Arbeitsmaterial

Algorithmen und Datenstrukturen

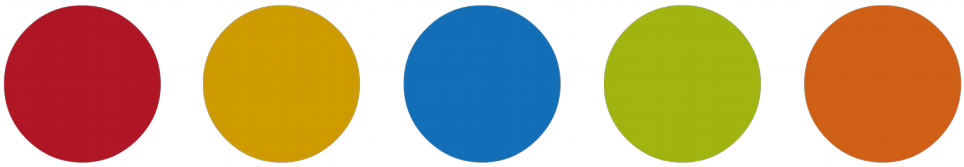
in Python II

Skript

Materialsammlung

Schulung:	Informatik und Wirtschaftsinformatik

Stand: 15. Sep 2020



Urquelle für die Aufgaben und Lösungen ist der Landesbildungsserver BW

überarbeitet und ergänzt:

Christine Janischek
Berufliches Schulzentrum Wangen

Inhaltsverzeichnis

1 Datenstrukturen L1 1

Datenstrukturen L1 1

Thema:	Datenstrukturen und Algorithmen in Python II– Arbeitsauftrag Quelle: L1 1 Datenstrukturen

Hinweis:

Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen das Informationsmaterial L1_1 Information_Datenstrukturen.docx.

1 Formulieren Sie einen Satz, der den Begriff 'Datenstruktur' definiert.

2 Beschreiben Sie die Merkmale folgender Datenstrukturen (Speicherstrukturen):

- 2.1 Array
- 2.2 Verkettete Liste
- 2.3 Stapelspeicher
- 2.4 Warteschlange
- 2.5 Baum

3 Begründen Sie für die folgenden Sachverhalte, welche Datenstruktur jeweils zu wählen ist.

3.1 Bei einem physikalischen Experiment zum Thema 'Freier Fall' werden Metallmuttern in bestimmten Abständen an eine Schnur gebunden, deren unteres Ende den Boden berührt. Die Schnur mit den Metallmuttern wird fallen gelassen und die jeweilige Zeit des Auftreffens der Muttern auf den Boden gemessen und erfasst.

Die Muttern befinden sich im ersten Versuch in den Abständen 5cm, 10cm, 20cm, 40cm, 80cm, 160cm (gemessen vom Boden).

Im zweiten Versuch in den Abständen 5cm, 10cm, 20cm, 40cm, 45cm, 80cm, 125cm, 160cm und 180cm.

Im dritten Versuch in den Abständen 5cm, 20cm, 45cm, 80cm, 125cm und 180cm.

3.2 Das Ergebnis der Ziehung der Lottozahlen soll digital erfasst werden. Die Ziehung der Superzahl soll dabei nicht berücksichtigt werden.

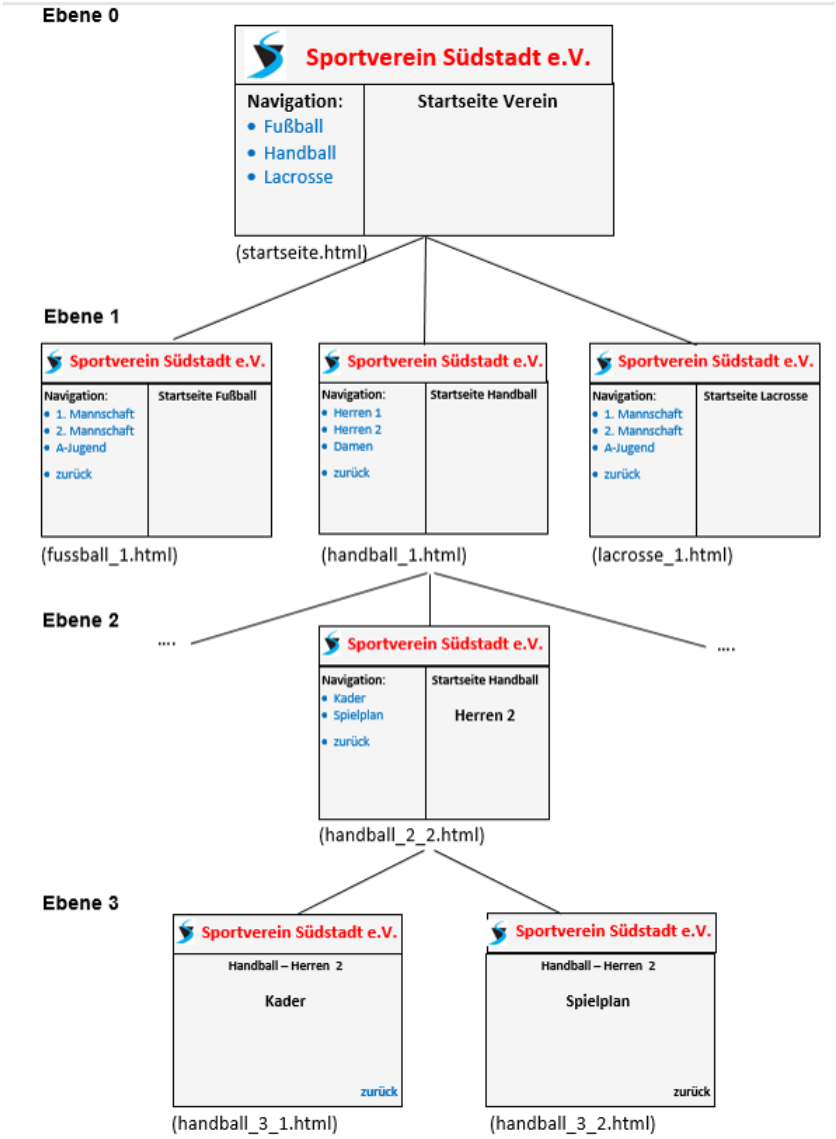
3.3 Der Sportverein Südstadt e.V. plant einen neuen Internetauftritt. Die Struktur des Seitenaufbaus finden Sie in der Anlage (Folgeseite).

Die einzelnen Seiten können mit den Navigationslinks aufgerufen werden. Mit Hilfe der zurück-Links sollen die jeweils zuvor aufgerufenen Seiten anwählbar sein. Dazu müssen die jeweiligen Seitennamen (startseite.html, fussbal_1.html etc.) gespeichert werden.

3.4 Der Sportverein Südstadt e.V. veranstaltet ein Fußballturnier mit acht beteiligten Mannschaften. Das Turnier soll im K.O.-Modus stattfinden. Der Spiel plan des Turniers soll digital erfasst werden.

3.5 Zur Kontrolle der Verkehrssicherheit im Elbtunnel (Hamburg) sollen die Kenn zeichen aller Fahrzeuge, die in den Tunnel einfahren, digital gespeichert wer den. Nach der Ausfahrt aus dem Tunnel werden die gespeicherten Kennzeichen sofort wieder gelöscht. Es wird unterstellt, dass der Tunnel einspurig ist und ein Überholverbot gilt.

Anlage: Struktur der Internetpräsenz des Sportvereins Südstadt e.V.



2 Die Datenstruktur Array L1 2

Datenstrukturen L1 1

Thema:	Datenstrukturen und Algorithmen in Python II– Arbeitsauftrag Quelle: L1 2 Array

L1 2 Aufgabe Array

Aufgaben:

1 Formulieren Sie den Code, der die nachfolgend beschriebenen Anweisungen ausführt.

1.1 Erzeugen Sie ein Array mit dem Namen weltmeister und den Werten 1954, 1974, 1990 und 2014.

1.2 Erzeugen Sie ein Array wmtrainer und den Werten „Sepp Herberger", „Helmut Schön" und „Franz Beckenbauer":

1.3 Fügen Sie in die Liste wmtrainer den Namen „Joachim Löw" am Ende ein:

1.4 Im Array weltmeister sind die Jahre gespeichert, in denen die deutsche Fußball-Nationalmannschaft Weltmeister geworden ist. Die Anzahl der Weltmeistertitel soll in der Konsole ausgegeben werden.

1.5 Alle im Array wmtrainer gespeicherten Namen sollen in der Konsole angezeigt werden.

LÖSUNG:

2 Das Array a = [1, 5, 20, 9, 4, 3, 1] ist gegeben und wird durch den folgenden Algorithmus verändert. Durchlaufen Sie den Algorithmus mit dem gegebenen Array und nennen Sie den Arrayinhalt nach jeder Anweisung. (Diese Aufgabe kann hier auf Papier gelöst werden.)

Algorithmus	Array
a = [1, 5, 20, 9, 4, 3, 1]	[1, 5, 20, 9, 4, 3, 1]
a[1] = a[2]	
a[5] = a[5] + a[6]	
a[6] = 4	
a.append(4)	
i = 3	
a[i] = i	

LÖSUNG:

Zum Umwandeln:

`array` — Efficient arrays of numeric values

This module defines an object type which can compactly represent an array of basic values: characters, integers, floating point numbers. Arrays are sequence types and behave very much like lists, except that the type of objects stored in them is constrained. The type is specified at object creation time by using a *type code*, which is a single character. The following type codes are defined:

Type code	C Type	Python Type	Minimum size in bytes	Notes
`'b'`	signed char	int	1	
`'B'`	unsigned char	int	1	
`'u'`	Py_UNICODE	Unicode character	2	(1)
`'h'`	signed short	int	2	
`'H'`	unsigned short	int	2	
`'i'`	signed int	int	2	
`'I'`	unsigned int	int	2	
`'l'`	signed long	int	4	
`'L'`	unsigned long	int	4	
`'q'`	signed long long	int	8	
`'Q'`	unsigned long long	int	8	
`'f'`	float	float	4	
`'d'`	double	float	8	

3 Implementierung von Arrays L1 3

Implementierung von Arrays L1 3

Thema:	Datenstrukturen und Algorithmen in Python II– Arbeitsauftrag Quelle: L1 3 1 1 Arbeitsauftrag Vereinsmeisterschaften

Ausgabe von Arrays – Vereinsmeisterschaften

Der Mühlberger SC ist ein Sportverein mit mehreren Abteilungen. Im Jahr 2016 eröffnete der Mühlberger SC die Abteilung DART. Seit dem Jahr 2017 finden auch Vereinsmeisterschaften statt.

Gegeben:

```
platzierungen = ["Matthias Schmitt", "Felix Holzmann", "Sabrina Eggers",
"Sebastian Wolf", "Niklas Eisenbaum", "Florian Kuster", "Jan Ackerman",
"Erika Ebersbacher"]
```

Hinweis:

Beachten Sie zur Bearbeitung der nachfolgenden Problemstellung das Informationsmaterial L1_2 Information Array.docx.

(I) Problemstellung

Die Vereinsmeisterschaft in diesem Jahr war ein großer Erfolg. Die Spieler lieferten sich einen spannenden Wettkampf vor vielen Zuschauern. Der Mühlberger SC möchte die Ergebnisse des Wettkampfs veröffentlichen. Nutzer sollen die Möglichkeit haben, sich an einem Computer eine Platzierung ausgeben zu lassen. Möchte der Nutzer den Erstplatzierten ausgegeben bekommen, muss er bspw. eine „1“ eingeben. Außerdem soll die Teilnehmerzahl angezeigt werden. Die Teilnehmer an der Meisterschaft sind in dem Array platzierungen in der Reihenfolge ihrer Platzierung gespeichert.

