Contents

1 Grundlagen

Wie bei jedem Handwerk gibt es auch beim Programmieren gewisse Werkzeuge die man benötigt um eine Applikation zu erstellen. Das erste Werkzeug, dass ein angehender Programmierer verwendet ist wahrscheinlich ein **Text-Editor**. In diesem Programm schreibt man dann wie schon der Name andeutet Text, so-genannten **Code**, in einer bestimmten **Programmiersprache**. Je nach Programmiersprache beschreibt der Code entweder

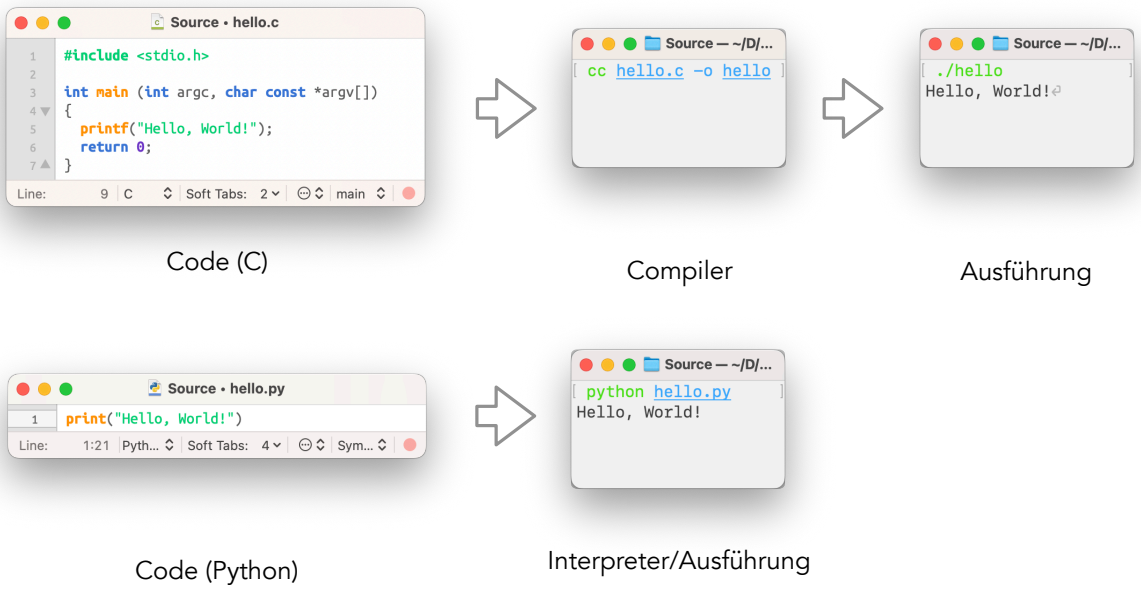
1. welche Schritte ein Computer ausführen soll (imperativ), oder
2. das Problem selber (deklarativ).

> Ein Programmiersprache ist wesentlich einfacher aufgebaut als eine menschliche Sprache. Das mag zu dem Gedanken verleiten, dass es einfacher ist einem Computer als einer Person beizubringen, wie ein bestimmtes Problem aussieht (deklarative Programmierung) oder wie es zu lösen ist (imperative Programmierung). Das ist aber ein Trugschluss. Im Endeffekt benötigt der Computer (durch die Einfachheit der Programmiersprache) eine wesentlich genauere Beschreibung als ein Mensch, führt diese aber dafür (im Normalfall) fehlerfrei und äußerst schnell durch.

Der „fertige“ Code wird entweder

1. von einem **Compiler** in Maschinen-Befehle übersetzt (**kompiliert**) und dann vom Computer ausgeführt, oder
2. direkt vom einem **Interpreter** ausgeführt (**interpretiert**).

