04b_Miniuebungen_Loesungen

0.1 Mini-Aufgaben zur Überprüfung des Verständnis: sets

Schreiben Sie ein Programm, das mit Hilfe von Mengen überprüft, ob in einem gegebenen String (der Einfachheit halber bestehend aus Kleinbuchstaben) alle 26 Buchstaben des Alphabets ohne Umlaute vorkommen. Überlegen Sie sich dazu, dass dies bereits möglich ist nur durch die Verwendung der beiden Befehle add() und len(). Erklären Sie, warum mit diesem Programm keinerlei Informationen gewinnbar sind, wie oft welcher Buchstabe vorkommt.

[2]:
```
demo_string = "ldksfldsafnsdfsakdjgkfsajlfdlsakhfkbcxkgdskjdfslkdhflskd"
#demo_string = "abcdefghijklmnopqrstuvwxyz"

letters = set()

# strings sind iterierbar
for c in demo_string:
    letters.add(c)

# Ergebnis
if len(letters)==26:
    print("Alle Kleinbuchstaben kommen vor.")
else:
    print("Nur die folgenden Kleinbuchstaben kommen vor")
    print(letters)
```

```
Alle Kleinbuchstaben kommen vor.
```

Programmieren Sie das Beispiel noch einmal, mit Hilfe eines Dictionaries.

[10]:
```
demo_string = "ldksfldsafnsdfsakdjgkfsajlfdlsakhfkbcxkgdskjdfslkdhflskd"
#demo_string = "abcdefghijklmnopqrstuvwxyz"

# Die schlaue Idee ist hier, die Eindeutigkeit von keys zu nutzen,
# und auf die Werte zu pfeiffen
letters = dict()
for c in demo_string:
    letters[c] = "irgendwas"

if len(letters)==26:
    print("Alle Kleinbuchstaben kommen vor.")
```

```
else:
    print("Nur die folgenden Kleinbuchstaben kommen vor")
    print(letters.keys())

# Alternativ können wir natürlich auch das Histogramm-Beispiel klonen,
# wir erhalten die Zusatzinformation, wie oft welcher Buchstabe vorkommt

# Gerade diese Zusatzinformation ist nicht gefordert!
```

```
Nur die folgenden Kleinbuchstaben kommen vor
dict_keys(['l', 'd', 'k', 's', 'f', 'a', 'n', 'j', 'g', 'h', 'b', 'c', 'x'])
```

Programmieren Sie das Beispiel noch einmal, mit Hilfe einer Liste.

[8]:
```
demo_string = "ldksfldsafnsdfsakdjgkfsajlfdlsakhfkbcxkgdskjdfslkdhflskd"
#demo_string = "abcdefghijklmnopqrstuvwxyz"

letters = list()

for c in demo_string:
    if c not in letters:   # Vorschau auf Effizienzdiskussion: das erfordert␣
 ↪einen Durchlauf durch die gesamte Liste, jedesmal!
        letters.append(c)

if len(letters)==26:
    print("Alle Kleinbuchstaben kommen vor.")
else:
    print("Nur die folgenden Kleinbuchstaben kommen vor")
    print(letters)
```

```
Nur die folgenden Kleinbuchstaben kommen vor
['l', 'd', 'k', 's', 'f', 'a', 'n', 'j', 'g', 'h', 'b', 'c', 'x']
```

Der erhoffte Erkenntnisgewinn ist, dass die Analyse einer Aufgabenstellung oft Hinweise liefert, welche Datenstruktur am besten geeignet ist für eine Aufgabe. Diese Beispiele zeigen, dass die Unterscheidung zwischen list, dict und set, d.h. die (Un-) Eindeutigkeit von Teilen der Datenstruktur, uns oft erheblich Arbeit ersparen kann, weil uns die Definition und damit die Garantie, die uns eine Datenstruktur gibt, bereits erheblich hilft. Und das wirklich Schöne ist, dass wir nicht wissen müssen, wie die Datenstruktur unter der Haube diese Garantie realisiert!

0.2 Impressum

0.2.1 Programmierkurs Python, Dominik Göddeke https://www.ians.uni-stuttgart.de, Universität Stuttgart

Version vom April 2023

Gefördert durch die Stiftung Innovation in der Hochschullehre. (alle Rechte am Logo vorbehalten)

Gefördert mit Mitteln der Deutschen Forschungsgemeinschaft (EXC 2075 - 390740016) im Rahmen der Exzellenzstrategie.

[ ]: