Programmierung mit Python für Einsteiger

Kapitel 1 - Einführung in die Programmierung

Autor: Dr. Christian Heckler

Vorbemerkungen

- Literatur: siehe Referenzen
- Verwendete Symbole:
  - ❗: Beispielprogramm
  - ❗: Weitere Erläuterungen im Kurs
  - ❓: Übung

Kursinhalt

- Einführung in die Programmiersprache Python für Programmier-Einsteiger
  - Algorithmen und allgemeine Programmierprinzipien werden nur gestreift
  - Nicht: Einführung in Werkzeuge zur Software-Entwicklung
- Themen:
  - Einführung und Datentypen
  - Kontrollstrukturen: Verzweigungen und Schleifen
  - Funktionen, Module und Pakete
  - Einführung in die Objektorientierung und Vererbung
- Ziel: Vermittlung der Grundlagen und Prinzipien für selbständiges Weiterarbeiten; keine vollständige Einführung in Python.

Programmieren: Was heißt das eigentlich?

- Ausgangspunkt: Problem / Aufgabe, z.B.:
  - Berechnung des Mitgliedsbeitrages der Mitglieder eines Vereins
  - Steuerberechnung
  - Sortierung von Namen
  - Berechnung des Durchschnittsverbrauchs
  - Lösche alle Cookies von der Festplatte
  - Sortiere die Urlaubsfotos unter Verwendung von EXIF-Daten
- Die Aufgabe soll automatisch (durch eine Maschine – in dem Fall also einen Rechner) gelöst werden

Algorithmus

- Um die Aufgabe automatisch zu lösen, benötigt man eine Bearbeitungsvorschrift („Schritt - für - Schritt - Anleitung“). Diese nennt man auch **Algorithmus**.
- Ein Algorithmus muss die folgenden Eigenschaften haben:
  - Endliche Beschreibung.
  - Beendigung nach endlich vielen Schritten.
  - Eindeutig.
- Beispiele:
  - Grundschule: Algorithmus zur Addition zweier Zahlen
  - Algorithmus zur Bestimmung einer Wurzel
  - Umrechnung einer römischen Zahl in eine Dezimalzahl
  - Algorithmus zur Berechnung des Mitgliedsbeitrages
  - Sortieralgorithmus

Darstellungen von Algorithmen

- Es gibt verschiedene Arten der Darstellung eines Algorithmus
  - Flußdiagramm,
  - Struktogramm,
  - Umgangssprache, Pseudocode

Beispiel: Nachschlagen im Lexikon

- Aufgabe: Nachschlagen eines Begriffes im Lexikon
- Algorithmus:
  - Schlage das Lexikon in der Mitte auf
  - Befindet sich dort der gesuchte Begriff: fertig
  - Steht der gesuchte Begriff im Alphabet vor den Begriffen auf der Seite, suche in der vorderen Hälfte des Lexikons, andernfalls in der hinteren Hälfte
- Könnte das ein Rechner schon automatisch ausführen? Warum? Warum nicht?

Beispiel: Nachschlagen im Lexikon II

- Setze den Suchbereich auf das gesamte Lexikon
- So lange der Begriff noch nicht gefunden ist:
  - Schlage das Lexikon in der Mitte des Suchbereichs auf.
  - Steht dort der Begriff: fertig
  - Suchbegriff lexikographisch kleiner als Begriffe auf Seite:
    - JA: Schränke den Suchbereich auf die vordere Hälfte des Suchbereiches ein.
    - NEIN: Schränke den Suchbereich auf die hintere Hälfte des Suchbereiches ein.

Beispiel: Nachschlagen im Lexikon III

- Fragen:
  - Was passiert, wenn der gesuchte Begriff nicht im Lexikon steht? Verfeinern Sie den Algorithmus, so dass auch dieser Fall beachtet wird.
  - Das Lexikon hat 1000 Seiten. Wie oft muss man maximal nachschlagen, um den Begriff zu finden?

Übung - Algorithmus Maximumbestimmung

Es soll der älteste Kursteilnehmer bestimmt werden:

- Beschreiben Sie einen Algorithums!
- Nach wievielen Ausführungsschritten ist der Algorithmus beendet?

Programm

- Ein Programm ist die Umsetzung eines Algorithmus (oder mehrerer Algorithmen) in einer bestimmten Programmiersprache.
- Das Programm kann von einem Rechner automatisch ausgeführt werden.

