2 Hello World Python

2.1 Kurze Geschichte der Programmiersprachen

1. Generation: Programmierung im Maschinencode.
2. Generation: Assemblersprachen und erste Entwicklung höherer Programmiersprachen wie FORTRAN, ALGOL und COBOL.
3. Generation: Entwicklung von Betriebssytemen mit Dialogbetrieb, Datenbanken, Methoden der strukturierten Programmierung, Programmiersprache Pascal
4. Generation: verteilte Systeme, Rechnernetze, Kommunikationsfähigkeit, gute Arbeits- und Programmierumgebungen;
5. Generation: Wissensverarbeitung, automatisches Schlussfolgern, deduktive Datenbanken, Expertensysteme, PROLOG.

Heute gibt es eine unübersehbare Fülle von Programmiersprachen, für die verschiedensten Anwendungsgebiete. Nur wenige Programmiersprachen haben jedoch als universell verwendbare Sprachen einen hohen Verbreitungsgrad in der industriellen Praxis erreicht.

Programmiersprachen werden üblicherweise Sprachen in folgende vier großen Klassen zu unterteilen:

1. Imperative Sprachen: Beispiele sind Basic, Algol, Pascal, FORTRAN, C, Modula
2. Funktionale Sprachen: Beispiele sind LISP, ML, Haskell
3. Logische Sprachen: Hier ist PROLOG das bekannteste Beispiel
4. Objektorientierte Sprachen: Smalltalk, Simula, C++ und Java gehören in diese Klasse.

Und Python? Dazu später.

Die Zuordnung einer Sprache zu einer dieser Klassen ist nicht immer eindeutig möglich. So haben beispielsweise die objektorientierten Sprachen C++ und Java einen imperativen Sprachkern und könnten so ebenso der Klasse der imperativen Sprachen zugeordnet werden. Die vier Sprachklassen kennzeichnen daher eher einen

gewissen Programmier-Stil und weniger die Ausdrucksfähigkeit einer bestimmten Sprache.

2.2 Kurze Geschiche von Python

- Python wurde von dem niederländischen Informatiker Guido van Rossum entwickelt
- Die Entwicklung begann Ende der 1980er Jahre als Hobbyprojekt

- Der Name sollte kurz, einprägsam und etwas unkonventionell sein, um die Philosophie von Python widerzuspiegeln. van Rossum war ein großer Fan der britischen Comedy-Show "Monty Python's Flying Circus", daher der Name "Python".
- Meilensteine:
  - **Python 0.9.0 (1991)** Die erste Version, Python 0.9.0, wurde im Februar 1991 veröffentlicht.
  - **Python 1.0 (1994):** Die erste offizielle Version von Python wurde veröffentlicht und legte den Grundstein für die weitere Entwicklung der Sprache.
  - **Python 2.0 (2000):** List Comprehensions wurden eingeführt, um das Erstellen von Listen auf eine elegante und kompakte Weise zu ermöglichen.
  - **Gründung der Python Software Foundation (PSF) (2001):** Die PSF wurde gegründet, um die Weiterentwicklung und Verbreitung von Python zu unterstützen. Sie spielt eine wichtige Rolle in der Förderung und Verwaltung der Python-Community.
  - **Python 3.x (2008):** Python 3.0 wurde veröffentlicht und führte tiefgreifende Veränderungen in der Sprache ein, um einige Probleme und Unklarheiten in Python 2.x zu beheben. Diese Versionen existierten eine Zeit lang nebeneinander, was zu Diskussionen in der Community führte.

2.3 Eigenschaften von Python

Allgemein

- einfach erlernbar und lesbar
- Interpreter-basiert
- plattformunabhängig
- Open Source

Sprachparadigmen: - imperativ: Ein Programm besteht aus einer Folge von Anweisungen (Befehlen), die den Zustand des Programms Schritt für Schritt verändern. - prozedural: Ein Programm wird in Prozeduren (auch Funktionen oder Routinen genannt) unterteiltS. Sie ist eine Unterform der imperativen Programmierung. Dabei wird der Code in wiederverwendbare, logisch zusammengehörige Blöcke aufgeteilt, die bestimmte Aufgaben übernehmen. - objektorientiert: Python unterstützt Klassen, Objekte, Vererbung usw. - funktional: Ein Programme kann durch das Anwenden und Kombinieren von Funktionen aufgebaut werden. Python ist aber nicht rein funktional. - dynamisch typisiert: Keine Typdeklarationen notwendig. Der Typ eines Variablen wird beim Verarbeiten des Programms bestimmt.

