Danny Rehl, Programmieren I - Python

Übungsblatt 08

Abgabefrist: Montag, den 16.12.2024, um 23:59 Uhr

Sie finden für dieses Übungsblatt eine Vorlage für Ihre Abgabe zum Herunterladen im Moodle.

Für die Bewertung Ihrer Abgabe zählen die von Ihnen vervollständigten Funktionen im oberen Teil der Vorlage. Schreiben Sie hier nur Ihre Lösungen hin, keine weiteren Befehle außerhalb der Funktionen.

Im unteren Teil finden Sie einen `if __name__ == '__main__'`-Abschnitt, welcher bereits einige Beispielaufrufe zu den Funktionen beinhaltet. Hier können Sie die Befehle beliebig ergänzen oder ersetzen, um Ihre Funktionen zu testen.

Hinweis: Trotz aller Tests muss die hochgeladene `.py`-Datei natürlich immer noch ausführbar bleiben. Beispielsweise wird ein `SyntaxError` im `if __name__ == '__main__'`-Teil in jedem Fall zur Pytest-Bewertung von 0 % führen.

Bitte achten Sie darauf, dass Ihr Code PEP 8 konform geschrieben ist. Das ist Voraussetzung für die abschließende Bewertung. Nutzen Sie die Hinweise des Pylint-Scorers, um die verbleibenden Style-Fehler zu finden.

Aufgabe 1 (2 Punkte)

Sie finden in der Vorlage folgendes Beispielobjekt `KNIGHTS`:

```
KNIGHTS = [
    ('Arthur', 34),
    ('Bedevere', 33),
    ('Galahad', 32),
    ('Lancelot', 36)
]
```

Ändern Sie die Funktion `sort_by_age`, sodass die per Parameter `knights` übergebene Liste von Rittern nach deren Alter in aufsteigender Reihenfolge sortiert wird. Benutzen Sie dafür die Funktion `sorted` mit ihrem Keyword-Argument `key`. Verwenden Sie allerdings direkt eine `lambda`-Funktion anstelle einer regulären `def`-Funktion als Wert für das Keyword-Argument.

Für `sort_by_age(KNIGHTS)` erwarten wir:

```
[('Galahad', 32), ('Bedevere', 33), ('Arthur', 34), ('Lancelot', 36)]
```

Aufgabe 2 (2 Punkte)

Ändern Sie die Funktion `young_knights`, sodass diese aus der als Parameter `knights` übergebenen Liste nur diejenigen Ritter (in unveränderter Reihenfolge) zurückgibt, welche jünger als 34 Jahre sind. Verwenden Sie hierfür keine reguläre `for`- oder `while`-Schleife, sondern `filter` und eine `lambda`-Funktion.

Hinweis: `filter` erzeugt einen Iterator. Vergessen Sie deshalb nicht, den Funktionsrückgabewert am Ende durch `list` in eine Liste umzuwandeln.

Für `young_knights(KNIGHTS)` erwarten wir:

```
[('Bedevere', 33), ('Galahad', 32)]
```

Aufgabe 3 (2 Punkte)

Ändern Sie die Funktion `get_knight_names`, sodass diese aus der als Parameter `knights` übergebenen Liste das erste Element der in der Liste enthaltenen Tupel extrahiert und als Liste zurückgibt. Verwenden Sie hierfür keine reguläre `for`- oder `while`-Schleife, sondern eine List-Comprehension.

Für `get_knight_names(KNIGHTS)` erwarten wir:

```
['Arthur', 'Bedevere', 'Galahad', 'Lancelot']
```

Aufgabe 4 (2 Punkte)

Ändern Sie die Funktion `convert_to_dict`, sodass diese die als Parameter `knights` übergebene Liste in ein Dictionary konvertiert, wobei das erste Element der in der Liste enthaltenen Tupel den Schlüssel darstellt und das zweite Element den Wert. Der zunächst als Integer vorliegende Wert (das Alter) soll dabei zu einem String umgewandelt werden. Verwenden Sie hierfür keine reguläre `for`- oder `while`-Schleife, sondern eine Dictionary-Comprehension.

Für `convert_to_dict(KNIGHTS)` erwarten wir:

```
{'Arthur': '34', 'Bedevere': '33', 'Galahad': '32', 'Lancelot': '36'}
```

Aufgabe 5 (2 Punkte)

Vervollständigen Sie die Funktion `prime_numbers`, sodass diese als Generator funktioniert, welcher - in aufsteigender Reihenfolge - eine Primzahl nach der anderen bestimmt und zurückgibt.

Die ersten 20 Primzahlen sollen dann beispielsweise so abgerufen werden können:

```
>>> NUMBERS = prime_numbers()
>>> [next(NUMBERS) for _ in range(20)]
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71]
```

(Der Generator soll natürlich genauso für die ersten 5 oder 100 Primzahlen funktionieren. 20 ist nur ein Beispiel.)

Hinweis: Wenn eine Zahl durch eine andere Zahl teilbar ist, dann ist bleibt kein Rest bei der Ganzzahldivision (Modulo). Der Modulo-Operator in Python ist `%`.

Sie müssen übrigens keine Optimierungen an Ihrem Algorithmus vornehmen. Solange das richtige Ergebnis produziert wird, gilt die Aufgabe als erfüllt.