Schiffe versenken

Allgemeines

- Verwenden Sie zur Programmentwicklung als Modulnamen (Dateinamen) für dieses Praktikum den Namen `p02NameVorname.py` und halten sich **strikt** an die in der Aufgabe **vorgegebenen Funktionsnamen und Parameterliste** (Grund: Unit-Test s. u.)

  `Name` ist durch Ihren Nachnamen und `Vorname` durch Ihren Vornamen zu ersetzen. Der Dateiname sollte weiterhin keine Leerzeichen enthalten.

- Bevor Sie Ihre Lösung den Betreuern vorstellen, können Sie mit dem in moodle bereitgestellten Unit-Test `p02unit.py` überprüfen, ob Ihre Lösung die richtigen Ergebnisse liefert.
  - Passen Sie die erste Import-Anweisung im Modul `p02unit.py` gemäß Ihrem Namen an.
  - Im Anaconda-Prompt können Sie den Unit-Test in Ihrem Projektverzeichnis über folgende Anweisung ausführen:
    `python p02unit.py`

- Sollten in der Aufgabenstellung Fragen zu beantworten sein, dann beantworten Sie diese bitte in einem Dokumentationskommentar am Anfang Ihre Python-Datei, z. B.:

```
"""
      Antworten zu den in der Aufgabenstellung gestellten
      Fragen:
      Frage 1: Ihre Antwort
      Frage 2: Ihre Antwort
      ...........
"""
```

Auf einem Raster mit n Zeilen und m Spalten werden (dem "Gebiet") werden mit "Schiffen" belegt. Die Schiffe bestehen dabei aus horizontal oder vertikal nebeneinanderliegenden Feldern. Die Anzahl der Felder wird auch "Länge" des Schiffs genannt.

	0	1	2	3	4
0		X	X	X	
1					
2		X			
3		X			

Die Zeilen und Spalten seien mit 0 startend indiziert. Es ist üblich, die Zeile zuerst zu nennen. Das vertikale Schiff belegt die Felder (2,1) und (3,1).

1 Das Spielfeld

Überlegen Sie sich eine Datenstruktur für die Repräsentation des Gebiets und seine Schiffsbelegung in obigem Beispiel. Definieren Sie eine Funktion `create_area(size)`, die ein leeres Gebiet zurückliefert. `size` ist dabei ein Tupel, das die Höhe und Breite des Gebiets angibt. Der erste Wert im Tuple entspricht dabei der Höhe, der zweite der Breite des Spielfeldes.
Für eine spätere einheitliche Ausgabe ist davon ausgehen, dass weder die Höhe noch die Breite den Wert 10 überschreitet und den Wert 2 unterschreitet. Sollte dies der Fall sein, so ist None

zurückzugeben, andernfalls wird ein Objekt in Form der Datenstruktur eines Spielfelds zurückgegeben.
Testen Sie die Funktion.

2 Die Schiffspositionierung

Definieren Sie eine Funktion `fill_area(area, p0, is_horiz, length)`, die ein Schiff der Länge `length` in das Gebiet `area` ab Position `p0` einträgt.
Das Tuple `p0` beinhaltet dabei den Zeilen- und Spaltenindex der Position. Der erste Wert entspricht dem Zeilenindex.
Der boolesche Übergabeparameter `is_horiz` zeigt an, ob das Schiff horizontal (nach "rechts") oder vertikal (nach "unten") eingetragen werden soll.

Hinweis: Die Funktion überprüft **nicht**, ob das Positionieren überhaupt möglich ist. Diese Aufgabe übernimmt die Funktion in Teil 4.

Tipp: Nutzen Sie für das Positionieren eines horizontalen Schiffes die Slice-Zuweisung. (Also die Angabe eines Slice auf der linken Seite des Zuweisungsoperators.)

Testen Sie die Funktion.

3 Die Spielfeldausgabe

Definieren Sie eine Funktion `print_area(area, title)`, die das Schiffsbelegungsschema durch ein `'X'` visualisiert. Für ein 4 x 5 Spielfeld mit zwei Schiffen in dem Feld, soll die Ausgabe wie folgt aussehen:

```
Spieler 1
  |01234|
0| XXX |
1|     |
2| X   |
3| X   |
  |01234|
```

In der ersten Zeile wird der Titel (Argument des 2. Parameters) ausgegeben, im Beispiel der String `'Spieler 1'`.
Für die Ausgabe der zweiten Zeile ist es sinnvoll ein Funktion `print_column_numbers(n)` zu erstellen, da die Ausgabe in der letzten Zeile wiederholt wird.

