Python – Testat 1

1 Einleitung

Für die HSRvote-Software besteht Bedarf für die automatische Verwaltung der Quiz-Resultate und die Erstellung einer Rangliste. Es soll dafür eine eigene Klasse `QuizRangliste` implementiert werden. Als Basis für die Datenspeicherung dient eine Textdatei, wie z.B. in Listing 1 dargestellt. Auf jeder Zeile wird der Name des Teilnehmers und sein persönliches Gesamtresultat (erreichte Punktzahl und benötigte Zeit) aufgelistet.

```
1 Noemi,7,40.0
2 Max,6,100.02
3 Anna,5,20.55
4 Laura,5,20.55
5 Fritz,5,39.3
6 Hans,2,45.24
```

Listing 1: Inhalt der Datei rangliste.txt

Die Klasse `QuizRangliste` soll als Schnittstelle zwischen der Textdatei und der restlichen Software dienen. Konkret bedeutet dies: man instanziert ein `QuizRangliste`-Objekt für jede Ranglistendatei, die man verwalten will, z.B. eine pro Teilnehmergruppe. Mit den dedizierten Methoden `als_dictionary()`, `als_liste()` und `als_string()` kann man die sortierten Daten im entsprechenden Format extrahieren. Mit den Methoden `resultat_addieren()` und `name_entfernen()` können die Daten modifiziert werden. Mit der Methode `speichern()` werden die Daten dann wieder zurück in die Textdatei gespeichert.

2 Aufgabenstellung und Bewertungskriterien

☞ *Implementieren Sie die Klasse `QuizRangliste`, wie unten im Detail beschrieben, und liefern Sie diese als Python-Datei dem Dozenten bis spätestens am 17. April 2019 um 10 Uhr per Email (nramagna@hsr.ch) ab.*

Bewertungskriterien: Für jedes Detail, das richtig implementiert wird, gibt es Punkte. Rechts neben jeder Detailbeschreibung, wird die Zahl der möglichen Punkte in Klammern angegeben. In diesem Testat 1 gibt es total 38 Punkte. Die unten angegebenen Klassen-, Methoden-, Parameter- und Key-Namen müssen exakt übernommen werden, sonst werden für den betroffenen Teil keine Punkte vergeben. Es wird empfohlen, den eigenen Code ausgiebig zu testen. Die Beispiele in den grauen Listing-Boxen eignen sich für einen ersten Vergleich.

Prüfungszulassung: Um an der Prüfung zugelassen zu werden, muss mindestens 50% der Summe der maximal erreichbaren Punkte aus beiden Testaten 1 & 2 erreicht werden.

3 Klasse `QuizRangliste`

- Nur die unten aufgelisteten Methoden dürfen *public* sein. (1 P)
- Es gibt keine *public* Instanzvariablen. (1 P)
- Die Klasse und deren Methoden besitzen aussagekräftige Docstrings. (4 P)
- Python-Codestil entspricht den PEP8-Empfehlungen. 1 P Abzug pro Stilfehler[1]. (6 P)

3.1 Methode `__init__(datei)`

- Die Methode liest die angegebene Textdatei (`encoding='utf-8'`) ein und speichert die Daten in eine nicht-öffentliche Instanzvariable ab. (1 P)
- Leere oder ungültige Zeilen werden ignoriert. (1 P)
- Erstellt eine neue leere Datei, falls diese noch nicht existiert. (1 P)

```
>>> qr = QuizRangliste(datei='rangliste.txt')
```

3.2 Methode `als_dictionary()`

- Die Methode gibt die intern gespeicherten Daten als Dictionary von Dictionaries zurück: **Key** = *name*, **Value** = {`'Punkte'`: *punkte*, `'Zeit'`: *zeit*} (1 P)
- Die Datentypen der Elemente sind: (1 P)
  `type(name) == str`,
  `type(punkte) == int`,
  `type(zeit) == float`

```
>>> d = qr.als_dictionary()
>>> print(d)
{'Noemi': {'Punkte': 7, 'Zeit': 40.0}, 'Max': {'Punkte': 6,
   'Zeit': 100.02}, 'Hans': {'Punkte': 2, 'Zeit': 45.24}, 'Anna':
   {'Punkte': 5, 'Zeit': 20.55}, 'Fritz': {'Punkte': 5, 'Zeit':
   39.3}, 'Laura': {'Punkte': 5, 'Zeit': 20.55}}
```

