



C-EXTENSIONS: WIE SIE PYTHON-CODE AUF DIE ÜBERHOLSPUR BRINGEN

DR. MAIK WEBER

11.03.2024

PROBELEHRVERANSTALTUNG AN DER HOCHSCHULE TRIER

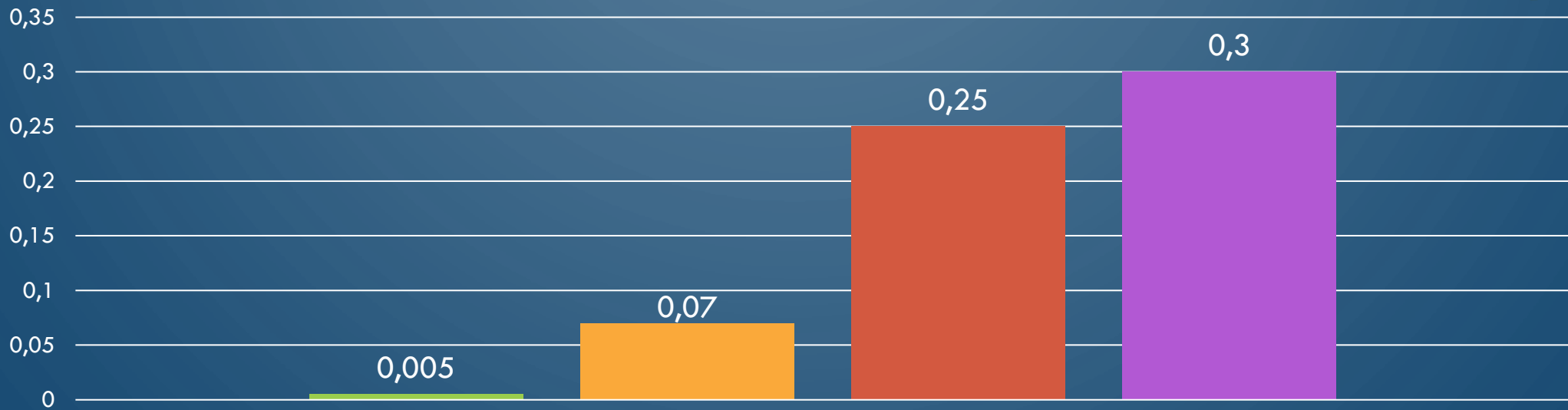
PROGRAMMIERSPRACHEN IM WETTSTREIT

QUELLE: [HTTPS://WWW.THOMASCHRISTLIEB.DE/](https://www.thomaschristlieb.de/) (2018)

Aufgabe: Approximation von π durch die Leibniz-Reihe (10^6 Iterationen)