Üblicherweise findet, zur Optimierung der Geschwindigkeit eines Programms, auch bei der Interpreter-Variante eine Übersetzung in maschinen-nahen Code statt. Als Programmierer muss man sich dabei aber – im Gegensatz zur Compiler-Variante – üblicherweise keine Gedanken machen.



1.1 Die Werkzeuge

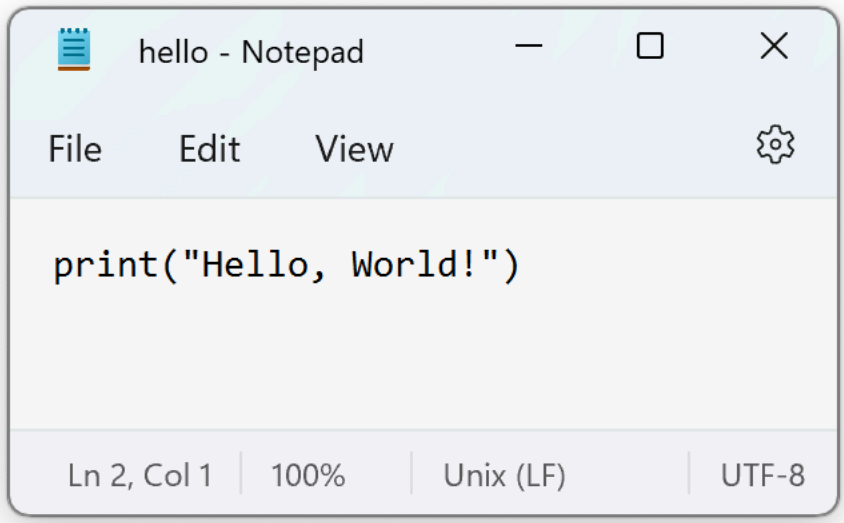
1.1.1 Editor

Wie schon in einführenden Teil beschrieben ist ein Text-Editor oder auch kurz Editor eines der wesentlichen Werkzeuge eines Programmierers. Von einer Textverarbeitung (wie z.B. “Word”, “LibreOffice” oder “Google Docs”) unterscheidet sich ein Texteditor, dadurch, dass man damit wirklich nur Text schreiben kann. Eine Formatierung oder das Einfügen von Bildern ist nicht möglich.

Als essentielles Werkzeug ist ein Texteditor Teil der Standard-Installation vieler Betriebssystemen:

- Linux: gedit, Kate, Vim, Emacs, ed
- macOS: TextEdit, Vim, Emacs, ed
- Windows Notepad

Theoretisch wäre es also möglich z.B. einfach Notepad zu verwenden und darin zukünftige Programmier-Projekte zu verwirklichen.



In der Praxis verwendet man aber üblicherweise einen speziell für die Programmierung angepassten Texteditor (manchmal auch Code-Editor genannt).

1.1.2 Command Line

Während man heutzutage als Anwender meist eine grafische Benutzeroberfläche (**GUI: Graphical User Interface**) verwendet, ist es als Programmierer üblich für viele Aufgaben in einer text-basierten Oberfläche (**CLI: Command Line Interface**) zu erledigen. Diese hat den Vorteil, dass man hier (wesentlich leichter als in einem GUI) Schritte, wiederum mittels Programmierung, automatisieren kann. Das heißt auch bei den Text den man bei einer textbasierten Oberfläche eingibt handelt es sich um Code in einer bestimmten Programmiersprache.

Um eine kleine Einführung in eine text-basierte Oberfläche zu geben schauen wir uns hier an wie man unter Windows Software mittels CLI installiert. Dazu installieren wir als erstes „Windows Terminal“ in der Powershell. Dabei handel es sich um einem Command-Line-Interpreter für die Programmiersprache PowerShell.

