Clemens Kaesler

Das große Python3 Workbook

Mit vielen Beispielen und Übungen -
Programmieren leicht gemacht!

2., überarbeitete Auflage

INHALTSVERZEICHNIS

Vorwort

Dieses Workbook soll Sie dabei unterstützen einen guten Einstieg in die Programmierung zu finden. Verwendet wird die Programmiersprache Python, die einen klaren und strukturierten Aufbau von Programmen einfordert und somit für das Erlernen von strukturierten Lösungsalgorithmen hervorragend geeignet ist. Zudem erfreut sich diese Programmiersprache in vielen Bildungseinrichtungen und Unternehmen zunehmender Beliebtheit.

Dieses Workbook ist analog zu den Workbooks zu verstehen, die Sie aus dem Sprachunterricht kennen. Sie erlernen hier ja schließlich auch eine neue Sprache, weshalb sich diese Analogie anbietet. Das Buch soll dazu anregen, direkt mitzuarbeiten. Im Buch sind viele Zwischenaufgaben gestellt, die sich auf den Lernstoff davor beziehen und direkt im Buch gelöst werden sollen. Dies dient dazu, dass das Erlernte direkt gefestigt und erprobt ist. Ein direktes Schreiben in das Buch hat den Vorteil, dass Sie interaktiv und direkt mit dem Lernstoff agieren und Ihr Wissen auch immer sofort auf den Prüfstand stellen.

Wichtig ist auch, dass Sie auch sämtliche Codierungen in der Python IDLE, dem Editor für diese Programmiersprache, abtippen. Durch das Abschreiben und durch die Eingabe neuer Problemlösungen werden Sie die Programmierung schnell erlernen und können allmählich in komplexere Projekte einsteigen.

Das Workbook beginnt mit den Basics der Programmierung, steigert sich jedoch in der Komplexität, so dass Sie am

Schluss kleinere GUI-Anwendungen programmieren können. Auch wenn bestimmte Inhalte aufgrund des Umfangs nicht in der Tiefe behandelt werden können (z.B. OOP) wird dennoch die Grundlage gelegt für anspruchsvolle Programmierprojekte, die selbständig erarbeitet werden können.

Didaktischer Hinweis für Lehrkräfte:
In den einzelnen Kapiteln sind sogenannte Zwischenübungen eingebaut. Nutzen Sie diese sowie auch die Übungsaufgaben am Ende der Kapitel um die Klasse immer wieder zu sammeln. Idealerweise präsentiert ein Schüler seine Lösung vor der Klasse. Erfahrungsgemäß driften die Schüler mit der Zeit aufgrund der unterschiedlichen Lerntempi weit auseinander, so dass diese Übungen und Aufgaben als Sammlungsphasen dienen können.

Nun viel Spaß beim Lernen einer neuen (Programmier-)Sprache!

2. überarbeitete Auflage,
Ludwigshafen am Rhein, Februar, 2020

Clemens Kaesler

1. Einführung in Python

Python ist eine moderne Programmiersprache und gehört zu den höheren Programmiersprachen, die aufgrund ihrer klaren Syntax und Struktur immer mehr Anhänger findet. Ein ausgesprochenes Prinzip der Entwickler von Python war es, die Programmlesbarkeit deutlich zu vereinfachen. Eine übersichtliche Struktur wird durch die erzwungene Einrücktiefe (siehe Beispiel XX) gebildet. Es läuft auf den verschiedensten Betriebssystemen (Linux, Mac, Windows u.v.m.) und verfügt über eine sehr umfangreiche Standardbibliothek. Standardbibliothek bedeutet, dass viele Probleme bereits gelöst sind und mit Hilfe einfacher Befehle (sog. Module) in Programme eingebunden werden können (z.B. **Generierung einer Zufallszahl).**

1.1 Die Programmiersprache Python

Wie bei den meisten Programmiersprachen ist ein Programm in Python eine Textdatei, welche im Prinzip zunächst mit einem beliebigen Texteditor bearbeitet werden kann. Die Ausführung des Programms besteht darin, dass Python die Anweisungen in der Textdatei Zeile für Zeile abarbeitet.

