Einführung in die Programmierung

Prof. Dr. Franziska Matthäus
Prof. Dr. Matthias Kaschube
Dr. Karsten Tolle
Kenan Khauto

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Übungsblatt ÜE-05

Ausgabe: 20.11.2024
Abgabe: 30.11.2024
13:00 Uhr

Inventarisierungssystem

Hinweis:

- *Es dürfen keine Lösungen aus dem Skript, dem Internet oder anderen Quellen abgeschrieben werden. Diese Quellen dürfen nur mit Quellenangaben verwendet werden und es muss ein hinreichend großer Eigenanteil in den Lösungen deutlich zu erkennen sein.*
- *Digitale Abgaben, die nicht im Format **.pdf** oder **.txt** für Texte oder **.py** für Code erfolgen, werden nicht bewertet. Bei Abgaben mehrerer Dateien müssen diese als **.zip** zusammengefasst werden.*
- *Achten Sie darauf die Variable **__author__** in allen Quellcode Dateien (.py) korrekt zu setzen (am Anfang des Quellcodes):* __author__ = "<Matr-Nr>, <Nachname>"
  Beispiel: __author__ = "1234567, Tolle"
  … Leerstellen vor und nach dem „=" und Leerstelle nach dem Komma beachten, sowie keine spitzen Klammern verwenden, die dienen bei der Schreibweise oben zum Definieren von Variablen.
- *Außerdem muss Ihr Name in jeder abgegebenen **.pdf** und **.txt** Datei zu finden sein.*
- *Abgaben der Dokumentation, die per Hand geschrieben und eingescannt werden, sind nur in zuvor abgesprochenen Ausnahmefällen erlaubt.*

Aktualisierung der Hinweise:

- *Datei- und Ordnernamen sollen keine Umlaute, diakritische oder Sonderzeichen, mit Ausnahme des Unterstrichs in der Namesmitte, enthalten! (Insb. MAC Nutzer sollten aufpassen, dass nicht zusätzliche Dateien im ZIP-Ordner hinzugefügt wurden! … diese ggf. löschen!)*
- *Beim Programmieren und Kommentieren halten Sie sich bitte an die Regeln im Programmierhandbuch, siehe Moodle-Kurs (Programmierhandbuch WiSe 24/25 (Style Guide)). Im Zweifelsfall gilt PEP 8.*
- *Bitte auch darauf achten, dass Ihre .py Dateien lauffähig sind. Es ist den Tutoren nicht zuzumuten, Abgaben zu Debuggen. Problematische Teile, z.B. wenn diese nicht wie gewünscht funktionieren, können auskommentiert und mit Kommentaren versehen werden.*

∑ 10 Punkte

Aufgabe ÜE-05-1: Inventarisierungssystem (10 Punkte)

Entwickeln Sie ein Python-Programm, das dabei hilft, Gegenstände zu organisieren und zu verwalten, die an verschiedenen Orten wie einem Keller, einem Schrank oder anderen Lagerplätzen aufbewahrt werden. Diese Lagerorte sowie die darin enthaltenen Gegenstände sollen mit **Dictionaries** dargestellt werden. Gegenstände, die in Kisten oder Behältern aufbewahrt werden, sollen ebenfalls durch **verschachtelte Dictionaries** abgebildet werden, um die hierarchische Struktur der Lagerung zu berücksichtigen.
Das Programm soll die folgenden Funktionalitäten umfassen:

1. **Hinzufügen von Gegenständen**
   Erstellen Sie eine Funktion, um neue Gegenstände an einen bestimmten Lagerort hinzuzufügen. Falls der Lagerort bereits existiert, sollen die Inhalte entsprechend aktualisiert werden. Falls er nicht existiert, soll er neu erstellt werden.
   Das Bild soll nur illustrieren, was die Funktion machen sollte. Es handelt sich nicht um ein Programm. Auch sind die gezeigten Eigenschaften wie „Zustand“, „Model“ nicht vorgegeben. Entscheiden Sie selbst und erläutern Sie dies in Ihrer Dokumentation entsprechend.

```
Ursprünglich:
{}
Nach dem Aufruf der Funktion:
{
    "keller": {
        "Hammer": {
            "Anzahl": 5,
            "Zustand": "neu"
        },
        "Bett": {
            "Anzahl": 2,
            "Zustand": "alt"
        }
    },
    "garage": {
        "Auto": {
            "Anzahl": 1,
            "Zustand": "neu",
            "Model": "Porsche"
        },
        "garageitem": {
            "something": "value1"
        }
    }
}
```

2. **Auflistung aller Gegenstände**
   Entwickeln Sie eine Funktion, die alle Gegenstände in allen Lagerorten alphabetisch auflistet.
   Hier Lagerort: Keller und Garage

```
['Auto', 'Bett', 'Hammer', 'garageitem']
```