Speichern Sie Ihre Lösung in Ihrem Ergebnisordner unter dem Namen

L1_3_1_1_vereinsmeisterschaften.py.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

(3) Welche Eigenschaften haben die Eingabe-, Verarbeitungs- und Ausgabedaten? (Variablenliste)

Bedeutung	Typ/Struktur	Variable/Größe

(4) Gewünschter Ablauf des Programms mit Beispieldaten:

Eingabe	`Welche Platzierung soll ausgegeben werden? 2`
Ausgabe	`Platzierung 2:  Felix Holzmann` `Teilnehmerzahl: 8`

(III) Struktogramm

(IV) Programmcode (Python-Code)

Thema:	Datenstrukturen und Algorithmen in Python II– Arbeitsauftrag Quelle: L1 3 1 2 Arbeitsauftrag Volleyball – Spieler anzeigen

L1_3.1.2 Volleyball: Spieler anzeigen

(I) Problemstellung

Die Abteilung Volleyball des Sportvereines Mühlbeger SC besteht zurzeit aus 12 Spielern. Die Namen der Stammspieler sind in dem Array *spieler* in der Reihenfolge ihrer Startpositionen erfasst.

```
spieler = ["Armin", "Batu", "Kai", "Sven", "Paul", "Milan"];
```

Die Ersatzspieler sind im Array *ersatz* in alphabetischer Reihenfolge erfasst.

```
ersatz = ["Chris", "Dennis", "Emin", "Goran", "Luca", "Nico"];
```

Der Trainer der Mannschaft möchte eine Software, mit deren Hilfe er sich die Namen der Stammspieler, die der Ersatzspieler und die des gesamten Kaders anzeigen lassen kann. Der Co-Trainier hat bereits mit der Implementierung der Eingabe begonnen (siehe Vorlage). Es steht nur noch die Implementierung der Funktionen aus.

Implementieren Sie

- eine Funktion `zeige_startaufstellung()`, mit der die Namen der Stammspieler
- eine Funktion `zeige_ersatzspieler()`, mit der die Namen der Ersatzspieler
- eine Funktion `zeige_kader()`, mit der die Namen aller Spieler im Kader der Mannschaft

in der Konsole angezeigt werden (siehe Folgeseite: (4) Gewünschter Ablauf des Programms).

> Hinweis: Die Inhalte mehrerer Arrays lassen mit folgender Anweisung einfach zu einem Array zusammenfügen:
>
> ```
> arrayNeu = array1 + array2
> ```

Optional:

Verwenden Sie für die Implementierung Ihrer Lösung die Datei *L1_3_1_2_Vorlage_volleyball_spieler_anzeigen.py*, die Ihnen im Ordner *Vorlagen* in digitaler Form vorliegt.

Speichern Sie Ihre Lösung in Ihrem Ergebnisordner unter dem Namen *L1_3_1_2_volleyball_spieler_anzeigen.py*.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

(3) Welche Eigenschaften haben die Eingabe-, Verarbeitungs- und Ausgabedaten?
(**Variablenliste**)

Bedeutung	**Typ/Struktur**	**Variable/Größe**

(4) Gewünschter Ablauf des Programms:

Eingabe	(1) Startaufstellung anzeigen (2) Ersatzspieler anzeigen (3) Kader anzeigen		
	Anzeigewunsch (1-3): 1	Anzeigewunsch (1-3): 2	Anzeigewunsch (1-3): 3
Ausgabe	---------- Startaufstellung ---------- Armin Batu Kai Sven Paul Milan	---------- Ersatzspieler ---------- Chris Dennis Emin Goran Luca Nico	---------- Kader ---------- Armin Batu Kai Sven Paul Milan Chris Dennis Emin Goran Luca Nico

(III) Struktogramm

Die Arrays *spieler* und *ersatz* sind bereits implementiert!

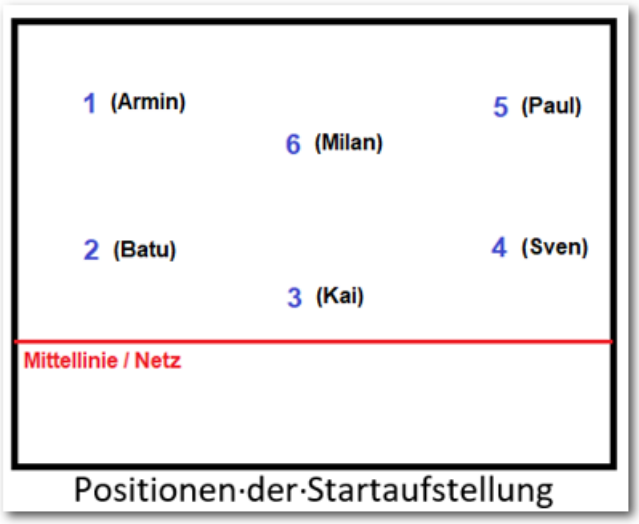
(IV) Programmcode (Python-Code)

Thema:	Datenstrukturen und Algorithmen in Python II – Arbeitsauftrag Quelle: L1 3 4 Arbeitsauftrag Volleyball Positionen tauschen

L1_3.4 Volleyball: Spielerpositionen tauschen – Index

(I) Problemstellung

Der Trainer der Abteilung Volleyball des Sportvereines Mühlberger SC möchte eine Erweiterung seiner Software.
Die Namen der Stammspieler sind in dem Array spieler in der Reihenfolge ihrer Startpositionen erfasst.
spieler = ["Armin", "Batu", "Kai", "Sven", "Paul", "Milan"]



Positionen·der·Startaufstellung

Beachten Sie:
Die Startposition 1 erhält den Index 0 des Arrays.

Die Software soll es ermöglichen, Spielerpositionen in der Startaufstellung zu tauschen. Nach der Eingabe von zwei Spielerpositionen sollen die entsprechenden Spielernamen im Array spieler getauscht und anschließend das Array mit der neuen Startaufstellung angezeigt werden. (Die Funktion zeige_startaufstellung() ist bereits implementiert!)

Z.B.: Tausch der Spielerpositionen 2 (Batu) und 4 (Sven).
Das Array spieler soll danach folgenden Inhalt haben:
spieler = ["Armin", "Sven", "Kai", "Batu", "Paul", "Milan"]

Implementieren Sie eine Funktion mannschaft_umstellen_pos(), die das beschriebene Problem löst.

Optional:
Verwenden Sie für die Implementierung Ihrer Lösung die Datei L1_3_4_vorlage_volleyball_positionen_tauschen_index.py, die Ihnen im Ordner Aufgaben/Vorlagen in digitaler Form vorliegt.

Speichern Sie Ihre Lösung in Ihrem Ergebnisordner unter dem Namen L1_3_4_volleyball_positionen_tauschen_index.html.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

(3) Welche Eigenschaften haben die Eingabe-, Verarbeitungs- und Ausgabedaten?
(Variablenliste)

Bedeutung	Typ/Struktur	Variable/Größe

(4) Gewünschter Ablauf des Programms mit Beispieldaten:

Funktion mannschaft_umstellen_pos()	
Eingabe	Startaufstellung umstellen Tausche Position: 2 mit Position: 4
Ausgabe	---------- Neue Startaufstellung ---------- Armin Sven Kai Batu Paul Milan

(5) Verarbeitung

(III) Struktogramm

Das Array spieler und die Funktion zeige_startaufstellung() sind bereits implementiert!

(IV) Programmcode (Python-Code)

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L1 3 5 Arbeitsauftrag Volleyball Inhalte einfügen

L1_3.5 Volleyball – Inhalte einfügen

Hinweis: Beachten Sie zur Bearbeitung der nachfolgenden Problemtellung das Informationsmaterial

L1_3.5 Information Elemente einfügen.docx

(I) Problemstellung

Die Software des Trainers der Abteilung Volleyball des Sportvereines Mühlberger SC enthält ein Array mit allen Spielernamen des Mannschaftskaders.

```
kader =     ["Armin", "Batu", "Kai", "Sven", "Paul",
             "Milan", "Goran", "Chris", "Nico",
             "Dennis", "Emin", "Luca"]
```

Mit Hilfe der Software soll es möglich sein, an einer bestimmten Stelle einen weiteren Spielernamen einzutragen, z.B. an der Stelle mit dem Index 6.

Implementieren Sie eine Funktion einfuegen_spieler(), mit der ein Spieler an einer vom Anwender einzugebenden Stelle, in das Array eingefügt wird.

z. B.: Spielername: Tom - Index: 6

Das Array kader soll danach folgenden Inhalt haben:

["Armin", "Batu", "Kai", "Sven", "Paul", "Milan", "Tom", "Goran", "Chris", "Nico",

"Dennis", "Emin", "Luca"]

Verwenden Sie für die Implementierung Ihrer Lösung die Datei L1_3_5_vorlage_volleyball_spieler

_einfuegen.py, die Ihnen im Ordner Aufgaben/Vorlagen in digitaler Form vorliegt.

Speichern Sie Ihre Lösung in Ihrem Ergebnisordner unter dem Namen L1_3_5_volleyball_spieler_einfuegen.py.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

(3) Welche Eigenschaften haben die Eingabe-, Verarbeitungs- und Ausgabedaten? (Variablenliste)

Bedeutung	**Typ/Struktur**	**Variable/Größe**

(4) Gewünschter Ablauf des Programms mit Beispieldaten:

Funktion **einfuegen_spieler()**	
Eingabe	Spielername: Tom Index: 6
Ausgabe	------------------------ Kader ------------------------ Armin Batu Kai Sven Paul Milan Tom Goran Chris Nico Dennis Emin Luca

(5) Verarbeitung

(III) Struktogramm

Das Array kader und die Funktion zeige_kader() sind bereits implementiert!

(IV) Programmcode (Python-Code)

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L1 3 6 Arbeitsauftrag Volleyball Inhalte entfernen

L1_3.6 Volleyball – Inhalte entfernen

Hinweis:

Beachten Sie zur Bearbeitung der nachfolgenden Problemstellung das Informationsmaterial L1_3.6 Information_Elemente_entfernen.docx

(I) Problemstellung

Die Software des Trainers der Abteilung Volleyball des Sportvereines Mühlberger SC enthält ein Array mit allen Spielernamen des Mannschaftskaders.

```
kader =     ["Armin", "Batu", "Kai", "Sven", "Paul", "Milan",
            "Goran", "Chris", "Nico",  "Dennis", "Emin",
            "Luca"];
```

Mit Hilfe der Software soll es möglich sein, den Spieler an einer bestimmten Stelle des Arrays zu entfernen.

z. B.: Spieler entfernen an der Stelle mit dem Index: 4

Das Array kader soll danach folgenden Inhalt haben:

kader = ["Armin", "Batu", "Kai", "Sven", "Milan", "Goran", "Chris", "Nico",
"Dennis", "Emin", "Luca"]

Implementieren Sie eine Funktion entferne_spieler(), mit der ein Spieler aus dem Array entfernt wird.

