

Aufgabenblatt 6

10 Der Python-Interpreter

1. Arbeiten Sie mit dem Terminal!

Sie haben bereits das Terminal kennen gelernt und können damit umgehen.

- a) Wo liegt der Python-Interpreter, der standardmäßig vom System aufgerufen wird, wenn Sie den Befehl `python` eingeben? Um welche Version von Python handelt es sich?

- b) Wo liegen weitere Versionen von Python? Um welche Version handelt es sich jeweils dabei? Finden Sie `python3.3` – die neueste Version von Python?

2. Der Python-Interpreter lässt sich ähnlich wie eine UNIX-Shell bedienen. Der Befehl `python`, ohne Argumente, startet den interaktiven Modus.

- a) Starten Sie nun den Python-Interpreter mit dem Befehl `python` in der Konsole und probieren Sie die folgenden Befehle aus!

```
>>> 4*5
>>> 64 / 16.0
>>> 64 / 15
>>> 64.0 / 15
>>> 64.0 // 15
>>> 7 % 3
>>> 7 * 5-9
>>> 7 * (5-9)
>>> -42
>>> --42
>>> ---42
>>> 10 * "Hallo"
```

- b) Weisen Sie nun einigen Variablen Werte zu und arbeiten Sie mit diesen Variablen.

```
>>> ganze_zahl = 4 + 2
>>> float_zahl = 7.3
>>> string1, string2 = "It's done!", 'Woohoo!'
>>> ganze_zahl + float_zahl
>>> neu = ganze_zahl * string1
>>> print(string2 * float_zahl)
>>> neu
>>> print neu
```

- c) Welche drei Möglichkeiten gibt es den Interpreter zu beenden / verlassen?
-

- d) Schließen Sie nun den Interpreter und starten Sie ihn erneut! Führen Sie den letzten Befehl `print neu` erneut aus! Was stellen Sie fest?
-

11 Variablen und Datentypen

Sie haben jetzt schon gesehen, wie Variablen in Python benutzt werden. Variablen, oder ihr Typ, müssen – anders als in herkömmlichen Programmiersprachen – vor ihrer Benutzung nicht deklariert werden. Sie können Variablen einfach benutzen, wenn Sie sie benötigen.

1. Variablen können beliebig überschrieben werden!

- a) Was ist Ihrer Meinung nach die Ausgabe dieser Codezeilen? Überlegen Sie zunächst und probieren Sie es anschließend aus!

```
>>> var1 = float(5)
>>> var2 = var1 + 2
>>> var1 = 6.0
>>> print (var1)
>>> print (var2)
>>> print (var1 + var2)
>>> var2 = "Hallo"
>>> print (var1 + var2)
```

b) Wozu dient die Funktion `float()`?

2. Überprüfen Sie Ihre Annahme im Python-Interpreter!

3. Arithmetische Operationen, also Operationen auf Zahlen, sind ebenfalls vorhanden.

```
>>> zahl = 1 + 2 - 3 * 4 / 5.0
```

a) Versuchen Sie die korrekte Belegung von `number` anzugeben. Überprüfen Sie Ihre Annahme anschließend im Interpreter!

b) In Welcher Reihenfolge arbeitet Python die Operatoren ab?

4. Ein weiterer Operator ist der Modulo-Operator. Modulo berechnet den Rest einer Division (meistens mit `mod` abgekürzt und in den meisten Programmiersprachen durch `%` repräsentiert).

Beispiel:

$13 \div 6 = 2 \text{ Rest } 1$ Also ist

$13 \bmod 6 = 1 \pmod{6}$ oder mathematisch $13 \bmod 6 \equiv [1]_6$

Mit welchem Wert ist die Variable `rest` jetzt belegt?

```
>>> rest = 11 % 3
```

5. Noch ein Operator ist `**` (das doppelte Mal-Zeichen). Was bewirkt es? Probieren die folgenden Zeilen aus und geben Sie eine kurze Antwort.

```
>>> square = 7 ** 2
```

```
>>> cube = 2 ** 3
```

6. Strings sind eine Folge von Zeichen. Strings werden von doppelten Anführungszeichen (`"`) oder von einfachen Anführungszeichen (`'`) definiert. *Die Maskierungszeichen funktionieren wie im UNIX-Dateisystem. Auch in Python ist das Backslash (`\`) nicht in den einfachen Anführungszeichen erlaubt.*

```
>>> mystring = 'Text'
```

```
>>> RNB = "R 'n' B steht für Rhythm and Blues"
```

7. Python nutzt + als Konkatenation von Strings:

```
>>> helloworld = "hello" + " " + "world"
```

Das kann auch mit Variablen kombiniert werden:

```
>>> first = hello
>>> second = world
>>> print first + second + "!"
```

Probieren Sie das mit Ihrem Vor- und Nachnamen aus (z. B. „Muster, Max“):

8. Python kann auch den *-Operator in Kombination mit Strings benutzen. Probieren Sie aus!

```
>>> nonsense = "bla" * 10
```

Was macht also der *-Operator?

