Danny Rehl, Programmieren I - Python

Übungsblatt 03

Abgabefrist: Montag, den 11.11.2024, 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.

Aufgabe 1 (1,5 Punkte)

Bisher haben wir für die Übungsaufgaben nur bereits existierende Built-in Functions oder Methoden von Objekten aufgerufen. Für dieses Übungsblatt sollen Sie Ihr neues Wissen aus der Vorlesung anwenden und eigene Funktionen schreiben. Sie finden für alle Funktionen immer schon einen passenden Funktionskopf in der Vorlage.

Vervollständigen Sie die Funktion `print_square`. Die Funktion soll keine Parameter haben, der Benutzer wird stattdessen um Eingabe einer Zahl per Tastatur gebeten. Diese Zahl soll dann quadriert und wieder ausgegeben werden.

Ein Aufruf könnte beispielsweise so aussehen:

```
Bitte geben Sie eine Zahl ein: 3
9
```

Hinweis: Machen Sie sich hier auch nochmal den Unterschied klar zwischen einem Rückgabewert einer Funktion gegenüber einer Ausgabe eines Werts per `print`. Die Funktion soll keinen Wert zurückgeben. (Das entspricht einem Rückgabewert von `None`, also `return None`, oder einfach `return` oder gar kein `return`.)

Aufgabe 2 (1,5 Punkte)

Vervollständigen Sie die Funktion `account_number`. Sie hat einen Parameter `number` mit welchem eine Kontonummer als Typ Integer übergeben wird. Die Funktion soll dieser Nummer so viele Nullen voranstellen, dass sich insgesamt 9 Ziffern ergeben. Dieses Ergebnis soll als String zurückgegeben werden. Sollte `number` eine negative Zahl sein, weniger als 3 Ziffern oder mehr als 9 Ziffern besitzen, dann soll `None` zurückgegeben werden. (Negative Kontonummern sowie Kontonummern von Länge kleiner 3 oder größer 9 werden von dieser Bank nicht vergeben.)

Ein paar Beispiele:

- `account_number(12345)` soll `"000012345"` zurückgeben
- `account_number(-999)` soll `None` zurückgeben
- `account_number(11)` soll `None` zurückgeben
- `account_number(1111111111)` soll `None` zurückgeben

Achtung: Bitte hier den Unterschied klar machen zwischen `None` (dem `NoneType`) und `"None"` (dem String).

Hinweis: Es gibt - wie so oft - mehrere Möglichkeiten, diese Aufgabe zu lösen. Welchen Weg Sie wählen ist egal. Allerdings ist mit Python Custom String Formatting eine besonders einfache Lösung möglich.

Aufgabe 3 (1,5 Punkte)

Vervollständigen Sie die Funktion `translate`. Sie hat zwei Parameter `dictionary` (ein Wörterbuch von Typ `dict`) und `word` (das zu übersetzende Wort von Typ `str`). Die Funktion soll die jeweilige Übersetzung zurückgeben. Ist das Wort nicht im Wörterbuch enthalten, soll `None` zurückgegeben werden.

In der Vorlage ist ein kleines Beispiel-Wörterbuch `DE_EN` enthalten, für welches sich folgende Beispiele ergeben:

- `translate(DE_EN, "Ritter")` soll `"Knight"` zurückgeben
- `translate(DE_EN, "Pferd")` soll `None` zurückgeben

Sie können beispielsweise den Operator `in` verwenden, um zu überprüfen, ob ein Key in einem `dict` belegt ist.

Alternativ: Diese Funktion ähnelt sehr stark einer bereits vorhandenen Methode von `dict`. Es lohnt sich, mal einen Blick in die Python Docs zu `dict` zu werfen.

Aufgabe 4 (1,5 Punkte)

Vervollständigen Sie die Funktion `invert_dict`. Sie hat einen Parameter `dictionary` für ein Wörterbuch vom Typ `dict`. Die Schlüssel-Wert-Paare von `dictionary` sollen invertiert und anschließend als neues Objekt vom Typ `dict` zurückgegeben werden.

Ein Aufruf von `invert_dict(DE_EN)` soll also beispielsweise folgenden Rückgabewert besitzen:

```
{"King": "König", "Knight": "Ritter", "Dragon": "Drache", "Aaaarrggh": "Aaaarrggh"}
```

Mit `DE_EN` funktioniert diese Operation ohne Probleme. Im Allgemeinen liefert die Anweisung "Invertieren eines Objekts vom Typ `dict`" jedoch keinen deterministischen Algorithmus - vereinfacht gesprochen hat der Algorithmus kein eindeutiges Ergebnis. Verständnisfrage: Welche notwendige Eigenschaft des Datentyps `dict` ist dafür verantwortlich? Notieren Sie Ihre Antwort per Kommentar in Ihrer Abgabe. Sie können die Lösung im Tutorium diskutieren.

Aufgabe 5 (4 Punkte)

Vervollständigen Sie die Funktion `word_mincer`. Sie hat zwei Parameter `word`, das zu modifizierende Wort vom Typ `str`, und `begin_upper`, der Einstellung von Typ `bool`, ob das Ergebnis mit einem Großbuchstaben beginnen soll. Die Modifikation besteht daraus, dass das Wort am Ende immer abwechselnd Groß- und Kleinbuchstaben enthalten soll. Das Ergebnis soll zurückgegeben werden und vom Typ `str` sein. Der Parameter `begin_upper` soll ein optionaler Parameter sein und einen Standardwert (default value) von `False` haben.

Ein paar Beispiele:

- `word_mincer("König")` soll `"kÖnIg"` zurückgeben
- `word_mincer("KÖNIG")` soll `"kÖnIg"` zurückgeben
- `word_mincer("kÖnIg")` soll `"kÖnIg"` zurückgeben
- `word_mincer("KöNiG")` soll `"kÖnIg"` zurückgeben
- `word_mincer("Kaugummiautomat")` soll `"kAuGuMmIaUtOmAt"` zurückgeben
- `word_mincer("Kaugummiautomat", begin_upper=True)` soll `"KaUgUmMiAuToMaT"` zurückgeben