Verwenden Sie für die Implementierung Ihrer Lösung die Datei L1_3_6_vorlage_volleyball_spieler_entfernen.py, die Ihnen im Ordner Aufgaben/Vorlagen in digitaler Form vorliegt.

Speichern Sie Ihre Lösung in Ihrem Ergebnisordner unter dem Namen L1_3_6_volleyball_spieler_entfernen.py.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

(3) Welche Eigenschaften haben die Eingabe-, Verarbeitungs- und Ausgabedaten? (Variablenliste)

Bedeutung	**Typ/Struktur**	**Variable/Größe**

(4) Gewünschter Ablauf des Programms mit Beispieldaten:

Funktion entferne_spieler()	
Eingabe	Entferne Spieler an der Stelle mit dem Index: 4
Ausgabe	---------- Kader ---------- Armin Batu Kai Sven Milan Goran Chris Nico Dennis Emin Luca

(5) Verarbeitung

(III) Struktogramm

Das Array kader und die Funktion zeige_kader() sind bereits implementiert!

(IV) Programmcode (Python-Code)

4 Grundlagen Algorithmik L2 1

Grundlagen Algorithmik L2 1

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L2 1 Arbeitsauftrag Einführung Algorithmik

L2_1 Einführung Algorithmik

Hinweis:

Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen das Informations-material L2_1 Information_Algorithmik.docx.

1 Formulieren Sie einen Satz, der den Begriff 'Algorithmus' definiert.

2 Erstellen Sie eine Übersicht der Eigenschaften von Algorithmen.

3.1 Der Möbelhersteller Holzwurm GmbH möchte für den Mitnahmetisch ALPHA eine Aufbauanleitung erstellen.

Formulieren Sie anhand der 'Abbildungen zum Aufbau des Mitnahmetisches ALPHA' (siehe Anlage, Seite 2) einen Algorithmus, der die Vorgehensweise zum Aufbau des Tisches beschreibt.

3.2 Die Kfz-Steuer für PKWs sollen mit Hilfe eines Programms ermittelt und ausgegeben werden.

Für die Berechnung der Steuer gelten folgende (vereinfachte) Regeln:

Für Dieselfahrzeuge werden pro 100 Kubikzentimeter Hubraum 9,50 EUR, für Benzinfahr-zeuge 2,00 EUR fällig.

Zusätzlich wird der CO2-Ausstoß des Fahrzeugs bei der Steuerberechnung berücksichtigt. Pro Gramm CO2/km werden 2,00 EUR berechnet. Dabei ist ein CO2-Freibetrag von 95 g/km zu beachten.

Für ein Diesel-Fahrzeug mit 2090 ccm und einem CO2-Ausstoß von 174 g/km ergibt sich somit ein Steuerbetrag von 348,00 EUR [20 * 9,50 + (174 – 95) * 2]

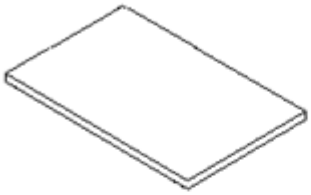
Formulieren Sie für die Berechnung der Kfz-Steuer einen Algorithmus in Form eines Struktogramms.

Lösung:

Anlage: Abbildungen zum Aufbau des Mitnahmetisches ALPHA

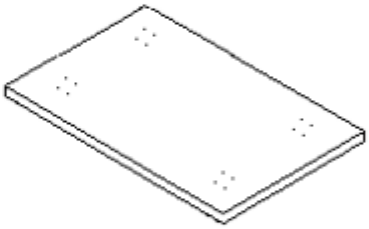
① Mitgelieferte Teile:

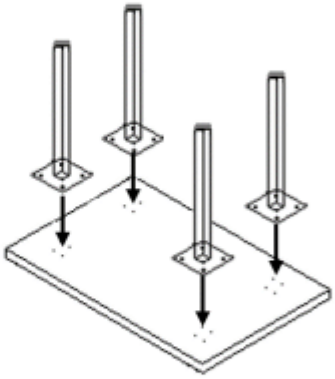
Tischplatte Ahornfurnier (1 Stück)	Tischbein Edelstahl (4 Stück)	Filzgleiter (4 Stück)	Befestigungsschrauben (16 Stück)

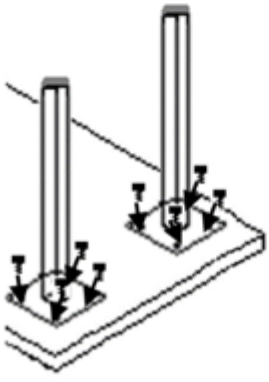


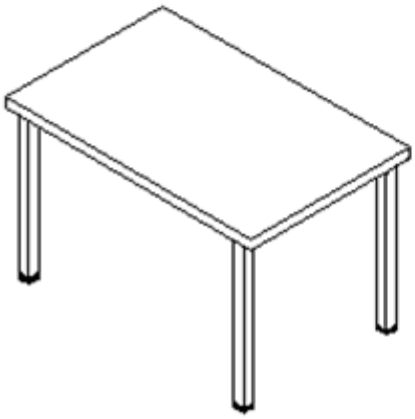

Aufbau:

② 

③ 

④ 

⑤ 

⑥ 

5 Sortieralgorithmen L2 2

Sortieralgorithmen L2 2

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L2 2 1 1 Arbeitsauftrag Bubble Sort Lotto

L2_2.1.1 Sortierung: Bubble Sort – Lottozahlen

Hinweis:

Beachten Sie zur Bearbeitung der nachfolgenden Aufgaben¬stellungen die Informationsmaterialien

- L2_2.1 Information_Bubble_Sort.docx
- L2_2.1 Präsentation_Prinzip_Bubble_Sort.ppsx.

(I) Problemstellung

Die Zahlen der aktuellen Lottoziehung liegen in der Reihenfolge der Ziehung in einem Array lotto vor (48, 5, 17, 32, 7, 29) und sollen noch sortiert werden. Sie erhalten den Auftrag, ein entsprechendes Programm zu entwickeln, das die Lottozahlen mit Hilfe des Sortieralgorithmus 'Bubble Sort' sortiert und die sortierten Zahlen in der Konsole ausgibt

Verwenden Sie für die Implementierung Ihrer Lösung die Datei L2_2_1_1_vorlage_bubble_sort_lotto.py, die Ihnen im Ordner Aufgaben/Vorlagen in digitaler Form vorliegt.

Speichern Sie Ihre Lösung in Ihrem Ergebnisordner unter dem Namen L2_2_1_1_bubble_sort_lotto.py.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

(3) Welche Eigenschaften haben die Eingabe-, Verarbeitungs- und Ausgabedaten? (Variablenliste)

Bedeutung	**Typ/Struktur**	**Variable/Größe**

(4) Gewünschter Ablauf des Programms mit Beispieldaten:

Ausgabe	`[5, 7, 17, 29, 32, 48]`

(5) Verarbeitung

(III) Struktogramm

(IV) Programmcode (Python-Code)

```
EIGENE LÖSUNG:
```

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L2 2 1 2 Arbeitsauftrag Bubble Sort Zahlenreihe

L2_2.1.2 Sortierung: Bubble Sort – Zahlenreihe

Hinweis:

Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen die Informationsmaterialien

L2_2.1 Information_Bubble_Sort.docx

L2_2.1 Präsentation_Prinzip_Bubble_Sort.ppsx.

(I) Problemstellung

Sie sollen ein Programm schreiben, das fünf Zahlen einliest und diese wieder sortiert ausgibt. Implementieren Sie für die Sortierung den Bubble Sort.

Verwenden Sie für die Implementierung Ihrer Lösung die Datei L2_2_1_2_vorlage_bubble_sort_zahlenreihe.py, die Ihnen im Ordner Aufgaben/Vorlagen in digitaler Form vorliegt.

Speichern Sie Ihre Lösung in Ihrem Ergebnisordner unter dem Namen

L2_2_1_2_bubble_sort_zahlenreihe.py.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

(3) Welche Eigenschaften haben die Eingabe-, Verarbeitungs- und Ausgabedaten? (Variablenliste)

Bedeutung	**Typ/Struktur**	**Variable/Größe**

(4) Gewünschter Ablauf des Programms mit Beispieldaten:

Eingabe	Zahl 1: 5 Zahl 2: 6 Zahl 3: 1 Zahl 4: 2 Zahl 5: 3

Ausgabe	1 2 3 5 6

(5) Verarbeitung

(III) Struktogramm

(IV) Programmcode (Py thon-Code)

```
EIGENE LÖSUNG KOMMT!
```

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L2 2 2 2 Arbeitsauftrag Selection Sort Zahlenreihe

L2_2.2.2 Sortierung: Selection Sort – Zahlenreihe

Hinweis:

Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen die Informationsmaterialien L2_2.2 Information_Selection_Sort.docx

(I) Problemstellung

Sie sollen ein Programm schreiben, das fünf Zahlen einliest und diese wieder sortiert ausgibt. Implementieren Sie für die Sortierung den Selection Sort.

Verwenden Sie für die Implementierung Ihrer Lösung die Datei L2_2_2_2_vorlage_selection_sort_zahlenreihe.py, die Ihnen im Ordner Aufgaben/Vorlagen in digitaler Form vorliegt.

Speichern Sie Ihre Lösung in Ihrem Ergebnisordner unter dem Namen

L2_2_2_2_selection_sort_zahlenreihe.py.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

(3) Welche Eigenschaften haben die Eingabe-, Verarbeitungs- und Ausgabedaten? (Variablenliste)

Bedeutung	Typ/Struktur	Variable/Größe

(4) Gewünschter Ablauf des Programms mit Beispieldaten:

Eingabe	Zahl 1: 5 Zahl 2: 6 Zahl 3: 1 Zahl 4: 2 Zahl 5: 3

Ausgabe	[1, 2, 3, 5, 6]

