



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

DR. MAIK WEBER

11.03.2024

PROBELEHRVERANSTALTUNG AN DER HOCHSCHULE TRIER

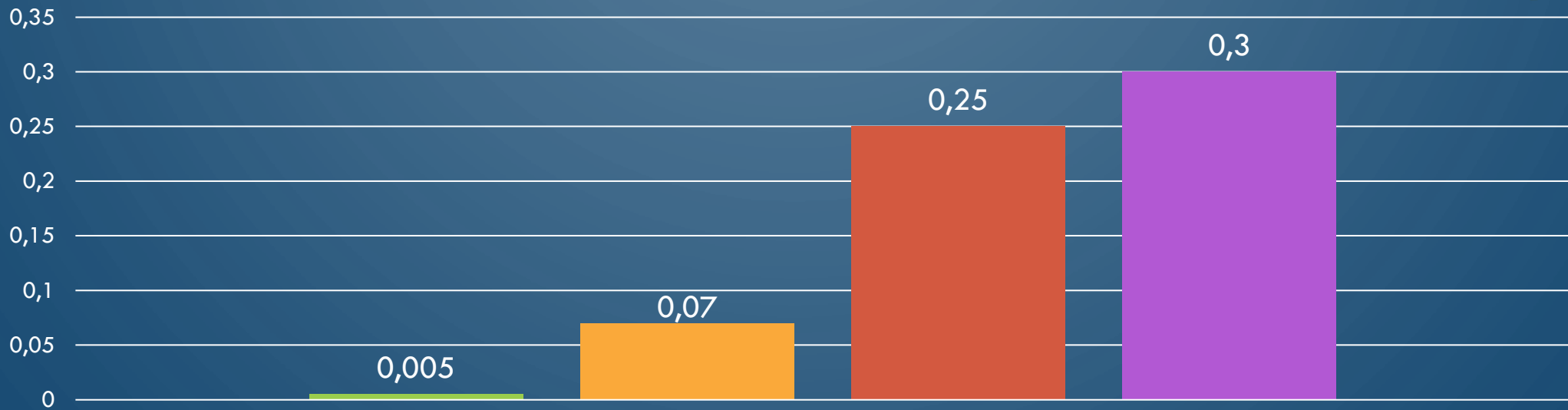
PROGRAMMIERSPRACHEN IM WETTSTREIT

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

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



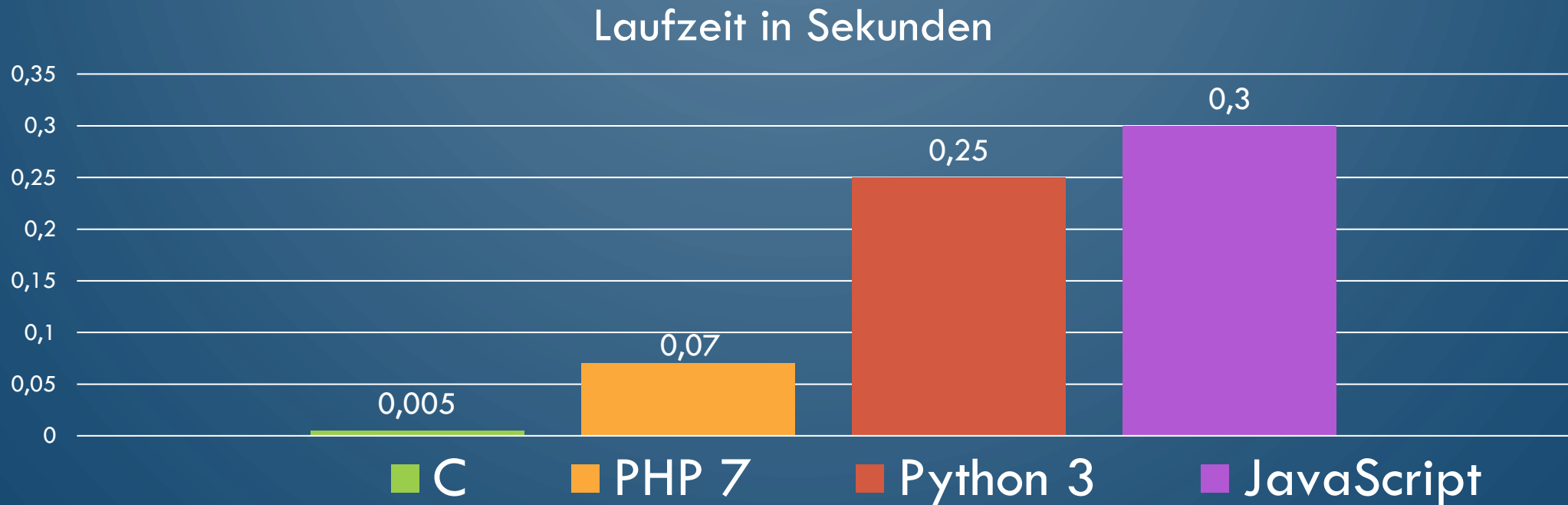
Laufzeit in Sekunden



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

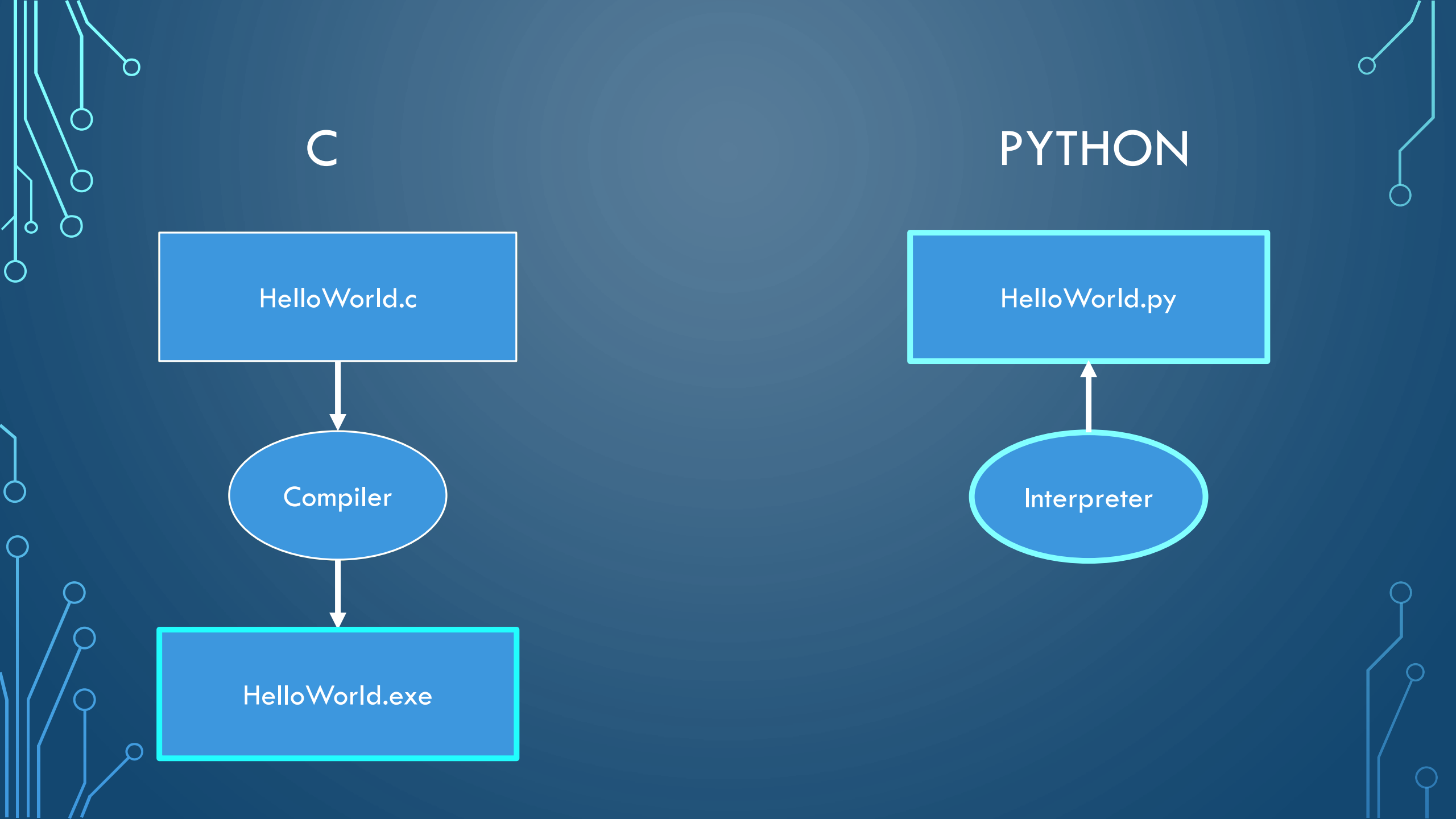
Compiler

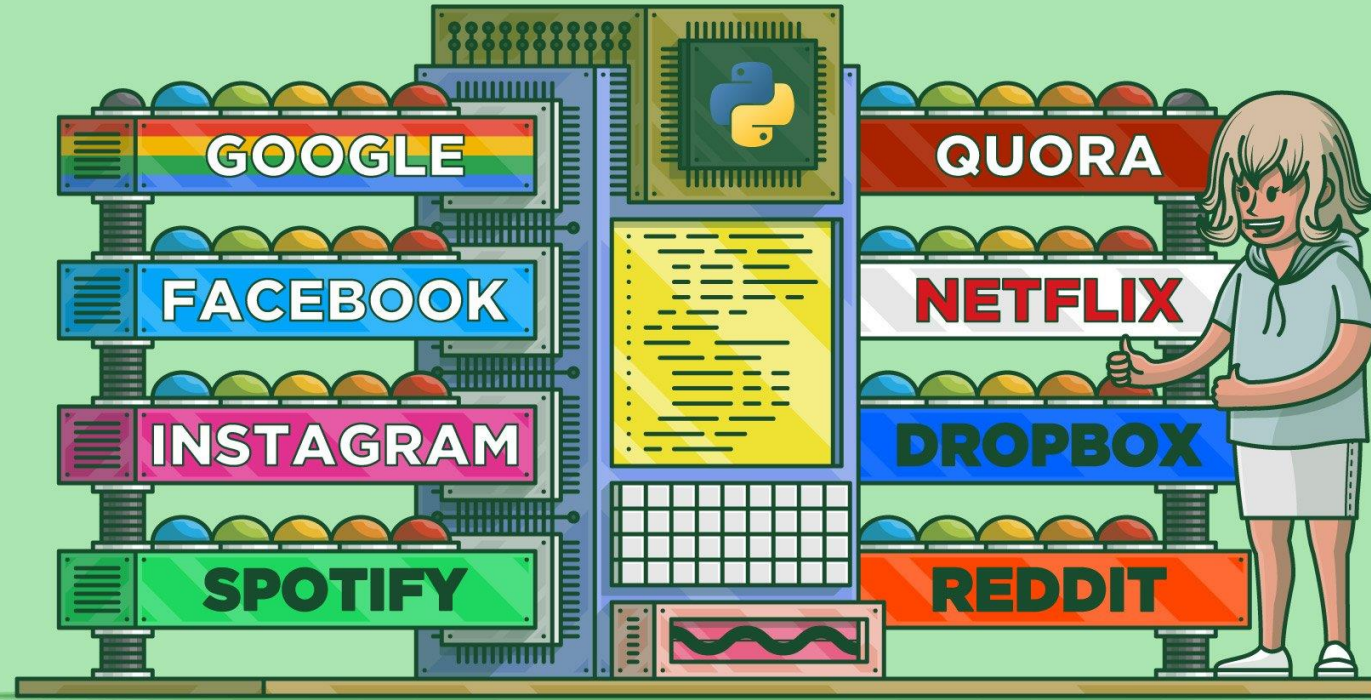
HelloWorld.exe

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 zur Verfügung stellen (mehrere Möglichkeiten)