Damit Sie direkt eine Vorstellung bekommen, wie so ein Programm aussieht, sei an dieser Stelle ein Beispiel aufgeführt. Sie müssen noch nicht jede Zeile verstehen (die englischen Begriffe sprechen jedoch meist für sich!).

Beispiel:
Im folgenden Programm wird der Nutzer gebeten, drei Zahlen einzugeben. Das Programm addiert die drei Zahlen und gibt sie als Summe aus.
Ist die Summe größer als 100, so gibt das Programm den Text aus: „Die Summe ist größer als 100!“ Ist die Summe kleiner oder gleich 100, so gibt das Programm den Text aus: „Die Summe ist kleiner oder gleich 100“.

```
1  print ("Dieses Programm berechnet Ihnen die
   Summe von drei Zahlen")
2  zahl1 = input ("Geben Sie die erste Zahl ein!")
3  zahl1 = int(zahl1)
4  zahl2 = input("Geben Sie die zweite Zahl ein!")
   zahl2 = int(zahl2)
5  zahl3 = input ("Geben Sie die dritte Zahl ein!")
   zahl3 = int(zahl3)
6
7  summe = zahl1 + zahl2 + zahl3
   print ("Die Summe ist", summe)
   if summe >100:
     print ("Die Summe ist größer als 100!")
   else:
     print ("Die Summe ist kleiner oder gleich 100!")
```

Geben Sie das Programm nun in Python-IDLE ein (vorher Einführung nächstes Kapitel lesen). Es ist wichtig, dass Sie immer alle Beispielprogramme in IDLE abtippen und ausprobieren.

1.2 Die Entwicklungsumgebung von Python - IDLE

Auch wenn einfache Texteditoren für die Erstellung eines Programms genügen, ist es besser, die eigens für die Programmiersprache entwickelten Entwicklungsumgebungen zu nutzen. Entwicklungsumgebungen heißen auf Englisch „Integrated Development Environment“ (IDLE) und sind eigene Programme, die mit zahlreichen Funktionen das Programmieren unterstützen. Die IDLE unterstützt das Programmieren insbesondere dadurch:

- Syntax der Programmiersprache wird farblich hervorgehoben, um den Code übersichtlicher zu machen
- Das Programm kann direkt in der IDLE ausgeführt werden
- Befehle werden oft automatisch vervollständigt
- Fehler werden erkannt und es werden Hinweise zur Fehlerbehebung gegeben

Laden Sie sich nun das komplette Python-Paket auf folgender Web-Site herunter:
https://www.python.org/downloads/
Nehmen Sie bitte den Download “Python 3.7.x“ oder falls es bereits wieder
Weiterentwicklungen gibt die neueste Version von Python 3.x.

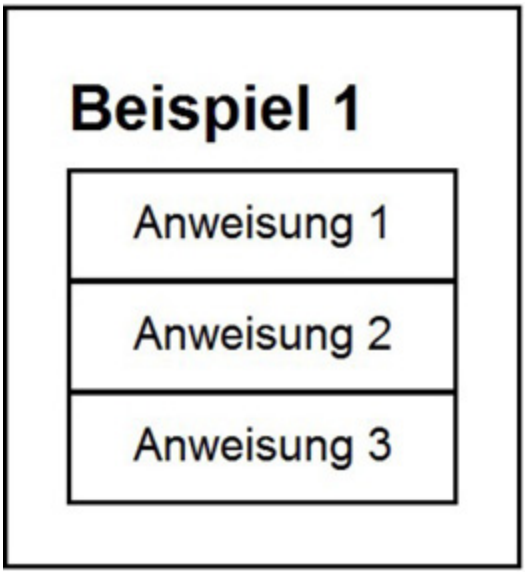
2. Der Algorithmus - Vom Problem zur Lösung