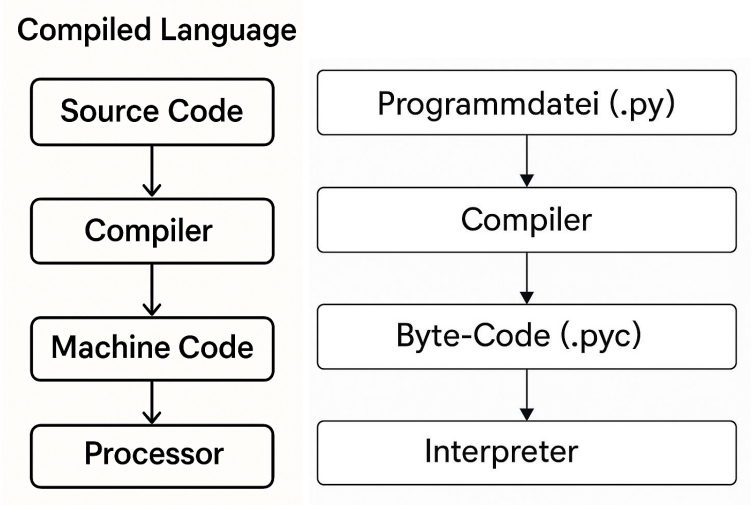
⇒ Python ist eine Mehrparadigmen-Programmiersprache, die verschiedene Programmieransätze unterstützt.

2.4 Der Python-Interpreter

Compilierte Sprachen vs interpretierte Sprache

Python ist eine interpretierte Programmiersprache. Im Gegensatz zu compilierten Sprachen, bei denen der Quellcode vom Compiler in Maschinencode übersetzt wird, der direkt vom Prozessor ausgeführt werden kann, wird bei Python com Compiler ein sogenannter Bytecode erzeugt. Dieser Bytecode wird anschließend vom Python-Interpreter zeilenweise ausgeführt. Das ermöglicht eine schnelle Entwicklung, einfaches Debugging und hohe Flexibilität während des Programmierens. Insbesondere kann man auf diese Weise plattformunabhängig programmieren, da der Python-Interpreter (PVM = Python-Virtual-Machine) auf verschiedenen Betriebssystemen verfügbar ist.

Ablauf compilierte Sprachen vs interpretierte Sprachen:



direkte Kommunikation mit dem Python Interpreter

Die direkte Kommunikation mit Python bezeichnet die unmittelbare Interaktion mit der Sprache. Dies geschieht beispielsweise über den Python-Interpreter oder interaktive Entwicklungsumgebungen wie Jupyter Notebooks. In solchen Umgebungen kann Python-Code direkt eingegeben und ausgeführt werden, wodurch sofort Ergebnisse sichtbar werden. Das ist besonders hilfreich zum Testen von Code, zum Experimentieren mit Funktionen und zum Erlernen der Sprache.

Aufruf des Python-Interpreters:

```
python3
```

2.5 Ganze Zahlen - `int`

Die ganzen Zahlen haben in Python den Datentyp int (=interger = ganzzahlig)

```
2
```

2

```
2+3
```

5

```
5-13
```

-8

```
12*324
```

3888

```
2*3.14
```

6.28

2.6 Realisierung der Ganzzahlwerte im Speicher - Zwei-Komplementdarstellung

Zur rechnerinternen Darstellung ganzer Zahlen im 1Bit-Speicher wird häufig die sogenannte **2-Komplementdarstellung** verwendet.

> **Definition**
>
> Ein **Bit** (=binary digit)ist die kleinste mögliche Informationseinheit. Mögliche Werte für ein Bit sind 0 oder 1. Dies entspricht dem Computerzustand Strom an oder Strom aus. In der logischen Interpretation wäre falsch = 0 und wahr = -1.

> **Definition**
>
> Der Wert einer Zahl $a = a_n a_{n-1} ... a_0$ mit n Bits in **2-Komplementdarstellung** ist definiert als:
>
> $$\text{Wert}(a) = -a_n \cdot 2^n + \sum_{i=0}^{n--1} a_i \cdot 2^i$$

Beispiel: 3-Bit 2-Komplementdarstellung $(n = 3)$

$\text{Wert}(101) = -0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 = 2$

Wert in 3-Bit 2-Komplementdarstellung	Wert in Vorzeichen-/Betragsdarstellung
000	0
001	1
010	2
011	3
100	-4
101	-3
110	-2
111	-1

Vorteile

Vorteilhaft bei dieser Art der Darstellung ist: - Es gibt **nur eine Darstellung der Zahl 0**. - Die **Subtraktion zweier Zahlen** lässt sich einfach auf eine **Addition** zurückführen.

```
8/4
```

```
2.0
```

