



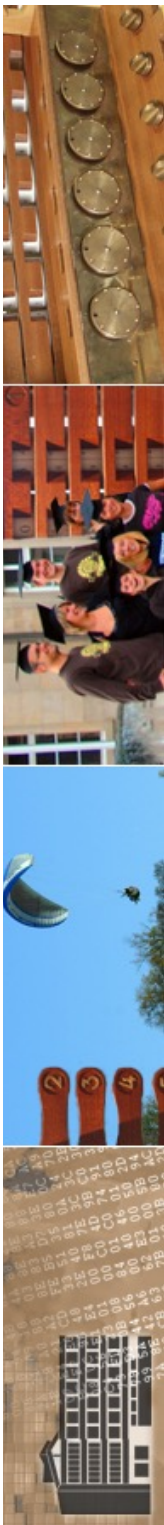
Grundlagen Internet-Technologien

INF3171

Serverseitige Web-Programmierung
mit CGI, Teil I: Einführung in Python

Version 1.0

22.05.2025



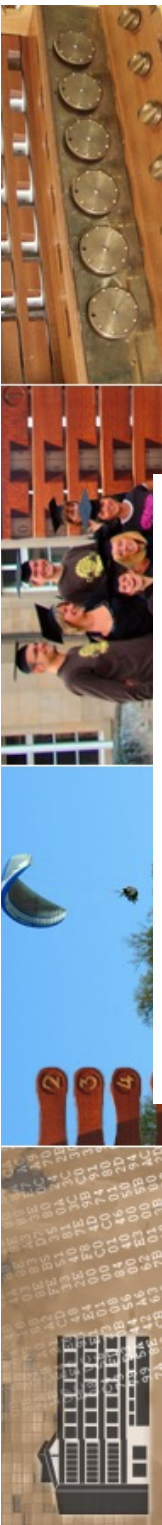


Sommerfest der Informatik

Hallo ihr Lieben,

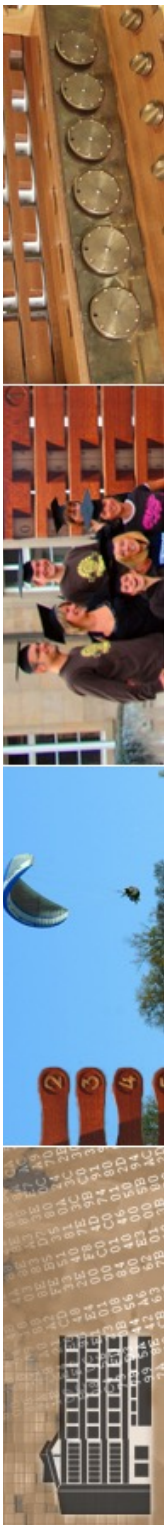
bald ist es endlich wieder so weit! Am ****27. Juni**** von ****17 bis 24 Uhr**** laden euch die Fachschaften Informatik und Kognitionswissenschaft herzlich zum Sommerfest auf dem Sand ein!

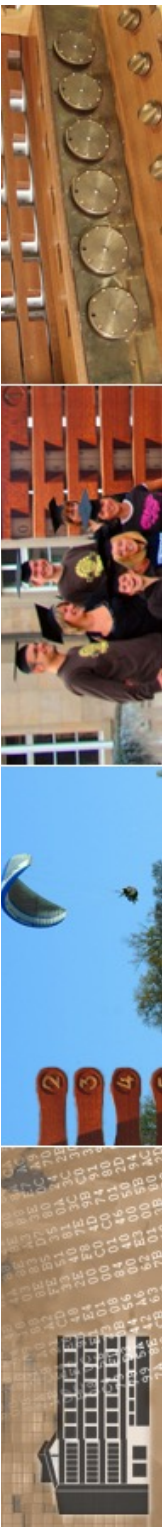
Ob ihr nur kurz vorbeischaut oder den ganzen Abend mit uns verbringt - ihr seid herzlich willkommen!



heute: wichtige Exzellenzentscheidung

- am 22.05. Bekanntgabe der Ergebnisse der Exzellenzcluster
 - Universität Tübingen hat sagenhafte neun (!!!) Anträge
 - nur mit 2 Clustern antragsberechtigt für Exzellenzuniversität





Missing Link: 5 Jahre DSGVO – "Die gezielte Panikmache hat sich gelegt"



Seit dem 25. Mai 2018 gilt die Datenschutz-Grundverordnung. Sie hat hiesige Grundsätze wie Datenminimierung bekannt gemacht und entzweit bis heute die Gemüter.

Lesezeit: 16 Min.  In Pocket speichern

   128



(Bild: mixmagic/Shutterstock.com)

08:05 Uhr

Von Stefan Krempf

In wenigen Tagen, am 25. Mai 2023, ist die Datenschutz-Grundverordnung (DSGVO) seit genau fünf Jahren in Kraft. Das Normenwerk mit 99 Artikeln und 173 erläuternden Erwägungsgründen polarisiert bis heute: Für die einen ist es ein Ausbund an Bürokratie, die aufgrund der enthaltenen Einwilligungsklausel im nervtötenden Abklicken von Cookie-Bannern gipfelt. Andere sehen in der Verordnung, deren Beschluss von einem heftigen Lobby-Streit begleitet war, einen weltweiten Goldstandard beim Absichern der Privatsphäre der Bürger.

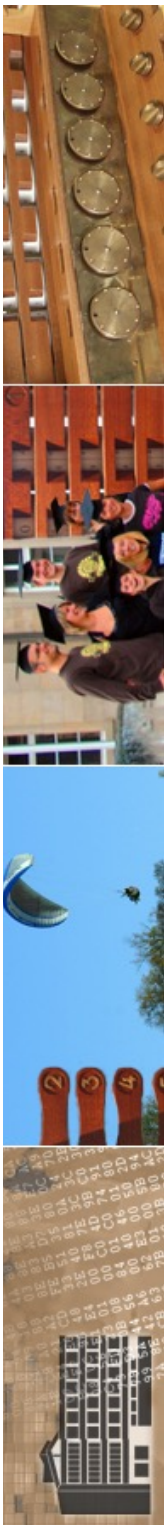
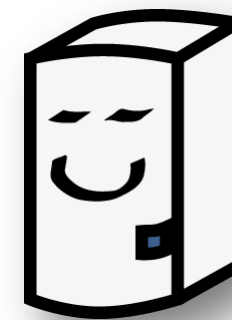


May 2024	May 2023	Change	Programming Language		Ratings	Change
1	1			Python	16.33%	+2.88%
2	2			C	9.98%	-3.37%
3	4	↑		C++	9.53%	-2.43%
4	3	↓		Java	8.69%	-3.53%
5	5			C#	6.49%	-0.94%
6	7	↑		JavaScript	3.01%	+0.57%
7	6	↓		Visual Basic	2.01%	-1.83%
8	12	↑↑		Go	1.60%	+0.61%
9	9			SQL	1.44%	-0.03%
10	19	↑↑		Fortran	1.24%	+0.46%
11	11			Delphi/Object Pascal	1.24%	+0.23%
12	10	↓		Assembly language	1.07%	-0.13%
13	18	↑↑		Ruby	1.06%	+0.26%
14	15	↑		MATLAB	1.06%	+0.18%
15	14	↓		Swift	1.01%	+0.09%
16	8	↓↓		PHP	0.97%	-0.62%



Dynamik im Web

- Dynamik beim Client
 - JavaScript
 - Flash
 - Silverlight
 - Java Applets
- Dynamik beim Server
 - CGI (mit Python, Perl, C, ...)
 - PHP
 - Java Servlets
- Klassiker google, ebay, amazon, facebook, ...



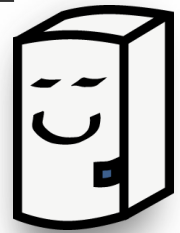
Client versus Server

• Client

- direkte Interaktion
- keine Netzbelastung
- CPU des Clients
- keine DB-Aktion
- Gefahr für Client
- verschiedene Clients führen zu verschiedenen Ergebnissen
- Sourcecode wird ausgeliefert: Kopie

• Server

- Diensteanbieter hat alles in der Hand
- Datenbankanbindung
- zum Client wird nur HTML übertragen
- Performance: alle teilen sich Server-CPU
- keine Interaktion

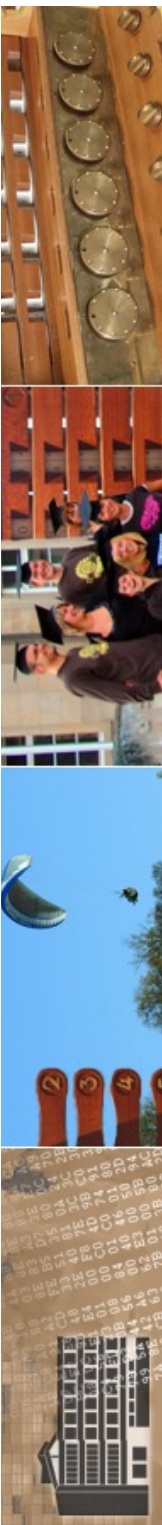


einfach(st)e Servertechnik

- **Common Gateway Interface (CGI)**

Möglichkeit, um im WWW serverseitig Programme bereitzustellen, die von "HTML-Seiten gestartet werden" und HTML-Code produzieren

– Entwicklung ab 1993 (!)



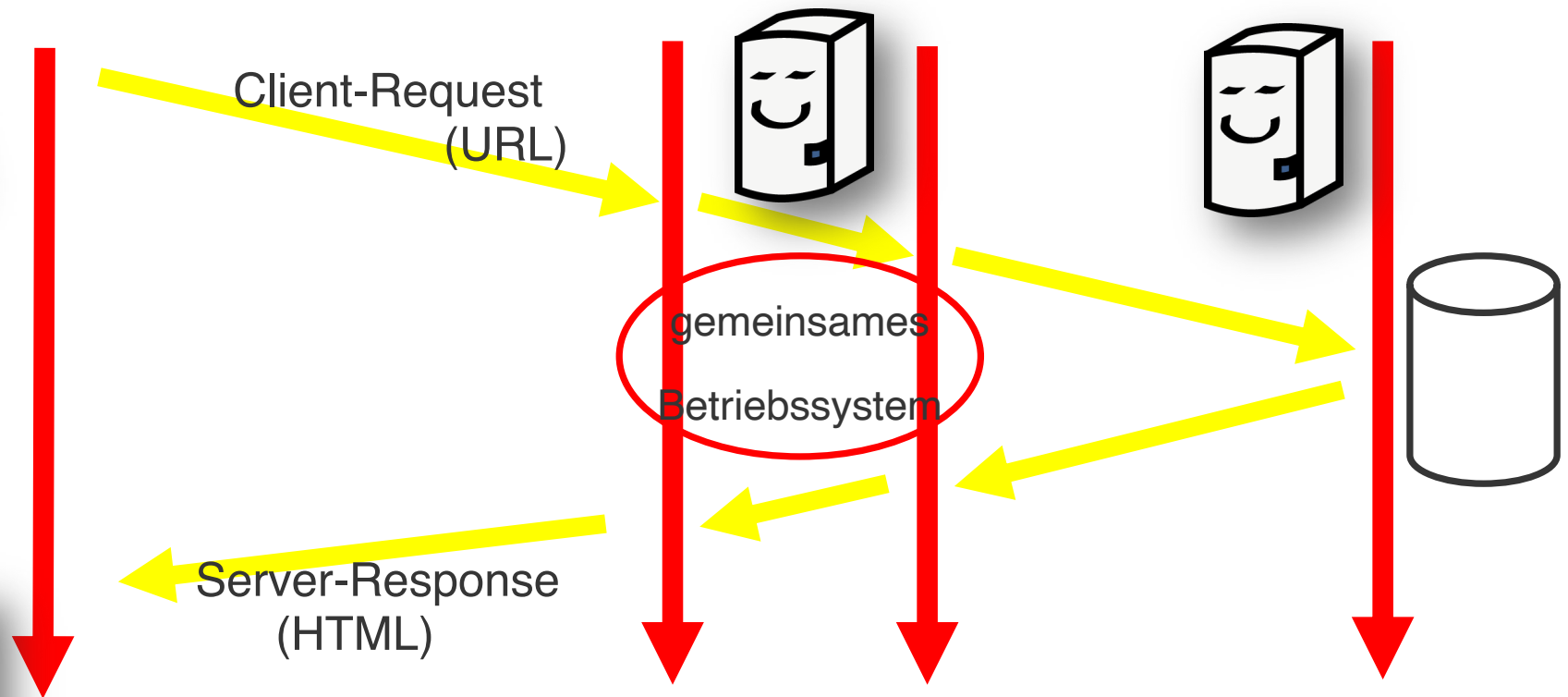
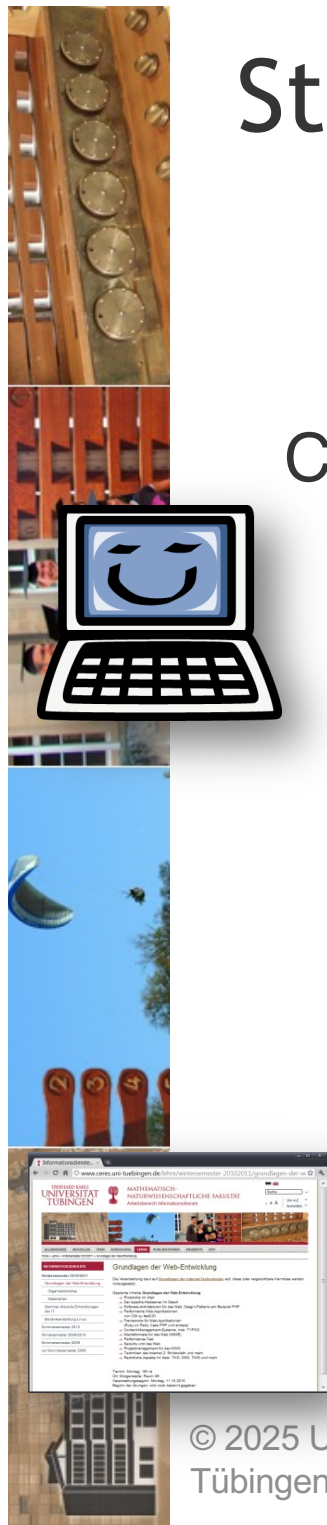
Strukturen einer HTTP-Transaktion mit cgi und Datenbank



Client

WWW-Server cgi-Script

DB-Server





cgi: Voraussetzungen

- Webserver ;-)
 - ...der cgi unterstützt (und auf dem cgi erlaubt ist...)
 - Verzeichnis für die Programme (default meist **cgi-bin**)
 - cgi-Umgebungsvariablen des Webserver

- aktuell cgi Version 1.1 (und das schon lange)
 - <http://www.w3.org/CGI/>
 - Version 1.2 seit November 1997 “in Diskussion”



CGI: Common Gateway Interface

Note: This page is no longer maintained. It is left here for historical purposes. Unfortunately, over time, some links may break that are not maintained by the entities managing those resources.

An [HTTP server](#) is often used as a gateway to a legacy information system; for example, an existing body of documents or an existing database application. The Common Gateway Interface is an agreement between HTTP server implementors about how to integrate such gateway scripts and programs.

It is typically used in conjunction with [HTML](#) forms to build database applications.

See also: [WWW and OOP](#) for more on building distributed applications on the web.

Specs and Documentation

[CGI 1.2 specification \(in progress\)](#)

This directory is the repository for the effort (reactivated in November of 1997) to turn the de facto Common Gateway Interface "standard" into an actual Informational RFC.

[The WWW Common Gateway Interface Version 1.1](#)

16th October 1995. David Robinson. An attempt to update the CGI spec.

[Apache Module mod_cgi](#)

[Using CGI in the CERN httpd](#)

Configuring CERN httpd to use CGI scripts with the Exec directive.

[Setting up CGI in NCSA httpd](#)

A description of using CGI scripts with the NCSA httpd, using ScriptAlias and CGI files.

Discussion

[comp.infosystems.www.authoring.cgi](#)

This newsgroup is a good place to find example scripts and discuss problems with other CGI developers.

[www-talk](#)

If you have technical comments or questions about the development of the CGI spec itself, www-talk is a good place for them.

Next-Generation APIs

See also: [WWW and OOP](#) for info on using CORBA and ILU in place of CGI.

[Fast-CGI](#)

using the CGI programming model in combination with multiplexed network connections.

