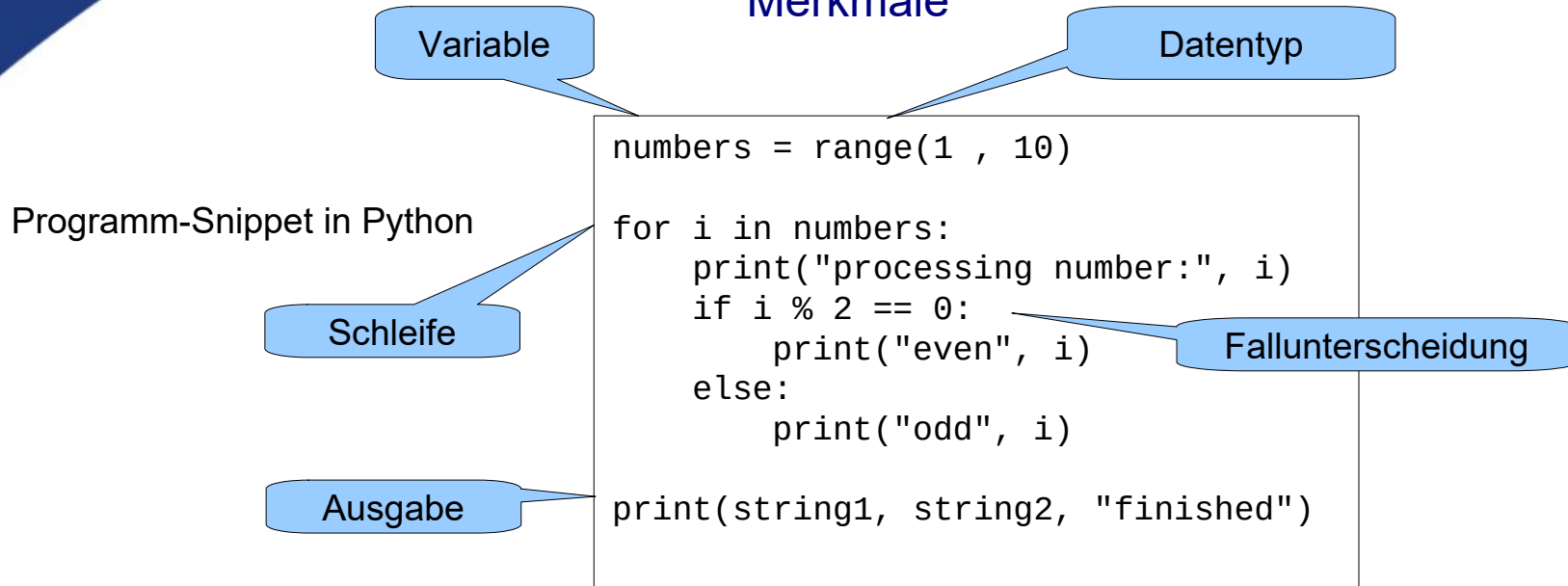




Python - Einführung

Stephan Karrer

Merkmale



- Dynamische Typisierung
 - Nachteil: mehr Typ-Fehler zur Laufzeit
 - Vorteil: weniger Schreibarbeit, Flexibilität
- Objekt-Orientierung (ist erstmal kaum sichtbar)
- Funktionale Sprachelemente
- Modular

Python: Historie und Versionen

Februar 1991: Guido van Rossum veröffentlichte die erste öffentliche Version von Python: die Version 0.9.0

1994: Python 1.0, die erste Version, die als stabil gilt.

2000: Python 2.0

2008: Python 3.0

- ▶ In wesentlichen Teilen nicht kompatibel mit Version 2

- Die Weiterentwicklung von Python liegt heute in den Händen der **Python Software Foundation (PSF)**:

The mission of the Python Software Foundation is to promote, protect, and advance the Python programming language, and to support and facilitate the growth of a diverse and international community of Python programmers.