9. Was bewirkt die Methode len()? Welche Funktion hat sie?

```
>>> astring = "hallihallo !"
>>> print(len(astring))
```

10. Welche verschiedenen Datentypen kennt Python? Finden Sie es mit der type-Funktion heraus!

```
>>> tmp = "str"
>>> type(4)
>>> type(int(4.0))
>>> type(tmp)
>>> type(float(7))
```

11. Zusätzlich zu den bisherigen Datentypen (Integer, Float, String) gibt es noch Listen. Listen sind sehr ähnlich zu Arrays; können jedoch jegliche Art von Variablen aufnehmen. Ein Beispiel, eine Liste zu erstellen, folgt:

```
>>> mylist = [7, 1, 5]
>>> mylist.append(3)
>>> mylist.sort()
>>> print(mylist)
```

- a) Listen können verbunden werden. Welche Ausgabe erwarten Sie beim Lesen des folgenden Codes? Notieren Sie!

```
>>> even_numbers = [2,4,6,8]
>>> odd_numbers = [1,3,5,7]
>>> all_numbers = odd_numbers + even_numbers
```

- b) Welchen Wert hat `all_numbers` tatsächlich? Prüfen Sie es nach!
-

12. Auf Listen kann wie auf Arrays zugegriffen werden.

```
>>> mylist = ["eins", 2, [3, 4], "fünf"]
>>> print(mylist[2])
```

- a) Welchen Wert erwarten Sie als Ausgabe? _____
- b) Prüfen Sie nach! Welcher Wert wird ausgegeben? _____
- c) Von welchem Variablentyp ist der ausgegebene Wert? _____
- d) Worin besteht der Unterschied zu Wortlisten in UNIX? _____
- e) Überprüfen Sie! Ist "eins" in mylist? _____

12 Ausgaben

Den *print*-Operator haben Sie bereits in seiner einfachsten Form kennen gelernt. Um Ausgaben mit Werten von Variablen zu kombinieren, wird der %-Operator verwendet.

```
>>> hello = 'Hallo du!'
>>> myvar = "Wie geht's dir?"
>>> print("%s %s") % (hello, myvar)
```

Diese Arbeitsweise ist der der *sprintf*-Anweisung aus der Programmiersprache C nachempfunden.

`%[flags][width][.precision]conversion`
oder
`%[schalter][breite][.genauigkeit]code`

Eine Tabelle aller möglichen Flags und Conversions ist auf der Homepage von Python unter <http://docs.python.org/2/library/stdtypes.html#string-formatting> zu finden.

1. Bei der Darstellung von Zahlen kann man angeben, wie viel Platz für die Zahl insgesamt eingeräumt werden soll:

```
>>> print('Zahl: %5d') %5
Zahl:      5
```

2. Bei Gleitkommazahlen kann man angeben, auf wie viele Nachkommastellen gerundet ausgegeben werden soll. Möchte man der Zahl auch vor dem Komma Platz einräumen, so muss bei **width** die komplette Länge der gewünschten Ausgabe angegeben werden. Möchte man also Platz für **drei** Stellen vor und **zwei** Stellen nach dem Komma einräumen, dann muss man folgendes eingeben:

```
>>> print('Zahl: %6.2f') %6.3774
Zahl:    6.38
```

Wie berechnet sich die Länge **width** bei Gleitkommazahlen?

3. Informieren Sie sich und füllen Sie die folgende Tabelle aus!

Flag	Bedeutung
'0'	
'_'	
' '	
'+'	
<hr/>	
Conversion	Bedeutung
'd'	
'o'	
'x' / 'X'	
'f'	
's'	
'%'	

4. Lösen Sie das Rätsel, indem Sie die richtigen Formatierungszeichen einsetzen! ¹
Warum verwechseln (englischsprachige) Informatiker Weihnachten und Halloween?

```
>>> day = 25
>>> print("Christmas is on DEC %_" % day
>>> print("Halloween is on OCT %_" % day
```

13 Skripting

Python-Skripte sind Textdateien mit Python-Quellcode. Ein Python-Skript kann dem Interpreter zur Ausführung übergeben werden.