Tipp: Die Zeile `width, height = len(area[0]), len(area)` zu Beginn der Funktion ist sinnvoll.

Tipp: Ein Zeichen `'X'` können Sie ohne anschließenden Zeilenvorschub durch `print('X', end='')`

Tipp: Sollten Sie mehrere Schiffe entdecken, als Sie positioniert haben, dann liegt das in den meisten Fällen an einer fehlerhaften `create_area`-Funktion.
Meist speichern Sie die **Referenz** einer Zeile für alle Zeilen im Spielfeld ab, anstelle jeweils eines neuen Objektes (*flache* oder *tiefe Objektkopie*).
Objektkopien können mithilfe einer geeigneten Konstruktion (*Comprehensions*), der Methode `copy()` oder eine Slice-Zuweisung auf `objekt[:]` ausgeführt werden. Probieren Sie alle drei Möglichkeiten aus.

4 Die Positionsprüfung

Definieren Sie eine Funktion `check_area(area, p0, is_horiz, length)`, die überprüft, ob ein Schiff der Länge `length` in das Gebiet `area` ab Position `p0` eingetragen werden kann, ohne sich mit einem

schon bestehendes Schiff zu überlappen. Außerdem ist vorab zu prüfen, ob der Zeilen- bzw. Spaltenindex gültig ist (also weder die möglichen Werte unter- noch überschreitet) und das Schiff an der angegebenen Position überhaupt in das Spielfeld passt.
Im Fall eines möglichen Eintrags soll die Funktion `True` zurückliefern, sonst `False`.

Für Profis: Die offiziellen Schiffe-Versenken-Regeln verlangen ein leeres Feld Abstand zwischen zwei Schiffen.
Ergänzen Sie die Funktion um den optionalen Parameter `profi_check`:
`check_area(area, p0, is_horiz, length, profi_check=False)`
Sollte für dieses Argument `True` übergeben werden, so wird eine Schiffsbelegung nur als gültig bewertet wird, wenn auch die Umgebung frei ist.

5 Die zufällige Schiffspositionierung

Definieren Sie eine Funktion `generate_boat(area, boat_spec)`, die ein Schiff der Länge `boat_spec` im Gebiet `area` zufällig belegt.

Tipp: Eine gleichverteilte zufällige ganze Zahl im Intervall von `a` bis einschließlich `b` erhalten Sie mittels

```
import random                                   # nur einmal zu Beginn der Datei
…
zahl = random.randint(a,b)
```

Sollte die zufällige Schiffspositionierung scheitern, so ist diese mit einer neuen zufälligen Position/Ausrichtung zu wiederholen. Nach einer – von Ihnen sinnvoll – festgelegten Anzahl von Versuchen ist die Funktion ergebnislos zu beenden.
Hinweis: Nutzen Sie die bereits erstellten Funktionen.

6 Die zufällige Mehrfach-Schiffspositionierung

Erweitern Sie die Funktion `generate_boat(area, boat_spec)` um die Möglichkeit mehrere Schiffe anzulegen. Hierzu soll `boat_spec` neben einer ganzen Zahl auch ein Dictionary enthalten dürfen, das die Länge und Anzahl der gewünschten Schiffe als Schlüssel und Inhalt des Dictionaries angibt. So soll z. B. `generate_boat(area, {2:2, 3:1})` zwei Schiffe der Länge 2 und ein Schiff der Länge 3 generieren.
Testen Sie die Funktion.

Hinweis: Nutzen Sie die Möglichkeit, `generate_boat` mit einem Integer als `boat_spec` aufzurufen. (Rekursiver Aufruf derselben Funktion)

Tipp: Denken Sie an die Möglichkeit, über Dictionaries mit

```
for (key,val) in dictionary.items():
```

zu iterieren.

Für Profis:
Erweitern Sie das Programm um die Möglichkeit Schiffe versenken gegen das Programm zu spielen.