direkte Installation: `setup.py → pip install`

(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 white, stylized circuit-like lines. These lines consist of straight segments connected by small circles, resembling a network or a map. The lines are more dense in the top-left and bottom-left corners and more sparse in the top-right and bottom-right corners.

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.


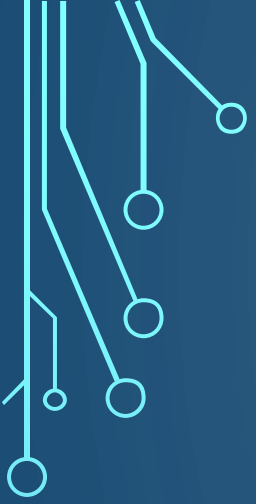
L. Oden (2020), Lessons learned from comparing C-CUDA and Python-Numba for GPU-Computing, 2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Västerås, Sweden, 2020, pp. 216-223, <https://doi.org/10.1109/PDP50117.2020.00041>

An abstract graphic on the left side of the slide, consisting of a network of light blue lines and small circles, resembling a circuit board or a neural network. The lines are vertical and horizontal, with some diagonal connections, and the circles are placed at various points along these lines.



QUIZ


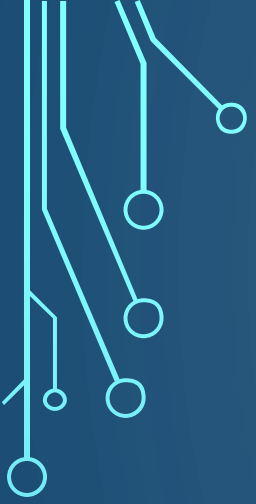
DAS QUIZ ZUR VORLESUNG

- 4 Antwortmöglichkeiten
- 1 richtige Antwort
- Abstimmung mit Fingern 1 - 4





Der Hauptgrund für die Verwendung von C-Extensions in Python liegt in der Regel in der Performancesteigerung. Welche Vorteile sind darüber hinaus denkbar?