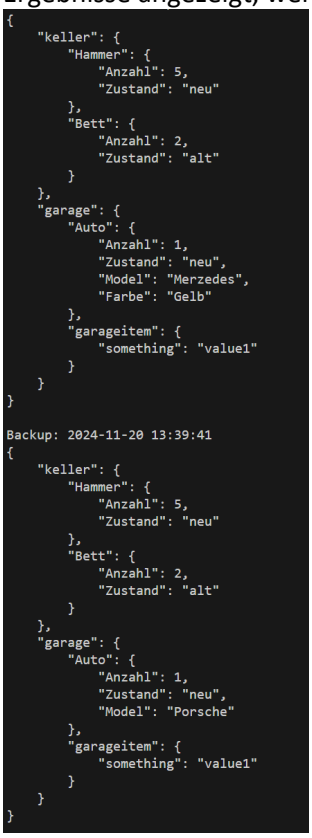
3. **Suche nach Gegenständen**
   Schreiben Sie eine Funktion, mit der Sie nach einem bestimmten Gegenstand suchen können. Die Funktion soll angeben, in welchem Lagerort (oder auch Lagerorten, man kann ja auch gleiche Dinge an verschiedenen Orten lagern) oder Behälter sich der Gegenstand befindet.
   Beispiel: Suche nach Bett: `(True, {'keller': {'Anzahl': 2, 'Zustand': 'alt'}})`

4. **Zusammenfassung nach Lagerort**
   Fügen Sie eine Funktion hinzu, die eine Übersicht über die in einem oder mehreren angegebenen Lagerorten enthaltenen Gegenstände erstellt. Falls ein Lagerort nicht existiert, soll dies entsprechend angezeigt werden.
   Beispiel: Zusammenfassung von: keller, garage, bedroom

```
{'keller': ['Hammer', 'Bett'], 'garage': ['Auto', 'garageitem'], 'bedroom': 'Location not found in inventory'}
```

5. **Aktualisierung von Gegenständen**
   Entwickeln Sie eine Funktion, die es ermöglicht, die Eigenschaften eines Gegenstands zu aktualisieren. Demonstrieren Sie in diesem Zusammenhang, wie Änderungen in einem verschachtelten Dictionary (z. B. durch flache Kopien) auf das ursprüngliche Dictionary wirken können.
6. **Erstellung/Anzeigen eines Backups**
   Schreiben Sie eine Funktion, die ein vollständiges Backup des Inventars erstellt (hier sollte man Deep Copy benutzen). Dieses Backup soll unabhängig vom ursprünglichen Inventar sein und zu einem späteren Zeitpunkt genutzt werden können (einfach gespeichert in einer anderen Variable, das Backup muss nicht in das Filesystem ausgelagert werden). Erfassen Sie dabei den Zeitpunkt der Backuperstellung (hier kann man **datetime.now()** nutzen). Über eine weitere Funktion soll der aktuelle Zustand zusammen mit dem Backup angezeigt werden können (s.u.).

   Beispiel: Nach Erstellung des Backups wird ein Gegenstand aktualisiert. Im Bild werden die Ergebnisse angezeigt, wenn man sie ausgibt.



```
{
    "keller": {
        "Hammer": {
            "Anzahl": 5,
            "Zustand": "neu"
        },
        "Bett": {
            "Anzahl": 2,
            "Zustand": "alt"
        }
    },
    "garage": {
        "Auto": {
            "Anzahl": 1,
            "Zustand": "neu",
            "Model": "Merzedes",
            "Farbe": "Gelb"
        },
        "garageitem": {
            "something": "value1"
        }
    }
}

Backup: 2024-11-20 13:39:41
{
    "keller": {
        "Hammer": {
            "Anzahl": 5,
            "Zustand": "neu"
        },
        "Bett": {
            "Anzahl": 2,
            "Zustand": "alt"
        }
    },
    "garage": {
        "Auto": {
            "Anzahl": 1,
            "Zustand": "neu",
            "Model": "Porsche"
        },
        "garageitem": {
            "something": "value1"
        }
    }
}
```

Zusätzliche Hinweise:

- Lösungen sind eigenständig zu erstellen, ohne Zuhilfenahme von generativer KI (z. B. ChatGPT). In der Klausur wird dieses Hilfsmittel ebenfalls nicht zur Verfügung stehen.
- Eine Tutorin/Tutor oder die Dozentin/der Dozent kann ein Code-Review verlangen, bei dem Sie Ihren Code erklären und rechtfertigen müssen.
- Rekursive Ansätze zur Lösung der Aufgabe sind nicht erlaubt!
- **Docstrings und Testfälle nicht vergessen!**