WDV1

Chrsitian Müller, Marc Johannes Honecker und Martin H. Müser

Apr 17, 2023

CONTENTS

In diesem Kurs soll Ihnen beigebracht werden, Computer zur Unterstützung diverser anderer Fächer zu verwenden. Als die dazu beste Programmiersprache hat sich (momentan) Python herauskristallisiert, auf das sich dieser Kurs fokussiert. Python ist bereits seit einigen Jahren State-Of-The-Art für viele Bereiche, sei es Scientific Computing oder (vor allem in den letzten Jahren) KI. Deshalb ist im Moment ein Ende seines Siegeszuges noch nicht absehbar.

Für den Kurs haben wir einen "ganzheitlichen" Ansatz gewählt, der von üblichen Programmierkursen abweicht. Sie lernen also im übertragenen Sinne zunächst ganze Sätze ohne mit vielen grammatikalischen Begriffen konfrontiert zu werden. Damit sollen Sie schnell in die Lage versetzt werden kontinuierlich durch das Skript an Funktionen verschiedener Bibliotheken herangefürht zu werden. Damit werdne an einigen Beispielen auch Hitergrund z.B. zu Numerik oder Statistik formlos gelehrt oder es wird aufgezeigt, wie die Probleme mit herkömmlichen Programmen, allerdings in Python Syntax gelöst würden. Am Ende von WDV-1 sollten Sie dann in der Lage sein, echte Python Bücher (quer) zu lesen, aber auch schnell andere Programmiersprachen wie z.B. C++ zu erlernen.

In den Hausaufgaben wird es zwei Testate geben, deren Bestehen Ihnen erlaubt an der finalen Leistungkontrolle, der Klausur, teilzunehmen. Weder bei Testaten, noch Klausuren sind Computer erlaubt. Das klingt paradox, aber es wäre sonst zu einfach zu mogeln. Leistungkontrollen werden vermutlich aus zwei, manchmal drei Teilen bestehen:

1. Ein Teil, in dem Ihnen der Code gegeben wird und Sie vorhersagen müssen, was bei dei dessen Ausführung passiert.
2. Ein Teil, in dem Sie selbst Python Code schreiben.
3. Unter Umständen werden Sie noch aufgefordert, Fehler in einem gegebenen Code zu markieren/verbessern.

Dieses Skript ist kein Lehrbuch. Details zur Syntax und andere Regeln werdne in der Vorlesung und den Übungen erklärt oder können in den angegebenen Referenzen nachgeschlagen werden. Dieses Skript ist eher eine Ansammlung von kleinen kommentierten Code-Schnipsel.

Ablauf einer Lehreinheit

Eine Lehreinheit besteht aus 3 Lehrstunden zu je 45 Minuten. In der 1. Stunde werden die Hausaufgaben besprochen, in der 2. Stunde neue Inhalte vermittelt und die 3. Stunde dient dazu, dass Studierende mit der Lösung des neuen Aufgabenblattes bginnen und dabei Unklarheiten oder andere Schwierigkeiten an die anwesenden HiWis stellen können.

CHAPTER

ONE

HELLO WORLD!

Das “Hello World! - Programm” ist traditionell das erste Programm, welches man in einer neuen Programmiersprache schreibt. Aber was ist überhaupt das “Hello World! - Programm”? Dieses Programm soll tatsächlich nur “Hello World!” ausgeben. Das klingt sehr simpel (ist es relativ schnell auch), es hilft aber dennoch sich bereits mit grundsätzlichen Strukturen vertraut zu machen. Deshalb wird dieses besondere Programm auch in diesem Kurs der Startpunkt für Python sein.

```
# Hier beginnt das Hello World! - Programm
def main():
    # wir geben "Hello World!" aus
    print("Hello World!")

# wir starten das Programm
main()
```

```
Hello World!
```

Diese kleine Programm besteht im wesentlichen aus drei Komponenten.

1. Einem **Kommentar:** Hierbei handelt es sich um eine Notiz des Programmieres. Eingeleitet wird dies durch das ‘#’-Symbol. Alles was hinter diesem Sybmol geschrieben steht, wird komplett bei der Übersetzung des Programms ignoriert.
2. Einer **Funktionendefinition:** Damit man Funktionen nutzen kann, muss man diese vorher definieren. Danach kann man diese beliebig oft benutzen. Man muss noch erwähnen, dass das main()-Programm ein besonderes Programm ist. Dieses dient im Allgemeinen als Startpunkt für die Übersetzung. Die Syntax ist hier immer die gleiche. Zuerst das Schlüsselwort `def`, danach der Name der Funktion, gefolgt von einem Doppelpunkt.
3. Einem **Funktionenaufruf:** Streng genommen sind hier zwei Aufrufe. Der erste in Zeile 3 und der zweite in Zeile 5. Der Aufruf an print() schreibt den Inhalt (hier: “Hello World!”) raus. Der Aufruf an main() startet das Programm und gibt dann “Hello World!” aus. Außerdem wird dem aufmerksamen Leser aufgefallen sein, dass wir `print()` nicht definiert hatten, obwohl doch eigentlich jede Funktion vorher definert sein müsste. Hier ist es allerdings ein wenig anders. Python liefert direkt die `print()`-Funktion mit aus, und sie damit einfach verfügbar. Es gibt noch weitere solcher Funktionen, die bei gegebener Zeit noch eingeführt werden.

