Algorithmik und Programmieren

Brückenkurs

Michael Wolf

22.9.2014 - 26.9.2014

Table of Contents

Organisatiorisches

Grundlagen

- Variablen
- Operatoren
- Kontrollstrukturen

Termine

Der Kurs findet an folgenden Terminen statt:

- Montag 22.9.14 11:15 - 13:00
- Dienstag 23.9.14 14:15 - 17:00
- Mittwoch 24.9.14 11:15 - 16:00
- Donnerstag 25.9.14 11:15 - 13:00
- Freitag 26.9.14 14:15 - 16:00

Für wen ist dieser Kurs

Und was lernt man hier

Der Kurs ist für:

- Programmieranfänger
- Informatikstudierende und die es noch werden wollen

Was lernt man hier?

- Grundlagen der Programmiersprache Python
- Grundlagen der Programmierung
- Implementierung von einfachen Algorithmen

Ablauf

Jede Einheit umfasst zwei Teile

Theorie Besprechung der Konzepte und Hintergründe, die benötigt werden um Programme zu schreiben.

Praxis Vertiefung der Theorie anhand zahlreicher Beispiele, die Sie alleine oder in Zusammenarbeit mit ihrem Sitznachbarn lösen sollten.

Ablauf der Übungen

Die praktischen Übungen machen den Haupteil dieser Lehrveranstaltung aus. Am effektivsten lernt man Programmieren, indem man es macht!
Zu jeder Übungseinheit gibt es einen Übunszettel, den man alleine oder mit dem Sitznachbarn löst. Die Aufgaben beziehen sich auf die Konzepte die zuerst im Theorieteil der Übung besprochen werden.

Was ist Python?

- wurde Anfang der 90ger von Guido van Rossum entwickelt
- legt großen Wert auf Programmlesbarkeit (kurzer, einfacher Syntax)
- Plattformunabhängig
- interpretierte Hochsprache
- sehr beliebt in der Hacker Community
- Frei und Quelloffen

Resourcen

- python homepage `https://www.python.org/`
- youtube
- `http://www.python-kurs.eu/`
- A Byte of Python, kostenlos zum download hier: `http://swaroopch.com/notes/python/`
- die Hilfe im Interpreter *help()*

Und los...

"Manchmal muss man rennen, bevor man laufen kann"

Tony Stark zu Jarvis, Iron Man

- Konsole öffnen
- python eingeben
- 1+1 eingeben
- schauen was passiert
- Abbrechen mit Ctrl+D oder exit() eingeben

Was ist da gerade passiert?

Das war der Interpreter, 1 und 1 wurden erfolgreich addiert und das Ergebnis war 2
Es gibt zwei Möglichkeiten ein Python Programm auszuführen:

- Interpreter - Nach jeder Zeile Code sieht man sofort das Ergebnis.
- Ein Programm kann man aber auch mittels *python programm.py* ausführen. Alle Kommandos in *programm.py* werden so ausgeführt, als ob man Zeile für Zeile im Interpreter eingibt.

Variablen

Was ist eine Variable

- Ein Programm verarbeitet **Daten**, die in sogenannten **Variablen** abgelegt werden.
- Ein Programm legt die Ergebnisse wieder in solchen **Variablen** ab.
- Eine Variable hat einen **Variablennamen** und einen **Typ**

Variablennamen

Gültige Variablennamen sind eine beliebige Kombination aus folgenden Zeichen: a-z, A-Z, 0-9 und '_'.
Ausnahmen:

- Variablennamen dürfen nicht mit Ziffern beginnen
- Reservierte Schlüsselwörter and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, with, yield

Variablennamen sollten selbsterklärend sein und englische Namen haben
Zum Beispiel: der Vorname sollte in eine Variable *firstname* gespeichert werden und nicht in eine Variable *f*

Datentypen

Python bietet Datentypen für die gängigsten Anwendungen:

- Strings (Zeichenketten): 'This␣is␣text!', "That's␣more␣text."
- Integer (Ganzzahlen): 1, 4294967296, 010, 0xFF, 0b100
- Float (Gleitkommazahlen): 1.35, 1.56e-5 ($1.56 * 10^{-5}$)
- Complex (Komplexe Zahlen): $2 + 3j$
- Bool (Wahr Falsch): True

Variablen cont.

Variablen haben in python keinen statischen Typ, d.h. es muss nicht vorher angegeben werden, welchen Typ sie haben und der Typ kann sich zur Laufzeit ändern.
Variablen haben aber einen dynamischen Typ, der bestimmt welche Operationen für diese Variable zulässig sind!

```
1 >>> a = 2
2 >>> a = "test"
3 >>> a + 3
4 Traceback (most recent call last):
5   File "<stdin>", line 1, in <module>
6 TypeError: cannot concatenate 'str' and
    'int' objects
```

Operatoren

Über Operatoren erhält man die Möglichkeit, z.B. Variablen mit Inhalten zu belegen oder die in ihnen gespeicherten Werte zu verändern.

Operatoren cont.

Operator	Erklärung	Beispiel
+, -	Addition, Subtraktion	10 - 3
*, /, %	Mult., Division, Rest	27 % 7 $\rightarrow$ 6
+x, -x	Vorzeichen	-3
~x	Bitweises Not	3 - 4 Ergebnis: -8
$<, <=, >, >=, !=, ==$	Vergleichsoperatoren	2 <= 3 $\rightarrow$ WAHR
or, and, not	ODER, UND, NICHT	(a or b) and c
**	Exponentiation	2**3 $\rightarrow$ 8
in	"Element von"	1 in [3, 2, 1]
|, &, ^	Bitoperatoren or, and, xor	6 $\hat{}$ 3
<<, >>	Shiftoperatoren	6 << 3

Operatioren Beispiele

```
1  a = 2
2  b = 3
3  c = a*b
4
5  print a, "*", b, "=", c
6
7  c = a**b
8  print a, "hoch", b, "ist", c
9
10 if a > b:
11     print "a ist groesser"
12 else:
13     print "a ist kleiner"
```

Kontrollstrukturen

- Bisher wurden die einzelnen Zeilen eines Programms sequentiell abgearbeitet.
- Sehr oft will man aber bestimmte Anweisungen auswählen oder wiederholen.
- Kontrollstrukturen definieren die Reihenfolge, in der Berechnungen durchgeführt werden.

Anweisungen und Blöcke

Anweisung

- Eine Anweisung ist zum Beispiel: x * 3
- anders als in manchen Programmiersprachen muss eine Anweisung nicht von einem Semicolon gefolgt werden

Block

- in Python bestimmt die Einrückung den Code Block
- Ein Block ist syntaktisch äquivalent zu einer einzelnen Anweisung.
- Blöcke kann man auch ineinander schachteln, indem man nochmals mittels Tabulator einrückt
- im Interpreter 4 Whitespaces benutzen, anstatt Tabulator
- Beispiele folgen noch.

Verzweigung mit if

Allgemeine Form:

```
1 if condition_1:
2 >>>print "condition 1 evaluated to true"
3 elif condition_2 and condition_3:
4 >>>print "condition 2 and condition 3
     evaluated to true"
5 elif condition_N:
6 >>>print "condition N evaluated to true"
7 else:
8 >>>print "none of them is true"
```

- >>> steht für einen Tabulator
- Der `else` und die `elif` Zweig(e) ist optional
- die Anweisungen nach den if Statements müssen in einem eigenen Code Block liegen

Exercise 1