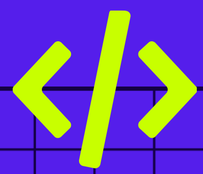
stackfuel

Python und andere Sprachen



Python

Python […] ist eine **universell nutzbare**, üblicherweise **interpretierte**, **höhere** Programmiersprache. Sie hat den Anspruch, einen gut lesbaren, knappen Programmierstil zu fördern. […]

Python unterstützt mehrere Programmierparadigmen, z. B. die **objektorientierte**, […] und die **funktionale** Programmierung. Ferner bietet es eine **dynamische Typisierung**. […]

- Wikipedia

Python

- höhere Programmiersprache
- Interpretierte
- Programmierparadigmen
  - Objektorientierte
  - funktionale Programmierung
- dynamische Typisierung
- universell nutzbare

stackfuel

Höhere Programmiersprachen

Bytecode

Assemblersprachen

ABSTRAKTION

CPU

```
61 A0 7E C3
67 0F 3C FA
8D 8D 47 0F
EE 9F 7D 58
E3 23 A5 B0
A1 EC 21 BE
A3 32 50 22
6C A2 43 99
7B 47 42 9A
70 7C FD 5F
45 F0 22 CF
D9 73 33 CB
C2 39 BE AB
E2 B1 8B 21
94 D7 23 AC
81 8A 01 84
CA E7 D3 25
6F BA 2B 19
FB 2C A2 30
69 75 05 07
F2 FA AC 45
00 D7 5C 3B
5E C2 5A 9D
25 55 C0 57
B6 CE 6B 98
56 BE 37 D8
2B ED FF 82
03 A0 11 D5
BB C0 2F 5C
```

```
section .data
    num1 db 56
    num2 db 98

section .bss
    result resb 1

section .text
    global _start

_start:
    mov al, [num1]
    mov bl, [num2]

euclidean_algorithm:
    cmp bl, 0
    je done
    xor ah, ah
    div bl
    mov al, bl
    mov bl, ah
    jmp euclidean_algorithm

done:
    mov [result], al
    mov eax, 60
    xor edi, edi
    syscall
```