Das main()-Programm in Python

Es ist nicht notwendig in Python mit dem main() Programm zu starten. Deshalb wird es bald nicht mehr genutzt werden. Da aber hiermit einfach die Syntax für eine Funktionen-Definition gezeigt werden kann wurde es hier gezeigt. Außerdem wird in den meisten anderen Programmiersprachen das main() Programm als Einstieg verlangt und somit haben Sie ein solches Programm dann schon einmal gesehen. Einige Beispiele wären (nur zur Illustration):

C

```c
# include <stdio.h>

int main() {
    printf("Hello World!\n");
    return 0;
}
```

C++

```cpp
#include <iostream>

int main()
{
    std::cout << "Hello World!" << std::endl;
    return 0;
}
```

Java

```java
class Main {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Rust

```rust
fn main() {
    println!("Hello World!");
```

1.1 Das Hello World! - Programm Improved

Zum Beispiel betrachten wir folgendes Prgramm:

```
def main():
    print("Hello World!")
    print("Hello World!")

main()
```

```
Hello World!
Hello World!
```

Wir können auch mehrfach Hello World! ausgeben.

Und sogar noch öfter:

```
def main():
    print("Hello World!")
    print("Hello World!")
    print("Hello World!")

main()
```

```
Hello World!
Hello World!
Hello World!
```

Aber auf die Dauer wird dies dann doch ganz schön schreibaufwendig. Aber Python (und alle anderen Programmiersprachen) hält hierfür etwas sehr vereinfachendes bereit: ***Schleifen***

1.1.1 Schleifen (engl. Loops)

Also schreiben wir unser letztes Programm ein wenig um.

```
def main():
    for i in range(3):
        print("Hello World!")

main()
```

```
Hello World!
Hello World!
Hello World!
```

Dies ist eine sogenannte `for`-Schleife. `for`-Schleifen sind bei weitem die meist genutzten Schleifen in Python.

Die Syntax ist im wesentlichen immer die wie oben gezeigt. Die beiden auffälligen Komponenten werden noch hier einzeln erläutert:

- `range()`: Diese Funktion gibt eine Reihe oder Folge (engl. range) von (Ganz-)Zahlen zurück. Hierbei gibt es insgsamt drei verschiedene Fälle:
  - `range(n)`: Hier wird eine Reihe von Zahlen beginnend bei 0 und endend bei **n-1** (also insgesamt n Zahlen) erzeugt
  - `range(a,b)`: Hier wird eine Reihe von Zahlen beginnend bei `a` und endend bei **b-1** erzeugt.
  - `range(a,b,inc)`: Hier wird eine Reihe von Zahlen erzeugt, die bei `a` startet, wobei immer um `inc` hochgezählt wird. Wenn das letzte Ergbnis $> b - 1$ ist, so wird die letzt mögliche Zahl, für die gilt, dass sie kleiner als `b-1` zurückgegeben. In Code sieht dies vereinfacht wie folgt aus:

```
def my_range(start, end, inc):
    # wir müssen uns alle berechneten Zahlen merken. Dies kann man in Python in einer␣
↪sogenannten
    # Liste machen. Später mehr dazu.
    range_list = list()
```

(continues on next page)

(continued from previous page)

```
    # while-Schleifen brauchen uns im Moment nicht wirklich zu interessieren. Diese␣
↪Art Schleife
    # führt solange einen bestimmten Programmteil immer und immer wieder aus, bis␣
↪eine bestimmte
    # Bedingung NICHT MEHR erfüllt ist (hier also: sobald start > end ist, brechen␣
↪wir ab)
    while start < end:
        range_list.append(start)
        start += inc

    # jetzt geben wir die Liste zurück
    return range_list
```

Die Schleife aus dem letzten Code-Schnipsel ist eine sogenannte `While`-Schleife. `While`-Schleifen führen einen Code-Abschnitt genauso lange aus, wie eine Bedingung gilt (also in unserem Beispiel solange `start < end`).

Jedoch Vorsicht mit `While`-Schleifen. Betrachten wir das folgende kleine Programm:

```
def my_never_ending_loop():
    # '==' steht für den Test auf Gleichheit; wir testen also, ob 1 gleich 1 ist
    while 1 == 1:
        myvar = 1
```

Es sollte ziemlich offensichtlich sein, dass diese Schleife niemals enden wird. Deswegen werden wir in den meisten Fällen eher selten `While`-Schleifen zu Gesicht bekommen und beschränken uns hauptsächlich auf `for`-Schleifen. Nur damit haben Sie schon einmal eine `While`-Schleife gesehen, und sind nicht komplett hilflos, falls Sie eine einmal sehen sollten.

1.1.2 Die `if-else` - Anweisung

Wir schauen uns noch eine weitere Erweiterung des Hello World! - Programmes an. Zum Beispiel möchte man an allen Tagen außer Montag `Hello beautiful World!` ausgeben, und montags `hello world`. Eine mögliche Implementierung wäre die Folgende:

```
def main():
    is_monday = False

    if is_monday:
        print("hello world")
    else:
        print("Hello beautiful World!")

main()
```

```
Hello beautiful World!
```

Das einzige wirklich neue/nicht direkt sich selbst erklärende Schlüsselwort sollte `False` sein. `False` ist englisch und steht einfach nur für *Falsch*. Ist also eine Bedinung *Falsch*, so wird in einer if - Anweisung der `else`-Zweig ausgeführt. Das Gegenstück zu `False` ist `True` (= wahr). Wäre, um in unserem Beispiel zu bleiben, heute Montag, so bekämen wir `Hello beautiful World!` als Ausgabe. Insgesamt wird dies unter dem Begriff der booleschen Algebra zusammengefasst.

Boolesche Algebra

Boolesche Algebra steht im Übrigen auch hinter der Bedingung in der `While`-Schleife! Denn solange die Bedingung der Schleife wahr ist (also zu `True` auswertet), wird der Schleifenrumpf ausgeführt. Sobald die Bedingung nicht mehr wahr ist (also zu `False` auswertet), wird die Schleife abgebrochen, und das restliche Programm nach der Schleife (welches auch leer sein kann) wird ausgeführt.