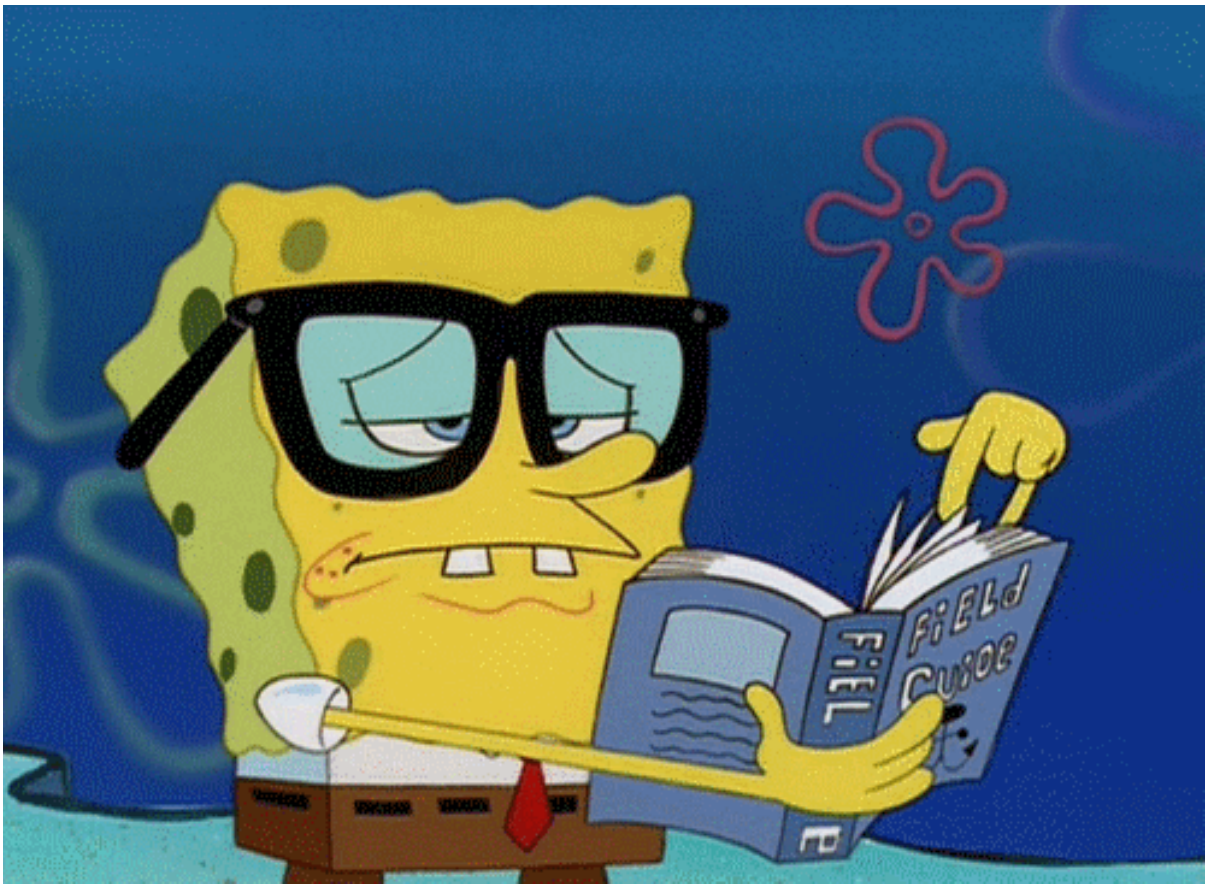
Editor-Trick des Tages: Keyboard Shortcut Reference

Wir haben in den letzten Wochen eine Menge Editor-Tricks besprochen, aber in der heutigen Sitzung geht es vor allem darum, dass Sie selbständig neue Informationen und Shortcuts finden. Deshalb besteht der heutige Editor-Trick darin, dass wir uns alle Shortcuts von VSCode ansehen.

Um die Übersicht über alle Shortcuts zu öffnen, können Sie in der Command Palette `>Keyboard Shortcuts Reference` eingeben. Dadurch wird der Browser geöffnet und Sie sehen ein PDF-Dokument (https://code.visualstudio.com/shortcuts/keyboard-shortcuts-windows.pdf), in dem verschiedene Arten von Shortcuts aufgelistet sind. Finden Sie einen Shortcut, den Sie gebrauchen könnten? Probieren Sie ab jetzt regelmäßig neue Shortcuts aus! So können Sie mit der Zeit immer professioneller mit unserem Editor umgehen.

