

## Übungsblatt 3

### Aufgabe 1: Rumkugeln

Schreiben Sie das Skript `KugelFunktionen.py` der Vorlesung und erweitern Sie das Skript so, dass das Volumen immer korrekt berechnet wird, auch für den Sonderfall  $90^\circ$  (für die Einheitskugel ist dieser Wert:  $4,19\text{m}^3$ ). Erweitern Sie das Skript um die Möglichkeit für eine Kugel bzw. ein Kugelsegment die Oberfläche immer korrekt zu berechnen.

### Aufgabe 2: Schleifen und „switch/case“

Schreiben Sie eine Funktion `displayListe2D(liste)`, die eine 2-dimensionale Liste von Integer-Werten 0-3 so ausgibt, dass jede null als ' ' (Leerzeichen), jede eins als '.' (Punkt), jede zwei als ':' (Doppelpunkt) und jede 3 als '!' (Ausrufungszeichen) ausgegeben wird.

Hinweis: Verwenden Sie zwei ineinander verschachtelte for-Schleifen:

<https://www.python-kurs.eu/for-schleife.php> oder

<https://www.youtube.com/watch?v=B5GhIXhDfoE> (in english)

Verwenden Sie die Möglichkeit von `print()`, einzelne Zeichen in eine Zeile zu schreiben.

Da es in Python keine switch/case-Anweisung gibt, bietet es sich an eine Wörterbuch mit Kombinationen wie `{1: ' . '}` anzulegen, um aus den Werten 0-3 Zeichen zu machen! mit `wörterbuch.get(num, '?')` können Sie sogar für nicht vorhandene Einträge einen Default-Wert (hier '?') definieren:

<https://www.geeksforgeeks.org/switch-case-in-python-replacement/>

### Aufgabe 3: Conways Universum

In dieser Aufgabe schreiben Sie die Erzeugung einer Zufallsmatrix und die Berechnung der Nachbarsumme als Funktionen die Sie in einem Modul abspeichern.

- a) Beginnen Sie zunächst ein neues Projekt, in dem Sie ein Skript `Universe.py` erzeugen.

Informieren Sie sich dann unter:

[https://www.python-kurs.eu/python\\_numpy\\_wahrscheinlichkeit.php](https://www.python-kurs.eu/python_numpy_wahrscheinlichkeit.php)

wie Sie mit Hilfe von `numpy` eine 2-dimensionales Array mit Zufallszahlen erzeugen können.

Schreiben Sie dann in das Skript `Universe.py` eine Funktion `create(rows, *cols)`, die mindestens den Parameter `rows` und bei Bedarf einen Parameter `cols` entgegen-nimmt. Die Funktion soll eine binäre (also ausschließlich Werte 0 oder 1 vom Typ `int`) Zufallsmatrix der Größe `rows`-auf-`cols` als Rückgabewert haben.

Wenn nur ein Parameter übergeben wird, dann soll eine quadratische Matrix mit Zufallswerten erzeugt werden.

Hinweis: `*cols` liefert ein Tupel welches erst in einen Integer gewandelt werden muss. Das Tupel hat die Länge 0 wenn für `*cols` nichts übergeben wurde!

- b) Machen Sie aus dem Skript der Aufgabe 4 (Wer sind meine Nachbarn) von Übungsblatt 2 eine Funktion `neighborCount()`, der Sie eine binäre Matrix sowie eine Zeilenposition und eine Spaltenposition übergeben.

Die Funktion soll die Anzahl (die Summe) der mit 1 besetzten, direkten Nachbarn als Rückgabewert vom Typ Integer haben (min. 0 Nachbarn, max. 8 Nachbarn).

Hinweis: Damit Sie auch die Nachbarn an den Rändern der Matrix ermitteln können, ist es notwendig, das Sie z.B. nicht einfach `row-1` rechnen, sondern `(row-1)%rows`, wobei `rows` die Gesamtanzahl der Zeilen angibt.

Eine Matrix `matrix` hat ihre Dimension in der Eigenschaft `matrix.shape` hinterlegt!

- c) Schreiben sie nun in das Skript `Universe.py` eine Funktion `getNeighbors()`, die für eine Matrix für jede Position die Anzahl der Nachbarn ermittelt und in einer weiteren Matrix `neighbor` speichert und als Rückgabewert zurückgibt, ohne die übergebene Matrix zu verändern.

Dazu erzeugen Sie am besten zu Beginn eine „leere“ (mit Integer-nullen initialisierte) Matrix gleicher Größe.

Verwenden Sie die Funktion `neighborCount()` in einer verschachtelten Schleife, um die Anzahl der Nachbarn für jede Position in der übergebenen Matrix zu ermittelt und in der „leeren“ Matrix zu speichern.

Hinweis: Das Modul `numpy` stellt die Methode `zeros()` zur Verfügung, um eine solche Matrix zu erzeugen siehe:

<https://numpy.org/devdocs/reference/generated/numpy.zeros.html>

- d) Um das Universum ausgeben zu können benötigen wir noch eine Funktion `display()` im Skript `Universe.py`, die eine übergebene Matrix (entweder des Universums oder der Nachbarn) auf der Console als Matrix darstellen kann.

Schreiben Sie dazu erneute zwei ineinander verschachtelte Schleifen, die für jeden Wert der Matrix, der größer null ist einen Stern ausgibt und für jede null der Matrix ein Leerzeichen.

Zur Erinnerung: `print('*', end='')` verhindert die Ausgabe eines Zeilenvorschubs!

Schreiben Sie nun ein neues Python Skript (z.B. `Main.py`), in dem Sie das Skript `Universe.py` importieren, ein Zufalls-Universum erzeugen, das Universum ausgeben und dann die Anzahl aller Nachbarn ermitteln. Die reale Anzahl der Nachbarn können Sie einfach mit `print(neighbor)` darstellen.

### Aufgabe 4: Leben und Sterben lassen!? (Zusatzaufgabe)

In dieser Übung geht es darum auf Basis der Matrix `neighbor` zu entscheiden, welche Sterne im Universum weiter leuchten bzw. neu entzündet werden und welche auskühlen bzw. verglühen!?

Wann ein Stern stirbt, weiterlebt oder zum Leben erweckt wird beschreiben folgende Regeln:

1. Ein toter Stern mit genau drei leuchtenden Nachbarsternen wird im nächsten Zeitschritt neu entzündet.
2. Leuchtende Sterne mit weniger als zwei leuchtenden Nachbarsternen sterben durch auskühlen bis zum nächsten Zeitschritt ab.
3. Leuchtende Sterne mit zwei oder drei lebenden Nachbarn sind in einer gesunden Konstellation und leuchten im nächsten Zeitschritt weiter.
4. Leuchtende Sterne mit mehr als drei leuchtenden Nachbarsternen verglühen bis zum nächsten Zeitschritt auf Grund von zu großer Hitze.

Nachbarsterne sind alle direkten Nachbarn (insgesamt acht) entsprechend der Information in der Matrix `neighbor`!

Versuchen Sie diese Regeln in einer Funktion `nextGeneration(universe, neighbor)` möglichst geschickt umzusetzen, um die nächste Matrizen-Besetzung im Universum zurückgeben zu können!

Wenn Sie wollen können Sie die Iteration der Generationen in einer Schleife laufen lassen und die Sterne des belebten Universums mit der Funktion `Display()` ausgeben!?