1. Das Programm PowerShell öffnen

    1. „Windows-Taste“ drücken
    2. „PowerShell“ eingeben
    3. Mit „Return“ ( ) bestätigen

2. Den folgenden Text (Code) in das PowerShell-Fenster einfügen

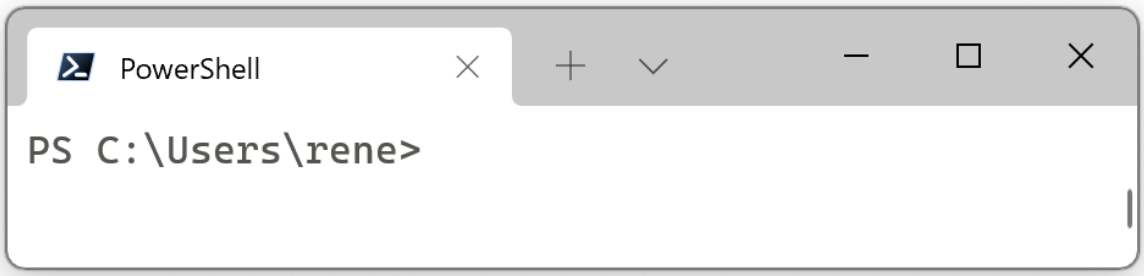
    ```
    winget install --id=Microsoft.WindowsTerminal -e
    ```

    und dann mittels (Return) ausführen.

    - Bei `winget` handelt es sich hierbei um ein Programm (**Befehl**) zum Suchen und Installieren von Software, die in diesem Zusammenhang auch als **Paket** bezeichnet wird.
    - Mit dem **(Sub-)Befehl** (Argument) `install` teilt man `winget` mit, dass es ein Paket installieren soll.
    - Der Name des Pakets wird mit der (Long-)**Option** (`--`) `id` mit dem **Argument** `Microsoft.WindowsTerminal` spezifiziert
    - Die (Short-)**Option** (`-`) `e` wiederum gibt an, dass das das genau (exakt) das Paket `Microsoft.WindowsTerminal` installiert werden soll

3. Zeit zum auf die Schulter klopfen, nachdem du im 2. Schritt (wahrscheinlich) dein erstes PowerShell-„Programm“ geschrieben hast

Um zu sehen ob die Installation von „Windows Terminal“ funktioniert hat suchen wird nach dem Programm Windows Terminal (einem mehr oder weniger modernerer Ersatz des Programms PowerShell) und führen es aus.



Nun wollen wir in einem zweiten Schritt einen Code-Editor installieren. Ein beliebtest Programm ist dabei Visual Studio Code von Microsoft.

1. Da wir nicht genau wissen wie das Paket für Visual Studio heißt suchen wir mit dem Befehl Kommando/Argument `search` nach dem Paket

    ```
    winget search 'Visual Studio Code'
    ```

    Die Anführungszeichen (Single Quotes) sorgen dabei dafür, dass der Text als als Ganzes interpretiert wird und nicht als die 3 Argumente `Visual`, `Studio` und `Code`.

2. Die Ausgabe des obigen Befehls/Code sollte nun die gefunden Pakete/Software anzeigen

```
Name                                   Id                                        Version Source
-------------------------------------------------------------------------------------------
Visual Studio Code                     XP9KHM4BK9FZ7Q                            Unknown msstore
Visual Studio Code - Insiders          XP8LFCZM790F6B                            Unknown msstore
Microsoft Visual Studio Code           Microsoft.VisualStudioCode                1.65.2  winget
Microsoft Visual Studio Code Insiders Microsoft.VisualStudioCode.Insiders 1.66.0  winget
```

3. Wir entscheiden uns für das Paket mit der Id `Microsoft.VisualStudioCode` und installieren dieses mit dem Befehl

   ```
   winget install -e --id Microsoft.VisualStudioCode
   ```

   Am obigen Befehl sehen wir, dass die Reihenfolge von Option (`e` und `id`) normalerweise keine Reihenfolge spielt. Weiters können wir feststellen, dass wir statt des Gleichheitszeichen zur Trennung der Option `id` und des dazugehörigen Arguments (`Microsoft.VisualStudioCode`) auch einfach Leerzeichen verwendet werden können.