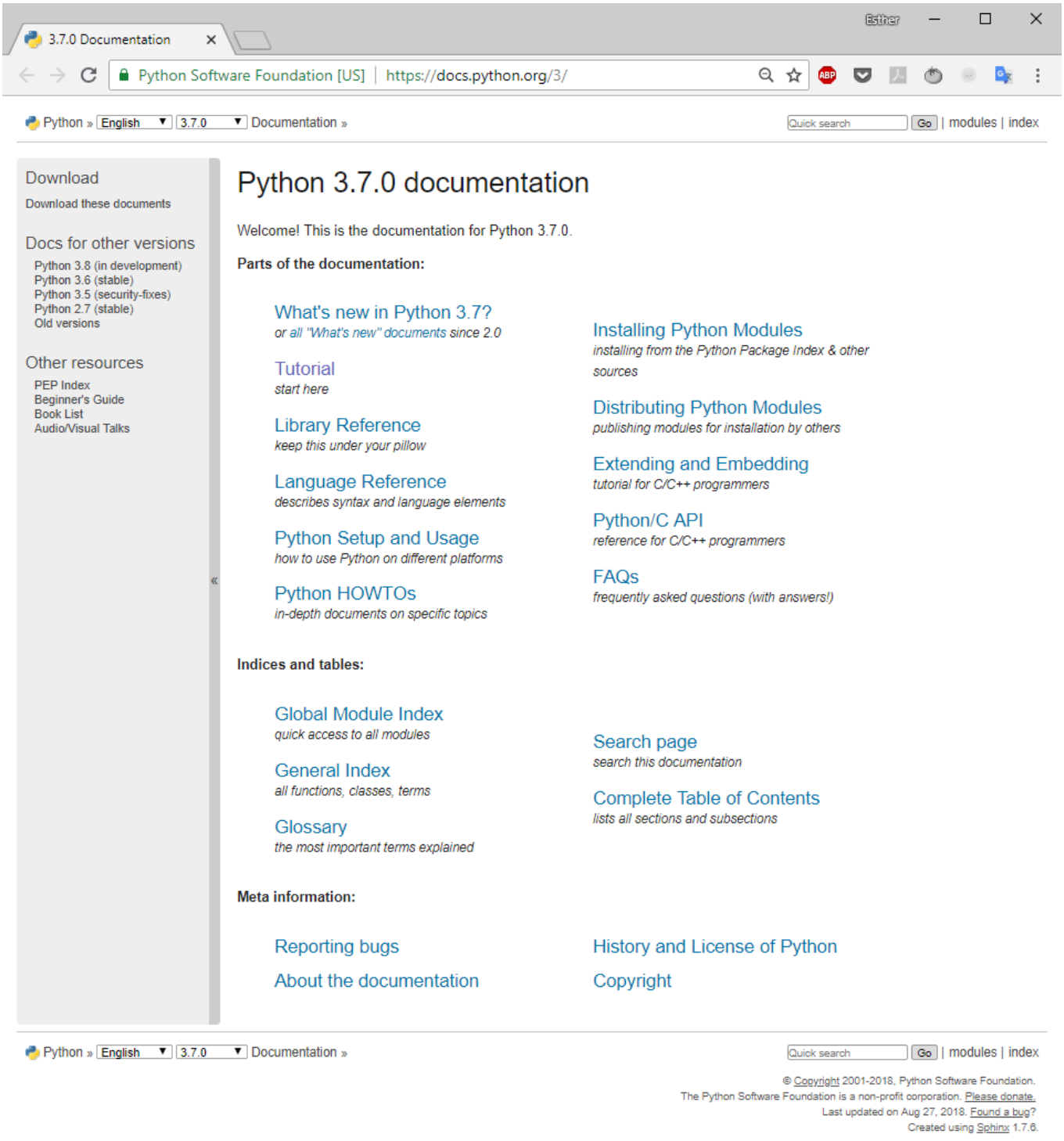
Recherchieren mit der Python-Dokumentation und StackOverflow



Inzwischen haben wir eine Reihe Funktionalitäten von Python besprochen, die Sie jederzeit in den Skripten der letzten Vorlesungen nachlesen können. Außerhalb dieses Kurses wird es Ihnen aber gelegentlich passieren, dass Sie etwas programmieren wollen, was im Kurs nicht behandelt wurde.

Die Python-Dokumentation beschreibt die Datentypen und dazugehörigen Methoden, die wir verwenden können, und erklärt, wie jeder Bestandteil der Sprache eingesetzt wird. Sie ist daher eine unverzichtbare Ressource für Ihre weitere Programmierkarriere (sofern Sie bei Python bleiben).