nasm the netwide assembler

julia

JS

Java

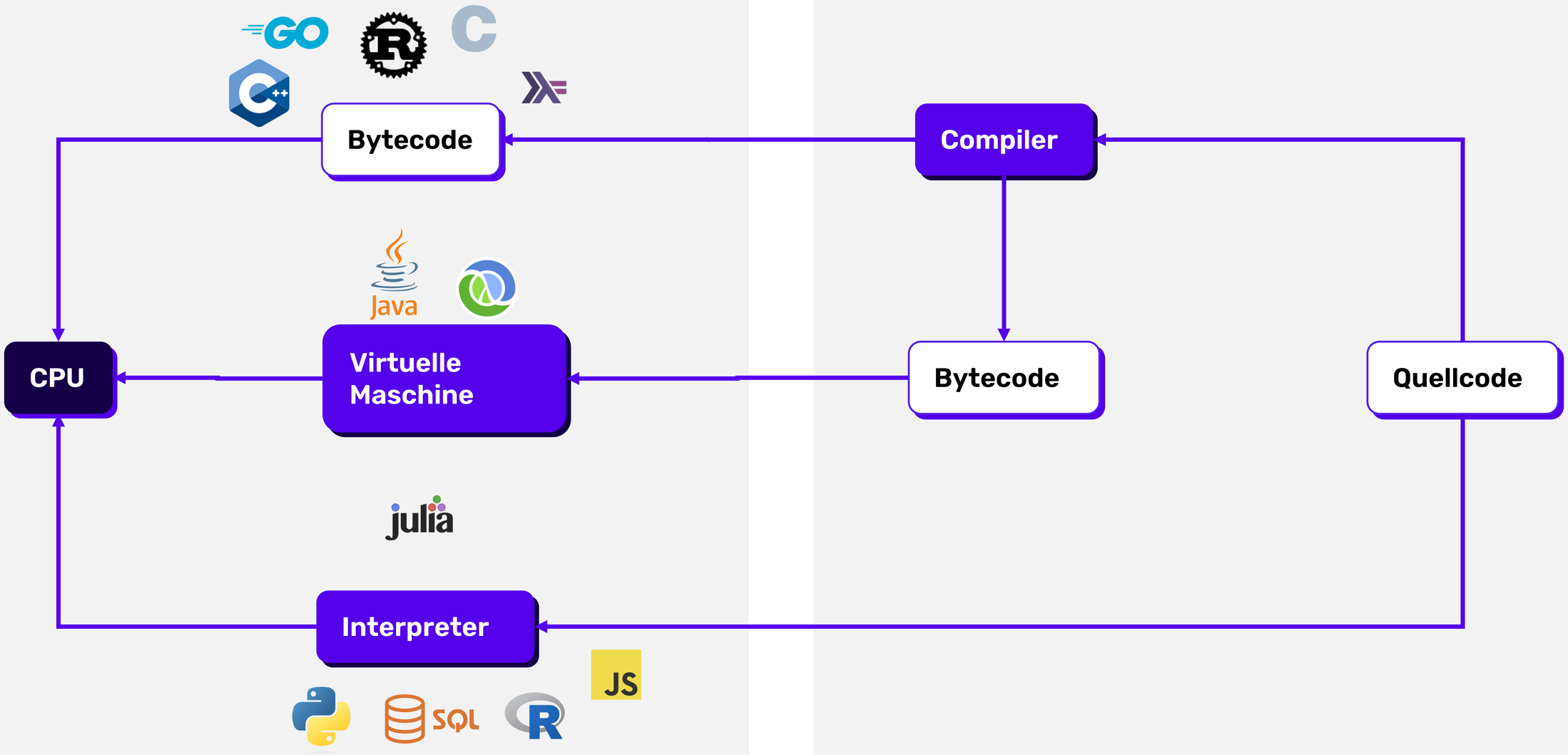
SQL

NIEDERE PROGRAMMIERSPRACHEN

HÖHERE PROGRAMMIERSPRACHEN

stackfuel

Interpretiert / Compiliert



Objektorientiert und Funktional

Imperative Programmierung

- Schritt für Schritt Anweisungen
- Man formuliert, **wie** etwas gelöst werden soll

Objektorientierte Programmierung

- Kapselung von **Datenstrukturen** und **Funktionalität** in Objekte
- Vererbung und Komposition für Komplexe Strukturen

Deklarative Programmierung

- Man beschreibt ein Problem
- Man formuliert **was** gelöst werden soll

Funktionale Programmierung

- **Funktionen** als zentraler Baustein
- **Rekursion** statt Schleifen
- Baut auf den Ideen des Lambda Kalküls auf

```python
def sum_imperative(n):
    total = 0
    for i in range(1, n + 1):
        total += i
    return total
```

```python
from functools import reduce

def sum_functional(n):
    return reduce(lambda x, y: x + y, range(1, n + 1))
```

Dynamisch Typisiert

Statisch Typisiert


```java
// Statisch typisierte Sprache - Java

public class Main {
    public static void main(String[] args) {
        // Deklaration und Initialisierung eines Integers
        int variable = 10;
        System.out.println(variable);  // Ausgabe: 10

        // Versuch, einen String zur selben Variablen zuzuweisen
        // (wird Kompilierungsfehler verursachen)
        variable = "Hallo, Welt!";
        // Fehler: Inkompatible Typen: String kann nicht in int konvertiert werden

        // Deklaration und Initialisierung eines Strings
        String text = "Hallo, Welt!";
        System.out.println(text);  // Ausgabe: Hallo, Welt!
    }
}
```

Dynamisch Typisiert

```python
# Dynamisch typisierte Sprache - Python

# Zuweisung eines Integers
variable = 10
print(variable)  # Ausgabe: 10

# Zuweisung eines Strings zur selben Variable
variable = "Hallo, Welt!"
print(variable)  # Ausgabe: Hallo, Welt!

# Zuweisung einer Liste zur selben Variable
variable = [1, 2, 3]
print(variable)  # Ausgabe: [1, 2, 3]
```

Universell einsetzbar

Universelle











Domänen spezifische



stackfuel

Danke