[ILU Requestor](#)

an approach using distributed objects

[Apache API](#)

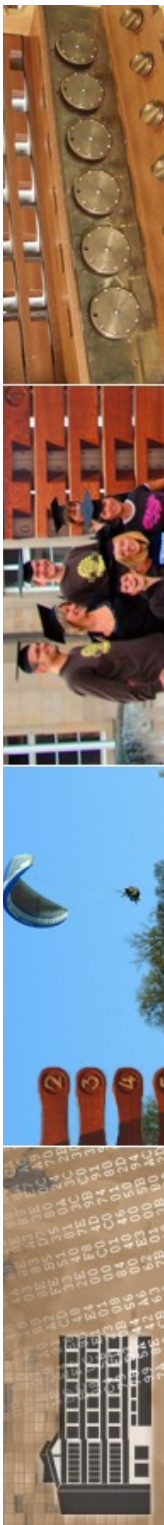


serverseitige Voraussetzungen

- für den Apache-Webserver: Modul `mod_cgi`
 - wird defaultmäßig integriert
- entsprechende Konfiguration in den Apache-Konfigurationsdateien: `httpd.conf`
- Scripte
 - entweder in einem cgi-bin-Verzeichnis (Standard)
 - oder Kennung durch Dateiendung bei entsprechender Konfiguration des Apache

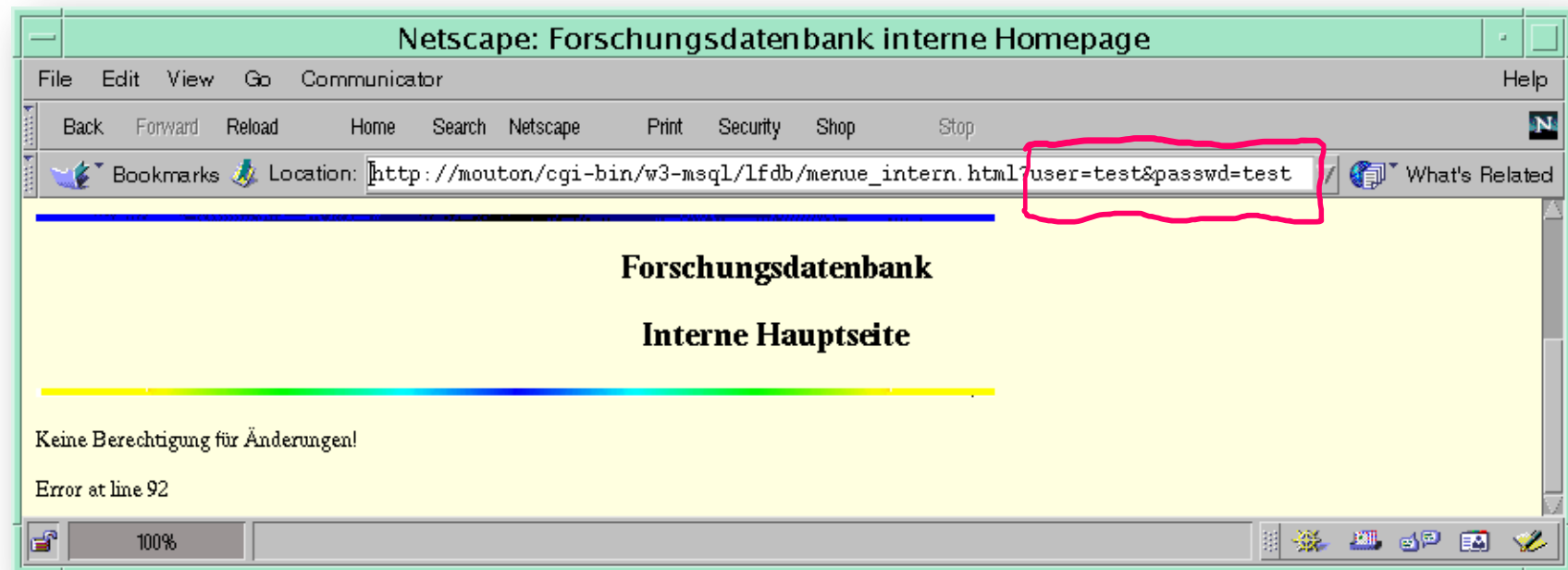
cgi: Aufruf aus HTML

- normaler Hyperlink
 - `<A HREF="/cgi-bin/counter.pl">Zähler</A>`
- über Formulare (→ Übungen)
- Grafikreferenzen (img, src)
- **SSI: Server-Side-Includes**
 - SSI ist eine einfache und effiziente Form für aktive Websites → später



cgi-Datenübergabe an den Server

- *unterschiedliches* Vorgehen bei den HTTP-Methoden **GET** und **POST**
 - GET: Codierung der Eingabefelder in URL; Syntax:
 - & (&) trennt Feldnamen
 - = trennt Name von Wert
 - ? leitet Datenteil ein



cgi-Datenübergabe an den Server

- Nachteile bei GET:

- URLs beliebiger Länge???
- Codierung der »Sonderzeichen«
- Speichern in Browser-History
(→ Kennwörter?!)

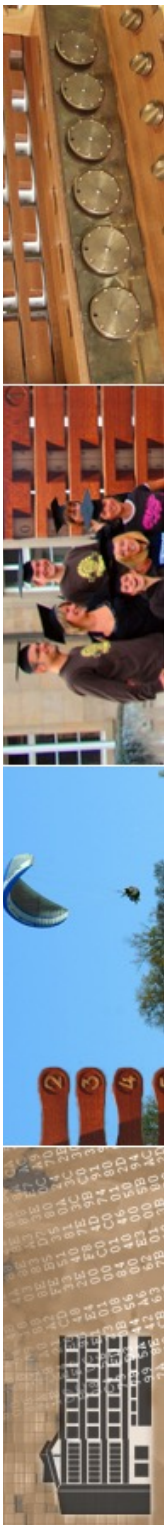
- POST: Daten werden über separate I/O-Ströme weitergeleitet, nicht über URL

Beispiel Zoom

- Kennwort kann verschlüsselt an Zoom-URL angefügt werden:

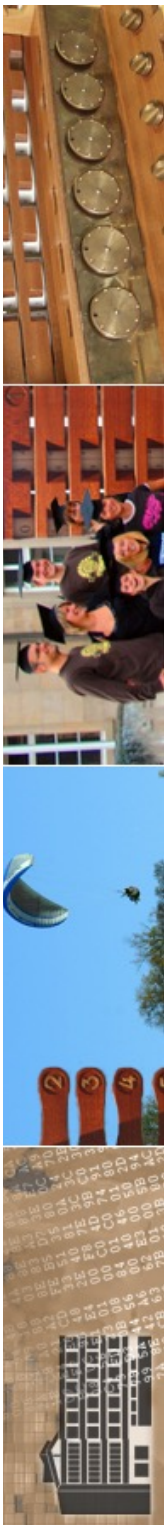
<https://zoom.us/j/96118146729?pwd=...>

– Optional zu aktivieren/deaktivieren



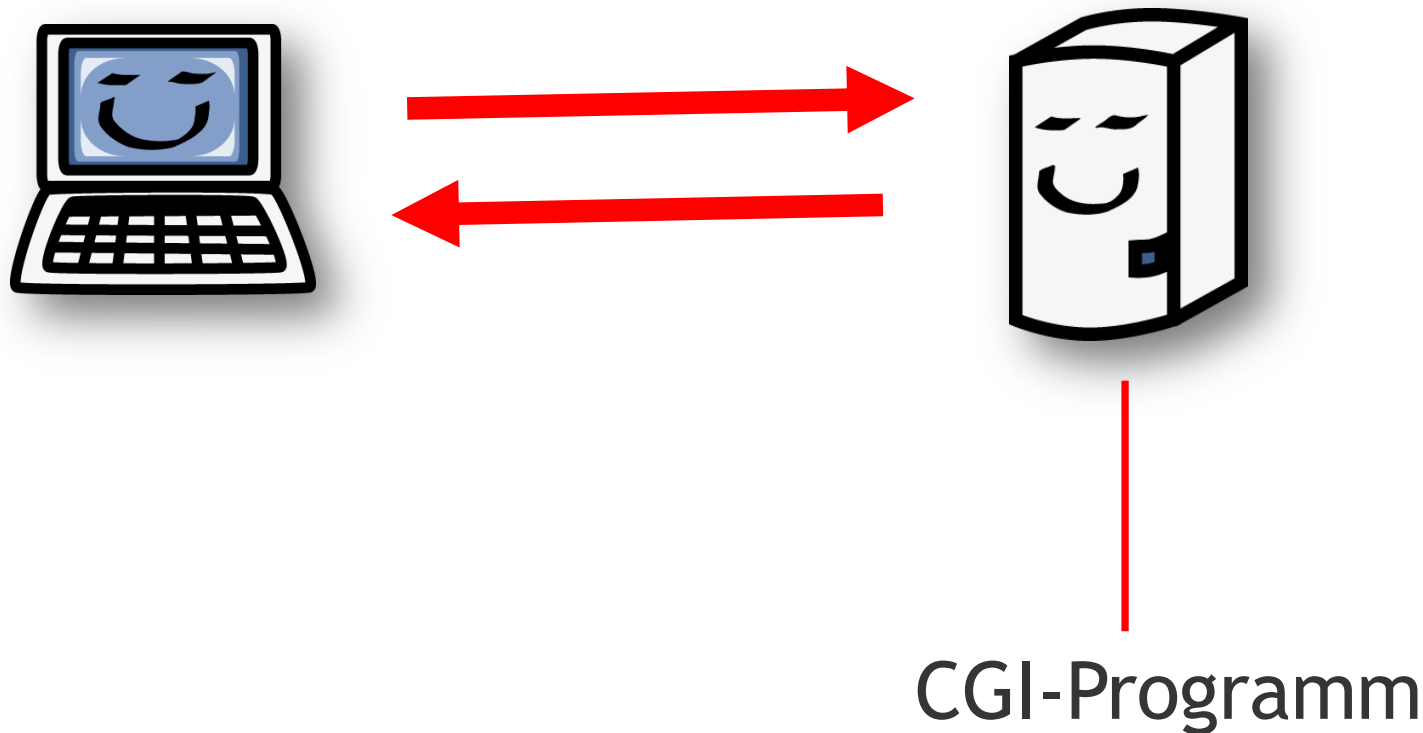
cgi Lifecycle

- ein cgi-Aufruf hat folgenden Lifecycle:
 - mit jedem http-Request wird ein cgi-Prozeß gestartet, der danach wieder beendet wird
- als Konsequenz ist cgi für viele Anfragen nicht performant, weshalb (viele) Alternativen entwickelt wurden
 - fastCGI, Servlets, Webserver-Module, cgid, ...



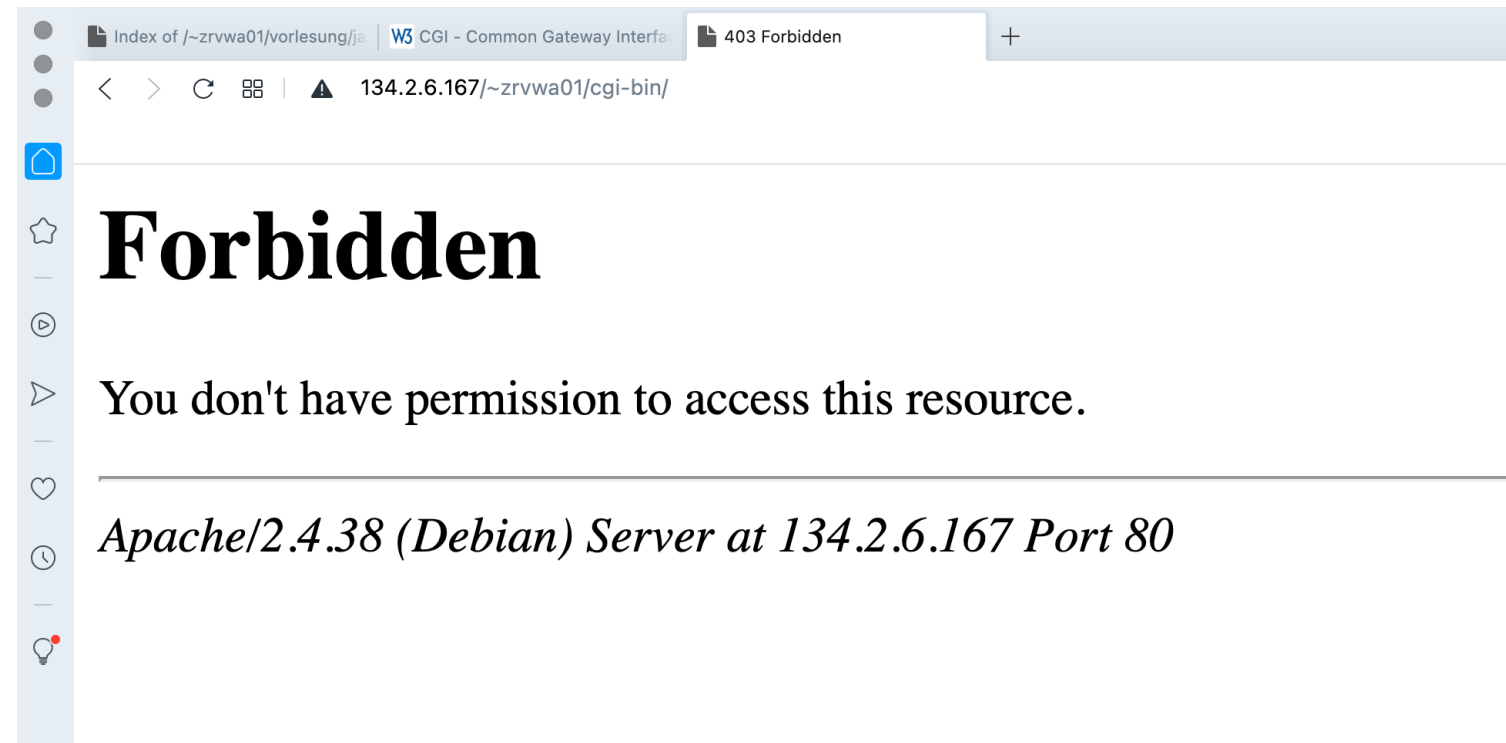
cgi - das Prinzip

- Web-Client „startet“ cgi-Programm auf Server



ganz wichtig...

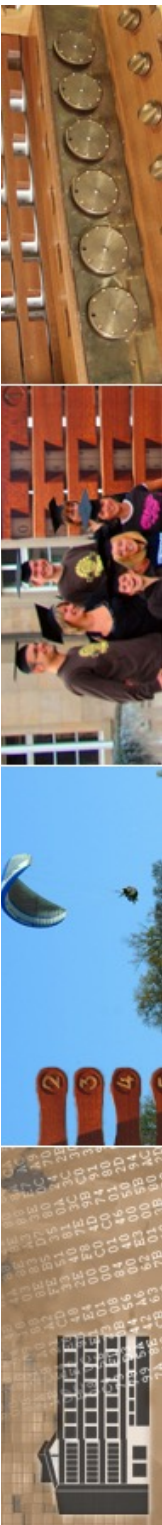
- cgi ist immer auch eine Gefahr (für den Server)



- *keine Übersicht* über Skripte ausliefern

cgi: Umgebungsvariablen

- Server legt bei Aufruf des cgi-Programms spezielle **Umgebungsvariablen** fest, die dem cgi-Programm die notwendigen Informationen liefern (interner Mechanismus der Datenübergabe)
- einige dieser Umgebungsvariablen:
 - `SERVER_NAME`, `SERVER_PROTOCOL`, `SERVER_PORT`
 - `PATH_INFO`, `SCRIPT_NAME`, `QUERY_NAME`
 - `REMOTE_HOST`, `REMOTE_ADDR`, `REMOTE_USER`