Wenn Sie im Browser die Seite https://docs.python.org (https://docs.python.org) öffnen, sehen Sie Folgendes:

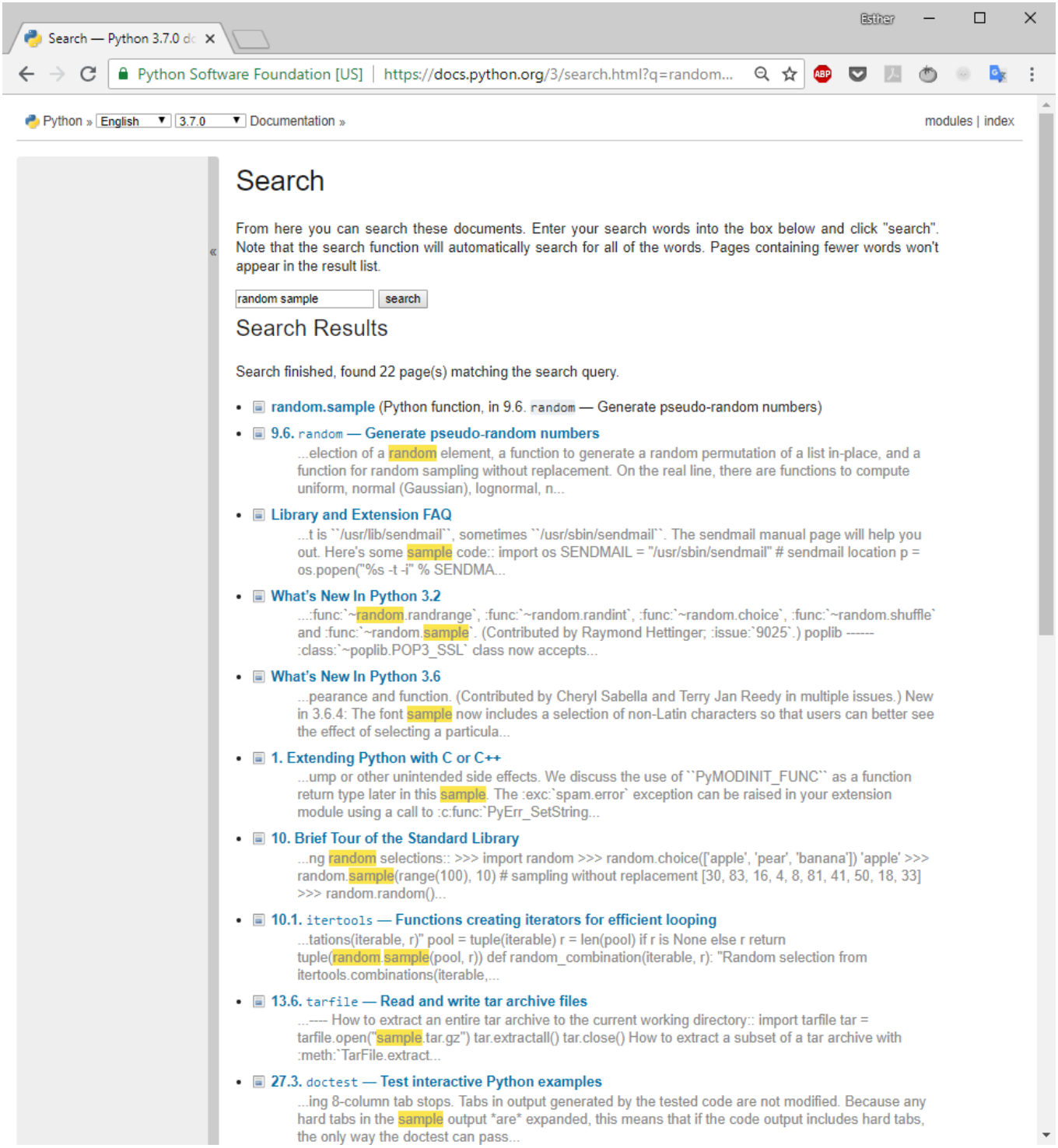


Da Pythonversionen sich teilweise stark unterscheiden, ist es wichtig, dass Sie auf die Doku für Ihre Version zugreifen. Dazu können Sie oben links die Version auswählen, in der Sie programmieren.