2.7 Kommazahlen - `float`

Es gibt auch Kommawerte (z.B. 2.0). Diese Zahlen habe den Datentyp `float` Die Datentypen sind nicht begrenzt.

```
5/4
```

```
1.25
```

```
7/3
```

```
2.3333333333333335
```

```
1.2 +3.34
```

```
4.54
```

```
-3.45 - 7.4
```

```
-10.850000000000001
```

```
4.6/4.3
```

```
1.069767441860465
```

```
-2.3*-3.6
```

```
8.28
```

2.8 Realisierung von Fließkommawerten im Speicher - Gleitkommadarstellung

Definition

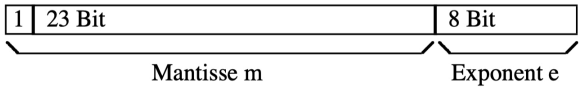
Eine reelle (eigentlich: rationale) Zahl wird im 1Bit-Speicher dargestellt durch die sogenannte **Gleitkommadarstellung**:

$$z = m \cdot b^e$$

mit - m: Mantisse - b: Basis - e: Exponent (zur Basis b)
In der Praxis wählt man für b meistens eine der Zahlen 2, 10, 16 und stellt dementsprechend die Zahl m als Ziffernfolge im Dual-, Dezimal-, oder Hexadezimalsystem dar.

Der IEEE 754-Standard für eine Zahl z in Gleitkommadarstellung legt die Größe der einzelnen Komponenten dieser Darstellung fest und hat folgendes Format (das erste Bit ist hierbei das Vorzeichen der Mantisse m):

32-Bit-IEEE 754-Gleitkommazahlen:



INFO

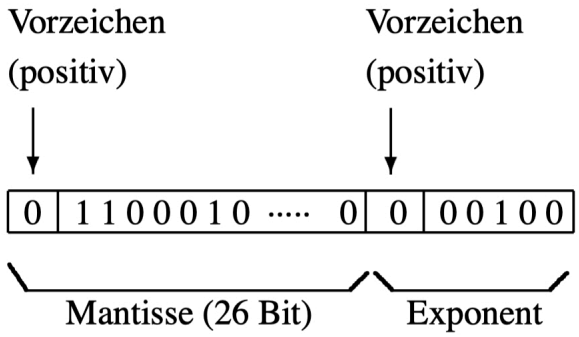
IEEE ist die Abkürzung für "**I**nstitute of **E**lectrical and **E**lectronics **Engineers**", eine Organisation, die Standards für verschiedene Technologien entwickelt und pflegt.

Bei der Darstellung der Zahlen im Rechner reserviert man einen festen Teil einer Folge von Speicherzellen (1-Bit Speichern) für die Mantisse und den Rest für den Exponenten (den Wert von b braucht man nicht zu speichern, da alle Rechnungen mit demselben Wert von b durchgeführt werden).

Beispiel:

Sei $b = 2$ und hat man insgesamt 32 Bit zur Verfügung

26 Bit werden für die Mantisse (incl. Vorzeichen) und 6 Bit für den Exponenten (incl. Vorzeichen) 3 Bit verwendet. Das erste Bit kennzeichnet jeweils das Vorzeichen; dabei kann man insbesondere für den Exponenten auch die 2-Komplementdarstellung benutzen. Die rechnerinterne Darstellung der Zahl 12,25 würde dann wie folgt aussehen:



Thema zum Weiterdenken: Die Gleitkommadarstellung ist nicht eindeutig. Die Zahle 12,26 läst sich auf zum Beispiel diese Arten darstellen.

- $1100,01 \cdot 2^0$
- $110,001 \cdot 2^1$
- $11,0001 \cdot 2^2$
- usw.

Daher wird in der Informatik die **normierte Gleitkommadarstellung** verwendet, bei der die Mantisse immer mit einer 1 beginnt (außer bei der Zahl 0). Dadurch wird die Darstellung eindeutig und es gibt keine Mehrdeutigkeiten mehr. Informiere dich über die normierte Gleitkommadarstellung.

> ! Folgerung
>
> 1. Es sind auf diese Weise nur endlich viele reelle Zahlen darstellbar.
> 2. Es gibt jeweils eine kleinste und eine größte darstellbare Zahl.
> 3. Es gibt ein endliches Intevall um den Nullpunkt, in dem keine darstellbare Zahl liegt.
> 4. Ein analoges Intervall gibt es selbstverständlich um andere Zahlen.

2.9 Skripte in Python

Neben der Möglihckeit im Terminal oder mit Jupyter direkt mit dem Python interpreter zu kummunizieren, gibt es die Möglichkeit eine Quellcodedatei zu erstellen, ein Skript und diese dann insgesamt zum Python-Compiler zu übergeben.

Datei: hello_world.py

```
1    print("Hello World")
```

Comililierung und Ausführung des Skripts:

```
> python3 hello.world.py
```