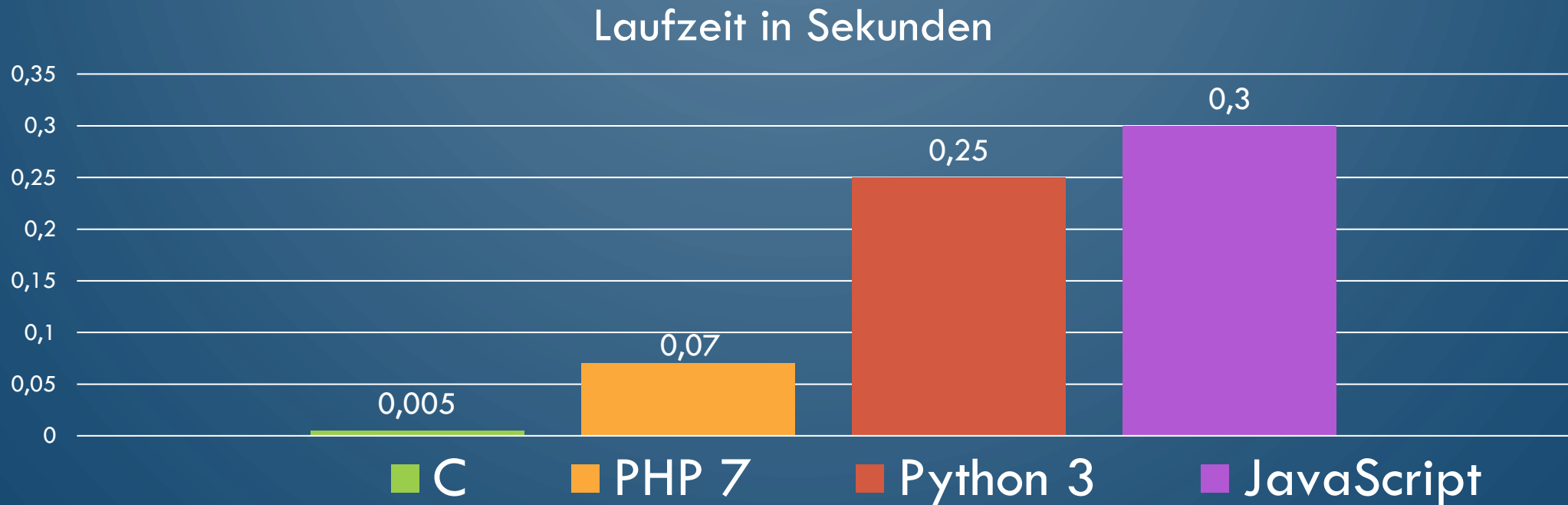
Laufzeit in Sekunden



PROGRAMMIERSPRACHEN IM WETTSTREIT

QUELLE: [HTTPS://WWW.THOMASCHRISTLIEB.DE/](https://www.thomaschristlieb.de/) (2018)

Aufgabe: Approximation von π durch die Leibniz-Reihe (10^6 Iterationen)



