

Aufgabe 6 numpy (ca. 23%)

Das Spielfeld von n-Sudoku besteht aus n^2 quadratisch angeordneten Feldern. Beispiel eines 4-Sudokus:

1		2	3
	2	1	4
4	1		2
2	3	4	1

Es gelten folgende Regeln, damit ein Spielfeld valide ist:

- Das Spielfeld hat n Zeilen und n Spalten.
- In jeder Zeile, in jeder Spalte und in jedem n-Teilquadrat (siehe gestrichelte Linien in der Abbildung) sind nur Zahlen zwischen 1 und n eintragen. Felder können auch leer sein.
- In jeder Zeile, in jeder Spalte und in jedem n-Teilquadrat kommt keine Zahl mehrfach vor.

- a) Zur Darstellung des Spielfeldes soll ein zweidimensionales numpy-Array verwendet werden. Überlegen Sie sich, wie leere Felder dargestellt werden und ergänzen Sie die folgende Zuweisung um ein numpy-Array zur Darstellung des obigen Beispiels:

sudoku =

Auch wenn oben ein Beispiel angegeben wurde, sollen die beiden Funktionen in den folgenden Teilaufgaben für beliebige n und für beliebige n-Sudokus funktionieren! Sie können in den Funktionen eine globale Variable n verwenden.

Folgende numpy-Funktionen könnten für die folgenden Teilaufgaben hilfreich sein. Dabei ist a ein beliebiges Array:

a.size: Anzahl der Zahlen in a. Beispiele: 16 bei einem 4×4-Array, 4 bei einem 1×4-Array.

a.reshape(k): Gibt ein eindimensionales Array zurück, das aus den Zahlen in a besteht.

a[i], a[i, :], a[:, i], a[i:j, k:m]: Slicing

- b) Ergänzen Sie folgende Funktion check_sudo, die ein eindimensionales numpy-Array sudo prüfen soll, ob nur Zahlen zwischen 1 und n eintragen sind (Felder können auch leer sein) und ob keine Zahl mehrfach vorkommt. Der Rückgabewert soll True oder False sein.

```
def check_sudo(sudo):
```

- c) Ergänzen Sie folgende Funktion `check_sudoku`, die ein zweidimensionales numpy-Array `sudo` prüft, ob es ein valides n-Sudoku-Spielfeld ist. Der Rückgabewert soll `True` oder `False` sein. Verwenden Sie die Funktion `check_sudo`.

```
def check_sudoku(sudoku):
```

Aufgabe 7 Dictionaries, JSON (ca. 27%)

Ein Kontakt kann vereinfachend als Dictionary mit den Schlüsseln name, fname, tel dargestellt werden. Beispiel:

```
{'name': 'Mustermann', 'fname': 'Max', 'tel': [49, 841, 211090]}
```

Die Werte zum Schlüssel tel sind Listen bestehend aus ganzen Zahlen. Eine Kontaktliste ist eine Liste von Kontakten.

Im Folgenden sei eine globale Kontaktliste contacts vorausgesetzt.

a) Erstellen Sie eine Funktion add_contact mit den Parametern

name, fname : Zeichenketten

tel : Liste von ganzen Zahlen

Die Funktion hängt einen Kontakt bestehend aus name, fname, tel an die Kontaktliste contacts an und hat keinen Rückgabewert.

b) Erstellen Sie eine Funktion update_contact mit den Parametern

p : Kontakt

name, fname : Zeichenketten (optionale Parameter)

tel : Liste von ganzen Zahlen (optionaler Parameter)

Falls p in der Kontaktliste contacts enthalten ist, werden name, fname, tel, sofern die Argumente vorhanden sind, ersetzt.

Die Funktion hat keinen Rückgabewert.

c) Erstellen Sie eine Funktion `replace_contact` mit den Parametern

`p, new_p` : Kontakte

Falls `p` in der Kontaktliste `contacts` enthalten ist, wird der Kontakt durch `new_p` ersetzt.

Die Funktion hat keinen Rückgabewert.

d) Erstellen Sie eine Funktion `find_contact` mit einem Parameter

`name` : Zeichenkette

Die Funktion gibt eine Liste alle Kontakte in der Kontaktliste `contacts`, dessen Name gleich `name` ist, zurück.

e) Erstellen Sie eine Funktion `write_contacts` mit einem Parameter

`file_name` : JSON-Dateiname

Die Funktion speichert die Kontaktliste `contacts` als JSON-Datei ab.

Die Funktion hat keinen Rückgabewert.

f) Erstellen Sie eine Funktion read_contacts mit einem Parameter

file_name : JSON-Dateiname

In der JSON-Datei ist eine Kontaktliste gespeichert. Die Funktion liest die JSON-Datei ein und hängt alle Kontakte, die in der Kontaktliste contacts noch nicht vorhanden sind, an. Bereits in contacts vorhanden Kontakte werden übersprungen.

Die Funktion hat keinen Rückgabewert.