IWGS1_WS1920_Zettel5

July 7, 2020

1 Informatische Werkzeuge in den Geistes- und Sozialwissenschaften I

1.1 Hausaufgabe 5 (Funktionen und Zahlensysteme, 100 Punkte)

Erschienen: 16.11.2019
Abgabe bis: 24.11.2019

Bitte laden Sie Ihre Notebooks bis 23:59 Uhr am Abgabetag in Ihrer Übungsgruppe bei StudOn hoch.

Wenn Ihnen einige der hier verwendeten Konzepte unbekannt sind oder Sie nicht wissen, wie Sie fortfahren sollen, können Sie die Vorlesungsunterlagen zu Rate ziehen oder jederzeit Fragen im Forum stellen, im Tutorium nachfragen, oder Ihrem Tutor eine Mail schreiben.

1.1.1 Aufgabe 5.1 (Römische Zahlen, 30 + 20 Punkte)

Wenn wir Werkzeuge der Informatik in Geistes- und Sozialwissenschaftlichen Kontexten verwenden, müssen häufig Daten von einem Format in ein anderes umgeschrieben werden. Dies wird oft notwendig, wenn Daten in einem anderen Kontext neu präsentiert werden sollen.

Aufgabe: Schreiben Sie eine Python-Funktion (ein Python-Programm, machen Sie sich mit dem Unterschied vertraut!) `int2roman`, die eine ganze Zahl (z.B. `2019`) als Parameter nimmt und einen String zurück gibt, der der gleichen Zahl, allerdings in römischen Zahlen entspricht.

Sie können dafür die einfache Darstellung (2019 = `"MMXVIIII"`) benutzen oder darüber hinaus auch die Subtraktionsregel (2019 = `"MMXIX"`) anwenden. Die einfache Darstellung gibt korrekt umgesetzt bereits volle Punkte, die Subtraktionsregel darüber hinaus noch 20 Bonuspunkte.

Hinweis: Beachten Sie, dass wenn Ihr Code über mehrere Zellen verteilt ist (wie hier eventuell mit Ihrer Funktion und den Tests in der übernächsten Zelle), erst alle Zellen (nochmal) ausgeführt werden müssen, bevor (Änderungen an) Zuweisungen und Definitionen effektiv sind.

[1]:
```
# Ihr Code hier! (Doppelklick zum Editieren, Shift + Enter um die Zelle␣
↪auszuführen)
```

[2]:
```
# Ein paar Tests, ob Ihre Funktion das richtige tut.
# Einkommentieren, wenn die Funktion definiert ist.

```

```
# print(int2roman(1500)) # sollte "MD" printen.
# print(int2roman(2019)) # sollte entweder "MMXVIIII" oder "MMXIX" printen.
# print(int2roman(1984)) # sollte entweder "MDCCCCLXXXIIII" oder "MCMLXXXIV"␣
 ↪printen.
# print(int2roman(599))  # sollte entweder "DLXXXXVIIII" oder "DXCIX" printen.
```

1.1.2 Aufgabe 5.2 (User Input Dictionaries, 40 Punkte)

Wir haben in vorherigen Aufgaben bereits diskutiert, dass es wichtig ist, Eingaben von dem Menschen vor dem Bildschirm bekommen zu können. Der nächste Schritt ist, diese Eingaben in sinnige Datenstrukturen zu überführen, die eine weitere Verarbeitung erleichtern. In dieser Aufgabe werden Sie daher ein Programm schreiben, dass ein beliebig großes Dictionary mit Eingaben aufbaut.

Für diese Aufgabe ist die `.update()` Methode (siehe: Dot-Notation) von Dictionaries hilfreich. Wenn Sie `d1.update(d2)` auf einem Dictionary `d1` aufrufen (wobei `d2` ein weiteres Dictionary ist), werden die Schlüssel/Wert-Paare von `d2` zu `d1` hinzugefügt (der Wert von `d1` wird also geändert, `d2` bleibt gleich). In der nächsten Zelle sehen Sie ein erklärendes Beispiel.

[3]:
```
# Beispiele für Dictionaries mit präzisen Wissenschaftlichen Tiernamen
# als Schlüssel und umgangssprachlichen Namen als Werten.

# Wir können Variablen zum Bau des Dictionaries verwenden.
key1 = "Corvus Corax"
val1 = "Kolkrabe"
animal1 = { key1 : val1 }

# Wir können auch alles direkt hinschreiben ("Literal Syntax").
animal2 = { "Bradypus pygmaeus" : "Zwergfaultier" }

# Jetzt führen wir alles zusammen, beginnend mit dem leeren Dictionary.
# Print-Statements machen den Verlauf des Variablenwerts klarer.
allAnimals = {}
print(allAnimals)
allAnimals.update(animal1) # Fügt die Paare aus animal1 hinzu.
print(allAnimals)
allAnimals.update(animal2) # Fügt die Paare aus animal2 hinzu.
print(allAnimals)
```

```
{}
{'Corvus Corax': 'Kolkrabe'}
{'Bradypus pygmaeus': 'Zwergfaultier', 'Corvus Corax': 'Kolkrabe'}
```

Aufgabe: Schreiben Sie ein Python-Programm, dass nach und nach ein Dictionary durch User-Input aufbaut, beginnend mit dem leeren Dictionary. Das Programm soll fragen, ob ein weiteres Schlüssel/Wert-Paar hinzugefügt werden soll. Ist die Antwort `ja`, soll separat sowohl nach einem Schlüssel als auch nach einem Wert gefragt werden. Dieses Paar (beides Strings) soll dann dem aktuellen Dictionary hintzugefügt werden (dafür kann die `.update()`-Methode benutzt werden).

