

Informatik I

6. Python-Programme schreiben, kommentieren, starten und entwickeln

Bernhard Nebel

Albert-Ludwigs-Universität Freiburg

05. November 2013

Informatik I

05. November 2013 — 6. Python-Programme schreiben, kommentieren, starten und entwickeln

6.1 Programme

6.2 Programme schreiben

6.3 Programme starten

6.4 Programme entwickeln

Programme

6.1 Programme

Programme

Programme

- ▶ Programme = konkretisierte Algorithmen?
- Ja, aber nicht immer! Oft eingebettet in Programme.
- ▶ Folge von Anweisungen und Ausdrücken, die einen bestimmten Zweck erfüllen sollen.
- ▶ Interaktion mit der Umwelt (Benutzer, Sensoren, Dateien)
- ▶ Unter Umständen nicht terminierend (OS, Sensorknoten, ...)
- ▶ Auf jeden Fall länger als 4 Zeilen!

6.2 Programme schreiben

Texteditoren

- ▶ Zum Schreiben von Programmen benutzt man einen **Texteditor**
 - ▶ *notepad* (Windows)
 - ▶ *notepad++* (Windows, Open Source)
 - ▶ *vim* (Open Source)
 - ▶ *emacs* (Open Source)
 - ▶ *gedit* (Open Source)
 - ▶ in IDE integrierter Editor (kommt noch)
 - ▶ Möglichst mit integriertem Syntaxchecker
- alle bis auf *notepad* haben dies oder unterstützen Plugins für Python

Zeilenumbruch

- ▶ Umbrechen, wenn Zeilen zu lang.
- ▶ Implizite Fortsetzung mit öffnenden Klammern und Einrückung (siehe PEP8):

Lange Zeilen

```
foo = long_function_name(var_one, var_two,
                          var_three, var_four)
def long_function_name(
    var_one, var_two, var_three,
    var_four):
    print(var_one)
```

- ▶ Explizite Fortsetzung mit *Backslash*:

Explizite Fortsetzung

```
foo = long_var_name1 + long_var_name2 + \
    long_var_name3
```

Kommentare im Programmtext

- ▶ Programme werden öfter **gelesen** als geschrieben!
- ▶ Auch für ein selbst: Erinnerungen daran, was man sich gedacht.
- ▶ Nicht das offensichtlich kommentieren, sondern Hintergrundinformationen geben.
- ▶ Möglichst englisch kommentieren.
- ▶ Der Rest einer Zeile nach # wird als Kommentar interpretiert.

Block- und Fließtext-Kommentare

- ▶ Blockkommentare: Zeilen, die jeweils mit # beginnen und genauso wie die restlichen Zeilen eingerückt sind beziehen sich auf die folgenden Zeilen.

Block-Kommentare

```
def fib(n):
    # this is a double recursive function
    # runtime is exponential in the argument
    if n == 0:
```

...

- ▶ Fließtext-Kommentare kommentieren einzelne Zeilen.

Schlechte und gute Kommentare

```
x = x + 1 # Increment x
```

```
y = y + 1 # Compensate for border
```

docstring-Kommentare

- ▶ #-Kommentare sind nur für den Leser.
- ▶ Möchte man dem Benutzer Informationen geben, kann man docstring-Kommentare nutzen.
- ▶ Ist der Ausdruck in einer Funktion oder einem Programm (Modul) ein String, wird dieses der docstring, der beim Aufruf der Funktion help ausgegeben wird.
- ▶ Konvention: Benutze den mit drei "-Zeichen eingefassten String, der über mehrere Zeilen gehen kann.

docstring

```
def fib(n):
    """Computes the n-th Fibonacci number.
    The argument must be a positive integer.
    """
```

...

Programme speichern

- ▶ Nachdem man ein Programm eingetippt hat, sollte man es abspeichern.
- ▶ Lege ein Verzeichnis in deinem *Home*-Verzeichnis an, und speichere alle deine Programme da.
- ▶ Füge dem Dateinamen immer die Dateierweiterung `.py` an, damit man weiß, dass es sich um ein Python-Programm handelt.
- ▶ *Windows*: Wähle immer *Alle Dateien* beim Sichern damit nicht `.txt` angehängt wird.

