Übungsblatt 5 (Objektive Betrachtungen)

Aufgabe 1: Komplexe Klasse

In dieser Aufgabe schreiben Sie eine Klasse mit der es möglich ist komplexe Zahlen anzulegen, auszugeben und mit ihnen Berechnungen anzustellen.

a) Schreiben Sie eine Klasse `Complex`, deren Objekte komplexe Zahlen mit folgenden privaten Eigenschaften (Attributen) sind:

- Realteil der komplexen Zahl
- Imaginärteil der komplexen Zahl
- Schriftzeichen für die imaginäre Zahl ('j' oder 'i')

Entwerfen Sie zunächst einen geeigneten Konstruktor, der ohne jeglichen Parameter eine Komplexe Zahl liefert: `0+0i`
Für den Fall, dass ein Zeichen für die imaginäre Zahl angegeben wird soll geprüft werden, ob es sich um ein korrektes Zeichen handelt, dass dann übernommen wird, ansonsten soll 'i' verwendet werden.

b) Schreiben Sie nun eine Methode `disp()` der Klasse `Complex`, welche die Ausgabe einer komplexen Zahl der Klasse in folgendem Format ermöglicht:

`1.00+2.00i` bzw. `-2.00-1.00j`

Verwenden Sie dazu die Methode `format()` der Stringklasse ähnlich wie:

```
cpxStr = "{0:f}{1:+f}{2}".format(a,b,c)
```

Hinweise: https://pyformat.info/#number_sign
Zusatzaufgabe: Überlegen Sie sich, wie Sie die Anzahl der Nachkommastellen per Parameter steuern können?

c) Die Addition zweier komplexer Zahlen erfolgt durch die Addition der Realteile und der Imaginärteile. Ermöglichen Sie mit einer weiteren Methode die Addition zweier komplexer Zahlen.

d) Schreiben sie nun ein Skript, dass mit Hilfe der Klasse `Complex` ermöglicht zwei Komplexe Zahlen anzulegen, diese zu addieren und auszugeben.

Zusatzaufgabe: Entwickeln Sie eine Methode, mit der Sie durch Eingabe von der Tastatur die Werte einer bestehenden komplexen Zahl überscheiben können. Da die Funktion `input()` immer einen String liefert müssen Sie diesen String analysieren und zerlegen. Mit Hilfe der Methode `endswith()` für Strings können Sie herausfinden, ob überhaupt eine komplexe Zahl eingegeben wurde (Endung ist dann ein `'i'` oder `'j'`). Verwenden Sie die Methode `split()` zum Zerlegen und `if/elif/else`-Anweisungen zur Analyse des Strings, um dann die Einzelteile mit der Funktion `float()` in die Zahlenwerte für den Real- und Imaginärteil zu wandeln. Versuchen Sie bei der Eingabe möglichst viele Eingabevarianten zu akzeptieren, jedoch unkorrekte Eingaben abzuweisen.

Aufgabe 2: Methoden von Standard-Operatoren

Die Verwendung der in Aufgabe 1 geschriebenen Methoden zur Ausgabe und zur Addition sind nicht besonders komfortabel in der Benutzung! Daher bietet Python die Möglichkeit die Methoden von Standard-Operatoren zu überschreiben.

a) Wenn Sie die Funktion `print()` verwenden wird zur Erzeugung der Zeichenkette zur Ausgabe die Methode `__str__()` des auszugebenden Objektes aufgerufen, die dann den String zur Ausgabe zurückliefert. Diese Methode kann in jeder Klasse angepasst (überschrieben) werden, damit das Objekt in der gewünschten Darstellung ausgegeben wird.
Schreiben Sie für die Klasse `Complex` eine Methode `__str__`, die eine Zeichenkette zur Darstellung von komplexen Zahlen liefert.
Danach können Sie einfach mit `print(c)` eine komplexe Zahl c ausgeben.
Hinweise: https://www.python-kurs.eu/python3_vererbung.php
Zusatzaufgabe: Überlegen Sie sich, wie Sie jetzt die Anzahl der Nachkommastellen beeinflussen können?

b) Schreiben Sie nun eine Methode für die Klasse `Complex`, die den Standard-Operator plus `'+'` für die Addition von komplexen Zahlen adaptiert. Danach lassen sich zwei komplexe Zahlen einfach mit `c=c1+c2` addieren!
Hinweise: https://docs.python.org/3.8/library/operator.html
Zusatzaufgabe: Schreiben Sie dementsprechend Methoden für alle Grundrechenarten unter Berücksichtigung aller Rahmenbedingungen!

c) Adaptieren Sie Ihr Script aus Aufgabe 1 so, dass Sie die neuen Berechnungsarten und `print()` zur Ausgabe für komplexe Zahlen verwenden.

Aufgabe 3: Klasse Universum

In dieser Aufgabe schreiben Sie eine Umsetzung des Conway Universums in objektorientierter Art und Weise. Legen Sie für jede Klasse ein eigenes Python-Skript an, das Sie dann später bei Bedarf importieren.

1. Schreiben Sie eine Klasse, die einen einzelnen Stern repräsentiert!
   a) Die Klasse hat einen Konstruktor ohne Parameter, der drei private Attribute setzt:
      - Den Helligkeitsgrad des Sterns, der zufällig auf 0 oder 1 gesetzt wird (später kann es evtl. auch Helligkeitsstufen geben).
      - Eine leere Liste in der acht Nachbarsterne eingehängt werden können
      - Eine Variable für die Anzahl der lebenden Nachbarsterne

   b) Um einen Stern sichtbar zu machen benötigt die Klasse eine Methode zur Ausgabe des Sterns, die '*' für lebende Sterne und ' ' für tote Sterne ausgibt.

   c) Die Klasse benötigt eine Methode, mit der Sie Nachbarsterne in Liste einhängen können, …

   d) eine weitere, um die Anzahl lebender Nachbarsterne zu ermitteln und …

   e) eine letzte, um zu berechnen, ob der Stern auf Grund der Nachbarkonstellation (Anzahl an Nachbarsternen) überleben wird.

2. Eine zweite Klasse soll das Universum beinhalten mit einer variablen Anzahl an Sternen.

   a) Der Konstruktor dieser Klasse hat zwei Parameter zur Angabe der Zeilen- und Spaltenanzahl und setzt folgende Attribute:

      - Anzahl der Zeilen des Universums
      - Anzahl der Spalten des Universums
      - Eine Liste in der Sie für jede Position einen Stern einhängen können

      Der Konstruktor erzeugt zunächst die Liste von Sternen als Matrix entsprechend der Zeilen- und Reihenangabe des Universums.
      Danach ermittelt er für jeden Stern die acht Nachbarsterne und setzt diese mit Hilfe der entsprechenden Methode in die Liste des Sterns ein.

   b) Das Universum benötigt eine Methode um sich komplett darstellen zu können. Diese Methode verwendet die Ausgabemethode der Sterne.

   c) Für die Berechnung der Folgezustände des gesamten Universums ist eine Methode zu schreiben, die:

      - zunächst für alle Sterne die Anzahl der lebenden Nachbarn ermittelt durch Aufrufen der entsprechenden Methode alle Sterne um …
      - dann für alle Sterne zu ermitteln, ob sie im Folgezustand weiterhin bestehen.

3. Ein weiteres Python-Skript erzeugt unter Verwendung des Konstruktors zunächst ein Universum, um dann im Wechsel das Universum darzustellen und zu aktualisieren.