Danach soll erneut nachgefragt werden, ob ein weiteres Paar hinzugefügt werden soll, sodass bei Programmstart nicht feststeht, wie viele Paare das Dictionary am Ende enthält. Wenn die Antwort schließlich irgendetwas außer `ja` ist, soll das aktuelle Dictionary geprintet werden.

[4]:
```
# Ihr Code hier! (Doppelklick zum Editieren, Shift + Enter um die Zelle␣
↪auszuführen)
```

1.1.3 Aufgabe 5.3 (Verständnisaufgabe 4, 30 Punkte)

Abermals tritt Ihre gute Freundin, Beatrice Beispiel, mit einem Problem in Python an Sie heran. Beatrice hat eine Beispieldatenbank mit vier Dictionaries mit Daten zu berühmten "Emmas" (siehe nächste Zelle). Sie hat außerdem ein kleines Programm geschrieben, dass alle Schlüssel/Wert-Paare aller Dictionaries mit etwas erklärendem Text printen soll. Das funktioniert allerdings aktuell nur für die erste Emma und hört danach dann mittendrin auf. Beatrice hat zu lange auf ihren eigenen Code gestarrt um den Fehler doch noch finden zu können und fragt deshalb Sie um Hilfe.

Aufgabe: Machen Sie sich mit Beatrices Daten und Programm vertraut. Finden Sie den Programmierfehler den Beatrice begangen hat. Beschreiben Sie Ihre Erkenntnisse in der letzten Zelle des Dokuments. Wie hätte dieses Problem verhindert werden können? Korrigieren Sie Beatrices Code, sodass er funktioniert wie geplant.

Hinweis: Wir möchten Sie dazu ermuntern, den Code und seine Funktionsweise näher kennen lernen indem Sie damit "spielen". Was passiert, wenn einzelne Variablen andere Werte annehmen? Würde eine andere Herangehensweise ebenfalls funktionieren? Ist die Reihenfolge der Befehle wichtig? Falls Sie feststellen, dass Sie den Code "kaputt gespielt" haben, können Sie jederzeit eine neue Version der Aufgabe runterladen.

[5]:
```
# Ein paar Dictionaries mit Daten von berühmten Emmas!
watson = { "Name" : "Emma Watson",
           "Geboren" : "15. April 1990",
           "Rolle" : "Hermione Granger"}

willard = { "Name" : "Emma Willard",
            "Geboren" : "23. Februar 1787",
            "Gestorben" : "15. April 1870",
            "Gründerin von" : "Troy Female Seminary"}

bonino = { "Name" : "Emma Bonino",
           "Geboren" : "09. März 1948",
           "Partei" : "Radicali Italiani"}

goldman = { "Name" : "Emma Goldman",
            "Geboren" : "27. Juni 1869",
            "Gestorben" : "14. Mai 1940",
            "Zitat" : "If I can't dance, it's not my revolution!"}

# Liste von dictionaries.
allEmmas = [watson, willard, bonino, goldman]
```

[6]:
```
i = 0
# Wir laufen über alle Emmas in der Liste allEmmas von oben.
while i < len(allEmmas):

    # Printet Linie zur Übersicht:
    aktuelleEmma = allEmmas[i]
    print("---------")

    # Das ist eine Methode, an eine Liste aller Schlüssel eines Dictionaries zu␣
 ↪kommen.
    keys = list(aktuelleEmma.keys())

    # Ich glaube sowas ähnliches ging auch mit einer for-Schleife
    # aber so hab ich es schonmal geschrieben.
    while i < len(keys):
        # Relevante Werte ermitteln und abspeichern.
        key = keys[i]
        val = aktuelleEmma[key]

        # Daten printen.
        print("Der Schlüssel", keys[i], "hat den Wert", val)

        # Laufvariable hochsetzen nicht vergessen.
        i = i + 1
```

```
---------
Der Schlüssel Rolle hat den Wert Hermione Granger
Der Schlüssel Geboren hat den Wert 15. April 1990
Der Schlüssel Name hat den Wert Emma Watson
---------
Der Schlüssel Name hat den Wert Emma Goldman
```

Ihre Beschreibung und Erläuterungen hier! (Doppelklick zum Editieren, Shift + Enter um die Zelle auszuführen)