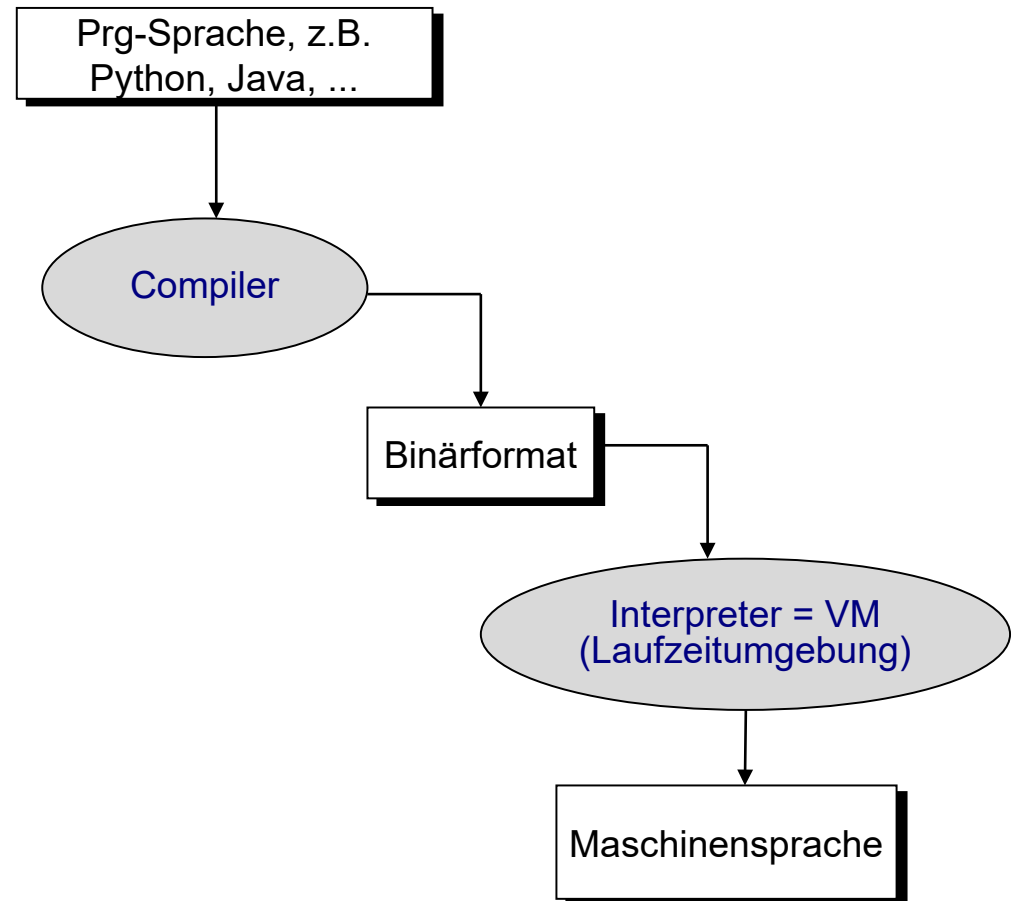
<https://www.python.org/psf/mission/>

Wesentliche Einsatzgebiete

- Web Development
 - Unterstützung durch Frameworks wie Django, Flask, FastAPI, ...
- Datenanalyse und Mathematik
 - Unterstützung durch Bibliotheken wie Pandas, NumPy, ...
- Künstliche Intelligenz (AI) und maschinelles Lernen
 - Unterstützung durch Bibliotheken wie TensorFlow, PyTorch, ...
- Automatisierung und Skript-Programmierung
 - selbstverständlich auch Online verfügbar

Trennung von Compiler und Interpreter

- Aktuell häufiger Ansatz bzgl. Umsetzung auf Binärformat des Rechners
- Plattform unabhängig, sofern der Interpreter auf mehreren Plattformen (Betriebssystemen) verfügbar ist.
- Das Zielsystem mit dem Interpreter ist oft nicht identisch mit dem Entwicklungssystem.
 - ▶ Entscheidend ist somit die Sprach-Version des Zielsystems !

