Aufgabenblatt

Thema: Lernfeld 5: Portfolio-Aufgabe

Conway's Game of Life

Conway's Game of Life ist ein von dem Mathematiker J.H. Conway entworfenes Spiel, dass die Entwicklung einer zellulären Population mittels einfacher Regeln simuliert.

Einen guten Eindruck von der Simulation kannst du unter https://playgameoflife.com/ bekommen.

Spielregeln

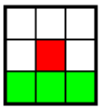
(Auszug aus Wikipedia: https://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens#Die_Spielregeln):

Die Folgegeneration wird für alle Zellen gleichzeitig berechnet und ersetzt die aktuelle Generation. Der Zustand einer Zelle (lebendig oder tot) in der Folgegeneration hängt nur vom aktuellen Zustand der Zelle selbst und den aktuellen Zuständen ihrer acht Nachbarzellen ab.

Die von Conway zu Anfang verwendeten Regeln sind:

- Eine tote Zelle mit genau drei lebenden Nachbarn wird in der Folgegeneration neu geboren.

  rot: Tote Zelle, die in der nächsten Generation geboren wird
  grün: Lebende Nachbarn der Zelle

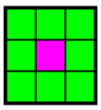


- Lebende Zellen mit weniger als zwei lebenden Nachbarn sterben in der Folgegeneration an Einsamkeit.

- Eine lebende Zelle mit zwei oder drei lebenden Nachbarn bleibt in der Folgegeneration am Leben.

- Lebende Zellen mit mehr als drei lebenden Nachbarn sterben in der Folgegeneration an Überbevölkerung.

  magenta: Lebende Zelle, die betrachtet wird
  grün: Lebende Nachbarn der Zelle



Mit diesen vier einfachen Regeln entsteht aus bestimmten Anfangsmustern im Laufe des Spiels eine Vielfalt komplexer Strukturen. Einige bleiben unverändert, andere oszillieren und wieder andere wachsen oder vergehen. Manche Strukturen, sogenannte Gleiter, bewegen sich auf dem Spielfeld fort.

Aufgabenblatt

Thema: Lernfeld 5: Portfolio-Aufgabe

Aufgaben

Es soll ein Programm geschrieben werden, das Conway's Game of Life umsetzt. Dafür soll eine Funktion `starte_simulation` implementiert werden. Die Details zu dieser Funktion sind der Schnittstellenbeschreibung zu entnehmen, die auf der nächsten Seite zu finden ist.

Herausforderungsstufe 1:

Implementiere die Schnittstelle `starte_simulation` in Python.

Herausforderungsstufe 2:

Implementiere die Schnittstelle `starte_simulation` sowie der Funktion lade_startkonfiguration zum Laden der Startkonfiguration aus einer Textdatei. Das Format der Textdatei sowie Details der Ladefunktion sind ebenfalls der Schnittstellenbeschreibung zu entnehmen.

Hinweise zur Abgabe

Die Abgabe kann bis zum Freitagabend der ersten Blockwoche des letzten Schulblocks dieses Schuljahres erfolgen und ist eine **Einzelleistung**.

Es ist der Quellcode einzureichen. Als Abgabe werden die Python-Dateien oder ein Link zu einem Git-Repository, unter dem die Python-Datei zu finden ist, akzeptiert. Die Datei bzw. der Link sind in der Aufgabenstellung unter der Kachel Portfolio einzureichen.

Die Aufgabe zählt als **optionale** Extraleistung für das Lernfeld 5, die bei Abgabe der Herausforderungsstufe 1 zu 25% bzw. bei Abgabe der Herausforderungsstufe 2 zu 33,3% in die Note mit einfließt.

Die Bewertung der Aufgabe erfolgt über den Test der Funktionalität. Dafür werden automatische Tests (Unit Tests) verwendet, die Funktionen erwarten, die genau der Schnittstellenbeschreibung entsprechen. Die Note ergibt sich aus dem prozentualen Anteil der bestandenen Tests. Grundlage bildet hier der IHK-Notenschlüssel.