1. Schreiben Sie Ihr erstes Python-Skript:
Öffnen Sie eine Texteditor Ihrer Wahl, z. B. `gedit`:

```
$ gedit hallo.py
```

Geben Sie ein:

```
1 print('Hallo Welt')
```

Speichern Sie Das Dokument ab. Das Skript kann nun ausgeführt werden:

```
$ python hallo.py
```

2. Kommentare dienen dazu den Quelltext für den Leser verständlich zu machen. Sie werden bei der Ausführung des Skripts ignoriert. Python kennt zwei Arten von Kommentaren:

- a) Mehrzeilige Kommentare

```
1 """Die folgende Funktion gibt
2     Hallo Welt!
3     auf der Kommandozeile aus"""
4 print("Hallo Welt!")
```

- b) Einzeilige Kommentare

```
1 print("Hallo Welt!") # gibt >Hallo Welt!< aus
```

¹Am 24. Dezember ist Heiligabend. Der Festtag Weihnachten selbst ist am 25. Dezember.

3. Schreiben Sie ein Skript!

Sie können sich an den folgenden Befehlen² orientieren:

<code>str.capitalize()</code>	<code>str.swapcase()</code>
<code>str.lower()</code>	<code>str.title()</code>
<code>str.replace(old, new[, count])</code>	<code>str.translate(table[, deletechars])</code>
<code>str.split([sep[, maxsplit]])</code>	<code>str.upper()</code>

Zugriffe auf Sequenzen sind über drei Operationen möglich:

S[i] Indizierung

Zugriff auf einzelnes Zeichen an bestimmter Stelle. Negative Indizes zählen vom Ende aus.

S[i:j] Slicing (Teilbereichsbildung)

Extrahieren von Bereichen. Die obere Grenze ist exklusiv.

Beispiele:

`S[1:]` geht von Index 1 bis zum Ende
`S[:-1]` selektiert alle bis auf das letzte Element
`S[n:m]` selektiert von n bis ausschließlich m
`S[:]` erstellt eine komplette Kopie

S[i:j:k] Extended Slicing

Jedes k -te Element wird selektiert.

Beispiele:

`S[::2]` extrahiert jedes zweite Element
`S[5:2:-1]` wählt ab Index 5 bis ausschließlich 2 die Elemente von rechts nach links

²Erklärungen auf <http://docs.python.org/2/library/stdtypes.html#string-methods>

Simulieren Sie die folgende Ausgabe mit Ihren Daten:

```
$ python skript.py
Erhard, Heinz
HEINZ erhard
Erste drei Buchstaben des Vornamens: Hei
Letzte zwei Buchstaben des Nachnamens: rd
H. Erhard
ein RHAR
Ein Rhar
```

- Es soll zwei Variablen **vname** und **nname** für den Vornamen und den Nachnamen definiert werden.
- Arbeiten Sie nur mit den beiden Variablen!
- Geben Sie den Nachnamen und Vornamen, getrennt durch ein Komma, aus!
- Geben Sie den Vornamen groß- und den Nachnamen kleingeschrieben aus!
- Geben Sie die ersten drei Zeichen des Vornamens aus!
- Geben Sie die letzten zwei Zeichen des Nachnamens aus!
- Geben Sie das erste Zeichen des Vornamens, gefolgt von einem Punkt und den Nachnamen aus!
- Geben Sie das zweite bis vierte Zeichen des Vornamens normal und das zweite bis vorletzte Zeichen des Nachnamens in Großbuchstaben aus!
- **Ergänzen** Sie die letzte Codezeile derart, dass der erste Buchstabe jeweils groß- und der Rest kleingeschrieben ist!

Nützliche Links

- Die offizielle Python Dokumentation
<http://docs.python.org/2/>
- Python - Das umfassende Handbuch. Galileo Computing <openbook>
<http://openbook.galileocomputing.de/python/>
- Python 3.3 Tutorial auf deutsch
<http://tutorial.pocoo.org/>
- Interaktives englischsprachiges Tutorial
<http://www.learnpython.org/>
- Wikibooks: Python Programming (engl.)
https://en.wikibooks.org/wiki/Python_Programming
- Wikibooks: Python unter Linux
https://de.wikibooks.org/wiki/Python_unter_Linux