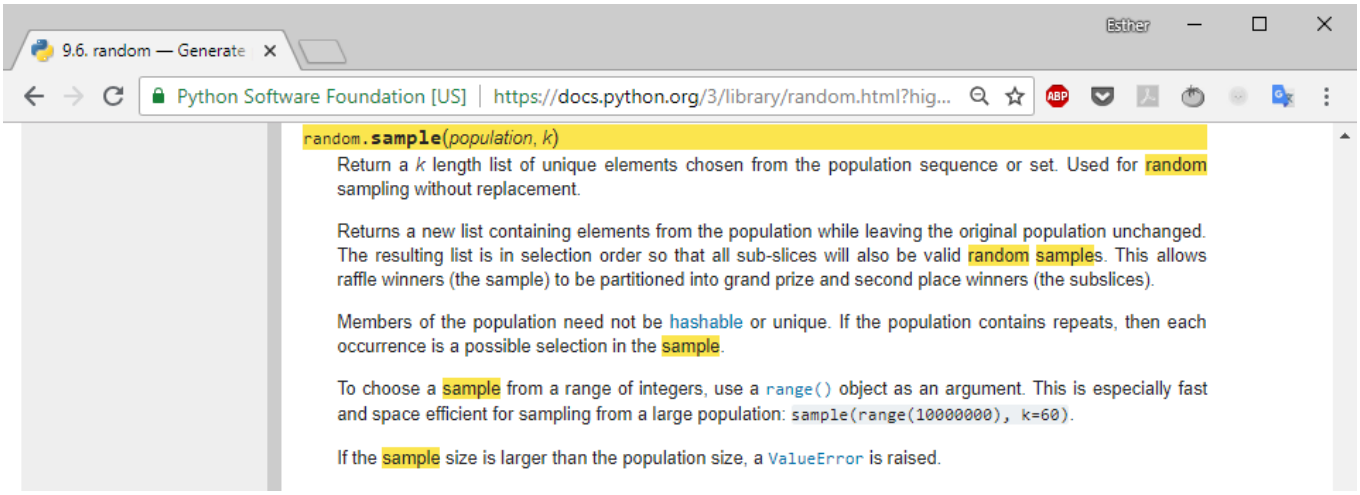
Oben rechts können Sie Stichworte eingeben, über die Sie etwas erfahren möchten. Nehmen wir beispielsweise an, dass wir aus einer Liste mit 10 Elementen genau 2 zufällige Elemente auswählen möchten. Wir kennen schon `random.randint()` für Zufallszahlen und `random.choice()` für die Auswahl genau eines zufälligen Elements aus einer Sequenz, aber können wir die Auswahl von 2 Elementen vielleicht auch mit einem einzigen Befehl umsetzen?

Zufällige Auswahl von n Elementen aus einer Liste

Da Methoden und andere Signalwörter in Python alle englischsprachig sind, müssen wir unser Ziel zunächst ins Englische übersetzen. Aus "zufällige Auswahl" wird "random sample".



Schon am Titel des ersten Treffers sehen wir, dass das Modul `random` eine Methode zur Verfügung stellt, die `sample` heißt. (Direktlink (https://docs.python.org/3/library/random.html?highlight=random%20sample#random.sample))



Wir sehen also, dass wir mit dem Befehl `random.sample(our_list, 2)` die Zufallsauswahl von 2 Elementen aus unserer Liste erreichen können. Im Text unterhalb des Befehls folgen Erklärungen, welche Typen die `population` haben kann und wie `random.sample()` sich verhält. Wir können ausprobieren, ob alles wie erwartet funktioniert:

In [ ]:

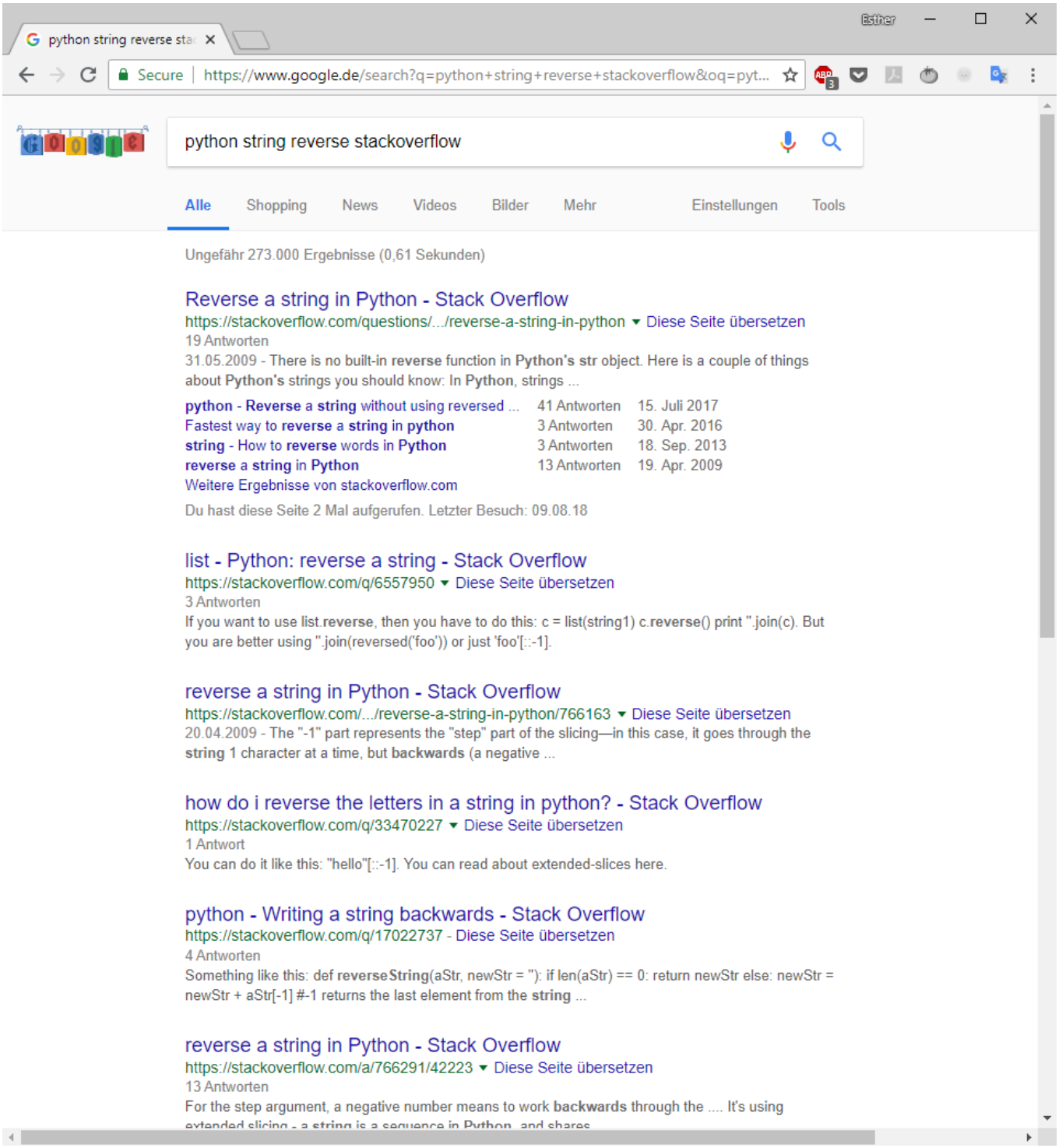
```
import random

our_list = [1,2,3,4,5,6,7,8,9,10]
for i in range(5):
    print(random.sample(our_list, 2))
```