[1] der Codestil wird mit der Flake8-Software geprüft, `http://flake8.pycqa.org/en/latest/`

3.3 Methode `als_liste()`

- Die Methode gibt eine Liste von Tupeln (*name*, *punkte*, *zeit*) zurück. (1 P)
- Die Datentypen der Elemente sind: (1 P)
  `type(name) == str`,
  `type(punkte) == int`,
  `type(zeit) == float`
- Die Liste ist nach den folgenden Prioritäten sortiert: (3 P)
  1. Punkte, absteigend
  2. Zeit, aufsteigend
  3. Name, alphabetisch aufsteigend: A→Z

```
>>> lst = qr.als_liste()
>>> print(lst)
[('Noemi', 7, 40.0), ('Max', 6, 100.02), ('Anna', 5, 20.55),
   ('Laura', 5, 20.55), ('Fritz', 5, 39.3), ('Hans', 2, 45.24)]
```

3.4 Methode `als_string()`

- Die Methode gibt die Daten als formatierten String zurück. (1 P)
- Das Zeilen-Format ist: `Name | Punkte | Zeit`, wobei `Punkte` als Ganzzahl und `Zeit` als Float (auf 1 Kommastelle gerundet) ausgegeben werden. (1 P)
- Die Spaltenbreite ist für alle Zeilen gleich und wird dynamisch anhand des längsten Elements in der jeweiligen Spalte ermittelt. Die Ausrichtungen sind: Name=linksbündig, Punkte=rechtsbündig, Zeit=rechtsbündig. (2 P)
- Die Reihenfolge ist gleich wie bei der `als_liste()`-Methode. (1 P)

```
>>> s = qr.als_string()
>>> print(s)
Noemi | 7 |  40.0
Max   | 6 | 100.0
Anna  | 5 |  20.6
Laura | 5 |  20.6
Fritz | 5 |  39.3
Hans  | 2 |  45.2
```

3.5 Methode `resultat_addieren(name, punkte, zeit)`

- Falls *name* bereits in den internen Daten existiert: *punkte* und *zeit* hinzuaddieren. (1 P)
- Falls *name* nicht existiert: neuen Namen anlegen. (1 P)
- Das Argument *name* darf keine Kommazeichen enthalten. (1 P)
- Die Datentypen der Argumente sind: (1 P)
  `type(name) == str`
  `type(punkte) == int` oder `str`,
  `type(zeit) == float` oder `str`
- Rückgabewert: `False` bei ungültigen Argumenten, `True` falls fehlerfrei. (1 P)

```
>>> qr.resultat_addieren(name='Noemi', punkte=7, zeit=50.4)
>>> qr.resultat_addieren(name='Noemi', punkte='7', zeit='50.4')
>>> print(qr.als_string())
Noemi | 21 | 140.8
Max   |  6 | 100.0
Anna  |  5 |  20.6
Laura |  5 |  20.6
Fritz |  5 |  39.3
Hans  |  2 |  45.2
```

3.6 Methode `name_entfernen(name)`

- Falls *name* in den internen Daten existiert: Name (inkl. Punkte und Zeit) löschen. (1 P)
- Falls *name* nicht existiert: nichts tun. (1 P)

```
>>> qr.name_entfernen(name='Fritz')
>>> print(qr.als_string())
Noemi | 21 | 140.8
Max   |  6 | 100.0
Anna  |  5 |  20.6
Laura |  5 |  20.6
Hans  |  2 |  45.2
```

3.7 Methode `speichern(als=None)`

- Schreibt die Daten als Komma-getrennte Werte in die ursprüngliche Datei, wie in Listing 1 gezeigt: Name,Punkte,Zeit (1 P)
- Falls das Argument `als` angegeben wird: die Daten in die angegebene Datei anstatt in die ursprüngliche Datei schreiben. (1 P)
- Die Reihenfolge ist gleich wie bei der `als_liste()`-Methode. (1 P)
- Die Datei wird mit `encoding='utf-8'` geschrieben. (1 P)

```
>>> qr.speichern()
>>> qr.speichern(als='neue_rangliste.txt')
```