Aufgabenblatt 5

Allgemeine Informationen zum Aufgabenblatt:

- Die Abgabe erfolgt in TUWEL. Bitte laden Sie Ihre Python-Datei bis spätestens **Freitag, 15.05.2020 20:00 Uhr** in TUWEL hoch.
- Beachten Sie bitte folgende Punkte
  - Ihre Programme müssen ausführbar sein, d.h. einzelne Programme sollten z.B. keine Syntaxfehler oder Typfehler enthalten.
  - Bei kleinen logischen Fehlern (falsche Ergebnisse) werden wir keine Punkte abziehen. Daher sollten Sie immer versuchen, alle Aufgaben vollständig abzugeben.
  - Ihre Programme sollten nur Konstrukte verwenden, die bisher in der Vorlesung vorgekommen sind. Sie sollten daher keine speziellen Aufrufe verwenden, die möglicherweise einzelne Aufgaben abkürzen. Bitte beachten Sie auch die weiteren Einschränkungen bei bestimmten Unteraufgaben.
  - Die Angabedatei muss nicht umbenannt werden.

In diesem Aufgabenblatt werden folgende Themen behandelt:

- Listen, Mengen, Dictionaries
- Ausnahmen
- Dateien

Aufgabe 1 (1 Punkt)

In dieser Aufgabe machen Sie sich mit Dictionaries vertraut. Dazu gibt es schon vorgegebenen Programmcode, zu dem Sie jeweils Erweiterungen implementieren müssen. Diese Erweiterungen stehen jeweils als Kommentar vor dem dazugehörigen Programmcode (Dictionary). Es wird auch die erwartete Ausgabe beschrieben.

Aufgabe 2 (1 Punkt)

Sie haben die Implementierung einer Funktion `find_root(x, power, epsilon)` gegeben. Zusätzlich gibt es die Funktion `print_vals`. Diese Funktionen testet `find_root` mit korrekten Aufrufen (Werten). Ihre Aufgabe ist es nun, die Funktion `find_root` so zu erweitern, dass bestimmte Situationen frühzeitig abgefangen und entsprechende Ausnahmen generiert werden. Folgende Ausnahmen werden in der Funktion geworfen (aber nicht behandelt):

- Sobald eines der Argumente nicht vom Typ `int` oder `float` ist, dann wird ein `TypeError` mit der Nachricht `Arguments should be of type int or float` geworfen.
- Falls `x < 0` und zusätzlich `power` eine gerade Zahl ist, dann wird ein `ValueError` mit der Nachricht `Negative number has no even-powered roots` geworfen.
- Falls `power < 1` ist, dann wird ein `ValueError` mit der Nachricht `power should be >= 1` geworfen.
- Falls `epsilon <= 0` ist, dann wird ein `ValueError` mit der Nachricht `epsilon should be > 0` geworfen.

Darauf aufbauend können Sie die Funktion `print_evil_vals` erweitern. Diese Funktion sollte die Werte in der Liste `evil_vals` zum Testen von `find_root` verwenden. Es sollten aber die von der Funktion generierten Ausnahmen abgefangen und ausgegeben werden.
Beschreiben Sie weiters in einem Kommentar:

- Was bewirkt der * beim Aufruf `find_root(*i)`?
- Bei welchen der Aufrufe werden die Ausnahmen nicht von ihrem Erweiterungscode generiert?

Aufgabe 3 (1 Punkt)

In dieser Aufgabe haben Sie schon Teile eines Programms gegeben, mit dem Sie das Spiel Tic-Tac-Toe (`https://de.wikipedia.org/wiki/Tic-Tac-Toe`) spielen können. Das gegebene Spielfeld hat fix 3 Zeilen mit jeweils 3 Spalten. Die einzelnen Funktionen arbeiten mit diesem Spielfeld. Eine der Funktionen ist zwar vorhanden, aber macht aktuell nichts[1] Es können schon Eingaben eingelesen werden und die zwei unterschiedlichen Zeichen (`X` oder `O`) platziert werden. Sie müssen nun das Programm fertig stellen:

- Vervollständigen Sie die Funktion `is_win(board, act_mark)`. Diese Funktion liefert `True` zurück, wenn das Zeichen `act_mark` dreimal in einer Zeile, oder Spalte oder einer der beiden Diagonalen des Spielfelds `board` auftritt. Also z.B. für `X` in der obersten Zeile bei

  ```
  X|X|X
  -----
   |O|
  -----
   | |O
  ```

  Sie dürfen hier auch zusätzliche eigene Hilfsfunktionen implementieren.

- Ergänzen Sie die Funktion `run_game` so, dass die Ausnahmen von der Funktion `mark` nicht zu einem Abbruch führen. Stattdessen sollte die Fehlermeldung ausgegeben und das Programm fortgesetzt werden.

- Stellen Sie weiters sicher, dass bei einem Unentschieden das Spiel auch korrekt beendet wird. Sie dürfen zusätzliche Variablen einführen.