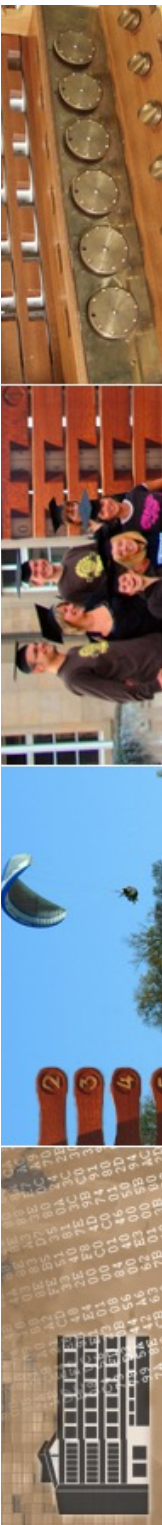
Standardscript

- mit Apache wird das Script

`printenv.pl`

ausgeliefert

- gibt Überblick über alle Umgebungsvariablen des Servers

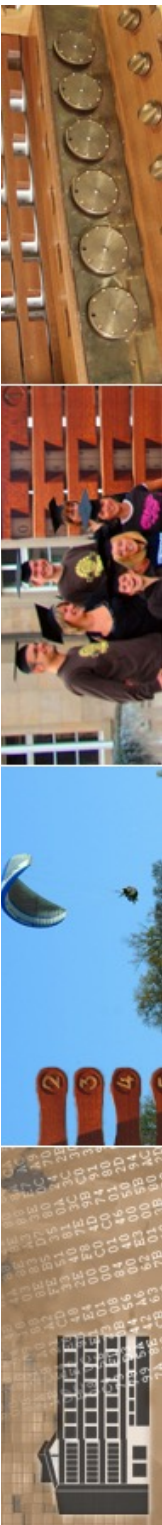


```
CONTEXT_DOCUMENT_ROOT="/home-link/zrvwa01/public_html"
CONTEXT_PREFIX="/~zrvwa01"
DOCUMENT_ROOT="/var/www/html"
GATEWAY_INTERFACE="CGI/1.1"
HTTP_ACCEPT="text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp
,image/apng,*/*;q=0.8"
HTTP_ACCEPT_ENCODING="gzip, deflate"
HTTP_ACCEPT_LANGUAGE="de-DE,de"
HTTP_CONNECTION="keep-alive"
HTTP_COOKIE="PHPSESSID=1te0125u54kql4015iephss4pu"
HTTP_HOST="134.2.6.167"
HTTP_SEC_GPC="1"
HTTP_UPGRADE_INSECURE_REQUESTS="1"
HTTP_USER_AGENT="Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36"
PATH="/usr/local/bin:/usr/bin:/bin"
QUERY_STRING=""
REMOTE_ADDR="134.2.163.18"
REMOTE_PORT="50601"
REQUEST_METHOD="GET"
REQUEST_SCHEME="http"
REQUEST_URI="/~zrvwa01/cgi-bin/perl/printenv.pl"
SCRIPT_FILENAME="/home-link/zrvwa01/public_html/cgi-bin/perl/printenv.pl"
SCRIPT_NAME="/~zrvwa01/cgi-bin/perl/printenv.pl"
SERVER_ADDR="134.2.6.167"
SERVER_ADMIN="webmaster@localhost"
SERVER_NAME="134.2.6.167"
SERVER_PORT="80"
SERVER_PROTOCOL="HTTP/1.1"
SERVER_SIGNATURE("<address>Apache/2.4.56 (Debian) Server at 134.2.6.167 Port
80</address>\n"
SERVER_SOFTWARE="Apache/2.4.56 (Debian)"
```



Beispiel für CGI mit C

- mit C bekommen wir eine **direkt ausführbare Binärdatei**
- Beispiel `webkompendium.c` gibt nur HTML-formatierte Ausgabe aus

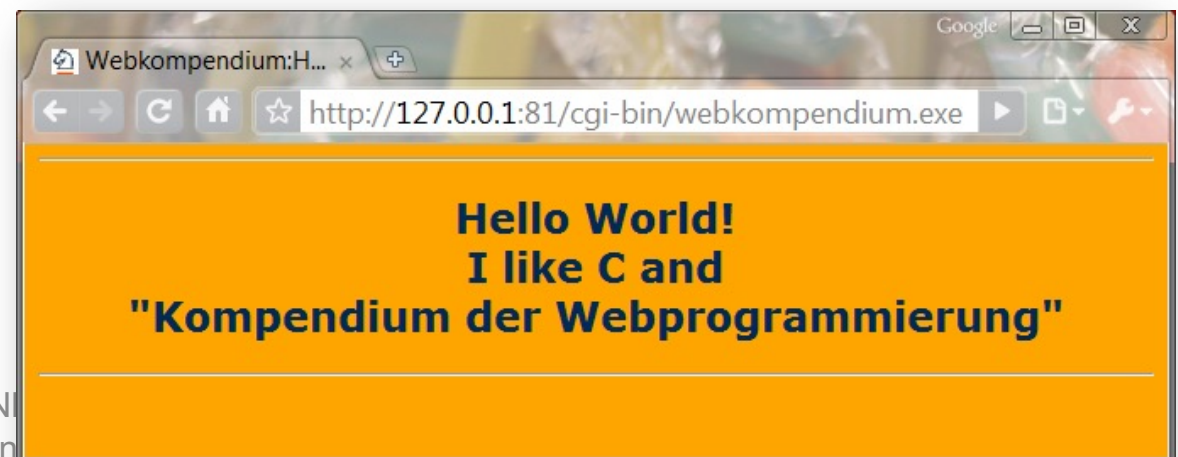




```

1 // Grundlagen Internet-Technologien
2 // CGI-Beispiel in C
3
4 #include <stdio.h>
5
6 main()
7 {
8     printf("Content-type: text/html\n\n");
9     printf("<HTML><HEAD><TITLE>Grundlagen Internet-Technologien:Hello World als C-cgi</TITLE>"
10     printf("<link rel=\"stylesheet\" type=\"text/css\"");
11     printf("href=\"/css/webkompendium.css\">");
12     printf("<BODY><HR><CENTER><H2>Hello World!<BR>");
13     printf("I like C and<BR>\"Grundlagen Internet-Technologien\"");
14     printf("</H2></CENTER><HR></BODY></HTML>");
15 }
16

```

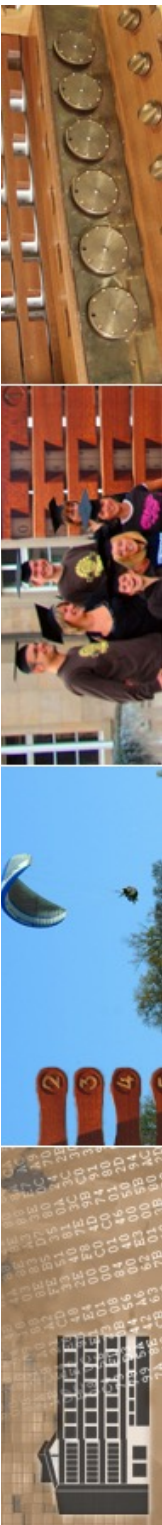


cgi Essentials

- cgi-Programm muss **ausführbar** sein
- liegt auf dem Server im Unterordner cgi-bin
- Ausgabe beginnt mit:

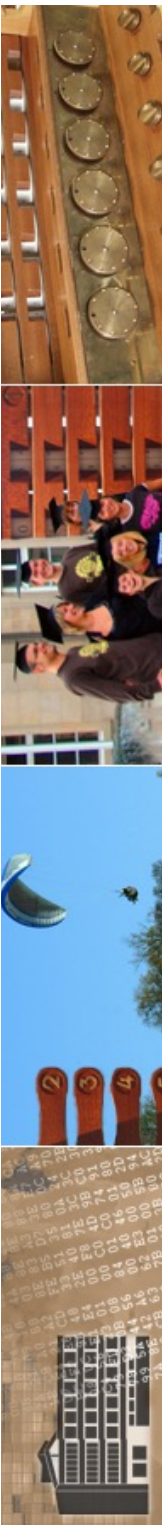
```
Content-type: text/html
\n
```

- Zugriff: URL beginnt mit **cgi-bin**



Sprachen für CGI

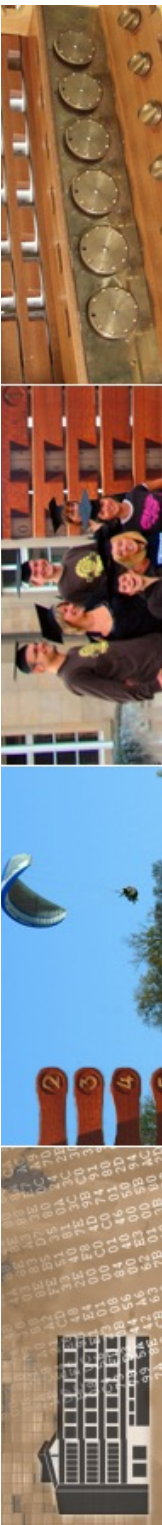
- alles „ausführbare“
- alle compilierten Sprachen (C, C++, C#, ...)
- beliebt: Scriptsprachen wie Perl, Python, Ruby
 - Java ist nicht direkt möglich - warum?





Scriptsprachen für CGI

- Scriptsprache: interpretierte, nichttypisierte Sprache
- relevant für CGI
 - Python
 - PERL: der ewige Klassiker
 - Ruby





TIOBE Index May 2025



[About us](#) [Knowledge](#) [News](#) [Coding Standards](#) [TIOBE Index](#) [Contact](#) [🔍](#)

[Products](#) [Quality Models](#) [Markets](#) [Schedule a demo](#)

May 2025	May 2024	Change	Programming Language		Ratings	Change
1	1			Python	25.35%	+9.02%
2	3	↑		C++	9.94%	+0.41%
3	2	↓		C	9.71%	-0.27%
4	4			Java	9.31%	+0.62%
5	5			C#	4.22%	-2.27%
6	6			JavaScript	3.68%	+0.66%
7	8	↑		Go	2.70%	+1.10%
8	7	↓		Visual Basic	2.62%	+0.61%
9	11	↑		Delphi/Object Pascal	2.29%	+1.05%
10	9	↓		SQL	1.90%	+0.45%
	10	↓		Fortran	1.78%	+0.53%



Tübingen

Technologien Sommersemester 2025





[About us](#)

[Knowledge](#)

[News](#)

[Coding Standards](#)

[TIOBE Index](#)

[Contact](#)



[Products](#) ▾

[Quality Models](#) ▾

[Markets](#) ▾

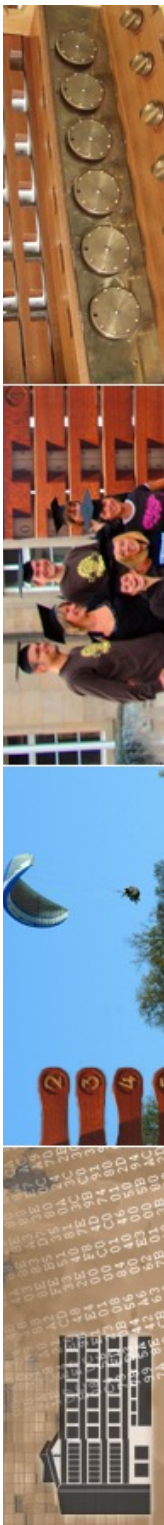
[Schedule a demo](#)

★ Language of the Year: 2007, 2010, 2018, 2020, 2021, 2024

TIOBE Index for Python

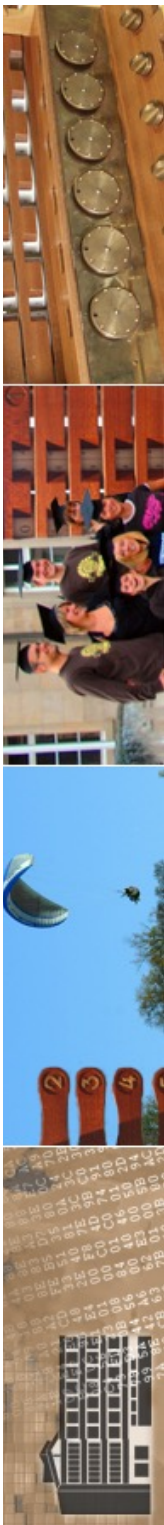
Source: www.tiobe.com





die Scriptsprache Perl

- **PERL: Practical Extraction and Report Language**
(Praktische Sprache für Datenextraktion und -ausgabe)
oder:
Pathologically Electric Rubbish Lister ; -)
(krankhaft zusammengeschustertes Auflistungsprogramm für
wirres Zeug)
- »Erfinder« Larry Wall, 1987
- aktuelle Version: 5.38.2 und (endlich) 6.0.x
(verwenden Sie mindestens 5.004)





Perl

ABOUT

5.38.2
DOWNLOAD

LEARN

DOCS

CPAN

COMMUNITY

That's why we love Perl 25,000 extensions on CPAN

Perl is a highly capable, feature-rich programming language with over 36 years of development.

↓ DOWNLOAD AND GET STARTED



Learning

With free online books, over 25,000 extension modules, and a large developer community, there are many ways to learn Perl.

Community

Perl has an active world wide community with over 230 local groups, mailing lists and support/discussion websites.

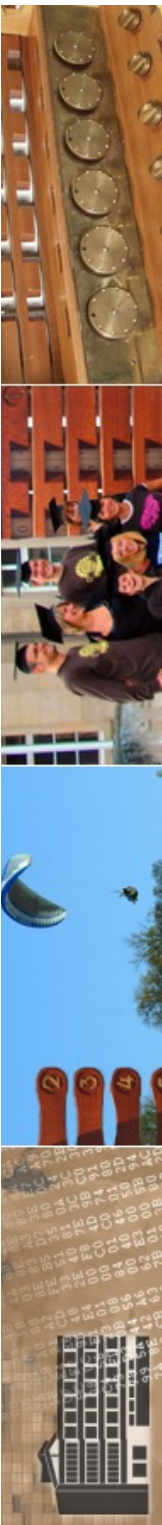
Docs

Core documentation, FAQs and translations.

Bildschirmfoto

die Scriptsprache RUBY

- Erfinder Yukihiro (Matz) Matsumoto
 - Entwicklung ab 1993, erste Veröffentlichung 1995
 - erst seit ~2000 englischsprachige Dokumentation
 - Name ist Anspielung auf PERL
 - stark objektorientiert
 - Grundlage für Ruby on Rails
 - aktuell 3.4.4





Ruby

A PROGRAMMER'S BEST FRIEND

Google™ Custom Search

Search

[Downloads](#) [Documentation](#) [Libraries](#) [Community](#) [News](#) [Security](#) [About Ruby](#)

Ruby is...

A dynamic, open source programming language with a focus on simplicity and productivity. It has an elegant syntax that is natural to read and easy to write.



Download Ruby

or [Read More...](#)

```
# The famous Hello World
# Program is trivial in
# Ruby. Superfluous:
#
# * A "main" method
# * Newline
# * Semicolons
#
# Here is the Code:

puts "Hello World!"
```

Ruby 3.4.4 Released

Ruby 3.4.4 has been released.

[Continue Reading...](#)

Posted by k0kubun on 14 May 2025

CVE-2025-43857: DoS vulnerability in net-imap

Get Started, it's easy!

[Try Ruby! \(in your browser\)](#)

[Ruby in Twenty Minutes](#)

[Ruby from Other Languages](#)

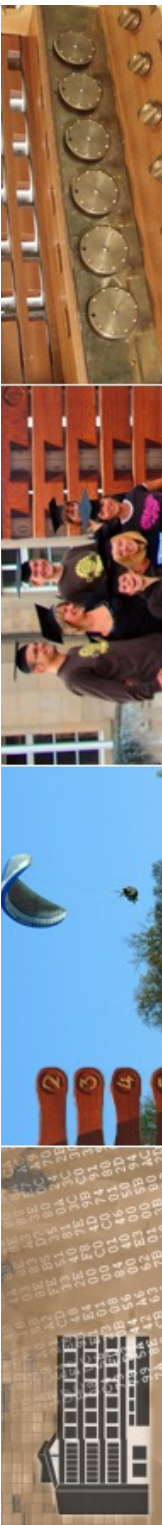
Explore a new world

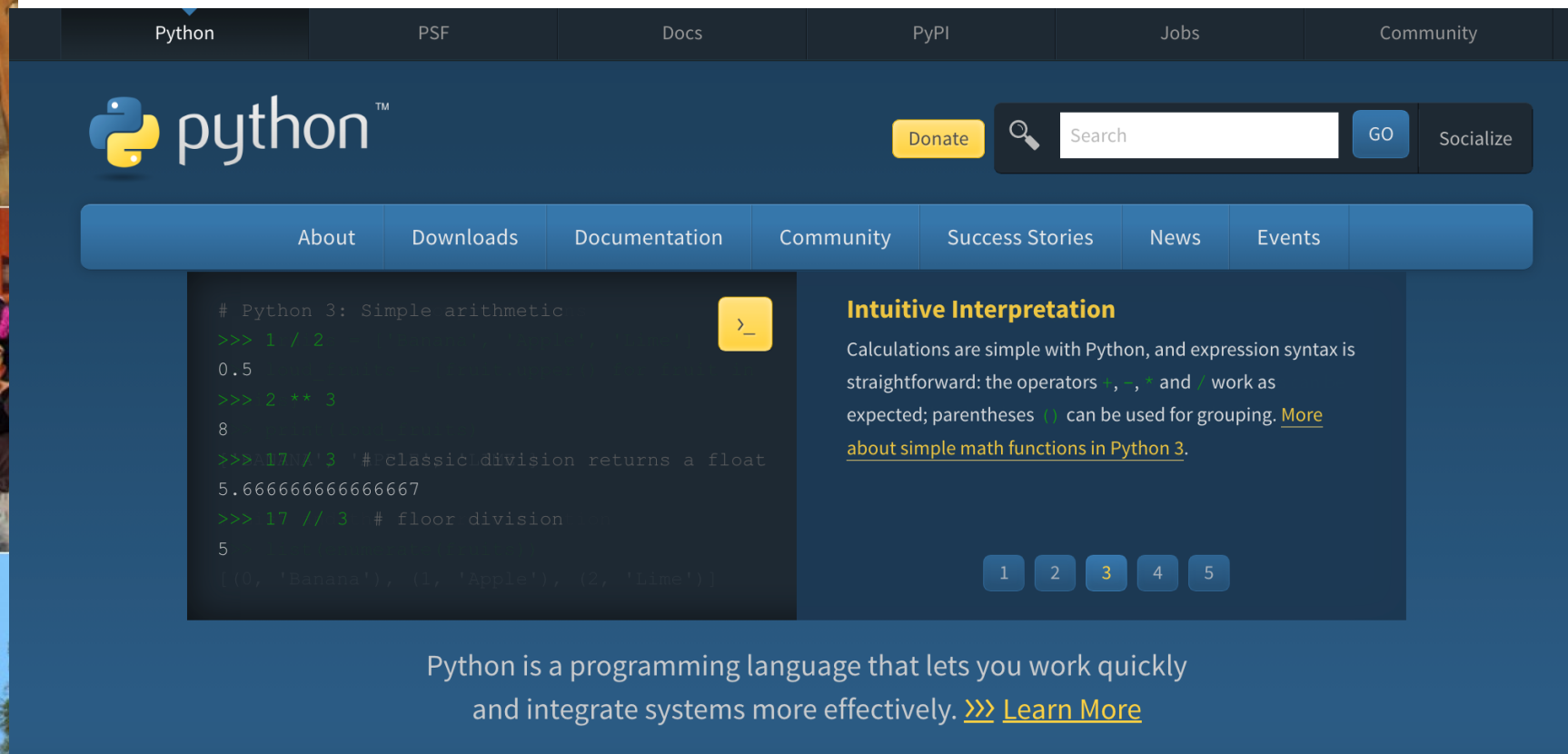




die Scriptsprache PYTHON

- Erfinder Guido van Rossum
 - Centrum voor Wiskunde en Informatica (CWI), Amsterdam <http://www.cwi.nl>
- Entwicklung ab 1989
- zentrale Quelle: www.python.org
- Pflege und Lizenz: Python Software Foundation PSF