(5) Verarbeitung

(III) Struktogramm

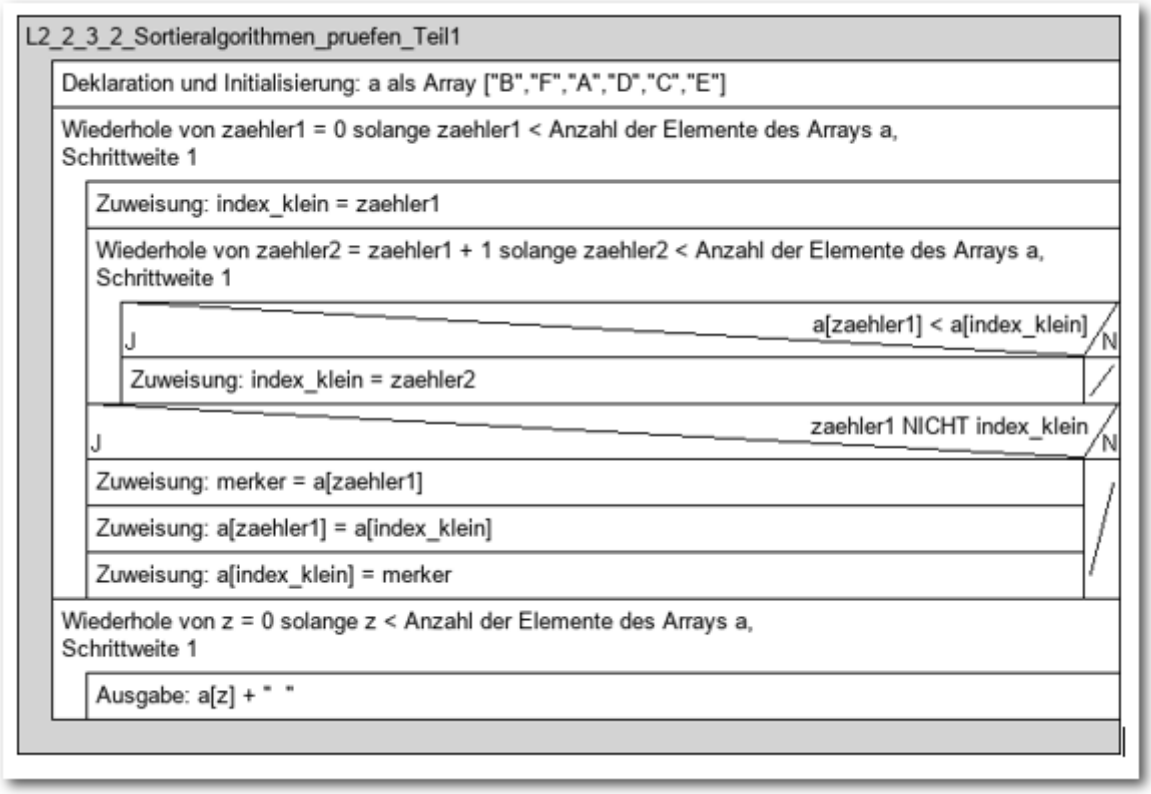
(IV) Programmcode (Python-Code)

```
EIGENE LÖSUNG KOMMT!
```

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L2 2 3 2 Arbeitsauftrag Sortieralgorithmen prüfen

L2_2.3.2 Sortieralgorithmen prüfen

1 Ihnen liegt nachfolgendes Struktogramm zur Analyse vor:



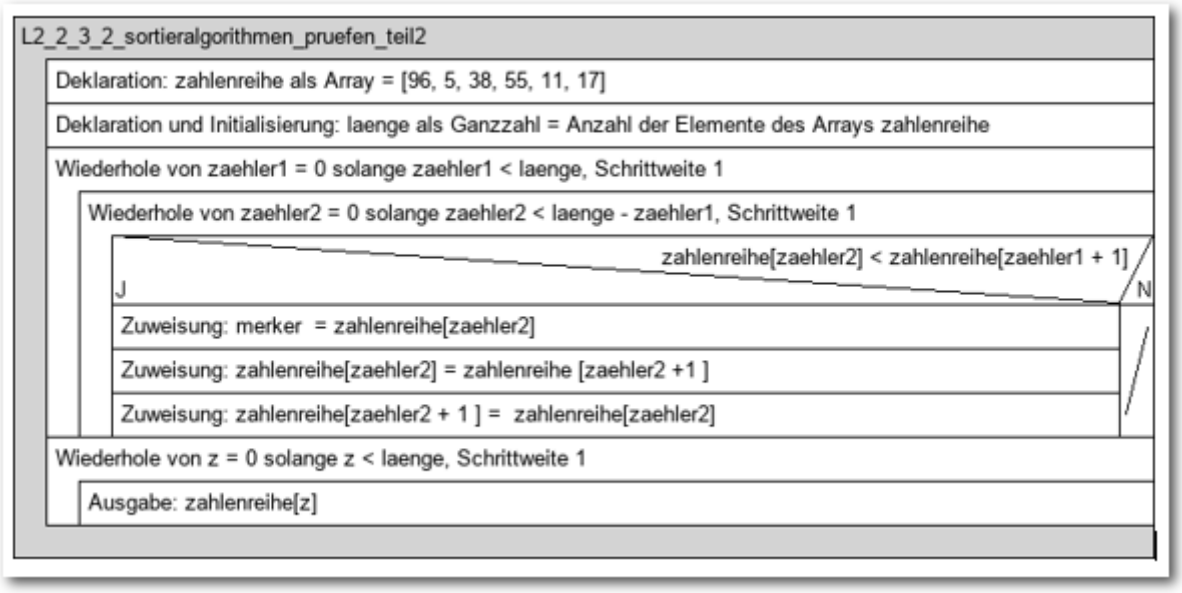
1.1. Analysieren Sie den im Struktogramm dokumentierten Programmablauf und beschreiben Sie das Ergebnis des Algorithmus.

1.2. Implementieren Sie den Programmcode gemäß des abgebildeten Struktogramms.

Speichern Sie Ihre Lösung in Ihrem Ergebnisordner unter dem Namen L2_2_3_2_sortieralgorithmen_pruefen_teil1.py.

```
EIGENE LÖSUNG KOMMT!
```

2 Ihnen liegt folgendes Struktogramm zur Analyse vor:



2.1 Nennen Sie die Zielsetzung des dargestellten Algorithmus.

2.2 Analysieren Sie die einzelnen Anweisungen des dargestellten Algorithmus und lokalisieren Sie vorhandene logische Fehler.

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L2 2 3 3 Arbeitsauftrag Volleyballspieler sortieren

L2_2.3.3 Volleyball – Spielernamen sortieren

(I) Problemstellung

Die Software des Trainers der Abteilung Volleyball des Sportvereines Mühlberger SC enthält ein Array mit allen Spielernamen des Mannschaftskaders.

```
kader =  ["Nico", "Batu", "Paul", "Kai", "Sven", "Milan", "Goran", "Chris", "Armin",
 "Dennis", "Emin", "Luca"]
```

Mit Hilfe der Software soll es möglich sein, die Spieler in alphabetischer Reihenfolge auszugeben.

Implementieren Sie ein Programm, mit dem das Array kader in alphabetischer Reihenfolge sortiert und in der Konsole ausgegeben wird.

Verwenden Sie für die Implementierung Ihrer Lösung die Datei L2_2_3_3_vorlage_volleyball_spieler _sortieren.py, die Ihnen im Ordner Aufgaben/Vorlagen in digitaler Form vorliegt.

Speichern Sie Ihre Lösung in Ihrem Ergebnisordner unter dem Namen L2_2_3_3_volleyball_spieler_sortieren.py.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

(3) Welche Eigenschaften haben die Eingabe-, Verarbeitungs- und Ausgabedaten?
(Variablenliste)

Bedeutung	**Typ/Struktur**	**Variable/Größe**

(4) Gewünschter Ablauf des Programms mit Beispieldaten:

Ausgabe	Kader alphabetisch sortiert: ['Armin', 'Batu', 'Chris', 'Dennis', 'Emin', 'Goran', 'Kai', 'Luca', 'Milan', 'Nico', 'Paul', 'Sven']

(5) Verarbeitung

(III) Struktogramm

(IV) Programmcode (Python-Code)

```
EIGENE LÖSUNG KOMMT!
```

6 Suchalgorithmen L2 3

Suchalgorithmen L2 3

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L2 3 1 1 Arbeitsauftrag Lineare Suche Mitgliedsnummer

L2_3.1.1 Suchen: Lineare Suche – Mitgliedsnummer

Der Sportverein Mühlberger SC hat seine Mitglieder zur Mitgliederversammlung eingeladen. Zugang zur Veranstaltung sollen nur Mitglieder haben. Deswegen wird am Eingang der Mitgliederausweis kontrolliert.

Hinweis:

Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen die Informationsmaterialien

- L2_3.1 Information_Lineare_Suche.docx
- L2_3.1 Präsentation_Prinzip_Lineare_Suche.ppsx.

(I) Problemstellung

Implementieren Sie ein Programm, das überprüft, ob ein Besucher der Mitgliederversammlung zugangsberechtigt ist. Dafür soll am Eingang die Mitgliedsnummer in das Programm eingegeben werden. Nach der Eingabe der Mitgliedsnummer wird geprüft, ob die eingegebene Nummer existiert. Alle vergebenen Mitgliedsnummern des Vereins sind im Array mnr erfasst.

mnr = [1001, 1019, 1014, 1009, 1005, 1002, 1018, 1008, 1003, 1010, 1007, 1004, 1020, 1013, 1015, 1011, 1017, 1012, 1006, 1016]

Wird die eingegebene Mitgliedsnummer gefunden, soll die Meldung „Zutritt gewährt" ausgegeben werden. Wird die Nummer nicht gefunden, soll die Meldung „Zutritt verweigert" erscheinen.

Verwenden Sie für die Implementierung Ihrer Lösung die Datei L2_3_1_1_vorlage_lineare_suche _mitgliedsnummer.py, die Ihnen im Ordner Aufgaben/Vorlagen in digitaler Form vorliegt.

Speichern Sie Ihre Lösung in Ihrem Ergebnisordner unter dem Namen L2_3_1_1_loesung_lineare_suche_mitgliedsnummer.py.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

(3) Welche Eigenschaften haben die Eingabe-, Verarbeitungs- und Ausgabedaten?
(Variablenliste)

Bedeutung	**Typ/Struktur**	**Variable/Größe**

(4) Gewünschter Ablauf des Programms mit Beispieldaten:

Mitgliedsnummer wird gefunden		Mitgliedsnummer wird nicht gefunden	
Eingabe	Mitgliedsnummer eingeben: 1001	**Eingabe**	Mitgliedsnummer eingeben: 20
Ausgabe	Zutritt gewährt	**Ausgabe**	Zutritt verweigert

(5) Verarbeitung

(III) Struktogramm

(IV) Programmcode (Python-Code)

```
EIGENE LÖSUNG KOMMT!
```

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L2 3 2 2 Arbeitsauftrag Binäre Suche Mitgliedsnummer

L2_3.2.2 Suchen: Binäre Suche – Mitgliedsnummer

Der Sportverein Mühlberger SC hat seine Mitglieder zur Mitgliederver¬sammlung eingeladen. Zugang zur Veranstaltung sollen nur Mitglieder haben. Deswegen wird am Eingang der Mitgliederausweis kontrolliert.

Hinweis:

Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen die Informations-materialien

- L2_3.2 Information_Binäre_Suche.docx
- L2_3.2 Präsentation_Prinzip_Binäre_Suche.ppsx.

(I) Problemstellung

Mit der Aufgabenstellung aus "L2_3.1.1 Arbeitsauftrag Lineare Suche Mitgliedsnummer.docx" wurde bereits eine Lösung mit Hilfe der linearen Suche erarbeitet.

In dieser Aufgabenstellung soll nach der Eingabe einer Mitgliedsnummer mit Hilfe der binären Suche geprüft werden, ob die eingegebene Nummer existiert.

