Python cheat sheet 1

1 Drucken

Die Tradition beim Erlernen einer neuen Programmiersprache ist, als allererste die Welt zu begrüßen. In Python:

```
print("Hallo Welt!")
```

Die Zeichen `""` bzw. `''` bedeuten, dass das Geschriebene ein `string`, also eine Zeichenfolge ist. Mehr dazu später. Das Befehl `print()` bedeutet, dass das Argument, dass im Klammern steht, gedruckt (im Terminal geschrieben) sein soll. Oft ist es nützlich, nicht nur ein Stück Text zu drucken, sondern dabei auch verschiedene Variablen zu benutzen (mehr dazu kommt auch bald). Das kann man in verschiedenen äquivalenten Formen tun. Die letzte ist meine Lieblingsform, aber jeder hat das Recht, sein eigenes (schlechtes) Geschmack zu haben.

```
name = "Hana"
alter = 23
print("Mein Name ist " + name + " und ich bin " + str(alter) + " Jahre alt.")
print("Mein Name ist", name, "und ich bin", alter, "Jahre alt.")
print(f"Mein Name ist {name} und ich bin {alter} Jahre alt.")
```

2 Rechnen mit python

Python ist eine sehr kraftvolle Sprache, in der man Webseiten, Computerspiele und wissenschaftliche Analysen entwickeln kann - es kann aber genauso gut als ein einfacher Taschenrechner funktionieren. Willst du wissen, wie viel 197123547 mal 5769103 ist? Einfach in Python eintippen. Hier sind die wichtigsten mathematischen Operationen.

`+`	*plus, z.B 13+5=18*
`-`	*minus, z.B. 70.5-10=60.5*
`*`	*mal, z.B. 3*5=15*
`**`	*potenziert, z.B. 2**4=16*
`/`	*geteilt durch, z.B. 13/8=1.625*
`//`	*geteilt bis auf ganze Zahlen, z.B. 13/8=1 weil 13 = 1*8 + 5*
`%`	*Rest der Division, z.B. 13%8=5 weil 13 = 1*8 + 5*
`round()`	*gerundet auf die nächste ganze Zahl, z.B. round(1.625)=2*

3 Variablen

Oft müssen wir ein Stück Information im Gedächtnis behalten - und uns außerdem erinnern, was für Bedeutung diese Information hat. Falls ich mich daran erinnern muss, dass ein Freund von mir, Demian, 20 Jahre alt ist, speichere ich die Zahl 20 in ein (imaginäres) Schachtel mit der Etikette Demian's Alter. Das Inhalt des Schachtels wird sich immer wieder ändern (er wird ja älter), aber die Etikette bleibt gleich. Im Programmiersprache heißt die Etikette Variable und die zugeordnete Zahl (oder anderes Stück Information) Wert der Variable. Die Variablennamen sollten möglichst informativ sein. Falls mehrere Wörter dafür notwendig sind, benutzen wir statt Leerzeichen Unterstreichungen, z.B. `alter_meines_freundes_demians = 20`. Wir vermeiden besondere Zeichen, z.B. ß. Die Variablen dürfen auch nicht mit einer Zahl starten (z.B. `20_geburtstag_geschenk` nicht erlaubt, `geschenk_20` schon).

Variablen können von verschiedenen Typen sein. Text wird als eine Zeichenfolge `string` (kurz `str`) gespeichert, ganze Zahlen als `integer` (kurz `int`), reelle Zahlen als `float` und Listen als `list`. (Ein Geheimnis: es gibt noch mehr.) Der Typ der Variable kann man mit dem Befehl `type()` nachprüfen. Manchmal ist es auch möglich, ein Typ ins andere umzuwandeln mit Befehlen wie `str()`, `int()`, `float()` oder `list()` (z.B. `int("13")` -> 13, `int("abc")` wäre natürlich nicht sinnvoll).