C

HelloWorld.c

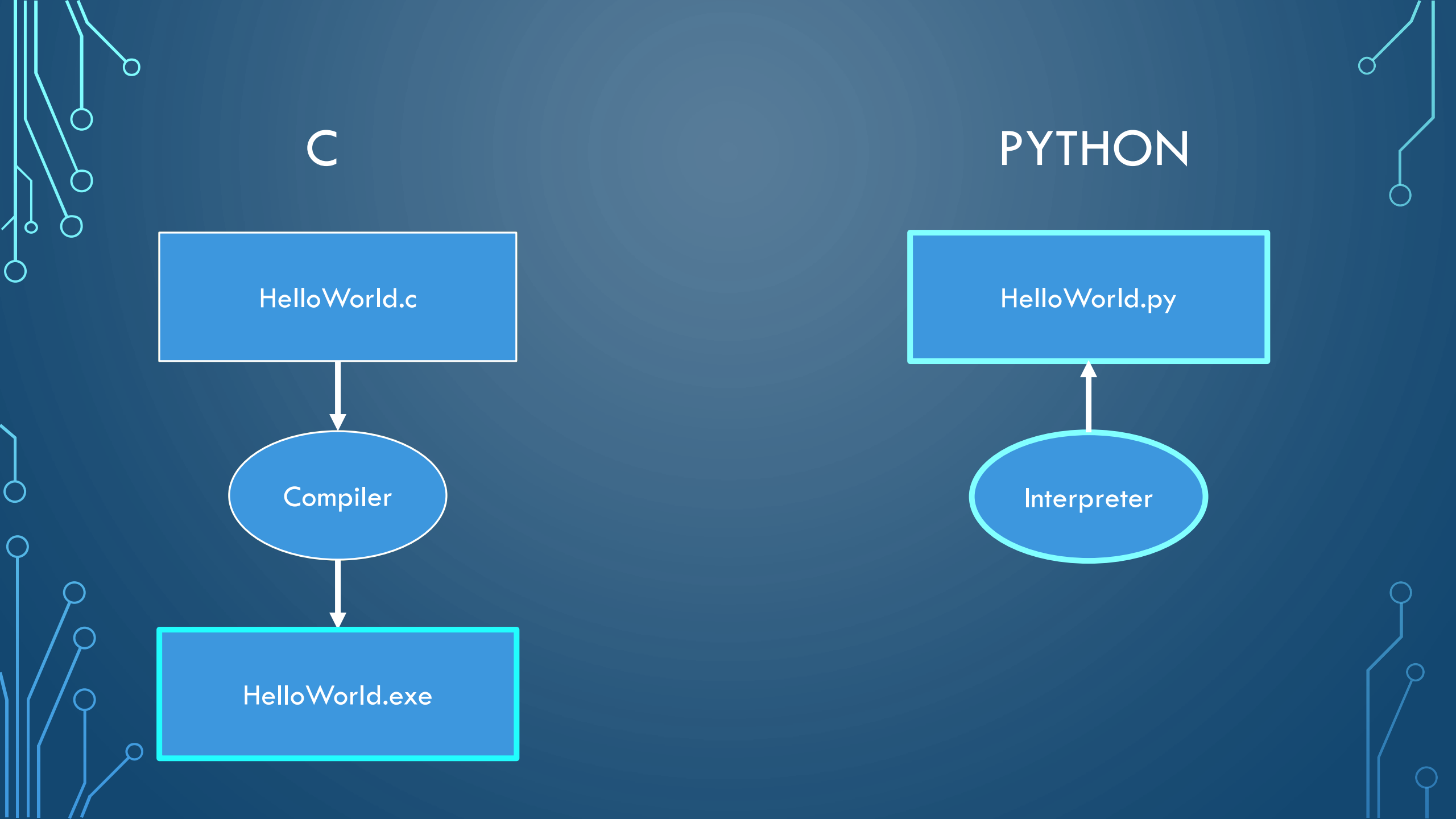
Compiler

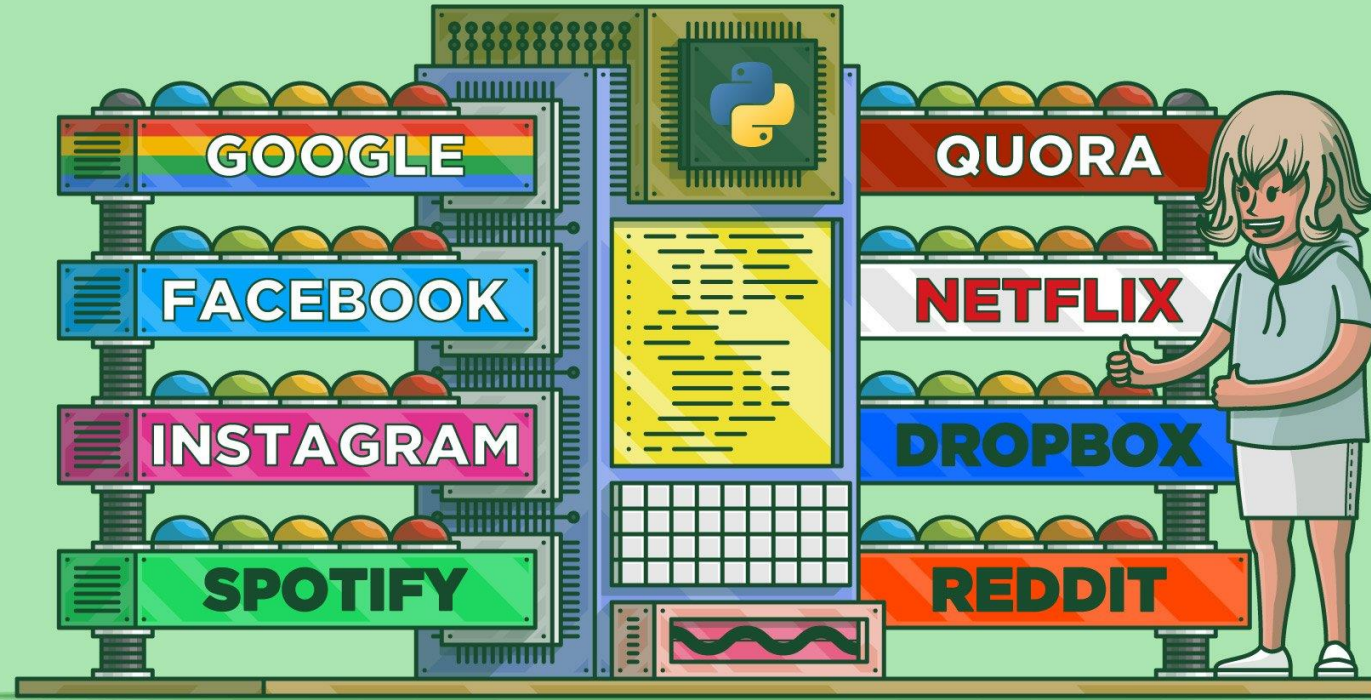
HelloWorld.exe

PYTHON

HelloWorld.py

Interpreter





<https://realpython.com/world-class-companies-using-python/>

Real Python



*“We initially chose to use Python because of its reputation for **simplicity and practicality**, which aligns well with our philosophy of ‘do the simple thing first.’”*