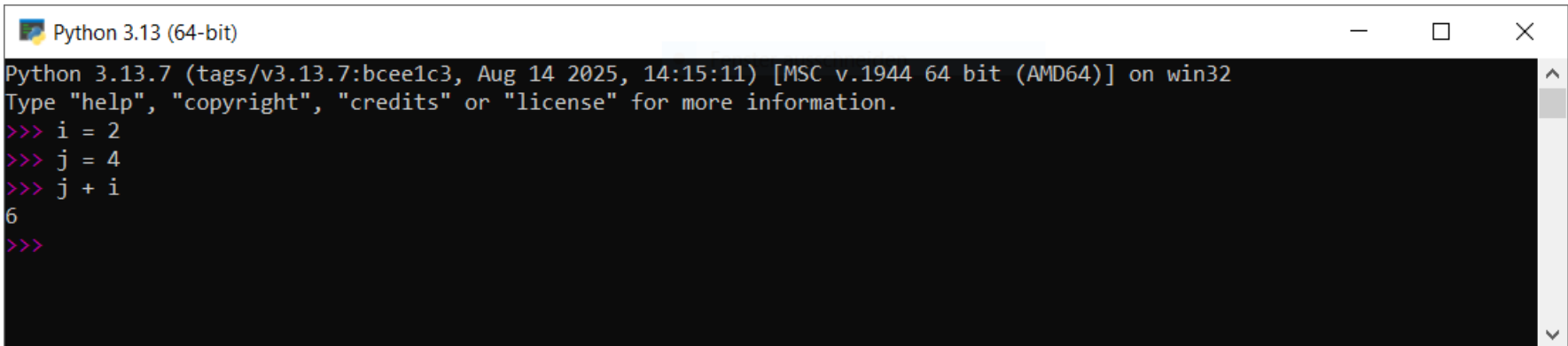


Verfügbarkeit

- Python ist frei erhältlich und Open Source
- Distributionen sind für alle relevanten Betriebssysteme verfügbar:
 - Linux
 - Windows
 - Mac
 - ...
- Distribution enthält auch die Dokumentation
 - selbstverständlich auch Online verfügbar
- Distribution enthält auch ein einfaches Werkzeug zum Ausführen des Codes, die Python-Shell (Read - Evaluate - Print – Loop = REPL)
- Viele Online-Tutorials bieten eine entsprechende Ausführungsumgebung an (z.B. <https://www.w3schools.com/python/>)

REPL-Shell

- REPL (Read - Evaluate - Print - Loop) : Wir tippen unseren Code direkt in das Tool (Kommando-Interpreter), der wird interpretiert und das Ergebnis ausgegeben.
- Beispiel: Python-Shell

A screenshot of a Python 3.13 (64-bit) REPL window. The window title is "Python 3.13 (64-bit)". The terminal text shows the Python version and build information: "Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (AMD64)] on win32". It then prompts the user to type "help", "copyright", "credits" or "license" for more information. The user has entered three lines of code: ">>> i = 2", ">>> j = 4", and ">>> j + i". The output of the last line is "6". The prompt ">>>" is shown again on the next line.

```
Python 3.13 (64-bit)
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> i = 2
>>> j = 4
>>> j + i
6
>>>
```

Editoren und IDEs

- Selbst einfache Editoren bieten mittels Plugins elementare Unterstützung für Syntax-Hervorhebungen etc. an.
 - Notepad++
 - UltraEdit
 - ...

- IDEs (Integrated Development Environment) integrieren zusätzlich:
 - Code Assist
 - Hilfefunktion und Dokumentation
 - Debugger
 - Testwerkzeuge
 - Code-Formatting
 - Anbindung an Versionsverwaltung

Beispiel: Python-IDE PyCharm (JetBrains)