Alle vergebenen Mitgliedsnummern des Vereins sind im Array mnr erfasst.

mnr = [1001, 1019, 1014, 1009, 1005, 1002, 1018, 1008, 1003, 1010, 1007, 1004, 1020, 1013, 1015, 1011, 1017, 1012, 1006, 1016]

Wird die eingegebene Mitgliedsnummer gefunden, soll die Meldung „Zutritt gewährt" ausgegeben werden. Wird die Nummer nicht gefunden, soll die Meldung „Zutritt verweigert" erscheinen.

Verwenden Sie für die Implementierung Ihrer Lösung die Datei L2_3_2_2_vorlage_binaere_suche

_mitgliedsnummer.py, die Ihnen im Ordner Aufgaben/Vorlagen in digitaler Form vorliegt.

Speichern Sie Ihre Lösung in Ihrem Ergebnisordner unter dem Namen

L2_3_2_2_binaere_suche_mitgliedsnummer.py.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

(3) Welche Eigenschaften haben die Eingabe-, Verarbeitungs- und Ausgabedaten? (Variablenliste)

Bedeutung	Typ/Struktur	Variable/Größe

(4) Gewünschter Ablauf des Programms mit Beispieldaten:

Mitgliedsnummer wird gefunden		Mitgliedsnummer wird nicht gefunden	
Eingabe	Mitgliedsnummer eingeben: 1001	**Eingabe**	Mitgliedsnummer eingeben: 20
Ausgabe	Zutritt gewährt	**Ausgabe**	Zutritt verweigert

(5) Verarbeitung

(III) Struktogramm

(IV) Programmcode (Python-Code)

```
EIGENE LÖSUNG KOMMT!
```

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L2 3 1 3 Arbeitsauftrag Volleyballspieler suchen

L2_3.1.3 Volleyball – Spieler suchen

(I) Problemstellung

Der Trainer der Abteilung Volleyball des Sportvereins Mühlberger SC möchte eine Erweiterung seiner Software.

Nach der Eingabe eines Spielernamens möchte er eine Information angezeigt bekommen, ob der Spieler im Mannschaftskader ist oder nicht.

Die Namen aller Spieler des Mannschaftskaders sind in dem Array kader erfasst.

```
kader = ["Armin", "Batu", "Kai", "Sven", "Paul", "Milan", "Goran", "Chris", "Nico",
"Dennis", "Emin", "Luca"]
```

Der erforderliche Algorithmus soll so optimiert sein, dass die Suche im Array kader beendet wird, sobald der gesuchte Spieler gefunden wurde.

Verwenden Sie für die Implementierung Ihrer Lösung die Datei L2_3_1_3_vorlage_volleyball_spieler_suchen.py, die Ihnen im Ordner Aufgaben/Vorlagen in digitaler Form vorliegt.

Speichern Sie Ihre Lösung in Ihrem Ergebnisordner unter dem Namen L2_3_1_3_volleyball_spieler_suchen.py.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

(3) Welche Eigenschaften haben die Eingabe-, Verarbeitungs- und Ausgabedaten? (Variablenliste)

Bedeutung	**Typ/Struktur**	**Variable/Größe**

(4) Gewünschter Ablauf des Programms mit Beispieldaten:

Spielername im Kader gefunden		Spielername im Kader nicht gefunden	
Eingabe	Gesuchter Spielername: Batu	**Eingabe**	Gesuchter Spielername: Timo
Ausgabe	Spieler ist im Mannschaftskader	**Ausgabe**	Spieler ist nicht im Mannschaftskader

(5) Verarbeitung

(III) Struktogramm

(IV) Programmcode (Python-Code)

```
EIGENE LÖSUNG KOMMT!
```

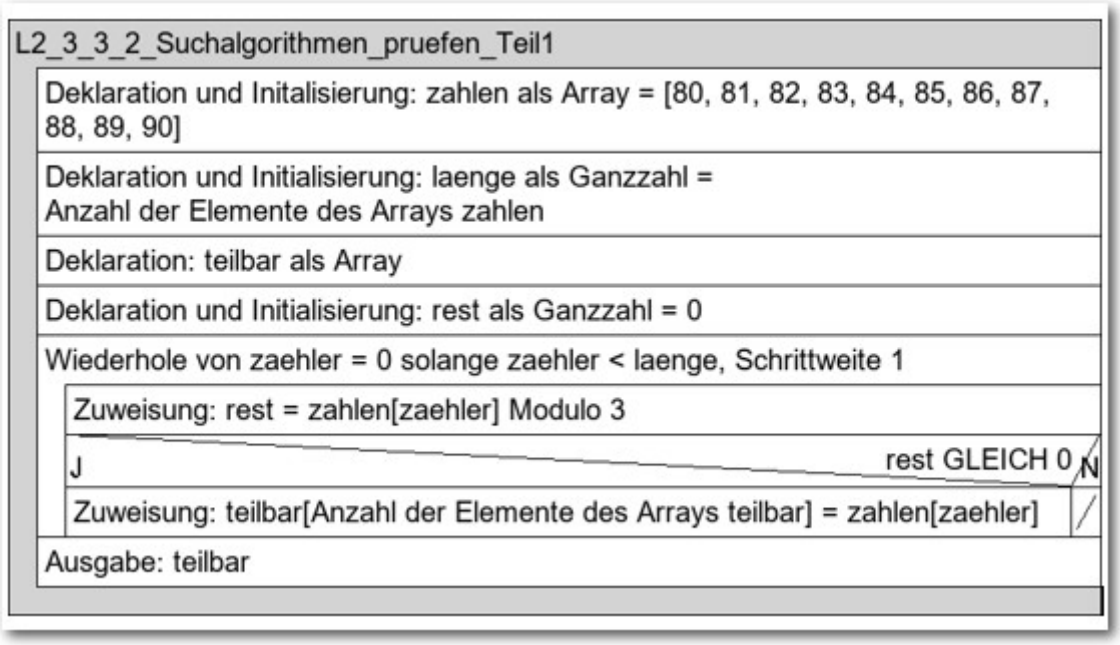
Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L2 3 3 2 Arbeitsauftrag Suchalgorithmen prüfen

L2_3.3.2 Suchalgorithmen prüfen

1 Gegeben ist das Array zahlen, das die natürlichen Zahlen von 80 bis 90 enthält.

zahlen = [80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90]

Mit Hilfe eines Programms soll geprüft werden, welche dieser Zahlen durch 3 teilbar sind. Zur Lösung des beschriebenen Problems wurde bereits folgendes Struktogramm entwickelt:



Hinweis:

Der Ausdruck zahl1 Modulo zahl2 liefert den Rest, den die Division von zahl1 geteilt durch zahl2 ergibt.

Beispiel: 20 Modulo 7 liefert den Wert 6

20 : 7 = 2, Rest 6 (7 * 2 + 6 = 20)

1.1 Führen Sie einen Schreibtischtest durch, indem Sie folgende Tabelle ausfüllen:

zaehler	zahlen[zaehler]	rest	teilbar

1.2 Implementieren Sie den Programmcode gemäß des abgebildeten Struktogramms.

Für die Modulorechnung wird in den meisten Programmiersprachen das % -Zeichen als Operator verwendet. Der Ausdruck zahl1 Modulo zahl2 wird somit folgendermaßen codiert:

```
zahl1 % zahl2
```

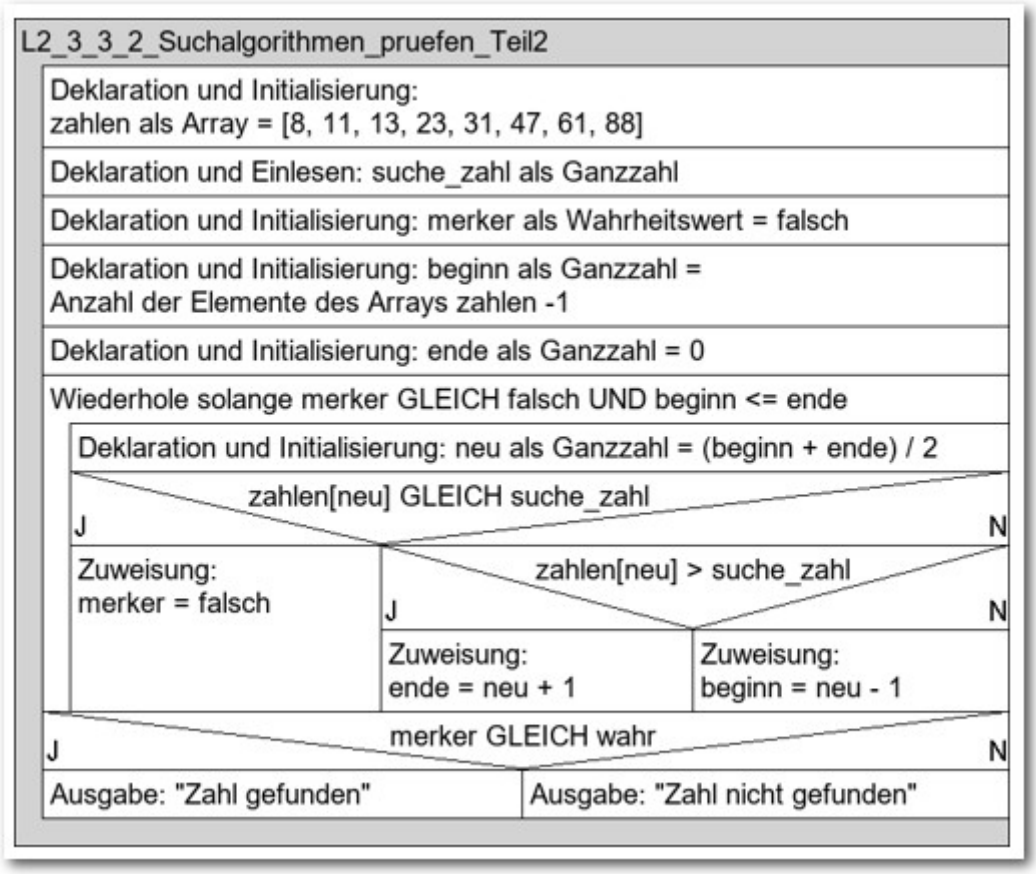
Verwenden Sie für die Implementierung Ihrer Lösung die Datei L2_3_3_2_vorlage_suchalgorithmen_pruefen_teil1.html, die Ihnen im Ordner Aufgaben/Vorlagen in digitaler Form vorliegt.

Speichern Sie Ihre Lösung in Ihrem Ergebnisordner unter dem Namen

L2_3_3_2_suchalgorithmen_pruefen_teil1.html.

EIGENE LÖSUNG KOMMT!

2 Ihnen liegt folgendes Struktogramm zur Analyse vor:



2.1 Nennen Sie die Zielsetzung des dargestellten Algorithmus.

2.2 Analysieren Sie die einzelnen Anweisungen des dargestellten Algorithmus und lokalisieren Sie vorhandene logische Fehler.

7 Dynamische Datenstrukturen – verkettete Liste L3 1

Dynamische Datenstrukturen verkettete Liste L3 1

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L3 1 1 Arbeitsauftrag verkettete Listen

L3_1.1 Dynamische Datenstrukturen: Verkettete Liste

Hinweis:

Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen das Informationsmaterial

L3_1 Information_verkettete_Liste.docx.

1 Supermarktkasse

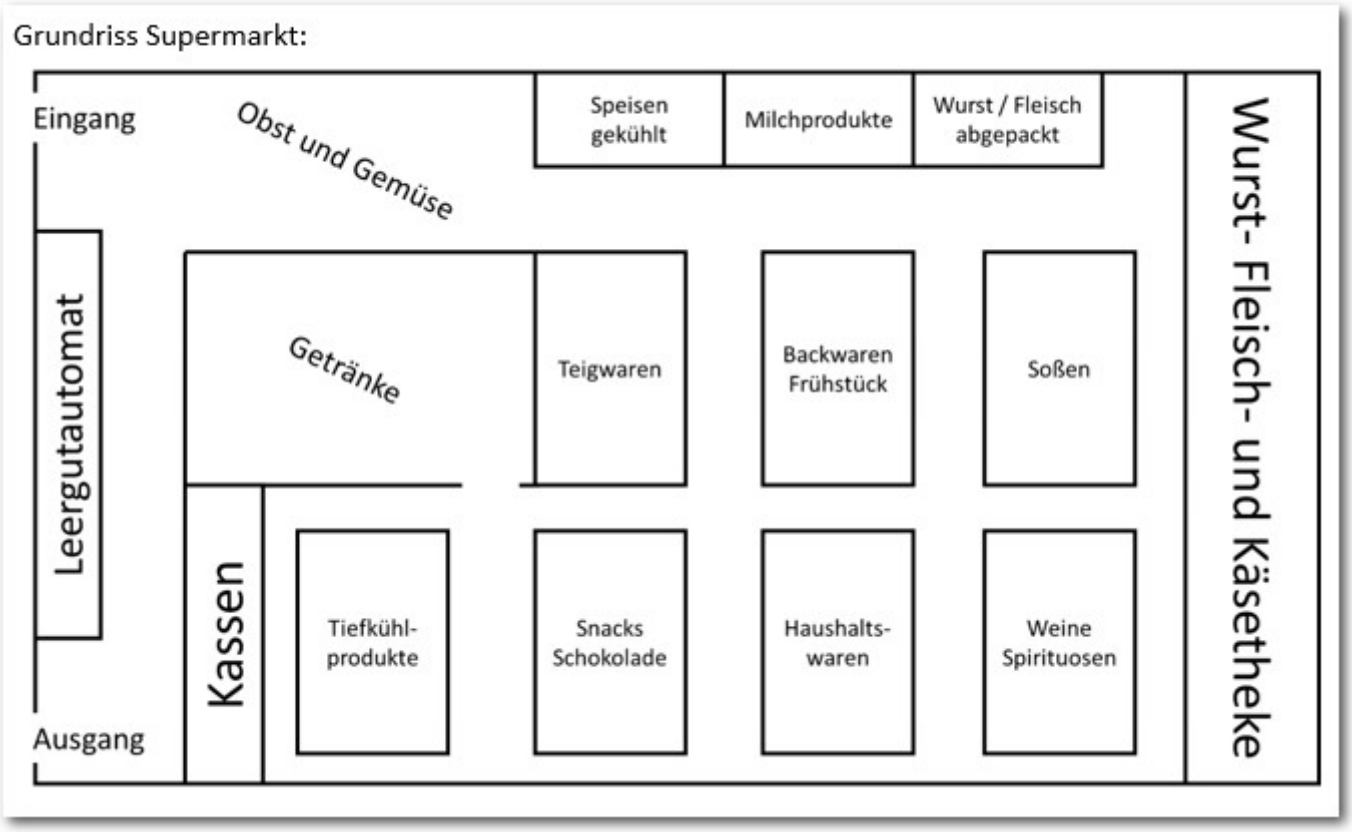
In einer Schlange stehen Personen an einer Kasse an. Leni ist die siebte Person in der Schlange und steht zwei Positionen vor Emma. Emma dagegen ist die zweitletzte Person in der Schlange.

Wie viele Personen sind in der Schlange?

2 Einkaufsliste Supermarkt

Timo hat seine Freundin Steffi zum Essen eingeladen. Sein Einkaufzettel, mit dem er in den Supermarkt geht, beinhaltet folgende Artikel:

Cola, Wein, Schokoladeneis, Milch, Tiefkühlpizza, Schokolade, Toilettenpapier, Kerzen



2.1 Erstellen Sie einen Einkaufszettel für Timo in Form einer verketteten Liste. Helfen Sie Timo alle Artikel in eine sinnvolle Reihenfolge zu bringen damit er schnell nach Hause kommt, um noch die Wohnung zu putzen.

Individuelle Schülerlösung:

2.2 Timo fällt bei der Fahrt in den Supermarkt auf, dass er noch Leergut zurück bringen kann. Außerdem glaubt er, dass Steffi die Kerzen vielleicht nicht gefallen könnten und streicht sie von der Einkaufsliste.

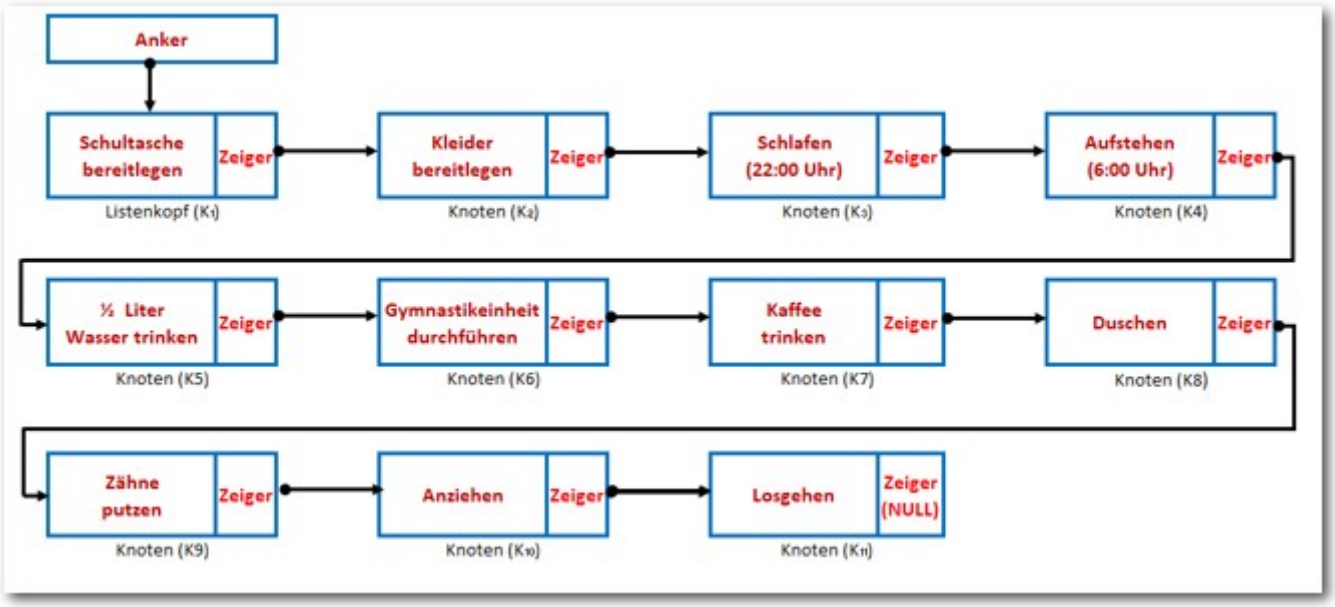
Überarbeiten Sie die verkettete Liste und beschreiben Sie ausführlich, wie Sie beim Löschen und Hinzufügen von Knoten vorgehen.

3 Morgenroutine

Um einen guten Start in den Tag zu haben, beschließt Steffi einer Morgenroutine nachzukommen. Die Morgenroutine fängt aber nicht erst am Morgen des Schultages an. Um möglichst wenig Stress am Morgen zu haben, möchte sie ihre Schultasche und die Kleider für den nächsten Tag schon am Vorabend bereitlegen. Steffi möchte um 22 Uhr schlafen gehen, so dass sie um 6:00 Uhr fit ist. Direkt nach dem Aufstehen nimmt sich Steffi vor, einen ½ Liter Wasser zu trinken und eine kurze Gymnastikeinheit durchzuführen. Außerdem muss Steffi noch folgende Tätigkeiten erledigen, bevor sie sich auf den Schulweg macht: Duschen, Zähne putzen, anziehen, Kaffee trinken, losgehen.

3.1. Erstellen Sie für Steffis Morgenroutine eine verkettete Liste in einer sinnvollen Reihenfolge.

Individuelle Schülerlösung:



3.2. Nach der ersten Woche möchte Steffi ihre Morgenroutine anpassen. Weil das Duschen zu viel Zeit am Morgen in Anspruch nimmt, duscht sie von nun an vor dem Schlafengehen. Anstatt wie bisher einen Kaffee zu trinken, möchte Sie auf Tee umsteigen.

Ändern Sie die verkettete Liste und beschreiben Sie genau, wie Sie beim Löschen und Hinzufügen von Knoten vorgehen.

Hinweis: Die Lösung ist auf die Musterlösung aus Aufgabenteil 3.1 bezogen

"Duschen" ändern:

"Kaffee trinken" löschen und "Tee trinken" hinzufügen:

8 Dynamische Datenstrukturen – Stapelspeicher L3 2

Dynamische Datenstrukturen
Stapelspeicher L3 2

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L3 2 1 Arbeitsauftrag Stapelspeicher

L3_2.1 Dynamische Datenstrukturen: Stapelspeicher (Stack)

Hinweis:

Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen das Informations-material

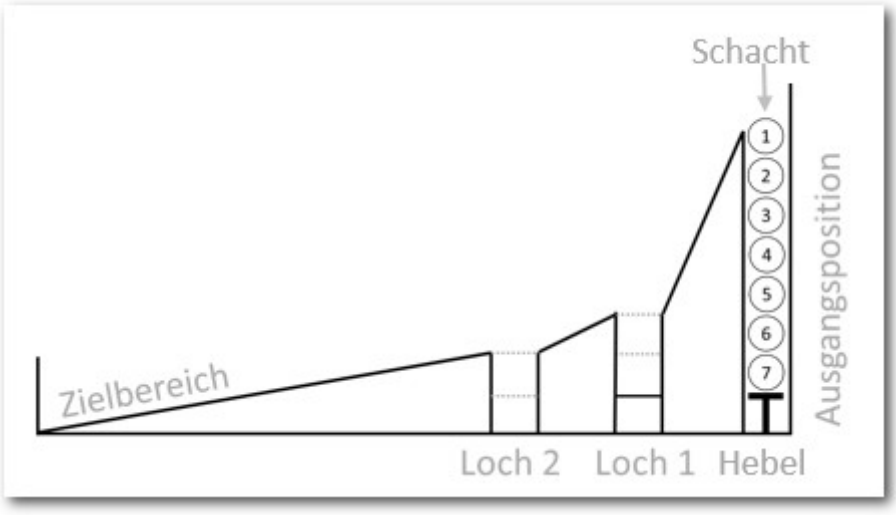
L3_2 Information_Stapelspeicher.docx.