Python PSF Docs PyPI Jobs Community

python™

Donate Search GO Socialize

About Downloads Documentation Community Success Stories News Events

```
# Python 3: Simple arithmetic
>>> 1 / 2
0.5
>>> 2 ** 3
8
>>> 17 / 3 '#classic division returns a float
5.666666666666667
>>> 17 // 3 # floor division
5
[(0, 'Banana'), (1, 'Apple'), (2, 'Lime')]
```

Intuitive Interpretation

Calculations are simple with Python, and expression syntax is straightforward: the operators `+`, `-`, `*` and `/` work as expected; parentheses `()` can be used for grouping. [More about simple math functions in Python 3.](#)

1 2 3 4 5

Python is a programming language that lets you work quickly and integrate systems more effectively. [>>> Learn More](#)

Join us in Pittsburgh, PA starting May 14, 2025. Grab your ticket today before we sell out! [REGISTER FOR PYCON US!](#)

Get Started

Whether you're new to programming or an experienced developer, it's easy to learn and use Python.

Download

Python source code and installers are available for download for all versions!

Docs

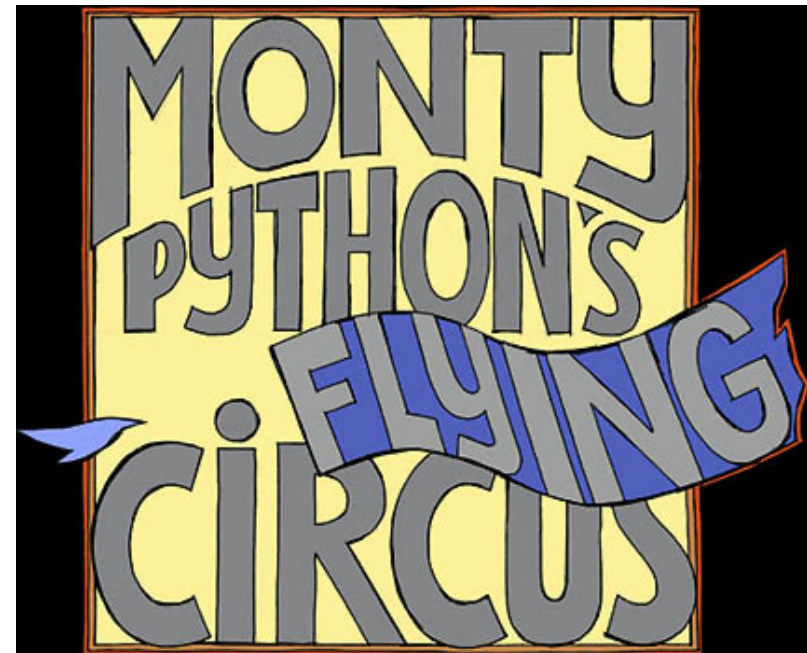
Documentation for Python's standard library, along with tutorials and guides, are available online.

Jobs

Looking for work or have a Python related position that you're trying to hire for? Our **relaunched community-run job board** is the

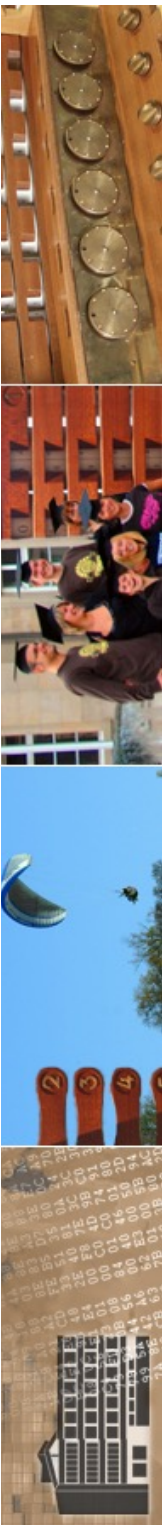
zum Namen

- der Name »Python« kommt *nicht* von der gleichnamigen Schlange, sondern ist ein Bezug zur englischen Comedy-Gruppy **Monty Python** (Das Leben des Brian, Monty Pythons Flying Circus, ...)



Warum PYTHON?

- moderne Scriptsprache ($\gg$ PERL++ $\ll$)
 - plattformunabhängig, da interpretiert
- objektorientiert
- netzwerkzentriert
- ...für CGI geeignet, aber auch für viel mehr!
- ähnlich wie in PERL sind viele Module verfügbar
- frei unter GPL





aktuelle Version

- aktuelle Version: PYTHON 3.13.3
 - für alle gängigen Betriebssysteme verfügbar

What's New In Python 3.10

Release: 3.10.4
Date: May 20, 2022
Editor: Pablo Galindo Salgado

This article explains the new features in Python 3.10, compared to Python 3.9.

For full details, see the [changelog](#).

Summary – Release highlights

New syntax features:

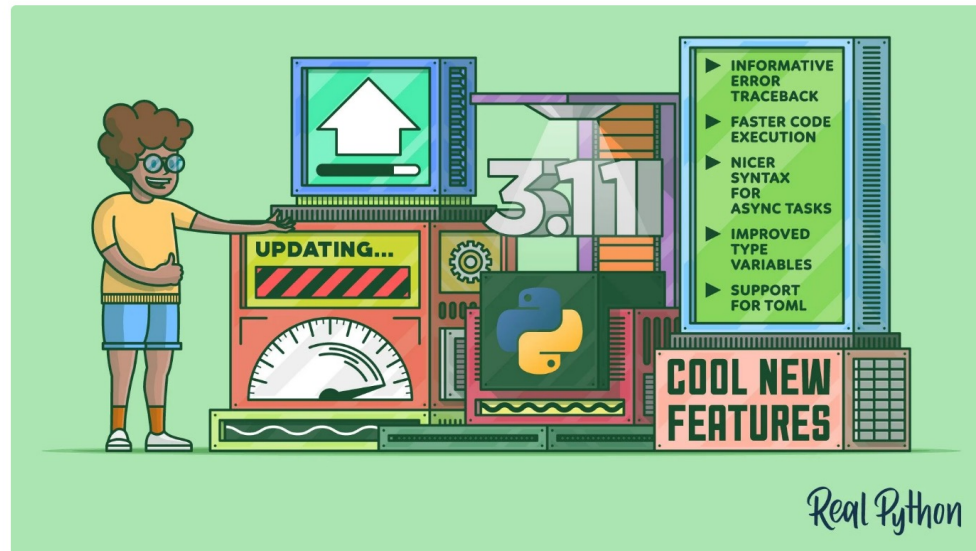
- [PEP 634](#), Structural Pattern Matching: Specification
- [PEP 635](#), Structural Pattern Matching: Motivation and Rationale
- [PEP 636](#), Structural Pattern Matching: Tutorial

```
C:\Users\thomas\Documents>python -V
Python 3.4.3
```

```
C:\Users\thomas\Documents>
```

```
thomas@PetitMouton =>
thomas@PetitMouton =>
thomas@PetitMouton => python3 -V
Python 3.13.2
thomas@PetitMouton =>
thomas@PetitMouton =>
```





Python 3.11: Cool New Features for You to Try

by Geir Arne Hjelle Oct 24, 2022 16 Comments

intermediate python

Mark as Completed



Tweet

Share

Email

Table of Contents

- [More Informative Error Tracebacks](#)
- [Faster Code Execution](#)
- [Nicer Syntax for Asynchronous Tasks](#)
- [Improved Type Variables](#)
- [Support for TOML Configuration Parsing](#)
- [Other Pretty Cool Features](#)
 - [Faster Startup](#)

— FREE Email Series —

Python Tricks

```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```

Email...

Get Python Tricks »

No spam. Unsubscribe any time.

Browse Topics Guided Learning Paths

Basics Intermediate Advanced

api best-practices community databases
data-science devops django docker
flask front-end gamedev gui
machine-learning projects python testing
tools web-dev web-scraping

DATAWARS

Practice Data Science

WITH AI ASSISTANCE

+100 real-life projects, online, ready-to-solve



Python 3.11			
Name	Änderungsdatum	Größe	Art
Python Documentation.html	Heute, 21:07	99 Byte	Alias
IDLE.app	Heute, 21:07	188 KB	Programm
Install Certificates.command	05.04.2023, 02:25	1 KB	Termina...ll-Skript
License.rtf	05.04.2023, 02:25	15 KB	RTF-Dokument
Python Launcher.app	Heute, 21:07	402 KB	Programm
ReadMe.rtf	05.04.2023, 02:25	3 KB	RTF-Dokument
Update Shell Profile.command	05.04.2023, 02:25	3 KB	Termina...ll-Skript
Macintosh HD > Programme > Python 3.11			

Python 3.10.4

Release Date: March 24, 2022



What's New In Python 3.13

Editors: Adam Turner and Thomas Wouters

This article explains the new features in Python 3.13, compared to 3.12. Python 3.13 was released on October 7, 2024. For full details, see the [changelog](#).

See also: [PEP 719](#) – Python 3.13 Release Schedule

Summary – Release Highlights

Python 3.13 is the latest stable release of the Python programming language, with a mix of changes to the language, the implementation and the standard library. The biggest changes include a new [interactive interpreter](#), experimental support for running in a [free-threaded mode](#) ([PEP 703](#)), and a [Just-In-Time compiler](#) ([PEP 744](#)).

Error messages continue to improve, with tracebacks now highlighted in color by default. The [locals\(\)](#) builtin now has [defined semantics](#) for changing the returned mapping, and type parameters now support default values.

The library changes contain removal of deprecated APIs and modules, as well as the usual improvements in user-friendliness and correctness. Several legacy standard library modules have now [been removed](#) following their deprecation in Python 3.11 ([PEP 594](#)).



Search

GO

Socialize

Events

Python's documentation, tutorials, and guides are constantly evolving.
Get started here, or scroll down for documentation broken out by type and subject.

Python Docs

See also [Documentation Releases by Version](#)



- [Beginner's Guide](#)
- [Python FAQs](#)



- Python Periodicals
- Python Books



- [Python Packaging User Guide](#)
- [In-development Docs](#)
- [Guido's Essays](#)



- PEP Index
- Python Videos
- Developer's Guide



- Browse Python 3.9.5 Documentation - (Module Index)
 - What's new in Python 3.9
 - Tutorial
 - Library Reference
 - Language Reference
 - Extending and Embedding
 - Python/C API
 - Using Python



- [FAQ: Sunsetting Python 2](#)
- [Final Python 2.7 Release Schedule](#)
- [Python 3 Statement](#)
- [Porting Python 2 Code to Python 3](#)
 - [Determine what projects are blocking you from porting to Python 3](#)
 - [Python 2 Support and Migration](#)



Download

Download these documents

Docs by version

Python 3.15 (in development)
Python 3.14 (pre-release)
Python 3.13 (stable)
Python 3.12 (security-fixes)
Python 3.11 (security-fixes)
Python 3.10 (security-fixes)
Python 3.9 (security-fixes)
Python 3.8 (EOL)
Python 3.7 (EOL)
Python 3.6 (EOL)
Python 3.5 (EOL)
Python 3.4 (EOL)
Python 3.3 (EOL)
Python 3.2 (EOL)
Python 3.1 (EOL)
Python 3.0 (EOL)
Python 2.7 (EOL)
Python 2.6 (EOL)
All versions

Other resources

PEP Index
Beginner's Guide
Book List
Audio/Visual Talks
Python Developer's Guide

Python 3.13.3 documentation

Welcome! This is the official documentation for Python 3.13.3.

Documentation sections:

What's new in Python 3.13?

Or all "What's new" documents since Python 2.0

Tutorial

Start here: a tour of Python's syntax and features

Library reference

Standard library and builtins

Language reference

Syntax and language elements

Python setup and usage

How to install, configure, and use Python

Python HOWTOs

In-depth topic manuals

Indices, glossary, and search:

Global module index

Installing Python modules

Third-party modules and PyPI.org

Distributing Python modules

Publishing modules for use by other people

Extending and embedding

For C/C++ programmers

Python's C API

C API reference

FAQs

Frequently asked questions (with answers!)

Deprecations

Deprecated functionality



Donate



Search

GO

Socialize

About

Downloads

Documentation

Community

Success Stories

News

Events

Python 3.12.3

Release Date: April 9, 2024

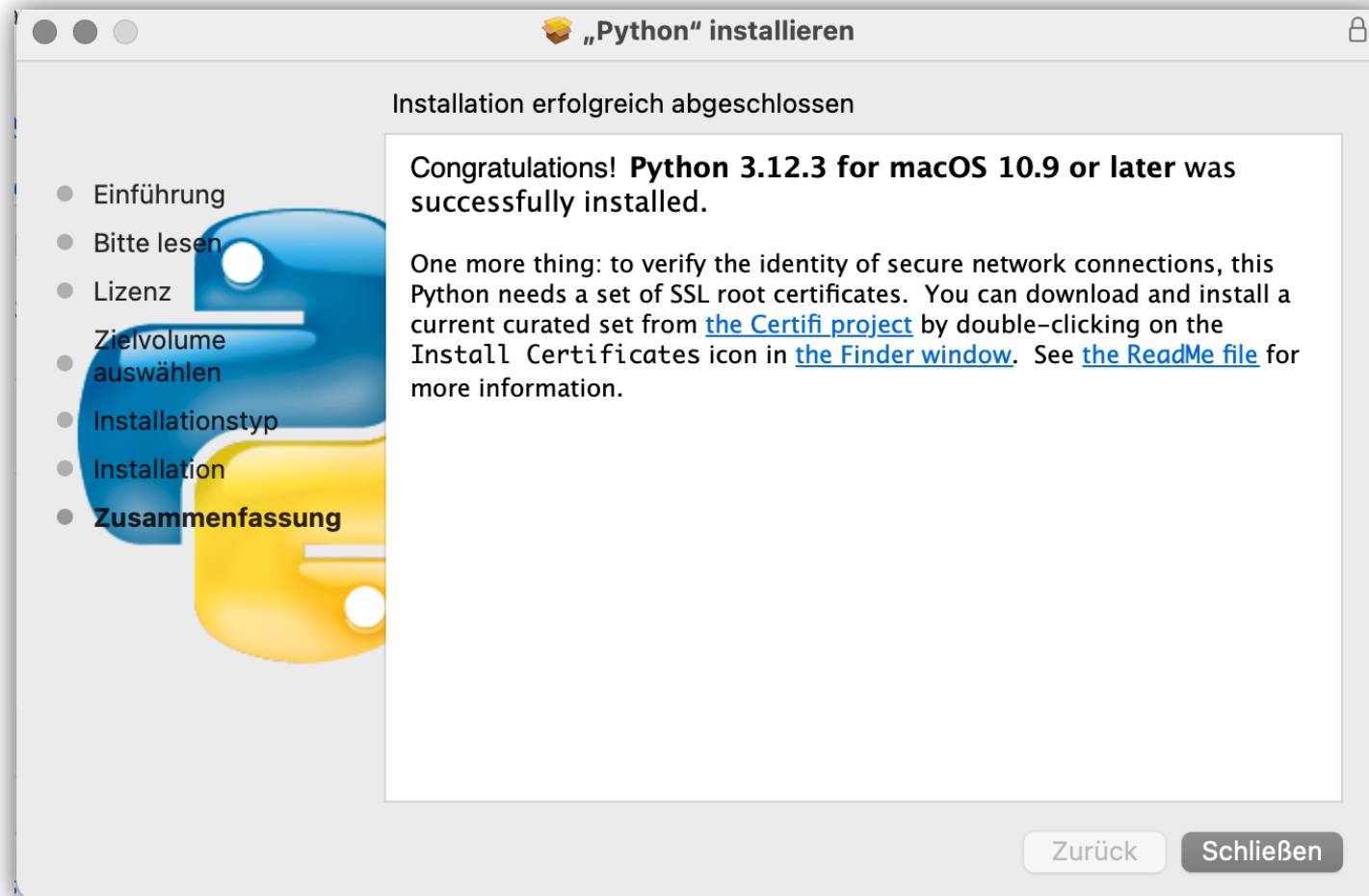
This is the third maintenance release of Python 3.12

Python 3.12 is the newest major release of the Python programming language, and it contains many new features and optimizations. 3.12.3 is the latest maintenance release, containing more than 300 bugfixes, build improvements and documentation changes since 3.12.2.

Major new features of the 3.12 series, compared to 3.11

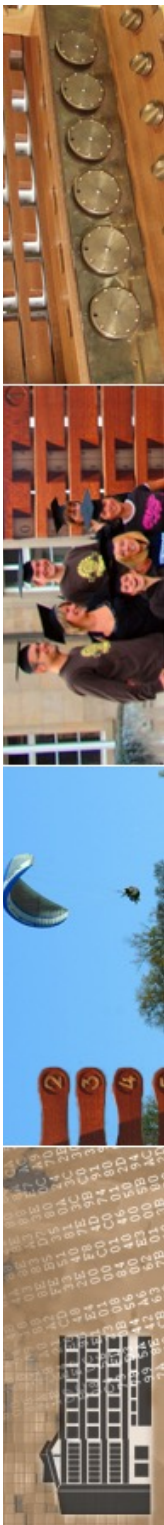
New features

- [More flexible f-string parsing](#), allowing many things previously disallowed ([PEP 701](#)).
- [Support for the buffer protocol](#) in Python code ([PEP 688](#)).
- [A new debugging/profiling API](#) ([PEP 669](#)).
- [Support for isolated subinterpreters](#) with separate Global Interpreter Locks ([PEP 684](#)).
- [Even more improved error messages](#). More exceptions potentially caused by typos now make suggestions to the user.
- [Support for the Linux `perf` profiler](#) to report Python function names in traces.
- [Many large and small performance improvements](#) (like [PEP 709](#) and support for the BOLT binary optimizer), delivering an estimated 5% overall performance improvement.