In diesem Buch geht es um Anwendungsprogramme, die helfen sollen eine Problemstellung zu lösen. Es genügt damit nicht, dass man die Befehle einer Computersprache kennt, sondern es erfordert Erfahrung, mathematisches bzw. logisches Denken und eines systematischen Vorgehens, um gegebene Problemstellungen tatsächlich mit Hilfe einer Programmierung zu lösen. Hierzu dienen sog. Algorithmen. Der Name „Algorithmus“ ist arabischer Herkunft und ist eine Abwandlung des Namens “Abu Dscha'far Muhammad ibn Musa al-Chwarizmi“, ein indischer Gelehrter, dessen Buch zur Technik des Rechnens (verfasst 825 n.Chr. in Bagdad) der heutigen Verwendung der arabischen (eigentlich indischen) Ziffern in unserem Kulturraum den Boden bereitete. Die verwendeten Rechenregeln wurden ab dem 16. Jahrhundert in Europa immer populärer und wurden als Algorithmen bezeichnet, ein Begriff, der sich als Allgemeinbegriff für Rechenverfahren durchgesetzt hat.

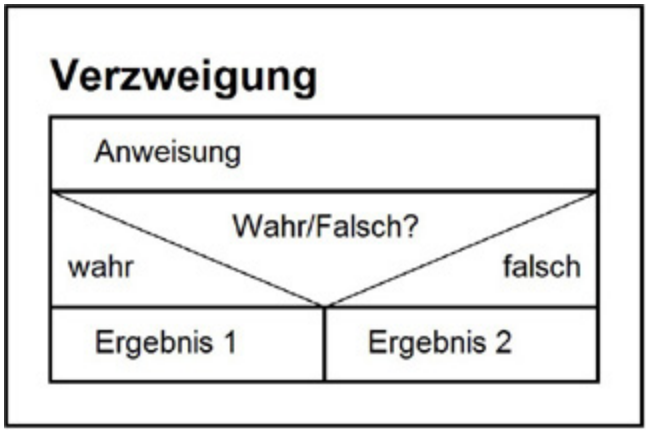
Algorithmen sind strukturierte Verfahren, wie eine Problemstellung gelöst werden kann. Die einzelnen Schritte müssen als Verarbeitungsvorschriften klar formuliert und absolut eindeutig sein. Eine Maschine muss sie sequentiell abarbeiten können.

2.1 Visualisierungshilfen für Algorithmen - Struktogramme

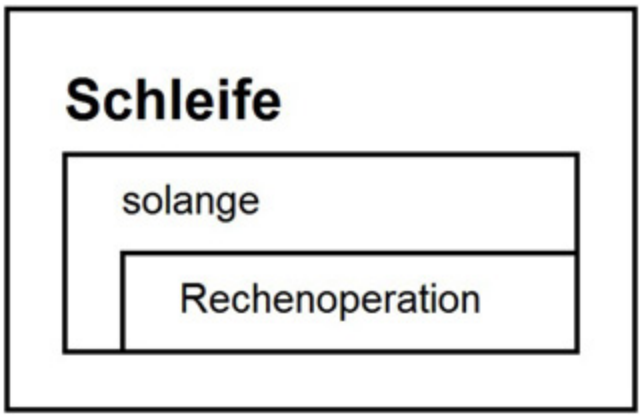
Algorithmen lassen sich sehr gut mittels Struktogrammen darstellen. In diesem Lehrbuch werden die sog. „Nassi-Shneidermann-Diagramme“ verwendet, die in der Programmierung eine lange Tradition haben. Diese Diagramme veranschaulichen die Umsetzung des Algorithmus in Form des Diagrammflusses.
Anweisungen, die aufeinander folgen, werden als Blöcke gestapelt:



Gibt es eine logische Ja/Nein-Situation (wahr/falsch), so handelt es sich um eine Verzweigung:



Beliebt sind in der Programmierung auch Schleifen, hier wird eine Rechenoperation so lange durchlaufen, bis eine bestimmte Schleifenbedingung erfüllt ist.



Wichtig: Sie werden sich fragen, warum Struktogramme erstellt werden müssen. Struktogramme visualisieren den Algorithmus und sind damit eine große Hilfe für die Programmierung, die ja letztlich einen geschriebenen Text darstellt an dem es nicht immer einfach ist, genau die algorithmische Abfolge zu erkennen.