4 for-Schleifen

Wenn Computer ein Teil des Codes mehrmals wiederholen sollte, gekennzeichnet man das mit einer Schleife. Die Zahl in Klammern von `range()` sagt uns dabei, wie viel mal das Befehl wiederholt werden soll. Anders als wir Menschen beginnen die meisten Programmiersprachen das Zählen mit 0 und enden beim eins weniger als die angegebene Zahl, also `range(15)` -> 0, 1, 2, ... 12, 13, 14. Wichtig: nicht das Doppelpunkt am Ende der Zeile vergessen und alle Befehle, die wiederholt sein sollten, 4 Leerzeichen nach recht verschieben. Die Variable, die hier `natuerliche_zahl` heißt, ist beliebig wählbar.

```
for natuerliche_zahl in range(15):
    print("Hallo Welt!")
    print(f"Ich drucke jetzt die Zahl {natuerliche_zahl}!")
```

5 Listen (`list`)

Eine Liste ist ganz einfach eine Sammlung mehreren Elementen in Ordnung. Die eckige Klammern sind ein Hinweis, dass es um eine Liste geht. Mit verschiedenen Befehlen kann man der Liste Elemente hinzufügen oder entfernen, sortieren usw. Die Elemente sind auch nummeriert, wir sagen, dass jeder sein `index` hat. Wenn ich z.B. eine Liste `zufaellige_sachen = [15, 'foo', 33.3, "Heute wird sonnig."]` habe, kann ich das erste Element (15) mit dem Befehl `zufaellige_sachen[0]` erhalten und das vierte Element ("Heute wird sonnig.") mit `zufaellige_sachen[3]`. Vorsicht! Dieses Nummerieren startet wieder mit 0, nicht mit 1.

Ein paar Befehle, die auf Listen funktionieren, z. B. auf `beste_liste = [3, 2.5, "Hallo"]`:

`.append()`	*ein Element am Ende hinzufügen, z.B.* `beste_liste.append(113)` *->* `[3, 2.5, "Hallo", 113]`
`.remove()`	*ein spezifisches Element entfernen, z.B.* `beste_liste.remove(2.5)` *->* `[3, "Hallo", 113]`
`.reverse()`	*die Ordnung der Elementen umkehren, z.B.* `beste_liste.reverse()` *->* `[113, "Hallo", 3]`
`.sort()`	*anordnen - macht nur Sinn, wenn alle Elemente entweder Zahlenfolgen oder Zahlen sind*
`sum()`	*berechnet die Summe der Elementen (alle müssen Zahlen sein), z. B.* `sum([5, 2.3, 8])` *-> 15.3*
`len()`	*ergibt die Anzahl der Elemente, z.B* `len(beste_liste)` *-> 3*

6 Zeichenfolgen (`str`)

Ähnlich wie für Listen gibt es auch für Zeichenfolgen recht viele Operationen, die man anwenden kann. Ich gebe hier nur drei Beispiele und wie sie sich auf die Variable `mein_name = "hana"` auswirken:

`.capitalize()`	*das erste Zeichen groß schreiben, also* `mein_name.capitalize` *->* `"Hana"`
`.upper()`	*alles groß schreiben, also* `mein_name.upper()` *->* `"HANA"`
`.center()`	*beide Seiten mit einem Zeichen erfüllen, z. B.* `mein_name.center(8, "!")` *->* `"!!hana!!"`

Es gibt noch recht viele Operationen auf Zeichenfolgen, die man auf verschiedene Wege entdecken kann. Die beliebteste Methode? So was wie 'python string operations' in deine Lieblingssuchmaschine eintippen.

7 Noch ein paar Hinweise

Programmieren lernt man durch machen. Einschließlich Fehler machen. Oft Fehlermeldungen zu bekommen ist die Regel, nicht die Ausnahme - auch für erfahrene Entwickler*innen. Die Fehlermeldungen liefern auch nützliche Informationen. Oft kann man erkennen, in welcher Zeile der Fehler passierte. Außerdem gibt es verschiedene Fehlermeldungen: ein `Identation Error` bedeutet, dass etwas mit Leerzeichen los ist; `Syntax Error` bedeutet oft, dass eine Klammer oder Doppelpunkt fehlt; `Index Error` mag bedeuten, dass man ein Element der Liste erreichen will, dass gar nicht existiert; und so weiter. Falls auch diese Überlegungen den Fehler nicht erläutern, kann es wieder helfen, die ganze Fehlermeldung zu googeln. Wahrscheinlich stand schon der eine oder andere vor einem ähnlichen Problem.

Kommentare sind auch nützliche Sachen. Das sind die Zeilen, die mit # anfangen und die der Computer einfach ignoriert. Für wen sind sie denn da? Für dich, wenn du nach ein paar Wochen, Monaten oder Jahren dein Code wieder liest und versuchst herauszufinden, was der verdammte Autor (also du aus Vergangenheit) im Sinn hatte, wenn er dieses Stück Code geschrieben hat. Gut gestellte Kommentare können diese Arbeit sehr erleichtern.

```
# ich weiß nicht, warum das folgende funktioniert, aber es funktioniert, also Finger weg!
```