StackOverflow als Ergänzung zur Python-Doku

Nicht immer ist es so leicht, in der Dokumentation nachzuschlagen, welche Befehle man benutzen muss, um bestimmte Dinge zu erreichen. Hier ein Beispiel:

Wir wollen einen String umdrehen. Wir verwenden die Doku, um herauszufinden, ob bzw. wie das geht. Egal, welche Suchbegriffe wir eingeben (reverse, string reverse, reverse string, revert string, ...), bekommen wir keinen offensichtlich richtigen Treffer.

In so einem Fall können wir Google bemühen. Idealerweise googeln Sie solche Fragen in etwa in der folgenden Form:



Wie Sie sehen, beginnt die Suchanfrage mit dem Stichwort `python`. Nichts ist frustrierender, als Zeit mit einer Erklärung zu verschwenden und erst nach einigen Minuten zu merken, dass die Erklärung sich auf die falsche Programmiersprache bezieht...

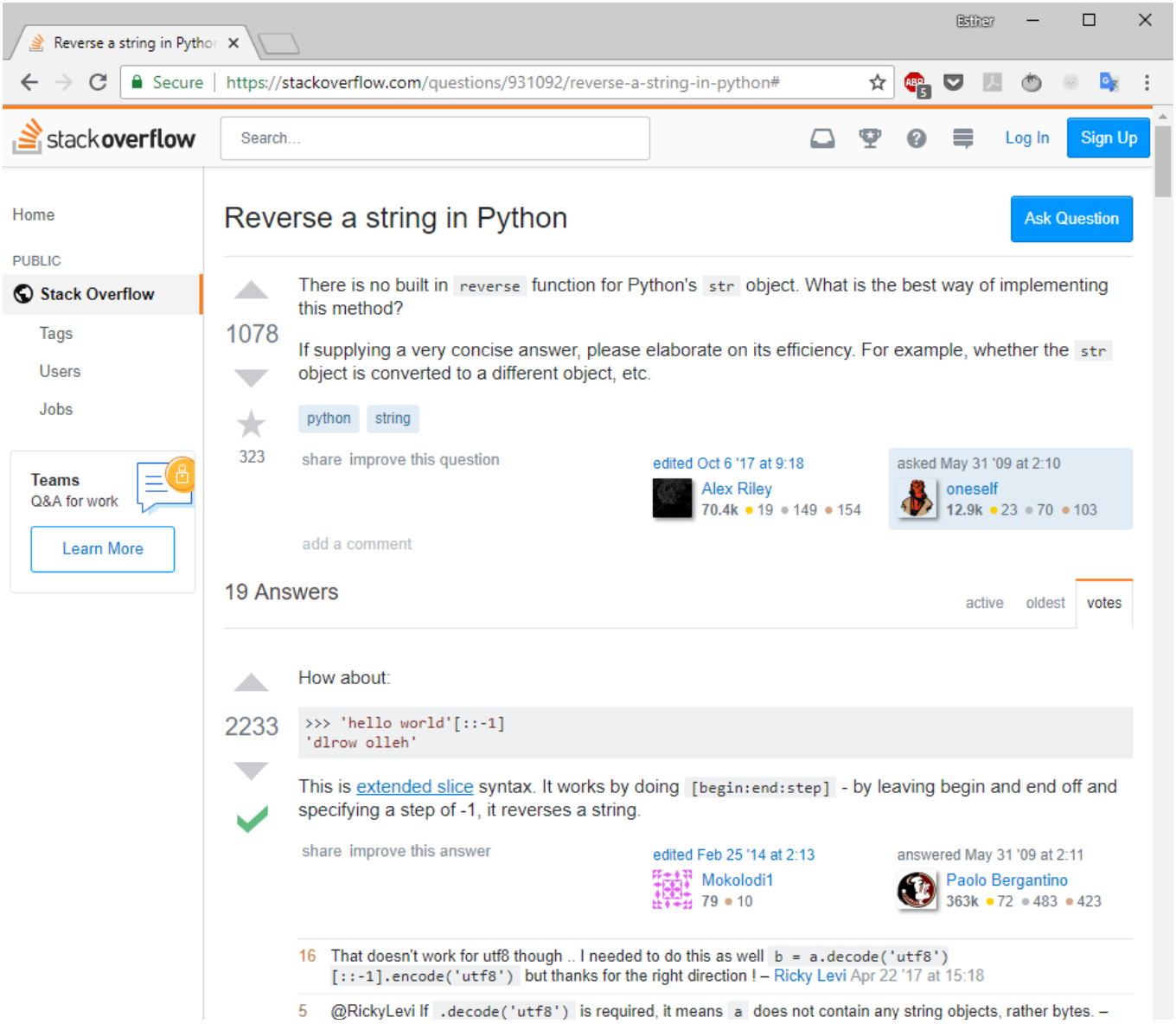
Danach folgt das Thema, das uns interessiert. Wir haben das Problem auf Englisch mit `string reverse` beschrieben.