Note	1	2	3	4	5	6
Prozente	92 - 100	81 - 91	67 – 80	50 - 66	30 - 49	0 - 29

Zudem sollte jeder den abgegebenen Code in einem Einzelgespräch erklären können. Diese Einzelgespräche erfolgen stichprobenartig im letzten Schulblock dieses Schuljahres. Nicht erklärbarer Code führt zu einer Verschlechterung der Note aus dem Funktionalitätstest.

Aufgabenblatt

Thema: Lernfeld 5: Portfolio-Aufgabe

Schnittstellenbeschreibung

Folge Funktion muss aufrufbar sein.

```
starte_simulation(
        startkonfiguration:List[List[int]],
        anzahl_schritte:int) -> List[List[int]]
```

Die Funktion erwartet als Parameter `startkonfiguration` die Zellenanordnung, mit der die Simulation beginnen soll, als eine geschachtelte Liste von Integer-Werten (siehe dazu die Anmerkungen zur Zellenanordnung). Eine 1 gibt dabei eine lebende Zelle an, ein 0 ein leeres Feld. Zu beachten sind die minimalen und maximalen Größen der Zellenanordnungen.

Zudem wird der Parameter `anzahl_schritte` erwartet, der angibt wie viele Folgegenerationen berechnet werden sollen.

Als **Rückgabe** wird die **Zellenanordnung nach dem letzten Schritt als geschachtelte Liste** erwartet. Das Format ist dabei identisch zur Startkonfiguration. Wird die Größe der Zellenanordnung unter- oder überschritten, soll die Funktion eine **leere Liste** zurückgeben.

Zellenanordnung

Zellen werden auf einem rechteckigen Gitter mit Zeilen und Spalten angeordnet. Folgende Eckdaten sind dabei zu beachten:

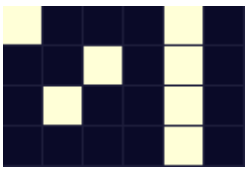
- Die minimale Zeilen- und Spaltenanzahl ist 5
- Die maximale Zeilen- und Spaltenanzahl ist 100
- Es müssen nicht gleichviele Zeilen und Spalten existieren.

Wir betrachten als Beispiel Zellen auf einem Gitter mit 4 Zeilen und 6 Spalten.

Zellenanordnung in Python

```
startkonfiguration = [[1, 0, 0, 0, 1, 0],
                      [0, 0, 1, 0, 1, 0],
                      [0, 1, 0, 0, 1, 0]
                      [0, 0, 0, 0, 1, 0]]
```

Zellenanordnung als Bild



Hierbei sind die hellen Quadrate lebende Zellen.

Aufgabenblatt

Thema: Lernfeld 5: Portfolio-Aufgabe

Dateiformat

Die Datei soll eine Textdatei mit folgendem Aufbau (Header, Daten, Trailer) sein. Die darin abgebildete Zellenanordnung ist identisch zum vorherigen Beispiel.

```
Conway
4 6
START
100010
001010
010010
000010
END
```

Im Header steht in der ersten Zeile das Wort „Conway“, in der zweiten Zeile sind die Dimensionen der Zellenanordnung (Anzahl Zeilen und Anzahl Spalten). In der dritten Zeile steht dann „START“. Im folgenden Datenteil stehen dann zeilenweise die Werte ohne Leerzeichen. Der Trailer besteht einfach aus dem Wort „END“.

Das Einlesen der Startkonfiguration soll über folgende Funktion erfolgen.

```
lade_startkonfiguration(pfad_zur_datei:str) -> List[List[int]]
```

Über den Parameter `pfad_zur_datei` bekommt die Funktion als String den Pfad zur Datei, die eingelesen werden soll.

Zurückgegeben wird eine **Zellenanordnung in dem oben beschriebenen Format**, falls die Datei dem vorgegebenen Dateiformat entspricht. Falls die Datei nicht dem Format entspricht, soll eine **leere Liste** zurückgegeben werden.