6.3 Programme starten

5 Wege ein Programm zu starten

- ▶ Starten mit explizitem Aufruf von Python3
- ▶ Starten als Skript
- ▶ Starten durch Klicken
- ▶ Starten durch Import
- ▶ Starten in einer IDE

Beispielprogramm: `example.py`

```
print("Hello world")
```

Starten mit explizitem Aufruf von Python3

Shell

```
# python3 example.py
Hello world
```

- ▶ Voraussetzungen:
 - ▶ Wir sind in dem Ordner, in dem die Datei `example.py` liegt.
 - ▶ Die Pfad-Variablen (PATH) wurde so gesetzt, dass der Python-Interpreter gefunden wird.
- ▶ Wird normalerweise bei der Installation geleistet.
- ▶ Kann „per Hand“ nachgetragen werden:
 - ▶ *Windows*: Systemsteuerung → System und Sicherheit → Erweiterte Systemeinstellungen → Erweitert → Umgebungsvariablen
 - ▶ *Unix*: Setzen der PATH-Variablen im entsprechenden Login-Skript oder in der Shell-Konfigurationsdatei (z.B. `~/.bash_profile`)

Starten als Skript

Shell

```
# example.py
Hello world
```

- ▶ Voraussetzungen:
 - ▶ Wir sind in dem Ordner, in dem die Datei `example.py` liegt.
 - ▶ *Windows*: `.py` wurde als Standard-Dateierweiterung für Python registriert.
 - ▶ *Unix*: Die erste Zeile in der Datei `example.py` ist:
`#!/usr/bin/env python3`
 und die Datei hat das `x`-Bit (ausführbare Datei) gesetzt.

Starten durch Klicken

- ▶ Wenn `.py` als Standard-Dateierweiterung für Python registriert ist (geht eigentlich bei allen Plattformen mit Desktop-Oberfläche), kann man die Datei durch Klicken (oder Doppelklicken) starten.
- ▶ Leider wird nur kurz das Shell-Fenster geöffnet, mit Ende des Programms verschwindet es wieder.
- ▶ *Abhilfe*: Am Ende die Anweisung `input()` in das Programm schreiben.
- ▶ *Allerdings*: Bei Fehlern verschwindet das Fenster trotzdem, und man kann keine Parameter beim Aufruf übergeben.
- ▶ Eigentlich nur für fertig entwickelte Programme mit GUI geeignet.

Starten durch Import

- ▶ Nachdem wir Python in Ordner aufgerufen haben, in dem `example.py` liegt:

Python-Interpreter

```
>>> import example
```

```
Hello world
```

- ▶ *Beachte:* Angabe ohne die Dateierweiterung!
- ▶ Funktioniert nur beim ersten Import.

Python-Interpreter

```
>>> import example
```

```
Hello world
```

```
>>> import example
```

```
>>>
```

6.4 Programme entwickeln

IDE = Integrated development environment

Einen Editor aufrufen, dann das Programm in der Shell starten, dann wieder den Editor starten, ...

Stattdessen kann man IDEs einsetzen für:

- ▶ Projektverwaltung
- ▶ Programm editieren
- ▶ Ausführen
- ▶ Testen und Debuggen
- ▶ Dokumentation erzeugen
- ▶ ...

Gibt es in den verschiedensten Komplexitäts- und Qualitätsabstufungen.

Pythons IDE: IDLE

Wohlmöglich benannt nach Eric Idle.

- ▶ Ist 100% in Python geschrieben und benutzt die *tkinter* GUI (graphical user interface).
- ▶ Läuft auf allen Plattformen.
- ▶ Multi-Fenster-Texteditor mit Syntaxkennzeichnung, multipler Zurücknahme, smartere Einrückung.
- ▶ Enthält ein Fenster mit Python-Shell.
- ▶ Rudimentäre Debug-Möglichkeiten.
- ▶ Beschreibung siehe:
<http://docs.python.org/3/library/idle.html>.

IDLE in Aktion

- ▶ File-Menü: New, Open und Recent File zum Öffnen einer neuen bzw. vorhandenen Programmdatei.
- ▶ File-Menü: Save und Save as abhängig davon, welches Fenster aktiv. Entweder die Shell-Interaktionen oder die Programmdatei wird gespeichert.
- ▶ Shell-Menü: Nur im Shell-Fenster aktiv. Hier kann man mit Restart den Interpreter neu starten.
- ▶ Run-Menü: Ist nur im Editorfenster aktiv. Hier kann man die Syntax überprüfen und das Programm starten, nachdem der Interpreter neu gestartet wurde.