Zuletzt folgt die Angabe des Stichworts `stackoverflow`.

StackOverflow ist eine Q&A-Seite für Programmierthemen, die seit 2008 existiert. Meist sind die Fragen sehr spezifisch und klar formuliert, oft enthalten sie kleine Codebeispiele, und dank dem Bewertungssystem für die Antworten sind richtige/empfehlenswerte Antworten leicht zu finden.

Wenn Sie die Antwort auf Ihre Frage auf anderen Seiten als SO finden, besteht immer die Gefahr, dass die Informationen veraltet sind oder sich auf eine andere Pythonversion beziehen! Auf SO wird die Pythonversion entweder direkt angegeben, oder in den Antworten wird darauf hingewiesen, falls es Unterschiede zwischen den Versionen gibt.

Schauen wir uns mal den ersten Treffer (https://stackoverflow.com/questions/931092/reverse-a-string-in-python) auf StackOverflow an:


Reverse a string in Pytho
Esther
Secure | https://stackoverflow.com/questions/931092/reverse-a-string-in-python#
stackoverflow
Search...
Log In
Sign Up
Home
PUBLIC
Stack Overflow
Tags
Users
Jobs
Teams
Q&A for work
Learn More
Reverse a string in Python
Ask Question
There is no built in reverse function for Python's str object. What is the best way of implementing this method?
If supplying a very concise answer, please elaborate on its efficiency. For example, whether the str object is converted to a different object, etc.
1078
323
python
string
share improve this question
edited Oct 6 '17 at 9:18
Alex Riley
70.4k • 19 • 149 • 154
asked May 31 '09 at 2:10
oneself
12.9k • 23 • 70 • 103
add a comment
19 Answers
active
oldest
votes
How about:
2233
>>> 'hello world'[::-1]
'dlrow olleh'
This is extended slice syntax. It works by doing [begin:end:step] - by leaving begin and end off and specifying a step of -1, it reverses a string.
share improve this answer
edited Feb 25 '14 at 2:13
Mokolodi1
79 • 10
answered May 31 '09 at 2:11
Paolo Bergantino
363k • 72 • 483 • 423
16 That doesn't work for utf8 though .. I needed to do this as well b = a.decode('utf8') [::-1].encode('utf8') but thanks for the right direction ! – Ricky Levi Apr 22 '17 at 15:18
5 @RickyLevi If .decode('utf8') is required, it means a does not contain any string objects, rather bytes. –

Die bestbewertete Antwort steht als erstes unter der Frage. Der Beispielcode und die dazugehörige Ausgabe zeigen uns, dass die Lösung tatsächlich das macht, was wir erwarten. In der Erklärung ist eine **Seite aus der Python-Dokumentation** verlinkt, auf der wir mehr erfahren können.

Einige Kommentare der besten Antwort erwähnen, dass es einen Unterschied macht, ob man **Python 2 oder Python 3** benutzt. Weiter unten folgen weitere ausführliche Antworten, in denen verschiedene Lösungen verglichen werden und wir lernen können, dass Slicing die beste Lösung für unseren Anwendungsfall ist.

Beachten Sie immer das **Datum** der Fragen und Antworten, die Sie bei der Recherche finden! Bei grundlegenden Fragen (im Erstsemesterkurs wahrscheinlich der häufigere Fall) ist es zwar zweitrangig, ob eine Antwort dieses Jahr oder vor drei Jahren gepostet wurde. Aber wenn Sie z.B. Probleme mit Ihrer Installation haben und eine Antwort auf SO finden, die mehrere Jahre alt ist, ist es sehr wahrscheinlich, dass die angegebene Lösung Ihnen heute nicht weiterhilft.