Nachfolgend ein Beispiel für einen Ablauf. Bitte beachten Sie, dass es natürlich viele unterschiedliche Abläufe gibt und Sie auch die Situation mit einem Unentschieden austesten sollten:

```
Player X choose your field:
Row number: 0
Column number: 0
X| |
-----
 | |
-----
 | |
Player O choose your field:
Row number: 3
Column number: 3
Invalid board position
Player O choose your field:
Row number: 2
Column number: 1
```

[1] Wird ein Funktionskopf angegeben, dann muss zumindest eine Anweisung im Funktionsrumpf stehen. In diesem Fall ist es die Anweisung **pass**, die eben nichts macht.

```
X| |
-----
 | |
-----
 |O|
Player X choose your field:
Row number: 1
Column number: 1
X| |
-----
 |X|
-----
 |O|
Player O choose your field:
Row number: 2
Column number: 1
Board position occupied
Player O choose your field:
Row number: 2
Column number: 0
X| |
-----
 |X|
-----
O|O|
Player X choose your field:
Row number: 2
Column number: 2
X| |
-----
 |X|
-----
O|O|X
X wins
```

Aufgabe 4 (2 Punkte)

In dieser Aufgabe implementieren Sie ein sehr einfaches Telefonbuch mit folgenden Funktionen:

- `create_telbook()`: In dieser Funktion wird versucht, eine Datei `tel.txt` zu öffnen, in der Telefonbucheinträge stehen. Jede Zeile enthält einen Namen (Typ String) und eine Telefonnummer (auch Typ String) getrennt durch ein Leerzeichen. Wir gehen davon aus, dass nur ein Vorname (ohne Whitespaces) und eine Nummer (ohne Whitespaces) gespeichert werden. Ist es möglich, die Datei zu öffnen, dann wird der Inhalt der Datei eingelesen und jede Zeile ergibt einen Eintrag in einem Dictionary, wobei der Name dem Schlüssel und die Telefonnummer dem Wert entspricht. Ist die Datei nicht vorhanden, dann wird ein leeres Dictionary erzeugt. Die Funktion gibt am Ende immer ein Dictionary zurück.

- `search(numbers)`: Mit dieser Funktion wird in einem Dictionary `numbers` gesucht. Dazu wird in der Funktion ein Name eingelesen. Danach werden zwei Fälle unterschieden:

  - Der Name ist in `numbers` vorhanden. Dann wird in einer Zeile die dazugehörige Telefonnummer ausgegeben.
  - Der Name ist nicht in `numbers` vorhanden. Dann wird eine Meldung am Bildschirm ausgegeben.

  Die Funktion liefert nichts zurück.

- `new_number(numbers)`: Mir dieser Funktion wird ein neuer Eintrag im Dictionary `numbers` erzeugt. Dazu werden der Name und die Telefonnummer von der Konsole eingelesen und eine Bestätigungsmeldung für die Aktion (z. B. New entry accepted!) ausgegeben. Gehen Sie dabei davon aus, dass Name und Nummer korrekt eingegeben werden. Sie müssen daher keine spezielle Behandlung dafür implementieren. Die Funktion liefert nichts zurück.

- `print_numbers(numbers)`: Diese Funktion gibt alle Einträge im Dictionary `numbers` aus. Es wird zunächst eine Kopfzeile ausgegeben, dann folgt eine Trennzeile und danach jeder Eintrag im Dictionary in einer eigenen Zeile (mögliche Formatierung siehe Beispiele).

- `menu(numbers)`: Mit dieser Funktion werden wiederholt Eingaben von der Konsole eingelesen. Dazu wird ein Menu ausgegeben, das folgende Einträge enthält:

  - (S)earch for number
  - (N)ew number
  - (A)ll numbers
  - (E)nd

  Abhängig von der Eingabe wird eine der oben beschriebenen Funktionen mit dem Dictionary `numbers` aufgerufen oder die Funktion beendet. Bei der Eingabe werden sowohl Groß- als auch Kleinbuchstaben akzeptiert. Bei falschen Eingaben wird das Menu wieder neu angezeigt.

- `print_goodbye(numbers)`: Diese Funktion gibt einen Verabschiedungstext aus und speichert alle aktuellen Einträge (ein Eintrag pro Zeile, Name und Nummer durch Leerzeichen getrennt) in der Datei `tel.txt`.

Das Hauptprogramm ist schon in der Angabedatei vorhanden und gibt den Ablauf vor. Bitte beachten Sie dabei, dass die Funktion `menu` die von ihr benötigten Funktionen aufrufen muss.
Ein möglicher Ablauf sieht so aus (`tel.txt` existiert noch nicht):

```
(S)earch for number
(N)ew number
(A)ll numbers
(E)nd

Your choice: n
Name: Bob
Number: 234567
New entry accepted!

(S)earch for number
(N)ew number
(A)ll numbers
(E)nd

Your choice: n
Name: Alice
Number: 5792938
New entry accepted!

(S)earch for number
(N)ew number
(A)ll numbers
(E)nd

Your choice: a
Name Number
------------------------------
Bob 234567
Alice 5792938

(S)earch for number
(N)ew number
(A)ll numbers
(E)nd

Your choice: e
Thank you for using this little phone book.
```

Ein weiterer Aufruf sieht so aus (`tel.txt` wird von vorher übernommen):

```
(S)earch for number
(N)ew number
(A)ll numbers
(E)nd

Your choice: s
Name: Bob
Number: 234567

(S)earch for number
(N)ew number
(A)ll numbers
(E)nd

Your choice: s
Name: Donald
Unknown name!

(S)earch for number
(N)ew number
(A)ll numbers
(E)nd

Your choice: n
Name: Donald
Number: 111222
New entry accepted!

(S)earch for number
(N)ew number
(A)ll numbers
(E)nd

Your choice: a
Name Number
------------------------------
Bob 234567
Alice 5792938
Donald 111222

(S)earch for number
(N)ew number
(A)ll numbers
(E)nd

Your choice: e
Thank you for using this little phone book.
```