1.1 Murmelbahn – Teil 1

Ihre kleine Schwester hat die Murmelbahn „Die wilde Lifo" zum zweijährigen Geburtstag geschenkt bekommen. Vereinfacht lässt sich die Murmelbahn und das Spielprinzip wie folgt darstellen:

Spielprinzip

Die Kugeln sind in der Anfangspo-sition in einem Schacht überei-nander gestapelt. Von dort wer-den sie durch einem Hebel in die Murmelbahn gedrückt. In der Bahn befinden sich zwei Löcher, in die jeweils zwei Kugeln passen. Ist ein Loch voller Kugeln, rollt die nächste Kugel über das Loch. Das Spiel endet, wenn sich keine Ku-geln mehr in der Anfangsposition befinden.



Aufgabe

Benennen Sie die Kugeln in der Reihenfolge, in der sie im Zielbereich ankommen. Verwenden Sie zur Dokumentation Ihrer Überlegungen folgende Tabelle.

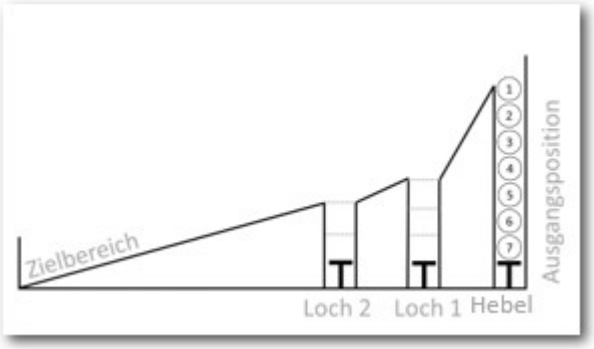
Schritt	Ausgangsposition	Loch 1	Loch 2	Zielbereich
	1,2,3,4,5,6,7			
1				
2				
3				
4				
5				
6				
7				

1.2 Murmelbahn – Teil 2

Nachdem Ihre Schwester viel Spaß mit der Murmelbahn hatte, entscheiden Sie sich, ihr die Weiter-entwicklung des Spiels, „Die noch wildere Lifo", zu schenken. Vereinfacht lässt sich die Murmelbahn und das Spielprinzip wie folgt darstellen:

Spielprinzip

Die Kugeln werden von der Ausgangsposition nacheinan-der durch einem Hebel in die Murmelbahn gedrückt. In der Bahn befinden sich zwei Löcher. In das erste Loch passen drei Kugeln, in das zweite Loch zwei Kugeln. Ist eines der Löcher voll, werden alle Kugeln aus diesem Loch nacheinander in die Murmelbahn zurück gedrückt. Erst wenn dieses Loch wieder leer ist, werden die Kugeln aus dem Schacht der Ausgangssituation in die Kugelbahn gedrückt, oder das andere volle Loch geleert.



Ansonsten gelten die bereits bekannten Regeln aus der Murmelbahn Teil 1.

Aufgabe

1.2.1 Stellen Sie den Spielverlauf in der nachfolgenden Tabelle dar.

Schritt	Ausgangsposition	Loch 1	Loch 2	Zielbereich
	1,2,3,4,5,6,7			
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				

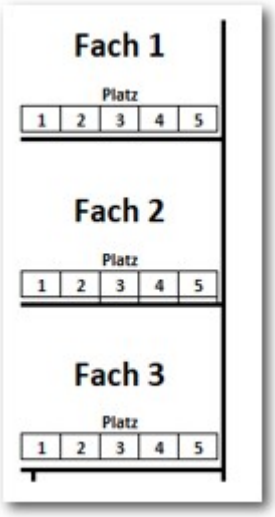
14				
15				
16				
17				

1.2.2 Nennen Sie die Ziffer der Kugel, die als erstes im Zielbereich ankommt und begründen Sie Ihre Entscheidung.

2 Lebensmittelmarkt SuperSpar

Der Lebensmittelmarkt SuperSpar kauft von verschiedenen Großhändlern stark verbilligte Restposten ein. So kann der Lebensmittelmarkt seinen Kunden günstige Preise anbieten und trotzdem Gewinne erzielen.

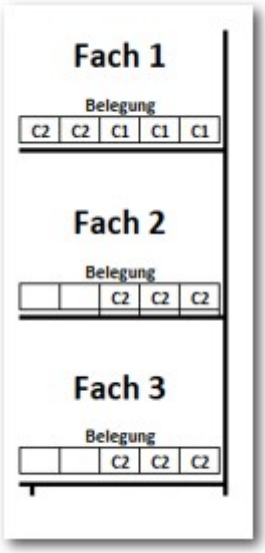
Die Ladenregale im Super-Spar verfügen über drei Fächer. In jedem Fach ist Platz für fünf Waren, die hinter-einander gestellt werden. Kommt neue Ware, wird diese von vorne aufgefüllt und die bereits vor-handene Ware wird nach hin-ten geschoben (Lifo-Prinzip). Sobald das erste Fach voll ist, werden die Waren in das zweite Fach nach dem Lifo-Prinzip gestellt.



Beispiel:

Am Montag kommt neue Ware im SuperSpar an. Es werden drei Flaschen Cola (C1) in das Ladenregal geräumt. Am Diens-tag kommt die zweite Lieferung mit zehn Flaschen Cola (C2).

Am Mittwoch entnimmt ein Kunde zwei Flaschen Cola aus dem Fach 2.



2.1 Der SuperSpar erhält am Montagmorgen eine Lieferung von fünf Sweeties (S1) und räumt diese in das Fach 1 des Regals ein. Am gleichen Tag kaufen die Kunden vier Sweeties.

Stellen Sie das Fach 1 nach der Befüllung mit den fünf Sweeties (S1) und nach Ladenschluss dar.

Nach der Befüllung	Nach Ladenschluss

2.2 Am Dienstagmorgen kommt eine weitere Lieferung von zehn Sweeties (S2). Die Süßigkeiten werden in das Regal eingeräumt. Begonnen wird im Fach 1, sofern dort noch Platz vorhanden ist.

Ist dieses voll, wird Fach 2 befüllt, anschließend Fach 3.

Im Laufe des Tages kaufen Kunden drei Sweeties aus Fach 1 und vier Sweeties aus Fach 2.

Stellen Sie das Regal nach der Befüllung der weiteren zehn Sweeties (S2) und nach dem Ladenschluss dar.

Nach der Befüllung	Nach Ladenschluss

2.3 Am Mittwochmorgen kommt erneut eine Lieferung von acht Sweeties (S3), die in das Regal eingeräumt werden. (Reihenfolge des Einräumens wie in Aufgabe b) beschrieben.)

Aus jedem Fach werden am Mittwoch zwei Sweeties gekauft.

Wie viele Sweeties aus der ersten (S1), aus der zweiten (S2) bzw. aus der dritten (S3) Lieferung sind noch im Regal?

Nach der Befüllung	Nach Ladenschluss

Es sind noch ein Sweety aus der ersten Lieferung (S1), zwei Sweeties aus der zweiten Lieferung (S2) und drei Sweeties aus der dritten Lieferung (S3) im Regal.

3 Internetrecherche

Das Internet bietet einen breiten Fundus an Informationen und wird deswegen gerne für die Recher-che benutzt. Wechseln Sie von der Internetseite A auf die Internetseite B wird die Adresse der Inter-netseite A in einen Stapelspeicher gelegt. Klicken Sie nun auf das „Zurück"-Icon, wird das vorherige Element aus dem Stack aufgerufen und Sie kehren zur Internetseite A zurück.

Der Browserverlauf eines Nutzers ist nachfolgend abgebildet.

Zeile	Adresse
1	**PUSH**("https://www.youtube.com/results?search_query=stapelspeicher")
2	**PUSH**("https://de.wikipedia.org/wiki/Stapelspeicher")
3	**POP**()
4	**PUSH**("http://deacademic.com/dic.nsf/dewiki/1323671")
5	**PUSH**("https://www.heise.de/")
6	**PUSH**("https://www.wim.uni-mannheim.de/de/informatik-und-wirtschaftsinformatik/")
7	**PUSH**("https://www.informatik.kit.edu/")
8	**PUSH**("http://www.warum-informatik.de")
9	**POP**()
10	**POP**()

3.1 Auf welcher Internetseite befindet sich der Nutzer derzeit?

3.2 Auf welchen Internetseiten war der Nutzer zweimal?

9 Dynamische Datenstrukturen – Warteschlange L3 3

Dynamische Datenstrukturen Warteschlange L3 3

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L3 3 1 Arbeitsauftrag Warteschlange

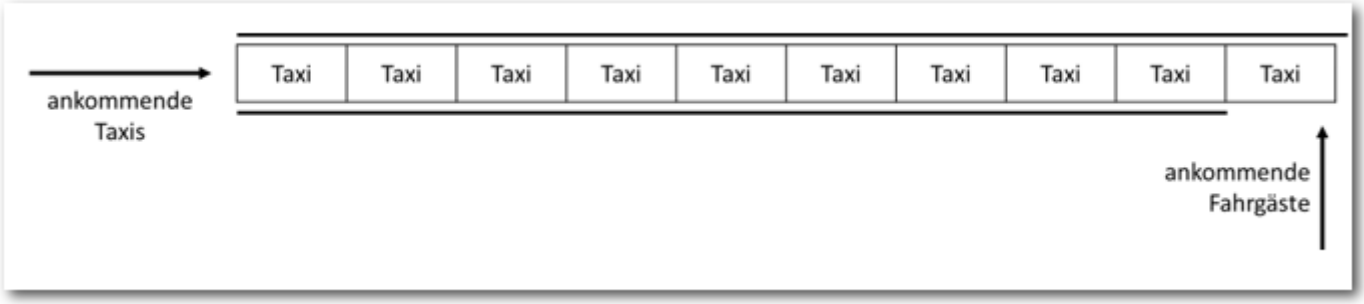
L3_3.1 Dynamische Datenstrukturen: Warteschlange (Queue)

Hinweis:

Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen das Informations-material L3_3 Information_Warteschlange.docx.

1 Taxistand – Teil 1

Am Flughafen gibt es einen Taxistand mit zehn Stellplätzen. Ankommende Taxis stellen sich in einer Schlange an. Im Taxistand kann nicht gewendet werden. Deswegen kann immer nur das vorderste Taxi Fahrgäste aufnehmen und abfahren. Das folgende Schaubild soll den Sachverhalt verdeutlichen.



Ankommende Taxis werden von einem elektronischen Erfassungssystem mit der Taxinummer erfasst. Verlässt ein Taxi den Taxistand wird dies ebenfalls erfasst. Ihnen liegen folgende Daten vor:

Zeit	Taxinummer	Ausfahrt/Einfahrt
14:01	T254	Einfahrt
14:01	T255	Einfahrt
14:02	T254	Ausfahrt
14:05	T432	Einfahrt
14:07	T432	Ausfahrt
14:10	T123	Einfahrt

Hinweis:

Kommen mehrere Taxis zur gleichen Zeit an (vgl. 14:01 Uhr), sind sie in der Tabelle nach ihrer Ankunft aufgelistet (T254 vor T255).

