Hochschule RheinMain
H. Arega, M.-A. Rohde, W. Weitz

Studiengang Medieninformatik
Wintersemester 2021/22

Programmieren 3

Prog3-Ueb-07

Aufgabe 1 (Interaktive Verwendung)

Starten Sie bitte Python (`python3`) in einer Shell und probieren Sie die linke Spalte der folgenden Ausdrücke aus. Machen Sie sich klar, was warum passiert.

Starten Sie nun `ipython3` und probieren Sie die rechte Spalte aus. `ipython3` bietet Ihnen einige Extrafeatures im interaktiven Betrieb an, z.B. Vervollständigungsvorschläge mit der Tabulator-Taste und weitere Unterstützungsfunktionen. Eigentlich möchte man im interaktiven Modus vorzugsweise mit `ipython3` arbeiten – wenn es verfügbar ist.

```
2 / 5
2 / 5.0
3 ** 4
int("42")
bin(42)
float("42")
"Hallo"
print("Hallo" + " Welt")
print("Hallo","Welt")
(i,j) = (1,2)
i
i,j
i,j = 3,4
i,j
True/2.
list(range(10))
list(range(1,1000,100))
s = "ham"
"eggs " + 2
"ham " "and " "eggs"
s * 5
s[:0]
s[0][0][0]
```

```
("x",)[0]
"eggs"[2]
("x","y")[1]
(1,2) + (3+4, 5)
(1,2) + (3)
(1,2) + (3,)
lst = [1,2,3] + [4,5,6]
lst[:]
lst[:0]
lst[-2]
lst[-2:]
([1,2,3] + [4,5,6])[2:4]
(lst[2],lst[3])
len(lst)
m = list("hallo"); m
"dio" in "He's an idiot, but he's ok"
0 or "" or [] or () or {} or None or "Ende"
(x,y) = (1,2)
(x,y) = (y,x)
bin(17),oct(17),hex(17), str(0x1dfc)
101, 0o101, 0x101
a,b = 0b001011,0b000110
a & b, a | b, a ^ b, b << 2
```

Aufgabe 2 (Slicing)

Extrahieren Sie bitte aus einer Liste der Zahlen von 0 bis 100:

- die ersten 10 Zahlen,
- die letzten 10 Zahlen,
- jede 10.te Zahl (beginnend mit 0),

- die mittlere Zahl,
- jede dritte Zahl, aber davon nicht die ersten vier und nicht die letzten fünf

Aufgabe 3 (Kontrollstrukturen)

Schreiben Sie folgende `for`-Schleife als `while`-Schleife um.

```
lst = [1,2,3]
for e in lst:
    print(e)
```

Schreiben Sie folgende **`while`**-Schleife als **`for`**-Schleife um.

```
m = [5,3,6,8,1]
i = 0
while i < len(m):
    z = m[i]
    print(z, "hoch zwei ist", z**2)
    i = i+1
```

Schreiben Sie die Programme so kompakt wie möglich.

Aufgabe 4 (Funktionen, Strings)

Schreiben Sie bitte eine Funktion `devocalize(s)`, die einen String `s` entgegennimmt und den gleichen String ohne Vokale als Ergebnis zurückliefert.

Beispiel: `devocalize("Das ist ein Baerenspass")` liefert `Ds st n Brnspss`.

Aufgabe 5 (Funktionen, Docstrings)

Haben Sie daran gedacht, Ihrer Funktion aus der vorigen Aufgabe einen aussagekräftigen Dokumentationsstring (Docstring) mitzugeben? Falls nein, sollten Sie das nun nachholen.

Überprüfen Sie in einer "Python Shell" mit `help(devocalize)`, ob das `help`-System aus Ihrem Docstring schlau wurde.

Aufgabe 6 (Funktionen, Listen)

Python-Funktionen können natürlich auch rekursiv sein.

Schreiben Sie bitte eine rekursive Funktion `dreh(lst)`, die eine Liste `lst` entgegennimmt und eine Liste als Ergebnis liefert, welches die Listenelemente von `lst` in umgekehrter Reihenfolge enthält. Es soll tatsächlich eine Liste *als Funktionsergebnis* zurückgegeben werden (die Elemente z.B. nur mit `print()` in der richtigen Reihenfolge im Terminal auszugeben genügt nicht; Natürlich können Sie sich aber den Rückgabewert von `dreh(lst)` zu Testzwecken mit `print()` anzeigen lassen).