Min Ni, Instagram



C-EXTENSIONS: WIE SIE PYTHON-CODE AUF DIE ÜBERHOLSPUR BRINGEN



- Motivation ✓
 - Praktisches Beispiel: Bildbearbeitung
 - Pure Python
 - C-Code als Shared Library
 - C-Code als Python-Modul (C-Extension)
 - C-Extensions in der Praxis
 - Quiz
- 
- 

5 SCHRITTE ZUR C-EXTENSION

1. **API einbinden:** `#include<Python.h>`

2. **Wrapper schreiben:**

Input: Python-Objects zu C-Typen konvertieren

`PyArg_ParseTuple(...)`

Funktionalität ausführen: C-Code

Output: Ergebnis(se) zu Python-Object konvertieren

`Py_BuildValue(...)`

3. **„Buchhaltung“:** Method Definitions und Module Definition für Interpreter anlegen

4. **Module Initialization Function anlegen**

5. **Packaging:** Modul bauen und installieren bzw. verteilen (mehrere Möglichkeiten)

direkte Installation: `setup.py → pip install ./`

(zum Verteilen: `python -m build → Wheel-Datei`)

Link zur Dokumentation: <https://docs.python.org/3/extending/extending.html>

C-EXTENSIONS IN DER PRAXIS

Bekannte Python-Module, die in C oder C++ implementiert wurden:

- Numpy (aufbauend: SciPy)
- Pandas
- TensorFlow und PyTorch
- Matplotlib
- PyCrypto
- ...

Praxistipps:

1. Verwenden Sie stets bereits verfügbare, optimierte Module in Ihren Programmen!
2. Falls nicht verfügbar: Identifizieren Sie langsame Routinen und lagern Sie ggf. einzelne Funktionen als C-Code in eine Bibliothek aus.
3. In größeren Projekten und bei häufiger Wiederverwendung: Erstellen Sie eine C-Extension als eigenständiges Python-Modul.

DOKUMENTE UND ÜBUNGSAUFGABE

Auf GitHub: <https://github.com/Maik-Weber/C-Extensions-Trier>

- Slides und alle Files zur Vorlesung.
- **Übungsaufgabe** „Caesar Verschlüsselung“ im Ordner Uebung-Caesar. Das Jupyter-Notebook `caesar-cipher.ipynb` führt Sie durch alle Aufgaben.

Ich kam, sah und siegte!

Lfk ndp, vdk xqg vlhjwh!

The background is a solid dark blue. In the corners, there are abstract, light blue line art designs that resemble circuit traces or neural network connections. These designs consist of straight lines of varying lengths and angles, some ending in small open circles. The designs are located in the top-left, top-right, bottom-left, and bottom-right corners, framing the central text.

APPENDIX

FÜR C TYPES: SHARED LIBRARIES UNTER WINDOWS, LINUX, MAC

Dateiendungen:

Windows: .dll (Dynamic Link Library)

Linux: .so (Shared Object)

Mac: .dylib (Dynamic Library)

Wichtige Compiler-Flags für GCC (GNU Compiler Collection):

Windows: `gcc -shared -o mylib.dll mylib.c`

Linux: `gcc -shared -o libmylib.so mylib.c`

Mac: `gcc -dynamiclib -o libmylib.dylib mylib.c`

BENCHMARKS

- High-Performance Computing (Lösen von Differentialgleichungen):
C gewinnt mit Faktor 2 – 5.

Langtangen, H.P., Cai, X. (2008). On the Efficiency of Python for High-Performance Computing: A Case Study Involving Stencil Updates for Partial Differential Equations. In: Bock, H.G., Kostina, E., Phu, H.X., Rannacher, R. (eds) Modeling, Simulation and Optimization of Complex Processes. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-79409-7_23

- Embedded Systems (z.B. FFT auf Microcontrollern):
C gewinnt mit Faktor 200 – 500.

Plaуска, I.; Liutkevicius, A.; Janaviciute, A. (2023). Performance Evaluation of C/C++, MicroPython, Rust and TinyGo Programming Languages on ESP32 Microcontroller. Electronics 2023, 12, 143. <https://doi.org/10.3390/electronics12010143>

- GPU-Programmierung: Numba(Python) vs C-CUDA:
C gewinnt mit Faktor 1,2 – 2.

L. Oden (2020), Lessons learned from comparing C-CUDA and Python-Numba for GPU-Computing, 2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Västerås, Sweden, 2020, pp. 216-223, <https://doi.org/10.1109/PDP50117.2020.00041>