Das EVA-Prinzip

- Ein Programm bekommt Eingabedaten, verarbeitet diese und liefert ein Ergebnis:

  ***E*: Eingabe**

  Z.B. über die Tastatur, aus einer Datei, aus einer DB, von einem anderen Programm …

  ***V*: Verarbeitung**

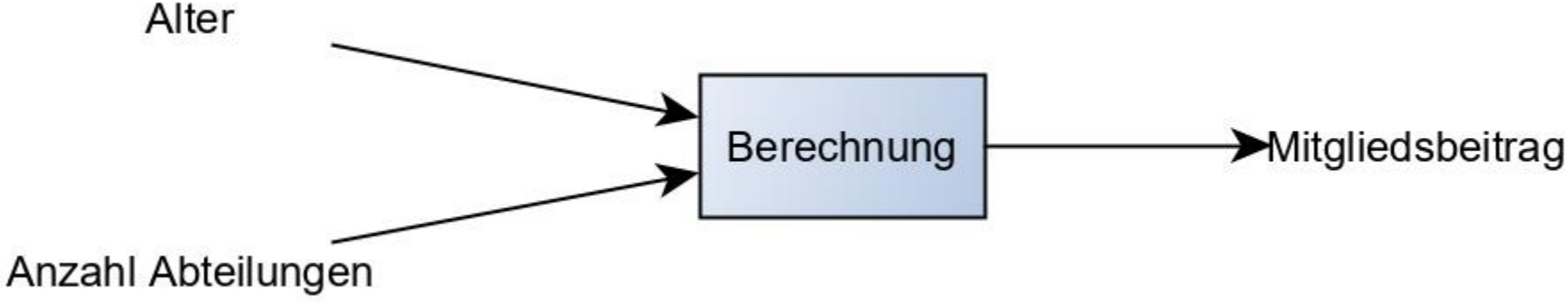
  Die Daten werden gemäß einem Algorithmus verarbeitet.

  ***A*: Ausgabe**

  Z.B. auf dem Bildschirm, dem Drucker, in eine Datei, in eine Datenbank, an ein anderes Programm …

Beispiel: Berechnung des Mitgliedsbeitrags

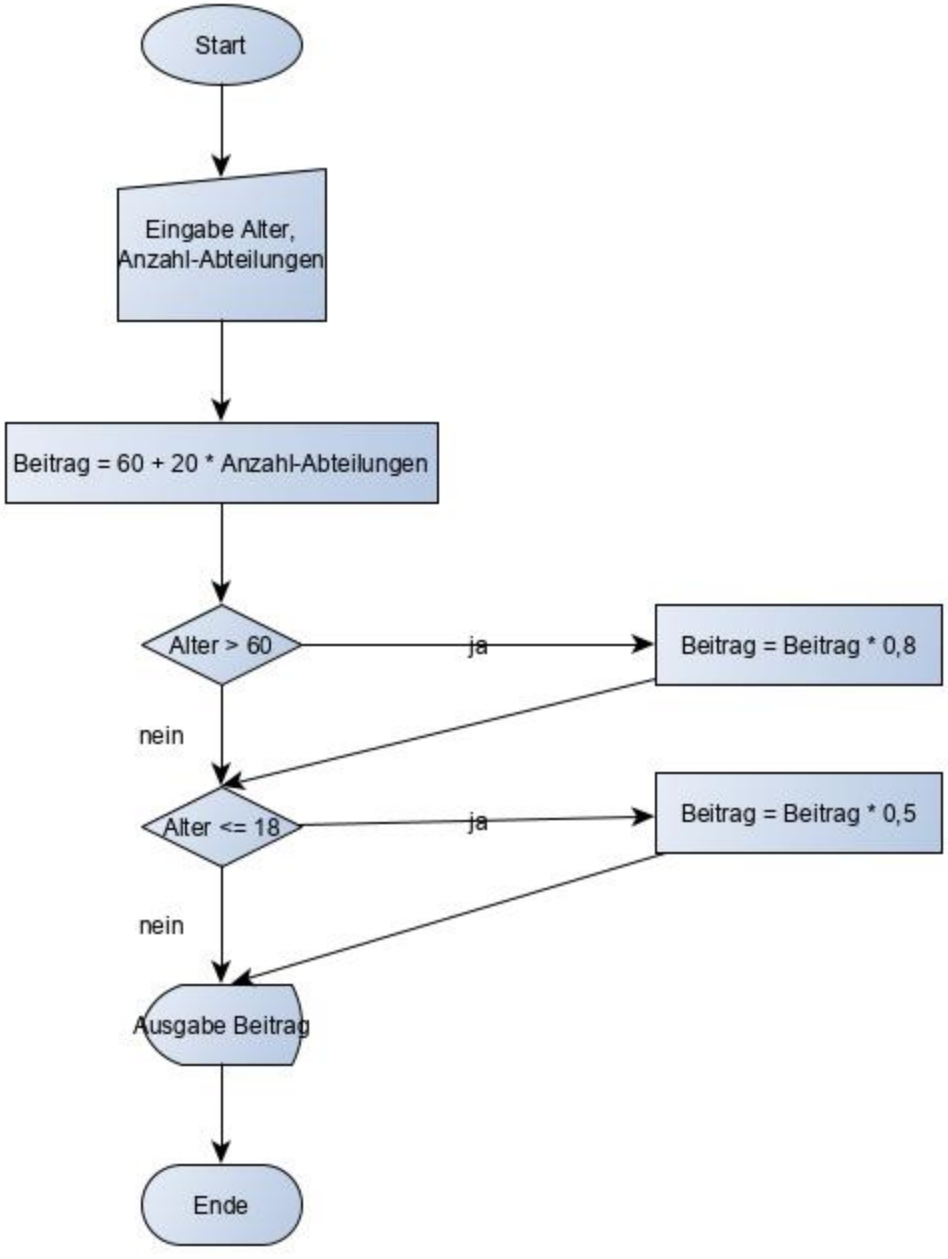
- Aufgabe: Berechnung des Mitgliedsbeitrages eines Vereins:
  - Der Beitrag beträgt 60 EUR im Jahr.
  - Für jede Abteilung, zu der das Mitglied gehört, weitere 20 EUR.
  - Senioren über 60 Jahren zahlen nur 80 %.
  - Jugendliche bis 18 Jahre nur 50 %.
- EVA:



Der Algorithmus (in Umgangssprache)

- Eingabe: Alter und Anzahl-Abteilungen
- Beitrag = 60 EUR + 20 EUR * Anzahl-Abteilungen
- Wenn Alter größer 60: Beitrag = Beitrag * 0,8
- Wenn Alter kleiner oder gleich 18: Beitrag = Beitrag * 0,5
- Ausgabe Beitrag

Der Algorithmus als Flussdiagramm



Programm in Python

- Textdatei („Quelltext“) mitgliedsbeitrag_einfuehrung.py mit dem folgenden Inhalt:

```
alter = int(input("Alter des Mitglieds: "))
anzahl_abteilungen = int(input("Anzahl der Abteilungen des Mitglieds: "))

beitrag = 60 + 20 * anzahl_abteilungen

if alter > 60:
    beitrag = beitrag * 0.8
if alter <= 18:
    beitrag = beitrag * 0.5

print("Mitgliedsbeitrag:", beitrag, "EUR")
```

- Das Programm wird sequentiell Zeile für Zeile ausgeführt.

Ausführung durch einen Rechner

- Ein Rechner versteht nur „Nullen und Einsen“.
- Wie kann also ein Programm, das in Textform vorliegt, durch einen Rechner ausgeführt werden?
- Prinzipiell gibt es zwei Möglichkeiten:
  - Übersetzer / Compiler
  - Interpreter

Übersetzer / Compiler

- Die (Text-) Datei mit dem „Quelltext“ wird in ein ausführbares Programm (eine binäre Datei) übersetzt. (Im Falle von Windows also in eine „exe-Datei“).
- Diese Aufgabe übernimmt ein entsprechendes Programm – der sogenannte Übersetzer oder Compiler.[1]

- Vorteil: schnelle Ausführung
- Nachteil: Läuft nur auf dem System, für das das Prg. übersetzt wurde.
- Beispiele für Prg-Sprachen: C, C++, Pascal, Fortran

Interpreter

- Ein „Interpreter“ (auch ein spezielles Programm) „interpretiert“ das Programm zur „Laufzeit“. D.h. der Interpreter liest das Programm Zeile für Zeile ein und führt die entsprechende(n) Anweisung(en) aus.
- Beispiel: Datei mitgliedsbeitrag_einfuehrung.py
- Ausführung mit dem folgenden Befehl:

```
python mitgliedsbeitrag_einfuehrung.py
```

(„python“ ist dabei das Interpreter-Programm)

Wenn man die Datei-Endung „.py“ mit dem Python-Interpreter verknüpft, kann man die Datei auch „direkt“ (z.B. per Doppelklick) ausführen.

Interpreter

- Vorteil eines Interpreters: Das Programm läuft unverändert überall dort, wo es den Interpreter gibt.
- Nachteile:
  - Programmausführung langsamer
  - Syntaxfehler werden erst zur Laufzeit erkannt
  - Auf der Zielplattform muss der Interpreter zur Laufzeit verfügbar sein. Dieser ist aber evtl. „groß“ (vgl. „Internet der Dinge“)
- Beispiele: Python, Basic

Es gibt auch Mischformen: Der Quelltext wird in ein (binäres) Zwischenformat übersetzt, das dann von einer „virtuellen Maschine“ (einem Interpreter für dieses Format) ausgeführt wird. Beispiel: Java.

Zusammenfassende Fragen

- Was ist ein Algorithmus?
- Welche Eigenschaften hat ein Algorithums?
- Wofür braucht man einen Algorithums?

Referenzen

- [Ste] Ralph Steyer: Programmierung Grundlagen, Herdt-Verlag
- [Kle] Bernd Klein: Einführung in Python 3, Hanser-Verlag
- [Kof] Kofler: Python – Der Grundkurs, Rheinwerk Computing
- [EK] Johannes Ernesti, Peter Kaiser: Python 3 – Das umfassende Handbuch, Rheinwerk Computing
- [Mat] Eric Matthes: Python Crashkurs – Eine praktische, projektbasierte Programmiereinführung, dpunkt.verlag
- [Swe] Sweigart: Eigene Spiele programmieren: Python lernen