Bevor wir nun Visual Studio Code öffnen und unser erstes Python-Programm erstellen sollten wir noch den Python-Interpreter installieren.

1. Eine Suche mittels

   ```
   winget search Python
   ```

   zeigt und, dass die richtige Id des Programms wohl `Python.Python.3` ist

2. Wir installieren Python 3 mittels:

   ```
   winget install Python.Python.3 -e
   ```

   Der obige Befehl zeigt uns, dass man die Option `id` auch weglassen kann und `winget` in diesem Fall annimmt, dass es sich beim Argument `Python.Python.3` um den Namen des Pakets handelt.

Zum Schluss unser ersten Ausflug in die Programmiersprache PowerShell wollen wir noch das Programm „Hello, World!“ schreiben. Dabei handelt es sich um ein ein typische erstes Programm, dass den Text (String) „Hello, World!“ auf dem Bildschirm ausgibt. Im Wikipedia-Artikel zur Programmiersprache PowerShell sehen wir, dass der Befehl `Write-Output` verwendet werden kann um einen Text auf dem Bildschirm auszugeben. Eine typische Implementierung (Realisierung eines Problems in Code) von „Hello, World!“ in PowerShell-Code könnte also z.B. so aussehen:

```
Write-Output "Hello, World!"
```

Die doppelten Anführungszeichen erfüllen dabei den gleichen Zweck wie die einfachen Anführungszeichen beim Befehl `winget search 'Visual Studio Code'`. Der Text `Hello, World!` wird als einzelnes Argument interpretiert und nicht als die zwei Argumente `Hello,` und `World!`.

1.1.3 Interpreter

Nachdem wir „Hello, World!“ schon in PowerShell geschrieben haben, wollen wir das gleiche Programm nun in Python implementieren. Der Code dafür kann z.B. so aussehen

```
print("Hello, World!")
```

Wir sehen, dass der übliche Befehl (Funktion) zum Ausgeben eines Texts (String) in Python `print` heißt. Das Argument des Befehls `"Hello, World!"` wird hier, wie auch in der PowerShell, unter doppelte (oder einfache) Anführungszeichen gesetzt. Damit wird in Python angezeigt, dass es sich um einen Text (String/Zeichenkette) und nicht um einen Befehl (wie z.B. bei der Funktion `print`) handelt.

Wir haben num verschiedene Möglichkeiten unser Programm auszuprobieren. Eine der Möglichkeiten ist die Ausführung direkt im Python-Interpreter.

1. Python-Interpreter öffnen
   - Möglichkeit 1: PowerShell öffnen, den Befehl `python` eingeben und mittels (Return) bestätigen
   - Möglichkeit 2: Mit „Windows-Taste“ die Suche öffnen, den Text “Python” eingeben um nach den Interpreter zu suchen und diesen öffnen
2. Den Code von oben in den Interpreter kopieren und dann mittels (Return) ausführen
3. Der Text „Hello, World!“ wird unter dem Code ausgegeben

```
>>> print("Hello, World!")
Hello, World!
```

Bisher haben wir Code nur direkt im Interpreter ausgeführt ohne diesen vorher in einer Text-Datei zu speichern. Angesichts dessen, dass der bisherige Code nur eine Zeile lang war – dieser Code wird auch oft als “one liner” bezeichnet, war das auch nicht wirklich nötig.

Nun wollen wir den Code unseres „Hello, World!“-Programms mittels Visual Studio Code speichern und ausführen.

1. Visual Studio Code öffnen
2. Eine neue Datei erstellen: Ctrl + N
3. Den Code von „Hello, World!“ einfügen
4. Die Datei mittels Ctrl + N unter dem Namen `hello.py` speichern
5. Mittels „Run“ → „Run Without Debugging“ ausführen

1.1.4 IDE