```
thomas@PetitMouton =>
thomas@PetitMouton =>
thomas@PetitMouton => python3 -V
Python 3.13.2
thomas@PetitMouton =>
thomas@PetitMouton =>
```

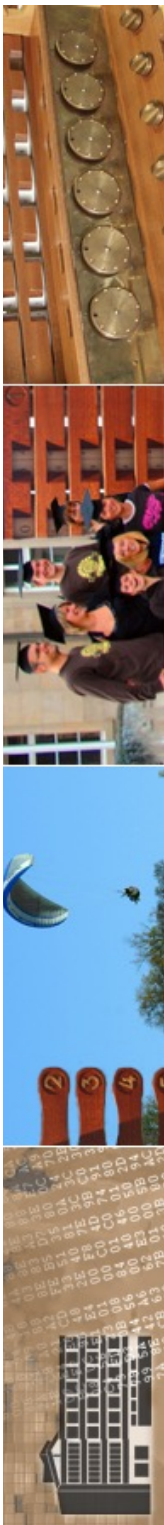


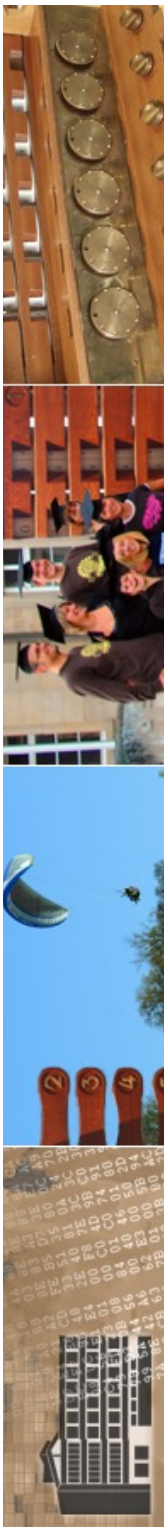


Dokumentation

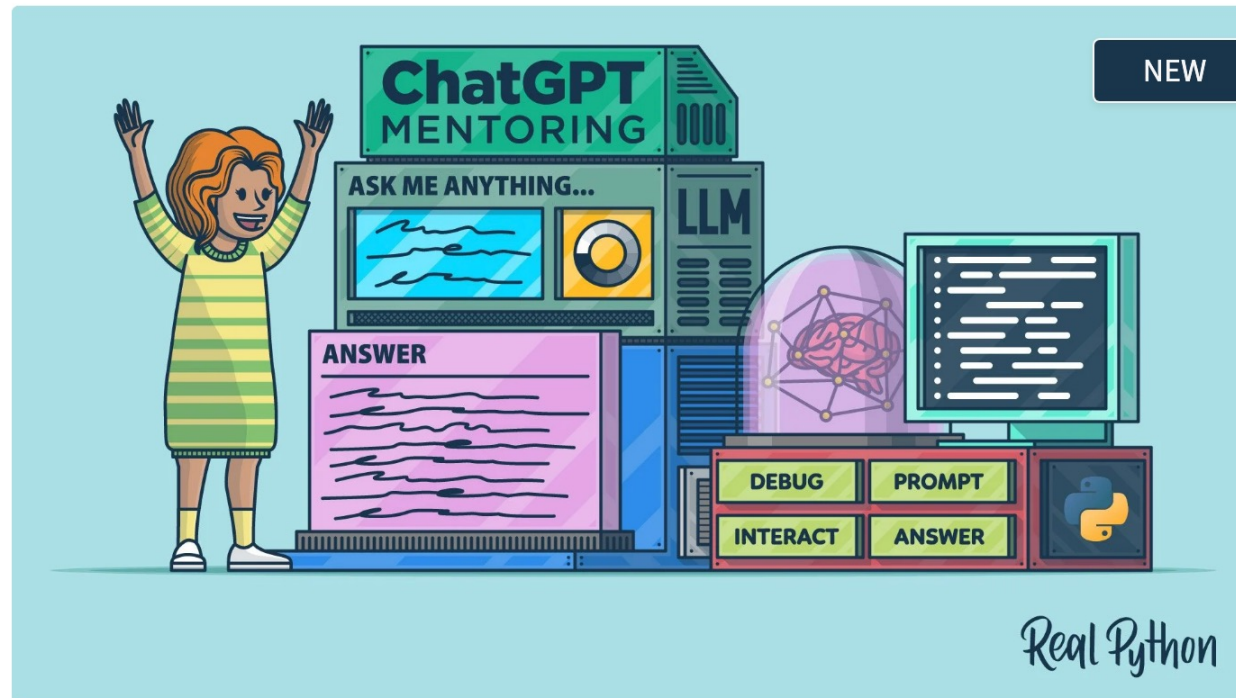
- zentral für Dokumentation:

<https://www.python.org/doc/>





Real Python Tutorials



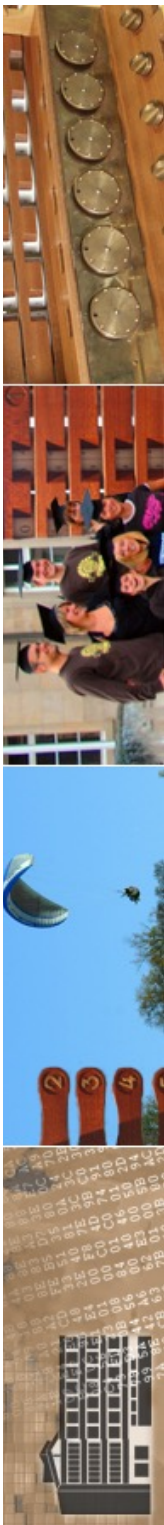
ChatGPT: Your Personal Python Coding Mentor

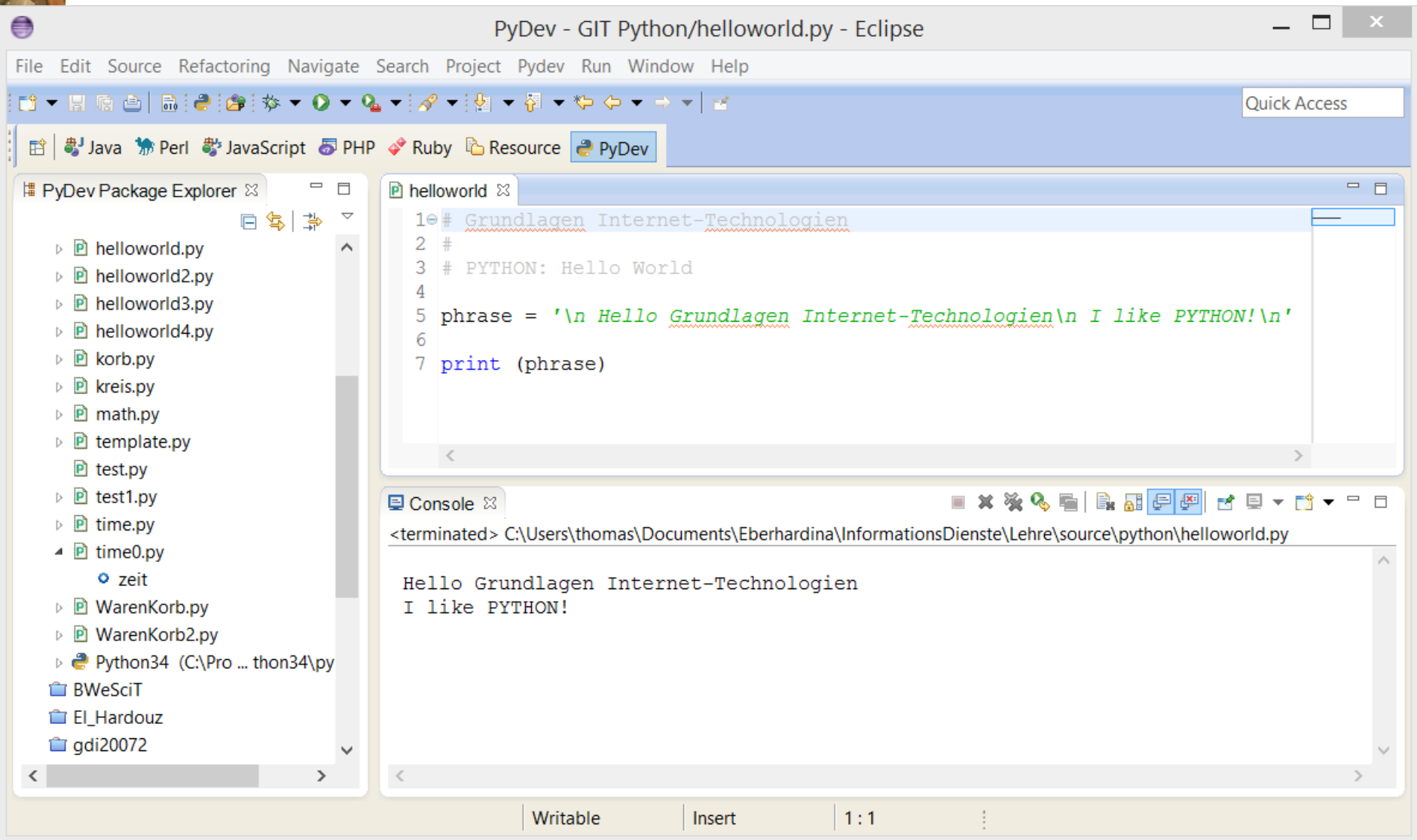
Large language models have gained popularity since OpenAI released ChatGPT. In this tutorial, you'll learn how to use ChatGPT as your Python coding mentor. You'll study a variety of use cases, learn how to interpret results, and see that you need to beware of incorrect and irrelevant responses.

May 17, 2023 basics best-practices

Entwicklungsumgebung

- es gibt (wenige) Python-Plugins für Eclipse
- frei und empfehlenswert: Pydev
<http://pydev.sf.net>
 - Installation über Eclipse-Update-Manager
 - damit pydev-Projekte
 - eigene Python-Perspektive







IDLE

- Python kommt mit einfacher IDE:
IDLE: Integrated DeveLopment Environment

```

Python 2.7.10 (default, Oct 23 2015, 19:19:21)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.0.59.5)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/mac/tcltk/ for current information.
    
```

Ln: 6 Col: 0



IDLE3

```

IDLE Shell 3.13.2

Python 3.13.2 (v3.13.2:4f8bb3947cf, Feb 4 2025, 11:51:10) [Clang 15.0.0 (clang-1500.3.9.4)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> |

```

Ln: 3 Col: 0

erste Schritte, Grundprinzip

- Python hat einen »interaktiven Modus« - direktes Arbeiten mit dem Interpreter
 - (Beenden durch <STRG>Z , <STRG>D oder exit())

```
thomas@u-163-c018 =>
thomas@u-163-c018 =>
thomas@u-163-c018 => python3
Python 3.11.3 (v3.11.3:f3909b8bc8, Apr  4 2023, 20:12:10) [Clang 13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
>>>
```



```
DOS Shell - python
C:\Users\thomas\Documents>
C:\Users\thomas\Documents>
C:\Users\thomas\Documents>python
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:44:40) [MSC v.1600 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> help()

Welcome to Python 3.4's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://docs.python.org/3.4/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics". Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help>
```





```
thomas@PetitMouton-2 =>
thomas@PetitMouton-2 => python3
Python 3.8.3 (v3.8.3:6f8c8320e9, May 13 2020, 16:29:34)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> help()
```

Welcome to Python 3.8's help utility!

If this is your first time using Python, you should definitely check out the tutorial on the Internet at <https://docs.python.org/3.8/tutorial/>.

Enter the name of any module, keyword, or topic to get help on writing Python programs and using Python modules. To quit this help utility and return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type "modules", "keywords", "symbols", or "topics". Each module also comes with a one-line summary of what it does; to list the modules whose name or summary contain a given string such as "spam", type "modules spam".

help>





```

*IDLE Shell 3.13.2*

Python 3.13.2 (v3.13.2:4f8bb3947cf, Feb  4 2025, 11:51:10) [Clang 15.0.0 (clang-15
00.3.9.4)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> help()
Welcome to Python 3.13's help utility! If this is your first time using
Python, you should definitely check out the tutorial at
https://docs.python.org/3.13/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To get a list of available
modules, keywords, symbols, or topics, enter "modules", "keywords",
"symbols", or "topics".

Each module also comes with a one-line summary of what it does; to list
the modules whose name or summary contain a given string such as "spam",
enter "modules spam".

To quit this help utility and return to the interpreter,
enter "q", "quit" or "exit".

help> |

```

Ln: 19 Col: 0



```
>>>
```

```
>>> help()
```

Welcome to Python 3.11's help utility!

If this is your first time using Python, you should definitely check out the tutorial on the internet at <https://docs.python.org/3.11/tutorial/>.

Enter the name of any module, keyword, or topic to get help on writing Python programs and using Python modules. To quit this help utility and return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type "modules", "keywords", "symbols", or "topics". Each module also comes with a one-line summary of what it does; to list the modules whose name or summary contain a given string such as "spam", type "modules spam".