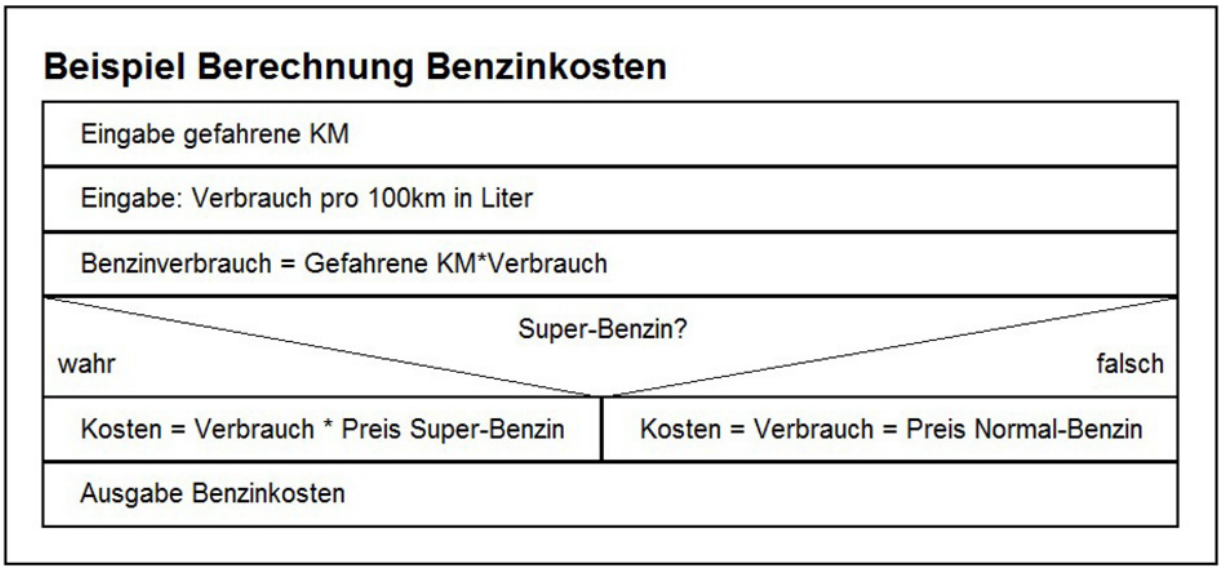
Beispiel:
Für ein Programm zur Berechnung der Kosten des Benzinverbrauchs soll ein Algorithmus erstellt werden. Bei dem Programm soll gewählt werden, ob es SuperPlus-Benzin ist oder Normal-Benzin

Algorithmus:

- ***Eingabe: Gefahrene Kilometer***
- ***Eingabe: Verbrauch pro 100 km (in Liter)***
- ***Benzinverbrauch: Gefahrene Kilometer * Verbrauch pro 100km (in Liter)***
- ***Wahl Super-Benzin oder Normal-Benzin***
- ***Falls Super-Benzin:***

- ***Kosten = Benzinverbrauch * Preis Super-Benzin/Liter***
- ***Falls Normal-Benzin:***
- ***Kosten = Benzinverbrauch * Preis Normal-Benzin/Liter***
- ***Ausgabe Kosten***

