**Python - Programmentwurf**

**Team Mitglieder:**

|  |  |
| --- | --- |
| **Name** | **e-Mail (DHBW)** |
| Thomas Gingele |  |
| Lukas Eisele | eisele.lukas-it21@it.dhbw-ravensburg.de |
| Jannik Künstler |  |
| Felix Schladt |  |
|  |  |

**Verwendete Python Version: 3.9.7, 3.10.4**

**Verwendete Pylint Version: 2.13.7**

**Verwendete nachinstallierte Bibliotheken: keine**

**Beschreibung des Dateiformats zum Speichern des Spiels:**

* {"turns": 2, "player\_0": {"name": "Lukas", "dices": [1, 1, 2, 5, 1], "flags": [false, true, true, true, true, false, false, true, true, false, false, false, true], "scores": [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], "active": true}, "player\_1": {"name": "Thomas", "dices": [6, 2, 4, 6, 5], "flags": [false, false, true, true, true, true, false, false, true, false, true, false, true], "scores": [0, 0, 0, 6, 0, 10, 12, 28, 0, 0, 0, 25, 0, 40, 0, 13, 106], "active": false}}
* Die Datei wird im json-Format gespeichert. Jeder Spieler besteht aus einem dictionary. In diesem dictionary steht jeweils der Name, die geworfenen Augenzahlen der Würfel und der Spielstand.

**Beschreibung des Benutzerinterfaces:**

* Wie sieht die Ein- und Ausgabe aus?

**Beschreibung der Architektur:**

* Grobe Architektur beschreiben (Textuell, oder Diagramm)

→ ./doc/depency\_diagramm.png

→ ./doc/structure\_diagramm.png

* Ca. halbe Seite bis eine Seite

**Dokumentation von einem kompletten Spielablauf:**

* Kopie der kompletten Ausgabe auf der Konsole. Evtl. eigene Datei

→ ./doc/played\_game\_record.log

* + Datei ist mittels tee erstellt → kopie von stdout
  + Ausgabe ohne color codes auf stdout:
    - cat ./doc/played\_game\_record.log > /dev/tty

Kann jedoch falsch aussehen, falls sich die terminal größe unterscheidet

**Log von den Tests:**

* Alle Tests über Konsole ausführen und Ausgabe hier einfügen (oder extra Datei)

→ ./doc/working\_game\_screenshot.png

→ played\_game\_record.log → this was recorded with tee and therfore contains all data printed to stdout → to gain a representative view of the ouptut please pip it into yout terminal → cat ./doc/played\_game\_record.log > */dev/tty*

**Bewertung der Testergebnisse:**

* Grund für fehlschlagende Tests

**Code-Coverage Ausgabe:**

* Tests mit Coverage ausführen und Ausgabe hier einfügen

**Bewertung der Coverage und Sinnvollheit der Tests:**

* Gewünscht ist eine Coverage von min 75% (pro Datei)
  + Wurde dies erreicht? Wenn nein, warum nicht?
* Begründet warum ihr euch sicher seid alles mit Tests abgedeckt zu haben
* Fehlerfälle in den Tests überprüft?

**Bewertung der Fehlersicherheit:**

* Werden Fehleingaben korrekt abgefangen?
  + Nenne 2 Beispiele wo ein Fehler abgefangen wird

**Pylint Ausgabe:**

* Pylint auf der Konsole Ausführen und Ergebnis einfügen

make pylint | tee doc/pylint.log

pylint ./src/

\*\*\*\*\*\*\*\*\*\*\*\*\* Module src.game\_engine

src/game\_engine.py:40:34: E1120: No value for argument 'self' in unbound method call (no-value-for-parameter)

src/game\_engine.py:62:40: E1120: No value for argument 'self' in unbound method call (no-value-for-parameter)

\*\*\*\*\*\*\*\*\*\*\*\*\* Module src.terminal

src/terminal.py:14:4: E0401: Unable to import 'msvcrt' (import-error)

src/terminal.py:24:0: R0903: Too few public methods (0/2) (too-few-public-methods)

------------------------------------------------------------------

Your code has been rated at 9.72/10 (previous run: 9.72/10, +0.00)

**Bewertung der Code-Qualität/Lesbarkeit:**

* Begründet noch vorhandene Pylint Warnungen

1. , 2. This is caused by us using the @staticmethod property on some functions in ./src/terminal.py.  
 This was done because those funcitons do access data of the class but its functionality is

platform dependend and therefore needs to be included in the class.

3. This error is specific to a non Windows environment. The msvcrt module is a built in pyhton module that

provides windows specific functionality.

A similar Error should appear about the Modules "tty" and "termios" on Windows (I can't check this).

4. The class in question is the Colors class in terminal.py which provides easy access to ASCII colorcodes.

This class is just a wrapper for constants and therefore does not have any functions related to it.

We believe this is the neatest way to bundle these constants and therefore ignored pylint's

requirement.

* Begründet warum euer Code gut lesbar ist

Wir haben uns intern auf einige Regeln für guten Code geeinigt.

1. Guter Code ist prägnanter Code → Kurzer Code ist guter Code und erhöht die Lesbarkeit.

1. Python gibt dem Programmierer dank inline if abfragen und list comprehensions die Möglichkeit. Möglcihst gut lesbaren und prägnanten Code zu schreiben.
2. Python ist nicht Java → Objektorientierung ist gut und schön, aber nicht immer Sinnvoll.   
   Desshalb haben wir absichtlich auf Vererbung und unnötigen Boilerplate-Code verzichtet.
3. Segmentierung → Wir haben versucht unserern Code in möglichst logische Unterteilungen aufzuteilen.
4. Consistency → Wir haben versucht unsere Libraries möglichst konstitent zu halten, um eine einfache Bentuzung aller Teammitglieder zu ermöglichen.

**Bewertungstabelle:**

* Ausgefüllte Bewertungstabelle