```
help> keywords
```

Here is a list of the Python keywords. Enter any keyword to get more help.

False	class	from	or
None	continue	global	pass
True	def	if	raise
and	del	import	return
as	elif	in	try
assert	else	is	while
async	except	lambda	with
await	finally	nonlocal	yield
break	for	not	

```
help> █
```



ein erstes Python-Script

```
thomas@uni-vpn229 =>
thomas@uni-vpn229 =>
thomas@uni-vpn229 => more helloworld.py
# Grundlagen Internet-Technologien
#
# PYTHON: Hello World

phrase = '\n Hello Einführung Internet-Technologien\n I like PYTHON!\n'

print (phrase)
thomas@uni-vpn229 =>
thomas@uni-vpn229 =>
```

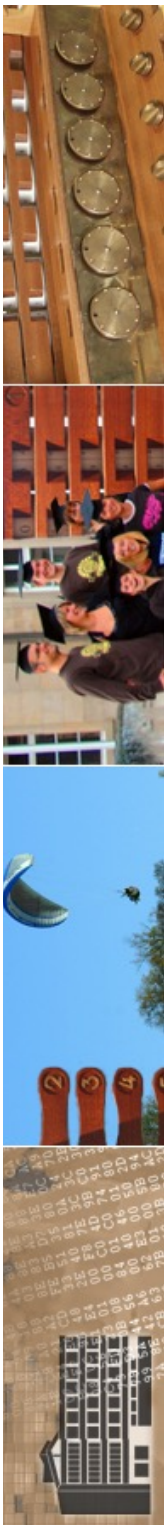




```
[thomas@uni-vpn229 =>
[thomas@uni-vpn229 =>
[thomas@uni-vpn229 =>
[thomas@uni-vpn229 => python3 helloworld.py
```

```
Hello Einführung Internet-Technologien
I like PYTHON!
```

```
[thomas@uni-vpn229 =>
[thomas@uni-vpn229 =>
[thomas@uni-vpn229 =>
thomas@uni-vpn229 => █
```





```
C:\Users\thomas\Documents\Eberhardina\InformationsDienste\Lehre\source\python>
C:\Users\thomas\Documents\Eberhardina\InformationsDienste\Lehre\source\python>py
thon helloworld.py
```

```
Hello Grundlagen Internet-Technologien
I like PYTHON!
```

```
C:\Users\thomas\Documents\Eberhardina\InformationsDienste\Lehre\source\python>
C:\Users\thomas\Documents\Eberhardina\InformationsDienste\Lehre\source\python>
```

The screenshot shows the PyDev IDE in Eclipse. The main editor window displays the contents of `helloworld.py`:

```
1 # Grundlagen Internet-Technologien
2 #
3 # PYTHON: Hello World
4
5 phrase = '\n Hello Grundlagen Internet-Technologien\n I like PYTHON!\n'
6
7 print (phrase)
```

Below the editor, the Console window shows the output of the program:

```
<terminated> C:\Users\thomas\Documents\Eberhardina\InformationsDienste\Lehre\source\python\helloworld.py

Hello Grundlagen Internet-Technologien
I like PYTHON!
```

The status bar at the bottom indicates the cursor is at line 1, column 1, in a writable state.



```
thomas@PetitMouton =>
thomas@PetitMouton => python3 helloGIT.py
```

```
Hello Grundlagen Internet-Technologien
Freedom and peace for Ukraine
```

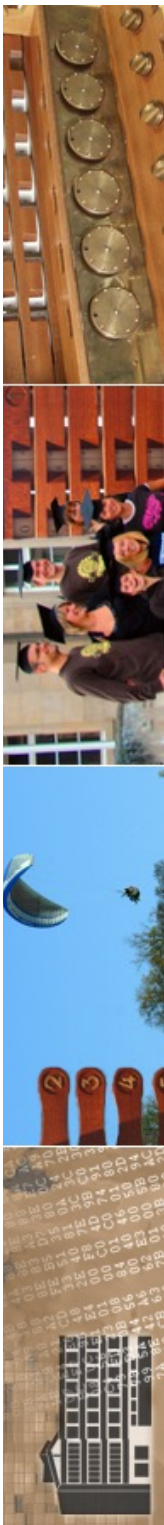
```
thomas@PetitMouton =>
thomas@PetitMouton =>
```

Beispielanwendung

- Eingabe des Radius eines Kreises, Berechnung von Fläche und Umfang

– Tastatur-Eingabe mittels

`input(prompt)`





```
thomas@uni-vpn229 =>
thomas@uni-vpn229 => more kreis.py
# Grundlagen Internet-Technologien
#
# Beispiel: Kreisberechnung von Python
#
```

```
### Tastatureingabe mit Prompt und Cast nach Gleitkomma
radius = float(input ("\n Bitte Radius eingeben: "));
```

```
pi = 3.1415926;
```

```
umfang  = 2.0 * pi * radius;
flaeche = pi * radius * radius;
```

```
print ("\n Die Flaeche des Kreises betraegt: " , flaeche , "\n");
print (" Der Umfang des Kreises betraegt: " , umfang , "\n\n");
```

```
thomas@uni-vpn229 =>
thomas@uni-vpn229 =>
thomas@uni-vpn229 => python3 kreis.py
```

```
Bitte Radius eingeben: 3
```

```
Die Flaeche des Kreises betraegt:  28.274333400000003
```

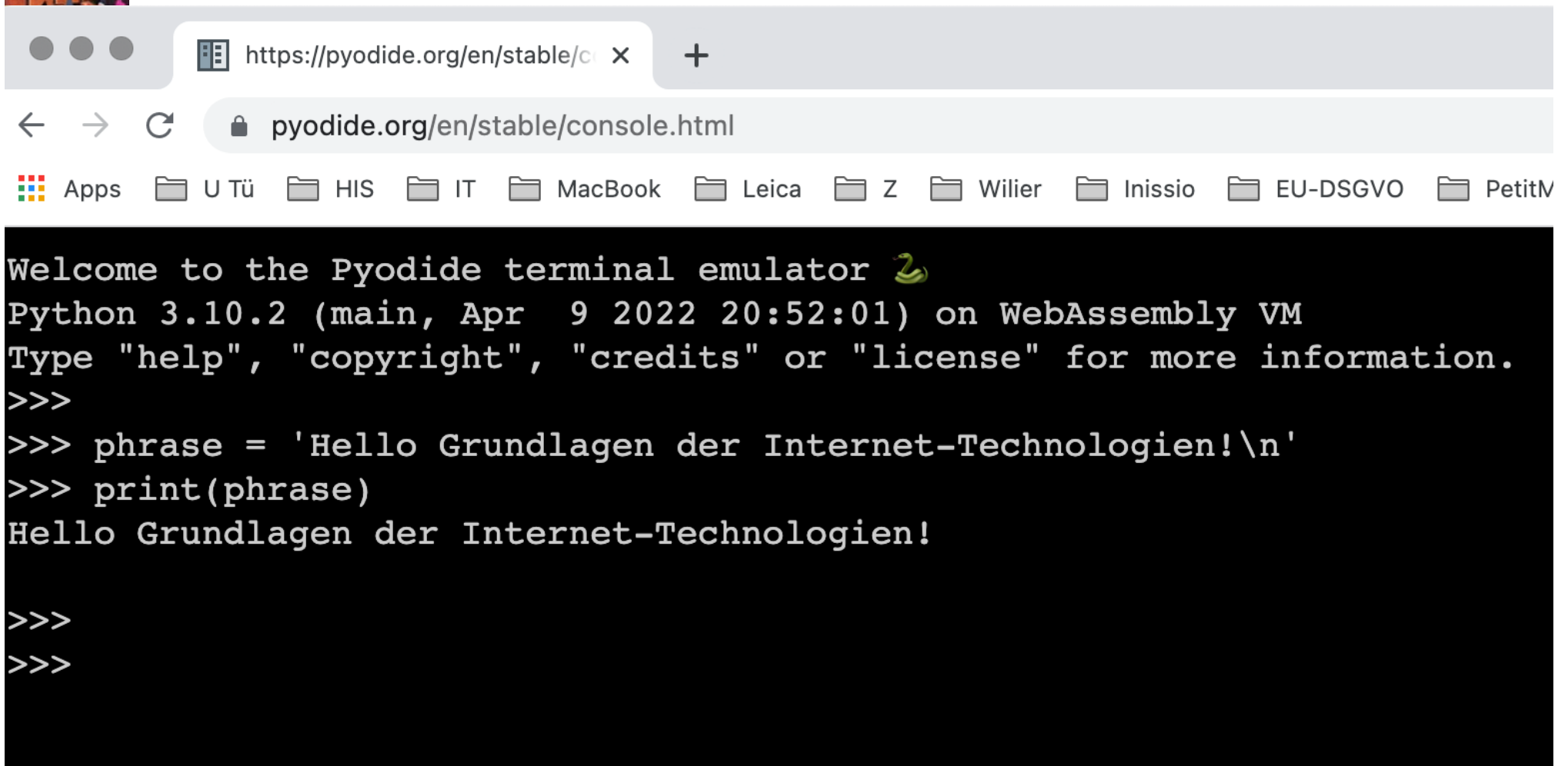
```
Der Umfang des Kreises betraegt:  18.849555600000002
```

```
thomas@uni-vpn229 =>
thomas@uni-vpn229 =>
```



Konsole im Browser

<https://pyodide.org/en/stable/>

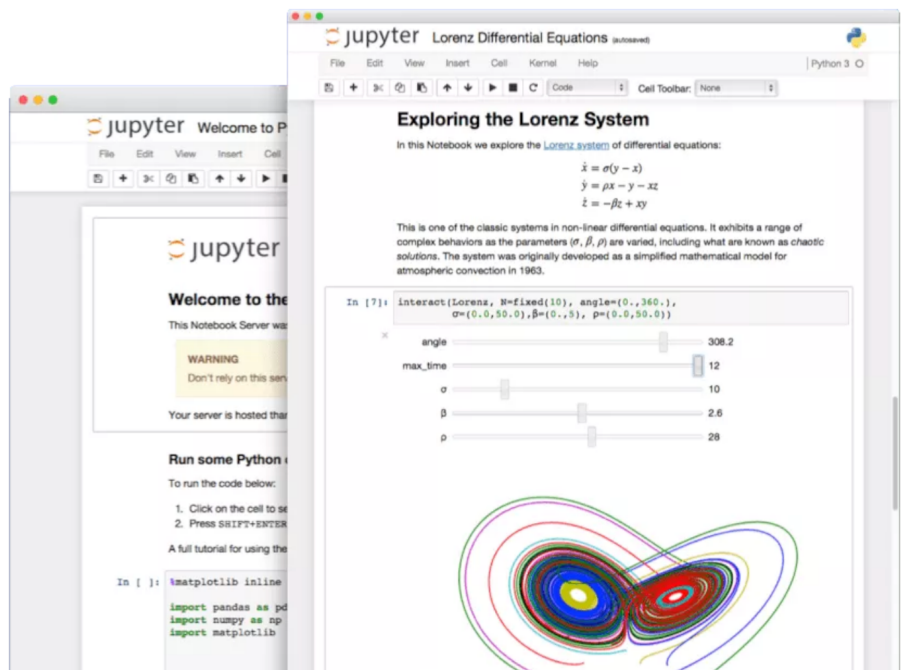




<https://jupyter.org/>

JupyterLab: A Next-Generation Notebook Interface

JupyterLab is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality.



Jupyter Notebook: The Classic Notebook Interface

The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

[Try it in your browser](#)

[Install the Notebook](#)

https://www.bwjupyter.de/



[Home](#) [Impressum](#) [Datenschutz](#) [Barrierefreiheit](#) [Sitemap](#) [KIT](#)

[Dokumentation](#) [Nutzungsbedi](#)

bwJupyter für die Lehre

Das KIT und die Uni Stuttgart entwickeln gemeinsam mit weiteren Pilothochschulen innerhalb des Dialogprozesses "Hochschulen in der digitalen Welt" des Baden-Württembergischen Ministeriums für Wissenschaft, Forschung und Kunst einen neuen Landesdienst **bwJupyter für die Lehre**.

Bei Jupyter handelt es sich um eine Webanwendung, die Nutzern den Zugang zu Rechenumgebungen und Ressourcen gewährleistet, ohne sie mit Installations- und Wartungsaufgaben zu belasten. Nachdem eine Ressource angefordert wurde, landet der Nutzer in einer Jupyterlab-Umgebung. Weitere Informationen zum Nutzen und der Funktionen des Jupyterlab Interfaces finden sie unter diesem [Link](#) .

Eine zentrale Komponente von Jupyter ist das Jupyter Notebook. Hierbei handelt es sich um ein Dokument, welches gleichzeitig formatierten Fließtext, ausführbare Code-Abschnitte und (interaktive) Visualisierungen (Bild, Ton, Video, 3D Ansichten) beinhalten kann. Weitere Informationen zum Jupyter Notebook finden sie unter diesem [Link](#) .

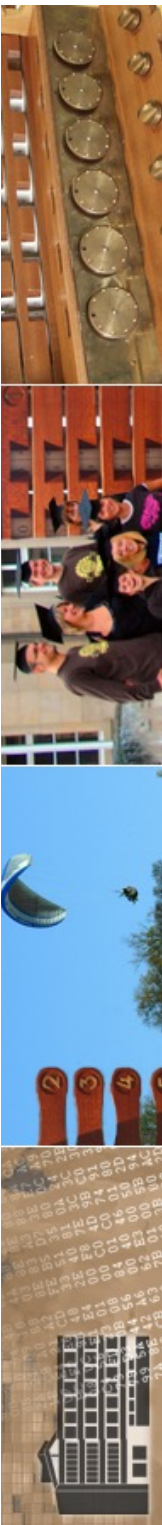
Nach einem erfolgreichen Pilotprojekt startet bwJupyter zu Beginn des Sommersemesters 2025. Der JupyterHub steht bereits ab dem **11. März** unter <https://hub.bwjupyter.de>  bereit.

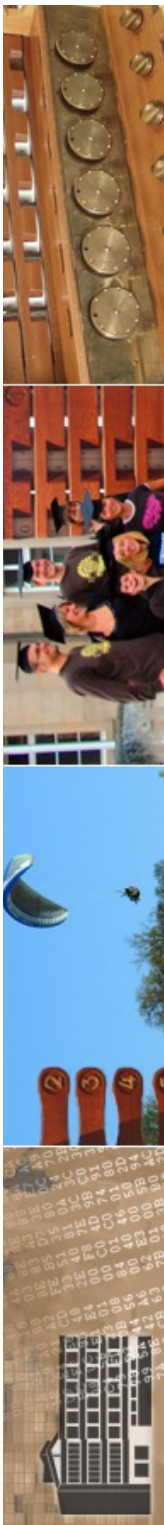
Ein- / Ausgabe

- Python liefert ein recht komfortables IO-System
- Python2
 - Methode `input('Prompt')`
 - Methode `raw_input('Prompt')` # Zeichenk.
- Python3
 - Methode `eval('Prompt')`
 - Methode `input('Prompt')` # Zeichenkette

Arbeitsweise des Interpreters

- Der Python-Interpreter arbeitet in 4 Schritten (vergleichbar dem PERL-»Interpreter«):
 - Einlesen der Datei (bzw. interaktive Eingabe) als 7-bit-ASCII-Strom
 - Lexikalische Analyse
 - Syntaktische Analyse durch Parser, Übersetzung in Bytecode
 - Interpretation des Bytecodes





```
thomas@uni-vpn229 =>
thomas@uni-vpn229 => python3 -h
usage: /Library/Frameworks/Python.framework/Versions/3.9/bin/python3 [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
-b      : issue warnings about str(bytes_instance), str(bytearray_instance)
         and comparing bytes/bytearray with str. (-bb: issue errors)
-B      : don't write .pyc files on import; also PYTHONDONTWRITEBYTECODE=x
-c cmd  : program passed in as string (terminates option list)
-d      : turn on parser debugging output (for experts only, only works on
         debug builds); also PYTHONDEBUG=x
-E      : ignore PYTHON* environment variables (such as PYTHONPATH)
-h      : print this help message and exit (also --help)
-i      : inspect interactively after running script; forces a prompt even
         if stdin does not appear to be a terminal; also PYTHONINSPECT=x
-I      : isolate Python from the user's environment (implies -E and -s)
-m mod  : run library module as a script (terminates option list)
-O      : remove assert and __debug__-dependent statements; add .opt-1 before
         .pyc extension; also PYTHONOPTIMIZE=x
-OO     : do -O changes and also discard docstrings; add .opt-2 before
         .pyc extension
-q      : don't print version and copyright messages on interactive startup
-s      : don't add user site directory to sys.path; also PYTHONNOUSERSITE
-S      : don't imply 'import site' on initialization
-u      : force the stdout and stderr streams to be unbuffered;
         this option has no effect on stdin; also PYTHONUNBUFFERED=x
-v      : verbose (trace import statements); also PYTHONVERBOSE=x
         can be supplied multiple times to increase verbosity
-V      : print the Python version number and exit (also --version)
         when given twice, print more information about the build
-W arg  : warning control; arg is action:message:category:module:lineno
         also PYTHONWARNINGS=arg
-x      : skip first line of source, allowing use of non-Unix forms of #!cmd
-X opt  : set implementation-specific option. The following options are available:

-X faulthandler: enable faulthandler
-X oldparser: enable the traditional LL(1) parser; also PYTHONOLDPARSER
-X showrefcount: output the total reference count and number of used
         memory blocks when the program finishes or after each statement in the
         interactive interpreter. This only works on debug builds
-X tracemalloc: start tracing Python memory allocations using the
         tracemalloc module. By default, only the most recent frame is stored in a
         traceback of a trace. Use -X tracemalloc=NFRAME to start tracing with a
         traceback limit of NFRAME frames
-X importtime: show how long each import takes. It shows module name,
         cumulative time (including nested imports) and self time (excluding
         nested imports). Note that its output may be broken in multi-threaded
         application. Typical usage is python3 -X importtime -c 'import asyncio'
-X dev: enable CPython's "development mode", introducing additional runtime
         checks which are too expensive to be enabled by default. Effect of the
         developer mode:
         * Add default warning filter, as -W default
         * Install debug hooks on memory allocators: see the PyMem_SetupDebugHooks() C function
         * Enable the faulthandler module to dump the Python traceback on a crash
         * Enable asyncio debug mode
         * Set the dev_mode attribute of sys.flags to True
         * io.IOBase destructor logs close() exceptions
-X utf8: enable UTF-8 mode for operating system interfaces, overriding the default
         locale-aware mode. -X utf8=0 explicitly disables UTF-8 mode (even when it would
         otherwise activate automatically)
-X pycache_prefix=PATH: enable writing .pyc files to a parallel tree rooted at the
         given directory instead of to the code tree
```



Python Bytecode

- Python speichert den Bytecode in einer Datei mit **Suffix .pyc**
 - Bytecode von hello.py in Datei hello.pyc
- Python-Bytecode ist *plattformunabhängig*
- Python Bytecode ist (i.a.) temporär
- Python-Module können als Bytecode verteilt werden
 - Performance-Gewinn, da Compilation entfällt
 - Quellen müssen nicht offengelegt werden
 - Vorteil gegenüber PERL und vielen anderen Scriptsprachen
 - Analogie: Java-class-Datei: Bytecode für die JVM





```
[thomas@PetitMouton-2 =>
[thomas@PetitMouton-2 => python3
Python 3.8.3 (v3.8.3:6f8c8320e9, May 13 2020, 16:29:34)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
>>> import py_compile
>>> py_compile.compile('WarenKorb.py')
'__pycache__/WarenKorb.cpython-38.pyc'
>>>
```





Table of Contents

What's New In Python 3.11

- Summary – Release highlights
- New Features
 - PEP 657: Fine-grained error locations in tracebacks
 - PEP 654: Exception Groups and `except*`
 - PEP 678: Exceptions can be enriched with notes
 - Windows `py.exe` launcher improvements
- New Features Related to Type Hints
 - PEP 646: Variadic generics
 - PEP 655: Marking individual `TypedDict` items as required or not-required
 - PEP 673: `Self` type
 - PEP 675: Arbitrary literal string type
 - PEP 681: Data class transforms
 - PEP 563 may not

Faster CPython

CPython 3.11 is an average of **25% faster** than CPython 3.10 as measured with the [pyperformance](#) benchmark suite, when compiled with GCC on Ubuntu Linux. Depending on your workload, the overall speedup could be 10–60%.

This project focuses on two major areas in Python: [Faster Startup](#) and [Faster Runtime](#). Optimizations not covered by this project are listed separately under [Optimizations](#).

Faster Startup

Frozen imports / Static code objects

Python caches [bytecode](#) in the `__pycache__` directory to speed up module loading.

Previously in 3.10, Python module execution looked like this:

```
Read __pycache__ -> Unmarshal -> Heap allocated code object -> Evaluate
```

In Python 3.11, the core modules essential for Python startup are “frozen”. This means that their [Code Objects](#) (and bytecode) are statically allocated by the interpreter. This reduces the steps in module execution process to:

```
Statically allocated code object -> Evaluate
```

Interpreter startup is now 10–15% faster in Python 3.11. This has a big impact for short-running programs using Python.

(Contributed by Eric Snow, Guido van Rossum and Kumar Aditya in many issues.)

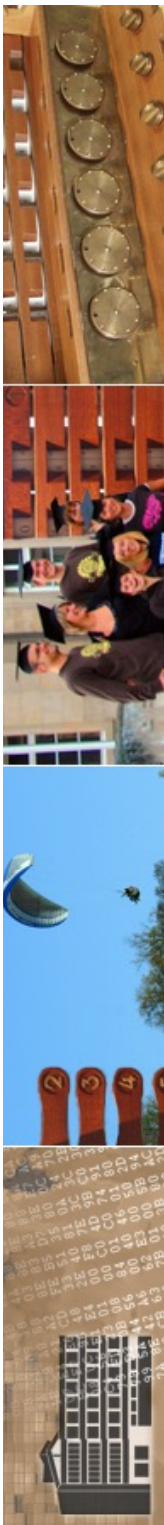
das Semikolon in Python

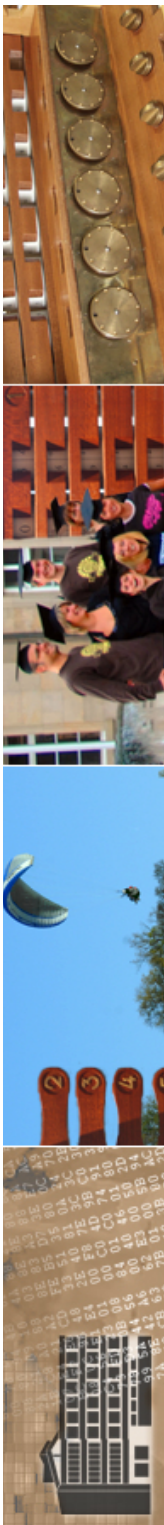
- das Semikolon ; ist in Python - im Gegensatz zu den meisten anderen Sprachen - *nicht* zum Abschluß einer Anweisung *notwendig*, es genügt das normale Zeilenende
- man kann es aber verwenden
- dafür müssen Anweisungen, die über mehrere Zeilen laufen, am Zeilenende durch ein \ zusammengefügt werden
- mit ; auch mehrere Anweisungen in einer Zeile



Kommentare in Python

- Kommentare können (wie etwa in PERL auch) durch `#` eingeleitet werden
- sie schaden nie...
- es sind immer zu wenig... ; -)
- die “üblichen” Kommentare `//` und `/* */` gibt es nicht, auch nicht in python3





numerische Literale

– Python3 unterscheidet *drei* (vier) Datentypen

- ganze Zahlen (beliebig groß)
42, 4711, 0o32 (oktal), 0x1a2 (hexadeximal)
- Gleitkommazahlen
3.1415, 5E24, 1.902e-19
- imaginäre/komplexe Zahlen (imaginary)
imaginär: 3.14j, -2j
komplex: 2 + 3j
- bool
True oder False

das j kommt von
der

Elektrotechnik

```
[>>>
[>>> c1 = 3 - 4j
[>>> c2 = 5 + 6j
[>>>
[>>> c1 * c2
(39-2j)
[>>>
[>>> c1 / c2
(-0.14754098360655735-0.6229508196721312j)
[>>>
```

hexal, oktal, binär

- in Python3 wird verwendet

- hexadezimal

0x1a2b

- oktal

0o1234

- binär

0b10101011

```
[>>>
[>>> a = 0b1010101
[>>> a
85
[>>>
[>>> b = 00717
[>>> b
463
[>>>
[>>> c = 0xCAFE
[>>> c
51966
[>>>
```

die Division

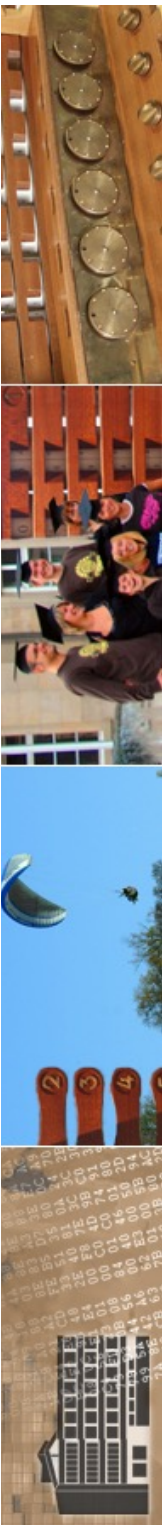
- bei zwei Ganzzahlargumenten
 - in Python2: / bedeutet div
 - in Python3:
Gleitkommadivision
div durch //

```
>>>
>>> 10/3
3.3333333333333335
>>>
>>> 10//3
3
>>>
```

Zeichenketten in Python

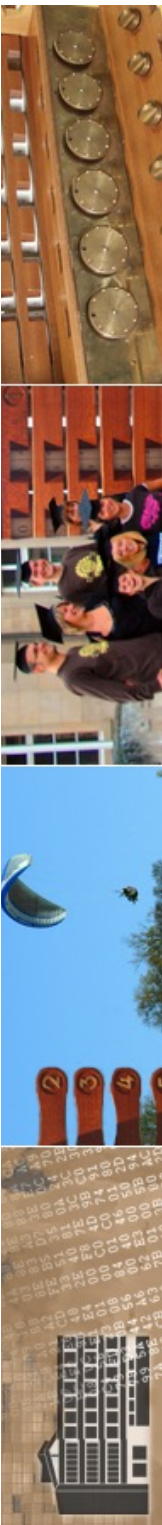
- Zeichenkettenlitterale

- kurze Zeichenkettenlitterale werden durch ' oder " begrenzt
- lange Zeichenkettenlitterale können über mehrere Zeilen reichen; sie werden durch dreifache Folgen von ' oder " begrenzt
- raw-Strings: `r"Zeilenumbruch: \n"`
- Escape-Sequenzen: wie gewohnt
 - `\\`
 - `\'`
 - `\"`
 - `\n`
 - `\a` (Ton)
 - `\t` (Tabulator)



Zeichenketten

- Verbinden von Zeichenketten mit +
- Python ist Unicode-fähig
- wie in Java Escape-Seqzenz \u
 - Unicode-String muss mit vorangestelltem „u“ gekennzeichnet werden





```
thomas@uni-vpn229 =>
thomas@uni-vpn229 => more unicode.py
# Grundlagen Internet-Technologien
#
# Beispiel: Unicode in Python

zeichenkette = u'100 \u20AC'

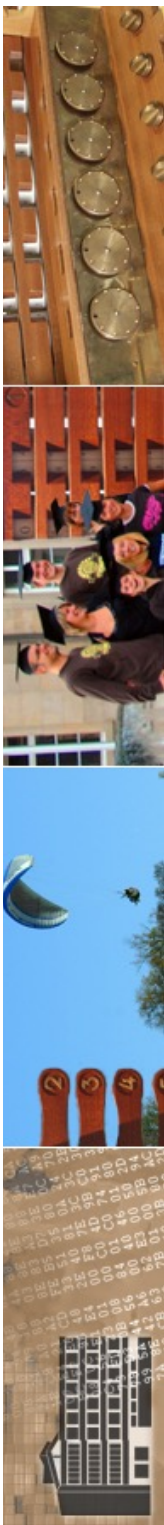
print ("\n" + zeichenkette+ "\n")
thomas@uni-vpn229 =>
thomas@uni-vpn229 =>
thomas@uni-vpn229 =>
thomas@uni-vpn229 => python3 unicode.py

100 €

thomas@uni-vpn229 =>
thomas@uni-vpn229 =>
```

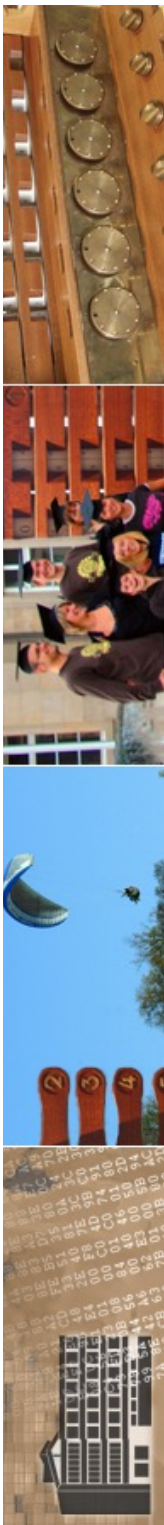
Zeichen

- ein Datentyp `char` ist in Python nicht implementiert
- sei `s` eine Zeichenkette der Länge `n`; dann ist
 - `s[i]`
 - für $i = 0, 1, \dots, n-1$ eine Zeichenkette der Länge 1, welche das Zeichen an der Position `i` der Zeichenkette `s` enthält
 - `s[i:j]`
 - die Teilzeichenkette von der Position `i` bis zur Position `j - 1`
 - `s[i:]` und `s[:j]`
 - die Teilzeichenketten ab der Position `i` und bis zur Position `j - 1`



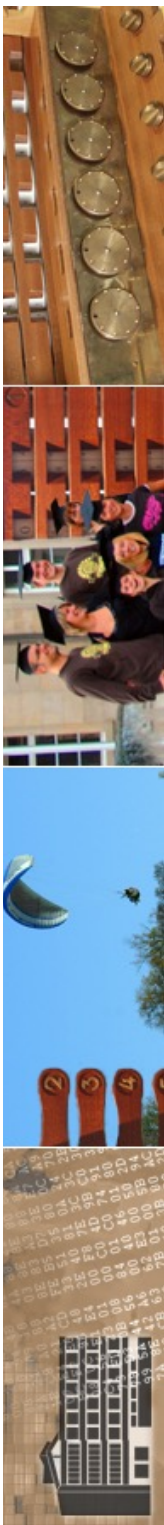
Datentypen in Python3

- Python ist (im Gegensatz zu den meisten anderen Scriptsprachen) objektorientiert
- Bezeichner sind Referenzen auf Objekte
- Datentype None
- Numerische Variablen:
 - integer - beliebige Größe
 - float
 - complex



Datentypen in Python3

- strukturierte Daten:
 - Listen (veränderbar)
 - Strings (nicht veränderbar)
 - assoziative Listen: Dictionary
- aufrufbare Typen
 - Funktionen & Methoden
 - Klassen
 - Klasseninstanzen
- andere Typen
 - Module
 - Klassen



arithmetische Ausdrücke

- wie »gewohnt«:

- $a + b$ $a - b$ $a * b$ a / b
 $a // b$
- $a ** b$ für die Potenz a^b
- $\text{abs}(a)$ für Betrag $|a|$
- $\text{int}(a)$ $\text{float}(a)$ $\text{long}(a)$:
 Typumwandlung
- $\text{complex}(a, b) = a + j b$
- nachgestelltes L für long: $a = 4711L$;
- $\text{hex}(i)$ $\text{oct}(i)$ $\text{str}(o)$



```
thomas@PetitMouton-2 =>
thomas@PetitMouton-2 => more math.py
# Grundlagen Internet-Technologien
#
# Verwendung der Python-Module math und cmath
# (math: mathematische Funktionen fuer R, cmath: mathematische Funktionen fuer C)

### Importe
###import math
import cmath

### Berechnung von exp(x)
eingabe = float(input('\n Bitte x zur Berechnung von e^x eingeben: '))
ergebnis1 = cmath.exp(eingabe)
print("\n exp(x) = "+str(ergebnis1))

### Berechnung der komplexen Exponentialfunktion
### nach Euler:
###  $e^{(x + iy)} = e^x * (\cos y + i \sin y)$ 

eingaber = float(input("\n Bitte Realteil von z zur Berechnung von e^z eingeben: "))
eingabei = float(input("\n Bitte Imaginaerteil von z zur Berechnung von e^z eingeben: "))
z = complex(eingaber, eingabei)
ergebnis2 = cmath.exp(z)
print("\n exp(z) = "+str(ergebnis2));
thomas@PetitMouton-2 =>
thomas@PetitMouton-2 =>
```





```
thomas@PetitMouton-2 =>
thomas@PetitMouton-2 =>
thomas@PetitMouton-2 => python3 math.py
```

Bitte x zur Berechnung von e^x eingeben: 2

$\exp(x) = (7.38905609893065+0j)$

Bitte Realteil von z zur Berechnung von e^z eingeben: 2

Bitte Imaginarteil von z zur Berechnung von e^z eingeben: -1

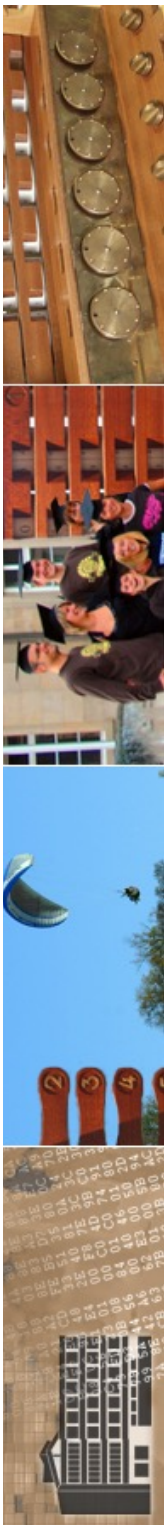
$\exp(z) = (3.992324048441272-6.217676312367968j)$

```
thomas@PetitMouton-2 =>
thomas@PetitMouton-2 =>
```

logische Ausdrücke

- ebenfalls wie üblich in Scriptsprachen gewohnt
- 0 entspricht false, alles andere entspricht true
 - $a < b$ $a > b$ $a == b$ $a != b$
 - and, or, not
 - jeder von 0 verschiedene Zahlenwert entspricht true
 - None und die leere Zeichenkette sind false, alle anderen true

$123 < 21$ false, aber $"123" < "21"$ true





Listen

- Python verfügt wie üblich für Scriptsprachen über einen lässigen Umgang mit Arrays, die direkt als Listen implementiert sind