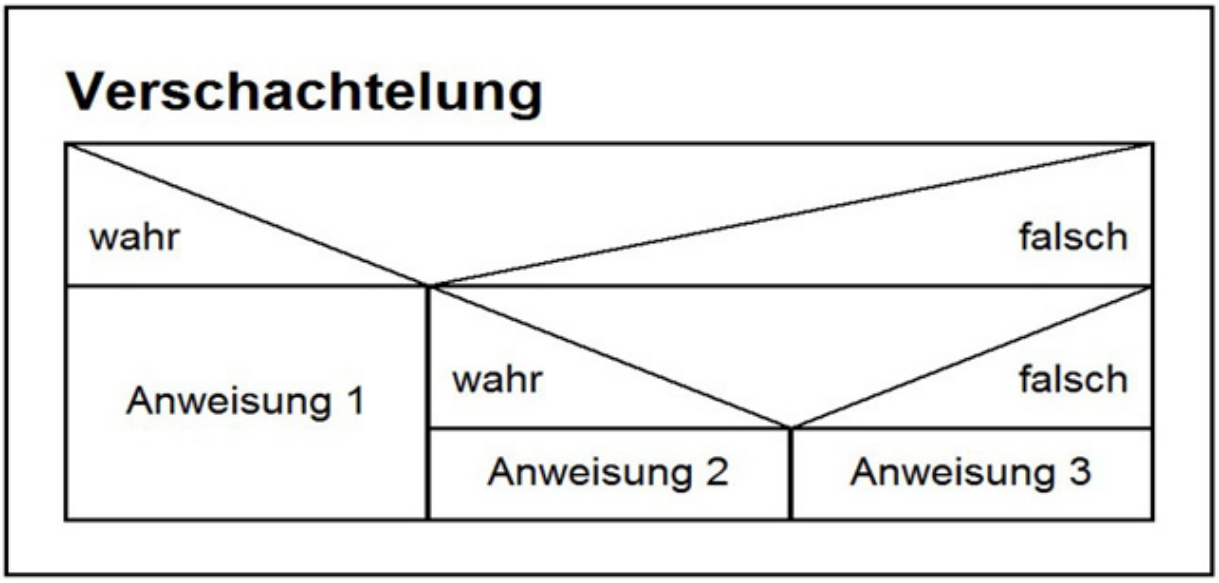
Nun die Darstellung als Struktogramm:



Anmerkung zum Struktogramm: Die Auswahl Super-Benzin oder Normal-Benzin kann durch die Frage nach Super-Benzin gelöst werden, da automatisch die Alternative in diesem Fall „Normal-Benzin“ ist.

Wichtig: Es sind auch Verschachtelungen möglich, so kann eine Verzweigung in eine Verzweigung eingebaut werden.

Beispiel:



Übungsaufgaben zu Kapitel 2

Erstellen Sie Struktogramme zu folgenden Problemstellungen:

Aufgabe 1:
Entwerfen Sie ein Struktogramm für folgende Mathematikaufgabe:
Es sollen drei Zahlenwerte eingegeben werden.
Diese drei Zahlenwerte werden multipliziert. Ist das Ergebnis größer 100, dann soll das Ergebnis durch 5 geteilt werden. Ist das Ergebnis kleiner 100 soll es mit 3 multipliziert werden. Ist das Ergebnis genau gleich 100, soll das Programm den Begriff „Volltreffer“ ausgeben!

Aufgabe 2:
Es soll ein Programm erstellt werden, dass die m^2 Fläche eines Rechtecks berechnet.

Aufgabe 3:
Es soll ein Programm erstellt werden, das erst abfragt, ob ein Rechteck oder ein Kreis berechnet werden soll. Dann

sollen die notwendigen Variablen abgefragt werden und Umfang und Fläche berechnet werden.

Aufgabe 4:
In einem Unternehmen wird unter bestimmten Bedingungen eine Mitarbeiterprämie gezahlt.

I. Hat der Mitarbeiter einen Jahresumsatz von 100.000,-€ oder weniger erreicht erhält er keine Prämie.
II. Für über 100.000,- € und weniger als 200.000,- € erhält er 2 % des Umsatzes.

Für einen Umsatz der größer ist als 200.000,- € erhält er 5 %.

Aufgabe 5:
Es soll ein Struktogramm für ein Programm erstellt werden, das den Zins pro Jahr einer Geldanlage ausgibt. Der Zins wird wieder neu angelegt (Zinseszinseffekt)!
In das Programm sollen der Anlagebetrag, die Anlagedauer sowie der Anlagezins eingegeben werden. Anschließend gibt das Programm für die Dauer die jeweiligen Jahreszinsen aus.

Aufgabe 6:
Erstellen Sie ein Struktogramm für ein Programm, das mitzählt, wie oft eine Zahl zwischen 1 und 10 mit 2 multipliziert werden muss, bis ein Ergebnis über 100.000 erreicht ist.

Aufgabe 7:
Erstellen Sie ein Struktogramm für ein Programm, das vom Nutzer abfragt, ob er Addition oder Multiplikation üben möchte. Dann sollen 5mal die Zahlen 1 bis 10 als Summanden oder Faktoren zufällig gewählt und der gewünschten Rechenoperation zugeführt werden.