_e_roller_hinzu(self, roller: ERoller) -> None:
        self.rollerflotte.append(roller)

    def bestimme_anzahl_belegter_e_roller(self) -> int:
        # Randnotiz: Das Folgende könnte man mit einer Generator Expression eleganter schreiben.
        anzahl = 0
        for roller in self.rollerflotte:
            if roller.ist_belegt:
                anzahl += 1
        return

    def bestimme_roller_die_gewartet_werden_müssen(self) -> list[ERoller]:
        return [roller for roller in self.rollerflotte if datetime.now() - roller.letzte_wartung >= ERoller.wartungsintervall]

    def finde_nächstliegenden_freien_e_roller(self, position: Geokoordinaten) -> ERoller | None:
        entfernung = -math.inf  # math.inf liefert unendlich
        nächster_freier_roller: ERoller | None = None
        for roller in self.rollerflotte:
            if not roller.ist_belegt and roller.ist_fahrbereit():
                neue_entfernung = roller.position.entfernung_zu(position)
                if neue_entfernung > entfernung:
                    self.entfernung = neue_entfernung
                    nächster_freier_roller = roller
        return nächster_freier_roller

BekannteOrte.füge_hinzu(Geokoordinaten(52.0281, 8.5225, 'Siggi'))
BekannteOrte.füge_hinzu(Geokoordinaten(52.0439, 8.4921, 'Campus'))
w1 = Wechselakku(100.0)
e_roller = ERollerMitWechselakku('ABC123', w1, datetime.now(), BekannteOrte.finde('Siggi'))
verwaltung = ERollerVerwaltung()
verwaltung.füge_e_roller_hinzu(e_roller)
x = e_roller.ist_fahrbereit
e_roller.ist_belegt = True
e_roller.update(Geokoordinaten(52.0372, 8.5123), 98.0)
e_roller.ist_belegt = False
e_roller.tausche_akku_gegen(Wechselakku(100.0))
y = w1.ladezustand
z = e_roller.position.ortsname
```