Merkmale einer guten StackOverflow-Antwort

- Gute Bewertung (die Zahl links vom Text der Antwort)
- Je nach Thema: Einigermaßen aktuelles Erstellungsdatum
- Mindestens ein kurzes, verständliches Codebeispiel mit Angabe des erwarteten Outputs
- Erklärung, warum der angegebene Code eine gute Wahl ist oder welche Alternativen es gibt
- Information dazu, ob die Lösung in Python 2 oder Python 3 oder in allen Pythonversionen funktioniert
- Idealerweise: Link auf offizielle Python-Dokumente, die weitere Infos zum Thema enthalten
- Bei komplexeren Aufgaben: Einschätzung oder Messung des Rechenaufwands der angegebenen Lösung
- Evtl. Vergleich verschiedener Möglichkeiten, das Problem zu lösen

Beim Recherchieren finden wir häufig Antworten, die wir gar nicht verstehen (weil wir erst seit wenigen Monaten programmieren). Das ist aber nicht schlimm! Wenn Sie weitersuchen oder im Thread etwas weiter nach unten scrollen, werden Sie verständlichere Posts zum Thema finden. So wird Ihr Verständnis der Programmiersprache nach und nach geschult.

Die Erfahrung zeigt, dass die Fragen-und-Antworten-Sammlung von StackOverflow für **fast alle unsere Programmierprobleme** ausreichend ist. Python ist so verbreitet, dass die allermeisten Probleme, die uns begegnen, auch schon jemand anderem als uns passiert sind. Das bedeutet, dass man viele Jahre lang darauf verzichten kann, selbst eine Frage auf SO zu posten - man muss ja nur die Frage finden, die die letzte Person gestellt hat, die das gleiche Problem hatte.

Sonderfälle treten dann auf, wenn Sie z.B. mit besonderen Modulen arbeiten, die nicht so verbreitet sind. Dann kann es Sinn machen, eine eigene Frage zu stellen. StackOverflow hat eine Anleitung zum Stellen von Fragen (https://stackoverflow.com/help/how-to-ask), die sehr viel Wert darauf legt, dass keine Duplikate erzeugt werden.

Python-Doku vs. StackOverflow

Während die Python-Dokumentation schnell weiterhilft, wenn es darum geht, wie bestimmte Methoden verwendet werden, ist StackOverflow pragmatischer orientiert und hilft uns vor allem in Fällen, wo wir nicht wissen, welche Methode wir anwenden sollen.

Die **Pythondokumentation** sollte Ihre erste Anlaufstelle sein, wenn Sie spezielle Fragen zu einzelnen Datentypen, Methoden oder Modulen haben. Vor allem wenn Sie einem neuen Datentyp oder einem neuen Modul begegnen, lohnt es sich, in der Dokumentation nachzuschauen, was man mit diesem Datentyp/diesem Modul alles machen kann.

Wenn Sie Hilfe für einen **konkreten, isolierbaren Anwendungsfall** brauchen, ist manchmal StackOverflow die bessere Wahl. Die Python-Doku enthält zwar ausführliche Informationen zu einzelnen Themen, aber hilft nicht dabei, unterschiedliche Teile von Python miteinander zu kombinieren, um Aufgaben zu lösen.

Achten Sie darauf, dass Sie **alles verstehen**, was Sie in einer StackOverflow-Antwort finden. Falls Sie es nicht verstehen und trotzdem die vorgeschlagene Lösung in Ihr eigenes Programm einbauen, ist es sehr sehr sehr wahrscheinlich, dass es später Probleme gibt, weil die verwendete Lösung nicht hundertprozentig zum Kontext passt, in dem Sie sie verwenden. Suchen Sie so lange nach einer Antwort, bis Sie eine finden, mit der Sie wirklich arbeiten können!

Aufgabe

In HTML gibt es einige Zeichen, die eine besondere Funktion haben, zum Beispiel die spitzen Klammern: `<>` . Um auf einer Webseite spitze Klammern als Symbole anzuzeigen, muss man sie deshalb anders schreiben, nämlich als `<>` . Genau wie der Backslash, den wir bereits kennen, ist das eine Methode, um bestimmte Zeichen zu *escapen*. Finden Sie mithilfe der Python-Dokumentation heraus, mit welcher Methode man diese Zeichen in Strings escapet. Suchen Sie auch eine StackOverflow-Antwort zu dem Thema. Welche der beiden Ressourcen finden Sie verständlicher? **Tipp**: Sie können die Google-Suchanfragen `python docs html escape` und `python html escape stackoverflow` verwenden.

Datentyp des Tages: Sets (Mengen)

Mengen kennen Sie schon aus der Vorlesung "Mathematische Grundlagen der Computerlinguistik". Mengen (englisch: *Sets*) ähneln in Python Listen, haben aber einige Eigenschaften, die sie von Listen unterscheiden:

- Sets sind unsortiert.
- Sets enthalten jedes Item nur einmal.