1. Einfachere Fehlerbehebung
 2. Bessere Lesbarkeit und Wartbarkeit
 3. Einfacherer Zugriff auf Systemaufrufe
 4. Verbesserte Portabilität
- 
- 



Der Hauptgrund für die Verwendung von C-Extensions in Python liegt in der Regel in der Performancesteigerung. Welche Vorteile sind darüber hinaus denkbar?

1. Einfachere Fehlerbehebung
 2. Bessere Lesbarkeit und Wartbarkeit
 - 3. Einfacherer Zugriff auf Systemaufrufe**
 4. Verbesserte Portabilität
- 
- 

Was ist ctypes?

1. Ein Modul zur Verwendung von Pointers in Python.
2. Ein Compiler und Interpreter, der Python in Bytecode kompiliert, bevor er ihn interpretiert
3. Eine Schnittstelle, die es erlaubt Python-Module in C zu schreiben
4. Ein Modul, das Pythonic-Bindungen zu C-Bibliotheken bereitstellt.

Was ist ctypes?


1. Ein Modul zur Verwendung von Pointers in Python.
2. Ein Compiler und Interpreter, der Python in Bytecode kompiliert, bevor er ihn interpretiert
3. Eine Schnittstelle, die es erlaubt Python-Module in C zu schreiben
4. **Ein Modul, das Pythonic-Bindungen zu C-Bibliotheken bereitstellt.**

Was ist KEINE Best Practice beim Einsatz von C-Erweiterungen für Python?

1. Vermeiden Sie Speicherlecks: In C muss Speicher manuell verwaltet und freigegeben werden!
2. Verwenden Sie C-Erweiterungen nur, wenn es notwendig ist: Prüfen Sie vorhandene Alternativen und wägen Sie Entwicklungsaufwand und Leistungssteigerung ab!
3. Schützen Sie Ihren Quellcode: Stellen Sie Projektpartnern Ihre C-Funktion als Shared Library zur Verfügung, die diese über CTypes einbinden können.
4. Behandeln Sie Exceptions korrekt: Wenn Sie in Ihrem C-Code eine Exception auslösen, stellen Sie sicher, dass diese in Python korrekt interpretiert wird.


Was ist KEINE Best Practice beim Einsatz von C-Erweiterungen für Python?

1. Vermeiden Sie Speicherlecks: In C muss Speicher manuell verwaltet und freigegeben werden!
2. Verwenden Sie C-Erweiterungen nur, wenn es notwendig ist: Prüfen Sie vorhandene Alternativen und wägen Sie Entwicklungsaufwand und Leistungssteigerung ab!
3. **Schützen Sie Ihren Quellcode: Stellen Sie Projektpartnern Ihre C-Funktion als Shared Library zur Verfügung, die diese über CTypes einbinden können.**
4. Behandeln Sie Exceptions korrekt: Wenn Sie in Ihrem C-Code eine Exception auslösen, stellen Sie sicher, dass diese in Python korrekt interpretiert wird.

The slide features a dark blue background with decorative light blue circuit-like lines. These lines, consisting of straight segments and small circles at junctions, are positioned along the left and right edges of the slide, framing the central text area.

PyPy ist ein in Python geschriebener Just-In-Time Compiler und Interpreter für Python-Code. Welche Nachteile erwarten Sie im Vergleich zur Standard-Python-Implementierung in C (Cpython)?

1. C-Extensions sind nicht kompatibel mit PyPy.
2. Nativer Python-Code läuft in der Regel schneller auf PyPy, aber C-Extensions können langsamer laufen als auf CPython.
3. Bekannte Module wie NumPy sind nicht verfügbar für PyPy, da sie in C/C++ geschrieben wurden.
4. Da die Performance von C-Extensions unabhängig von Plattform und Interpreter ist, sind keine Unterschiede zu erwarten.

The slide features a dark blue background with decorative white circuit-like lines and nodes. These lines are positioned along the left and right edges, with some extending into the central area. The lines consist of straight segments connected by small circles, resembling a stylized electronic circuit.

PyPy ist ein in Python geschriebener Just-In-Time Compiler und Interpreter für Python-Code. Welche Nachteile erwarten Sie im Vergleich zur Standard-Python-Implementierung in C (Cpython)?

1. C-Extensions sind nicht kompatibel mit PyPy.
- 2. Nativer Python-Code läuft in der Regel schneller auf PyPy, aber C-Extensions können langsamer laufen als auf CPython.**
3. Bekannte Module wie NumPy sind nicht verfügbar für PyPy, da sie in C/C++ geschrieben wurden.
4. Da die Performance von C-Extensions unabhängig von Plattform und Interpreter ist, sind keine Unterschiede zu erwarten.