Wie Sie sich sicherlich erinnern, dreht man die leere Liste nicht um, das gibt also ein gutes Abbruchkriterium. Eine nichtleere Liste kann man herumdrehen, indem man sich das erste Listenelement merkt, den Rest der Liste herumdreht, und dann das gemerkte Listenelement an die gedrehte Restliste anhängt.

Hinweise:

Eigene Python-Installation

Python wird bei vielen Betriebssystemen mitgeliefert oder ist leicht nachinstallierbar, entweder über den Paketmanager Ihres Systems oder von der Python-Homepage unter `https://www.python.org/downloads/`. Wie in der Vorlesung erwähnt verwenden wir (aus Kompatibilitätsgründen zu den Pool-Installationen) den Sprachstand Python 3.6. Bitte achten Sie darauf, wirklich mit Python Version 3.x zu arbeiten. Es kann auch eine höhere 3er-Version als 3.6 sein, beschränken Sie sich dann bitte auf die in der Vorlesung behandelten Sprachmittel und Bibliotheksmodule der 3.6er-Version.

Manche Systeme werden noch von Hause aus mit der alten Python Version 2.x ausgeliefert, diese ist *nicht* in allen Aspekten (auch syntaktisch nicht) kompatibel. Unter Linux gibt es z.B. gelegentlich ein Kommando `python`, das oft ein altes Python 2 ist, und ein separates `python3`, das wir nutzen würden. Wenn Sie Python interaktiv starten, wird Ihnen die Version ausgegeben, so dass Sie sich leicht vergewissern können. Wenn Sie Python unter Windows aus dem passenden Paket installieren, genügt ggf. `python` (statt `python3`).

VirtualBox/Box.MI und x2go

Eine weitere Möglichkeit ist die Nutzung einer virtuellen Linux-Installation z.B. mit VirtualBox und dem downloadbaren Image auf `https://box.mi.hs-rm.de/` So können Sie eine virtuelle Linux-Installation, auf der die nötigen Tools schon vorinstalliert sind, auf Ihrem (gerade auch Nicht-Linux-)Rechner nutzen (4 GB RAM oder besser mehr sind dabei ratsam).

Auch können Sie sich z.B. mit X2go auf einem der internen MI-Server `linux001.intern.mi.hs-rm.de` oder `linux002.intern.mi.hs-rm.de` einloggen und auf einer Hochschul-Installation (entspricht von der Ausstattung den Rechnerpools) arbeiten. X2go ist ein "Viewer", den Sie auf Ihrem Rechner installieren können und der sich über `ssh` mit einem Hochschulserver verbinden kann. Sie sehen im X2go-Fenster dann den Linux-Desktop (und die Linux-Tools), die in tatsächlich aber auf dem gewählten Server laufen. X2go auf Ihrem Rechner dient nur der Anzeige (also eine Art "grafisches Terminal").

Weitere Hinweise sowohl zur VirtualBox/Box.MI als auch zu X2go finden Sie im MI-Portal unter Dokumentation/HOWTOs.

Entwicklungsumgebung: VS Code mit Python-Extension

Für Python gibt es natürlich auch eine Extension für VisualStudio Code. Hierzu sollten Sie die Erweiterung mit dem Namen "Python" vom Anbieter Microsoft (Vorsicht – es gibt mehrere "irgendwas mit Python", wir nehmen die genannte) installieren. Wenn Sie eine "`.py`"-Datei anlegen, wird der entsprechende Support aktiviert und Sie können wie gewohnt das Programm starten (z.B. mit F5), Breakpoints setzen und debuggen, ähnlich, wie Sie es von C schon kennen. Sie finden `https://code.visualstudio.com/docs/python/python-tutorial` weiter Infos zu dieser Erweiterung.

Der für die Ausführung von Python-Programmen zuständige Python-Interpreter wird *nicht* als Teil der VS-Code-Python-Extension mitinstalliert (häufiges Missverständnis). Da VSCode nach installierten Pythons sucht, sollten Sie *zuerst* Python auf Ihrer Maschine installieren und *danach* die VSCode-Erweiterung. Aus den gleichen Gründen wie bei C wird auch im Python-Teil der Veranstaltung durchgängig dazu geraten, VSCode als Entwicklungsumgebung zu nutzen und den Umgang damit zu üben. Natürlich spricht nichts dagegen, gerade zum schnellen Ausprobieren auch hin und wieder Python im interaktiven Modus ohne Entwicklungsumgebung zu nutzen. Diese Möglichkeit ist ja gerade eine der Stärken interpretierter Sprachen. Ich nutze z.B. Python im interaktiven Modus regelmäßig als "Taschenrechner-Ersatz".