Genau wie Listen sind Sets ebenfalls **mutable**, das bedeutet, sie können "in place" verändert werden.

Sets können auf mehrere Arten erzeugt werden:

In [ ]:

```
set1 = set()                # leere Menge erstellen...

for i in [1,2,3,3,4,4,5]:
    print(i)
    set1.add(i)             # ... und nach und nach füllen

print("Set 1: " + str(set1))
```

In [ ]:

```
set2 = set([1,2,3,3,4,4,5]) # Menge direkt beim Erstellen mit Elementen füllen

print("Set 2: " + str(set2))
```

In [ ]:

```
set3 = {1,2,3,3,4,4,5}      # Keine Verwechslungsgefahr mit Dictionarys:
                            # Sets haben keine Values, nur Keys

print("Set 3: " + str(set3))
```

Der Unterschied zwischen `set2` und `set3` ist, dass `set2` mithilfe eines Funktionsaufrufs erzeugt wurde. Wir erkennen Funktionen inzwischen an den runden Klammern: `set()`, analog zu den schon bekannten Funktionen `str()` oder `bool()`.

`set3` dagegen wurde direkt mithilfe einer bestimmten Art von Klammern als Menge erstellt, analog zu den Datentypen, die wir schon länger kennen, z.B. Listen (dort verwenden wir eckige Klammern, hier sind es geschweifte Klammern).

Aufgabe

1. Führen Sie die folgenden Codebeispiele aus. Entspricht die Ausgabe Ihren Erwartungen? Haben Sie eine Erklärung dafür, warum das passiert, was passiert?

In [ ]:

```
set4 = set("This is the boss, and I'm sick of waiting")

print(set4)
```

In [ ]:

```
set5 = {}

print(type(set5))
```

In [ ]:

```
set6 = set("This is the boss, and I'm sick of waiting".split())

print(set6)
```

Durch die Art, wie Sets im Arbeitsspeicher angelegt werden, kann mit ihnen effizienter gearbeitet werden als mit Listen. Wir verwenden also immer dann Sets, wenn die **Sortierung und die Anzahl der Vorkommen einzelner Elemente** keine Rolle spielen.

Operationen auf Sets

Beachten Sie, dass Sets mutable sind. Änderungen müssen also nicht explizit dem Variablennamen zugewiesen werden, sondern verändern automatisch sofort den Inhalt des Sets. Beispielsweise können wir mit `set1.add("hallo")` das Wort `"hallo"` in die Menge `set1` einfügen.

Operation	Bedeutung
`len(s)`	Anzahl der Elemente in `s`
`s.add(e)`	Füge dem Set `s` das Element `e` hinzu
`s1.update(s2)`	Ergänze `s1` um alle Elemente aus `s2`
`s1.intersection(s2)` **oder** `s1 & s2`	Erzeuge die Schnittmenge von `s1` und `s2`
`s1.union(s2)` **oder** `s1 | s2`	Erzeuge die Vereinigungsmenge von `s1` und `s2`
`s1.difference(s2)` **oder** `s1 - s2`	Erzeuge eine Menge, die alle Elemente enthält, die in `s1` , aber nicht in `s2` enthalten sind
`s1.issubset(s2)`	`True` , wenn `s1` eine Teilmenge von `s2` ist
`e in s`	`True` , wenn das Element `e` in `s` enthalten ist; sonst `False`
`s.discard(e)`	Entferne das Element `e` aus `s` , falls es enthalten ist

Aufgabe

1. Auf StackOverflow finden Sie einen Thread (https://stackoverflow.com/questions/17373161/using-curly-braces-to-initialize-set) über die Frage, wie man am besten Mengen initialisiert. Passt der Inhalt der Antworten zu Ihren Überlegungen vorhin? Wie wollen Sie in Zukunft Mengen in Ihrem Code anlegen - so wie `set1`, `set2` oder `set3`?

Zusammenfassung

Sie haben heute gelernt,

- wie Sie die Python-Dokumentation und StackOverflow nutzen können, um mehr über Python zu lernen und um Probleme zu lösen, bei denen Sie alleine nicht weiterkommen
- wie Sie mit `random.sample()` eine zufällige Auswahl einer bestimmten Anzahl von Elementen aus einer Kollektion extrahieren können
- wie Sie in Python einen String umdrehen können
- wie Sie in Python einen String *html-escapen* können
- wie Mengen in Python verwendet werden und worin sie sich von Listen unterscheiden

Und falls noch Zeit ist...

... können wir uns weiter mit dem Cheat Sheet beschäftigen! Es steht noch die Wiederholung der folgenden Themen aus:

- if/elif/else (Bedingungen)
- for-Schleifen
- Kombination von Bedingungen und Schleifen
- (Funktionen)
- Module
- Dateien lesen
- Dateien schreiben
- User-Input mit input()