1.1 Aus der Auflistung wird ersichtlich, dass im automatischen Erfassungssystem ein Fehler sein muss. Beschreiben Sie diesen Fehler.

Sie erhalten neue Daten vom 01.08. aus dem automatischen Erfassungssystem.

Zeit	Taxinummer	Ausfahrt/Einfahrt
08:00	T021	Einfahrt
08:02	T099	Einfahrt
08:05	T085	Ausfahrt
08:08	T432	Ausfahrt
08:10	T021	Ausfahrt
08:12	T123	Einfahrt
08:16	T099	Ausfahrt
08:18	T525	Einfahrt
08:21	T456	Einfahrt

1.2 Wie viele Taxis befanden sich vor der Einfahrt des Taxis T021 im Taxistand?

1.3 Welche Taxis befinden sich in welcher Reihenfolge um 08:22 Uhr im Taxistand? Vervollständigen Sie zur Beantwortung der Frage die folgende Tabelle.

	10	9	8	7	6	5	4	3	2	1
08:00										

1.4 Um 08:01 Uhr kommt Sandra Meier am Taxistand an. Es warten keine Fahrgäste auf ein Taxi. Sandra Meier möchte aber ausschließlich mit dem Taxi T021 fahren. Wann ist Sandra Meier abgefahren und wie viele Fahrgäste hat sie vorgelassen?

1.5 In der Zeit von 08:30 bis 09:00 arbeitete das automatische Erfassungssystem fehlerhaft. Es wurden zwar die Uhrzeiten und die Taxinummern erfasst, jedoch nicht, ob es sich um eine Ausfahrt oder Einfahrt handelte.

Vervollständigen Sie die nachfolgenden Lücken unter Beachtung der dynamischen Datenstruktur Warteschlange.

Zeit	Taxinummer	Ausfahrt/Einfahrt
08:30	T050	Einfahrt
08:32	T599	Einfahrt
08:33	T123	
08:35	T525	
08:38	T456	
08:40	T050	
08:42	T321	
08:46	T099	
08:48	T599	
08:51	T321	
08:53	T555	
08:56	T099	

Beachte:

Um 08:30 befanden sich am Taxistand noch die Taxis

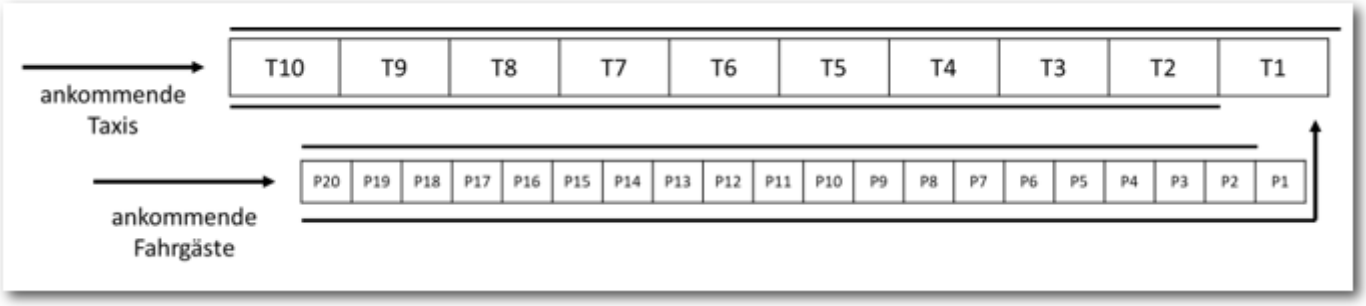
08:12 T123 Einfahrt

08:18 T525 Einfahrt

08:21 T456 Einfahrt

2 Taxistand – Teil 2

Aufgrund einiger Unstimmigkeiten der Fahrgäste bei der Taxivergabe hat der Flughafen neben dem Taxistand auch eine Wartelinie für Fahrgäste eingerichtet (P1 – P20).



Als zusätzliche Information können zukünftig die Sitzplätze eines Taxis gespeichert werden. So hat das Taxi T432.3 bspw. 3 Sitzplätze, das Taxi T009.6 hat 6 Sitzplätze. Außerdem muss auch die Anzahl der zusammengehörigen Fahrgäste (= Fahrgastgruppe) beachtet werden. So hat die Fahrgastgruppe F1.2 beispielsweise 2 Personen und die Fahrgastgruppe F2.8 sogar 8 Personen.

Zeit	Taxinummer	Ausfahrt/Einfahrt
08:00	T021.5	Einfahrt
08:02	T021.5	Ausfahrt

Zeit	Ankommende Fahrgastgruppen
08:01	F1.3
08:01	F2.2

In diesem Beispiel kommt um 08:00 das Taxi T021.5 an. Um 08:01 kommen die Fahrgastgruppen F1 mit drei Personen und die Fahrgastgruppe F2 mit zwei Personen. Die Fahrgastgruppe F1.3 kann direkt abfahren, sodass um 08:02 nur noch die Fahrgastgruppe F2.2 auf ein Taxi wartet. Folglich können mehrere Fahrgastgruppen nicht mit einem Taxi fahren (z. B. aufgrund unterschiedlicher Fahrtziele).

Eine Fahrgastgruppe kann sich dagegen auf zwei oder mehrere Taxis verteilen.

Die nachfolgende Tabelle verdeutlicht den Sachverhalt.

Taxi	....		5		4		3		2		1	
Person	...	...	10	9	8	7	6	5	4	3	2	1
08:00	....										T021.5	
	...	...										
08:01	....										T021.5	
	...	...						F2.2	F2.2	F1.3	F1.3	F1.3
08:02	....											
	...	...									F2.2	F2.2

Sie erhalten aus dem automatischen Erfassungssystem neue Daten vom 02.08.

Zeit	Taxinummer	Ausfahrt/Einfahrt
08:00	T021.5	Einfahrt
08:02	T009.6	Einfahrt
08:07	T021.5	Ausfahrt
08:09	T009.6	Ausfahrt
08:11	T010.6	Einfahrt
08:13	T010.6	Ausfahrt
08:14	T432.3 T258.3 T554.5	Einfahrt Einfahrt Einfahrt
08:16	T432.3	Ausfahrt
08:18	T599.3	Einfahrt

Zeit	Ankommende Fahrgastgruppen
08:02	F1.3
08:02	F2.9
08:09	F3.1
08:12	F4.2

2.1 Vervollständigen Sie die nachfolgende Tabelle mit den Plätzen im Taxistand und der Warteschlange für die Fahrgastgruppen. Beachten Sie die Regeln für die dynamische Datenstruktur Warteschlange.

2.2 Wie viele Taxis und wie viele Fahrgäste sind um 08:06 Uhr im Taxistand?

2.3 Ermitteln Sie die Fahrgastzahl des Taxis T010.6, wenn es um 08:13 Uhr den Taxistand verlässt.

2.4 Wie viele Fahrgäste können mindestens und höchstens bis 08:30 Uhr abfahren, wenn kein weiteres Taxi mehr kommt?

10 Dynamische Datenstrukturen – Baum L3 4

Dynamische Datenstrukturen
Baum L3 4

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L3 4 1 Arbeitsauftrag Baum

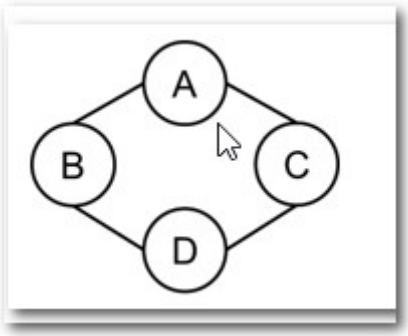
L3_4.1 Dynamische Datenstrukturen: Baum

Hinweis:

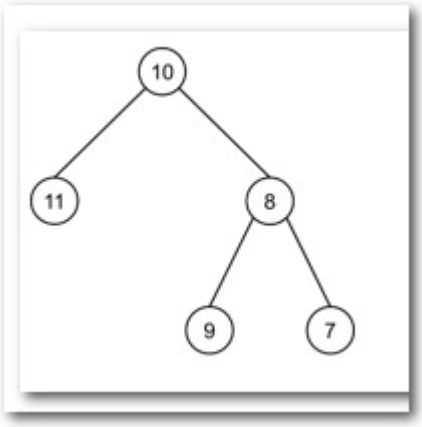
Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen das Informations-material L3_4 Information_Baum.docx.

1 Nennen Sie Alltagsbeispiele für die Datenstruktur Baum.

2 Begründen Sie, ob es sich beim nachfolgenden Schaubild um einen Baum handelt.

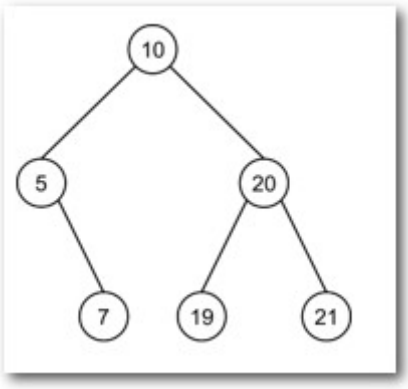


3 Markieren Sie einen Teilbaum im nachfolgenden Baum.

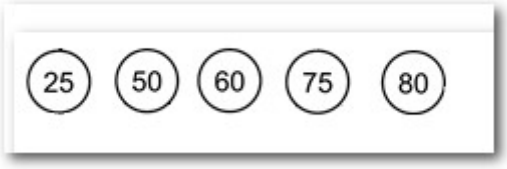


4 Beurteilen Sie, ob der nachfolgende Baum vollständig, voll und/oder geordnet ist.

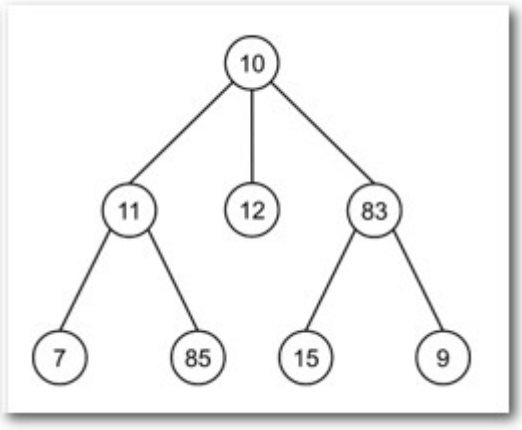
GEORDNET: VOLL: VOLLSTÄNDIG:



5 Überführen Sie die Knoten {25, 50, 60, 75, 80} in einen vollen, geordneten Binärbaum.



6 Überführen Sie den folgenden Baum in einen Binärbaum.



7 Fügen Sie nacheinander die Knoten {B, A} in den gegebenen Binärbaum ein. Achten Sie darauf, dass der Binärbaum weiterhin geordnet ist.