```
– a = [1, 2, 3, 4]
```

```
– len (a)          ###          4
```

```
– a [1] = 'zwei' ;  ###          [1, 'zwei', 3, 4]
```

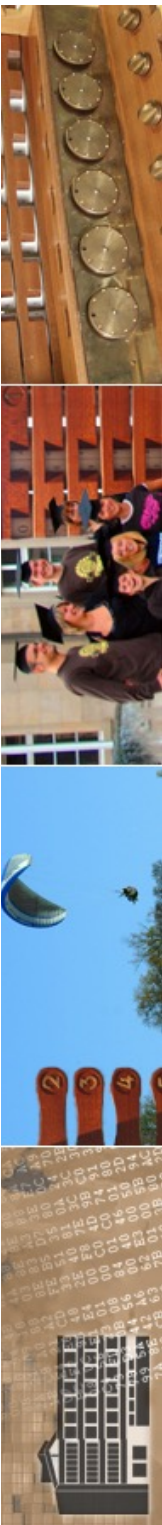


```
Python 3.8.3 (v3.8.3:6f8c8320e9, May 13 2020, 16:29:34)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information
>>>
>>>
>>>
>>>
>>> liste = [1, 2, 3, "Hello GIT", 3 - 4j]
>>> print(liste)
[1, 2, 3, 'Hello GIT', (3-4j)]
>>> print(liste[3])
Hello GIT
>>>
```



Listen

- im Gegensatz zu Arrays können Python-Listen erweitert werden:
 - `a.append('fünf')` `### push`
- weitere Listenoperation
 - `a.pop()` : Gibt zuletzt abgelegtes Element zurück
- es stehen somit pop und push zur Verfügung
 - ➔ LIFO





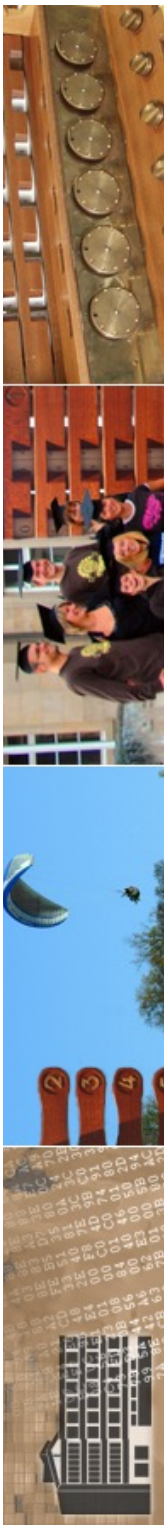
```
>>>
>>> liste.append("Tuebingen")
>>>
>>>
>>> print(liste)
[1, 2, 3, 'Hello GIT', (3-4j), 'Tuebingen']
>>>
>>> liste.append("Corona")
>>>
>>> liste.append("Corona")
>>>
>>>
>>> print(liste)
[1, 2, 3, 'Hello GIT', (3-4j), 'Tuebingen', 'Corona', 'Corona']
>>>
```





mehrdimensionale Listen

- ein Listenelement selbst kann auch wieder eine Liste sein → mehrdimensionale Listen





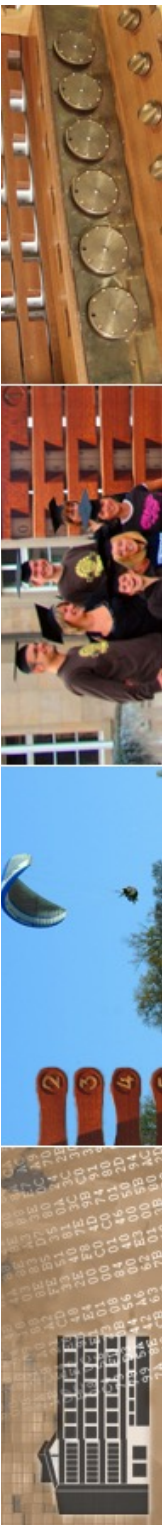
```
>>>
>>>
>>>
>>> dim2 = ["Medizininformatik", "SARS-CoV2"], 4711, 313, liste]
>>>
>>> print(dim2)
[['Medizininformatik', 'SARS-CoV2'], 4711, 313, [1, 2, 3, 'Hello GIT', (3-4j),
'Tuebingen', 'Corona', 'Corona']]
>>>
>>>
```





Methoden für Listen

- es gibt die „üblichen“ Methoden für Listen wie
 - `reverse()`
 - `sort()`
 - ...



assoziative Listen (Dictionary)

- Syntax:
 - `hash = { key1:value1, key2:value2, ... , keyN:valueN }`
- Methoden
 - `items ()` : Liste mit 2-Tupeln
 - `keys ()` : Liste der Keys
 - `values ()` : Liste der Values
 - `has_key (k)` : Überprüft, ob Key k vorhanden
 - `update (hash2)` : fügt hash2 an



Dictionary

```
>>>
>>>
>>> hash = {'tuebingen': 'Medizininformatik', 'stuttgart': 'Medizintechnik',
313:4711}
>>>
>>> print(hash)
{'tuebingen': 'Medizininformatik', 'stuttgart': 'Medizintechnik', 313: 4711}
>>>
>>> print(hash['tuebingen'])
Medizininformatik
>>>
>>>
```





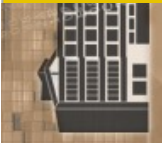
```
thomas@uni-vpn229 =>
thomas@uni-vpn229 => more assozListe.py
# Grundlagen Internet-Technologien
#
# Beispiel: assoziative Liste mit Python

hashBeispiel = {"liebling": "MediaFotografie", "auchGut": "Web-Kompendium"}
[
### erweiterung des Hash
hashBeispiel["klassiker"] = "Brecht: Dreigroschenoper"

print ("\n Anzahl der Elemente in Liste: ", len(hashBeispiel))
print ("\n ein Element: ", hashBeispiel["liebling"])
print ("\n Die komplette Liste: ", hashBeispiel)
thomas@uni-vpn229 =>
thomas@uni-vpn229 =>
thomas@uni-vpn229 => python3 assozListe.py
[
Anzahl der Elemente in Liste:  3

ein Element:  MediaFotografie

Die komplette Liste:  {'liebling': 'MediaFotografie', 'auchGut': 'Web-Kompendium',
'klassiker': 'Brecht: Dreigroschenoper'}
thomas@uni-vpn229 =>
thomas@uni-vpn229 =>
```





Mengen

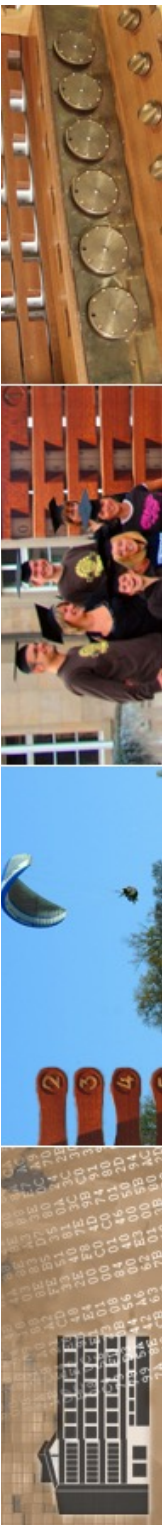
- in Python Datenstruktur „Menge“
(Datentyp set)
 - set enthält jedes Element nur ein mal

```
>>>
[>>> mySet = {"Informatik", "Medizininformatik", "Bioinformatik", "Medieninformatik"}
[>>>
[>>> print(mySet)
{'Bioinformatik', 'Informatik', 'Medizininformatik', 'Medieninformatik'}
[>>>
[>>> "BWL" in mySet
False
[>>>
[>>> "Medizininformatik" in mySet
True
[>>>
[>>>
```



Mengen

- zahlreiche Operationen
 - $\&$: Schnittmenge
 - $|$: Vereinigungsmenge
 - in : Element in Menge enthalten?
 - $\leq$: Teilmenge
 - $<$: echte Teilmenge





```
>>>
>>>
>>> myOtherSet = {"Medizin", "Medizintechnik", "Medizininformatik"}
>>>
>>>
>>> mySet & myOtherSet
{'Medizininformatik'}
>>>
>>> mySet | myOtherSet
{'Informatik', 'Medieninformatik', 'Medizintechnik', 'Bioinformatik', 'Medizininformatik', 'Medizin'}
>>>
>>> {"Informatik"} <= mySet
True
>>>
>>>
```



noch zwei Methoden

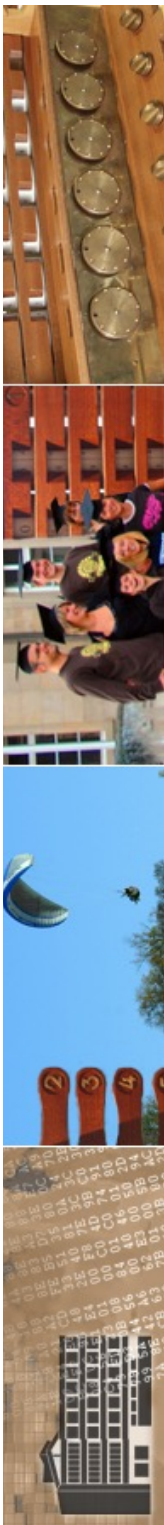
- **type** : gibt Datentyp zurück
 - `>>> type ("Hello World")`
`<type 'string'>`

- **in** : Überprüft, ob Element in Struktur vorhanden ist

– `>>> if "zwei" in a: print "ja"`

ja

```
>>>
>>>
>>>
>>> type(mySet)
<class 'set'>
>>>
>>> z = 1 - 2j
>>> type(z)
<class 'complex'>
>>>
>>>
>>>
```



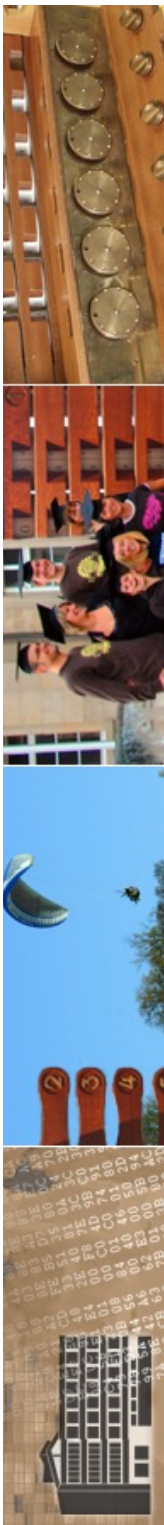
Kontrollstrukturen

- Blöcke (sequentielle Komposition)
 - Blöcke werden in Python nicht wie üblich gebildet:

Blöcke werden durch *einfaches Einrücken* gebildet!

“Sequenzkopf” am Anfang

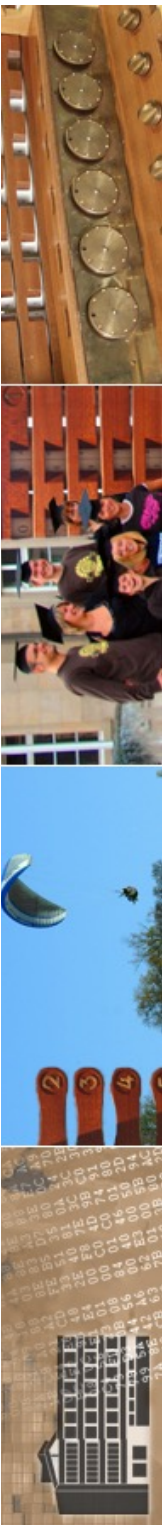
Vorsicht beim Konvertieren Windows <-> Linux





Entscheidung

- Kontrollstruktur
 - `if (boole) :`
 `wahr_anweisungen`
 `else :`
 `falsch_anweisungen`
 - es gibt noch zusätzlich `elif (bool2)`





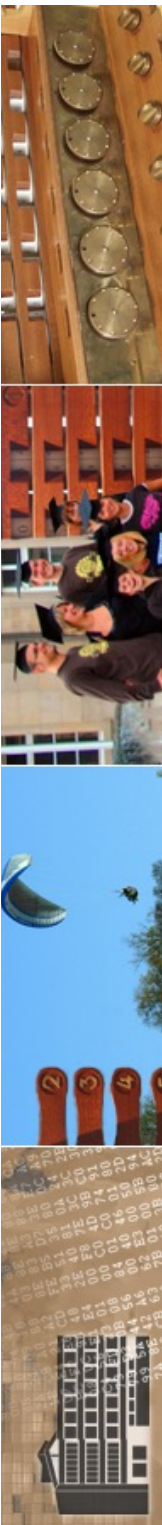
Entscheidung

```
>>>
>>> x = 42
>>> y = 4711
>>>
>>> if (x < y) :
...     print("x ist kleiner als y")
... else :
...     print ("x ist groessergleich y")
...     y = x
...
x ist kleiner als y
>>>
>>>
>>>
```

Schleifen

- die klassische Schleife: `while`
 - Syntax »wie zu erwarten«:

```
while (boole) :  
    anweisungen;
```

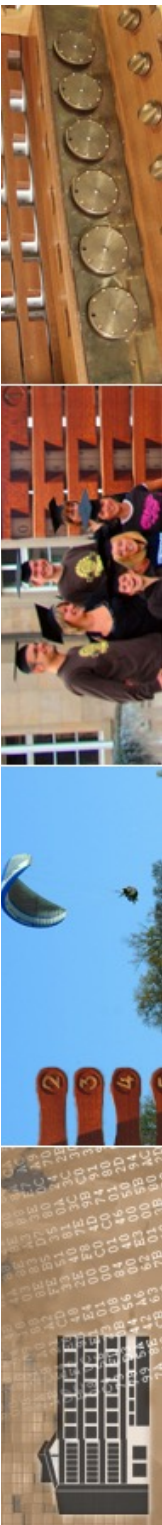




while-Schleife

```
maximum = 100

summe = 0; i = 1;
while (i <= maximum):
    summe += i
    i+=1
print ("Summation mit while: ", summe,"\n")
```





while-Schleife im Browser

```

>>>
>>>
>>>
>>> maximum = 100
>>> summe = 0; i = 0 ### nicht notwendig, aber sinnvoll
>>>
>>> while (i <= summe):
...     summe += i
...     i += 1
...
>>> print("Summation bis 100 mit while: ", summe)
Summation bis 100 mit while:  0
>>>
>>>
>>>
>>>

```

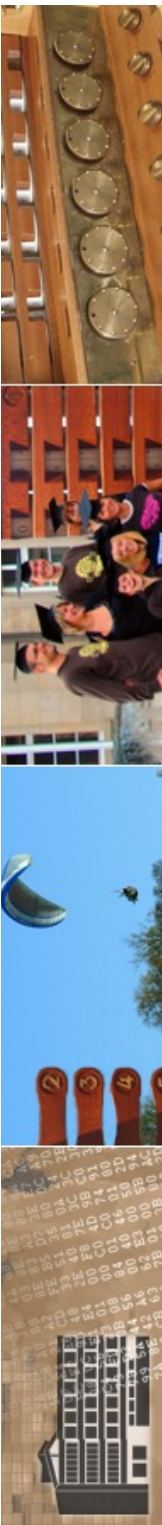
Schleifen

- for hat ebenfalls eigene Implementierung

– Syntax:

```
for variable in sammlung :  
    anweisungen
```

(liste wird mit steigendem Index
durchlaufen)



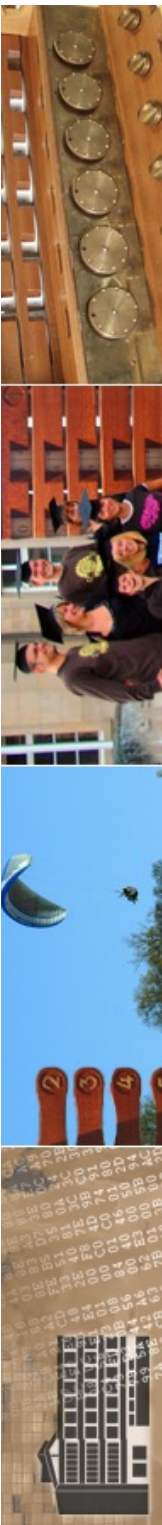


for-Schleife

```

summe = 0
for i in list(range(1,101)):
    summe += i;
print ("Summation mit for:  ", summe,"\n")
[thomas@PetitMouton-2 =>
[thomas@PetitMouton-2 =>

```





```
[thomas@uni-vpn229 => more schleifen.py
# Grundlagen Internet-Technologien
#
# Beispiel: Schleifen in Python

maximum = 100

summe = 0; i = 1;
while (i <= maximum):
    summe += i
    i+=1
print ("Summation mit while: ", summe,"\n")

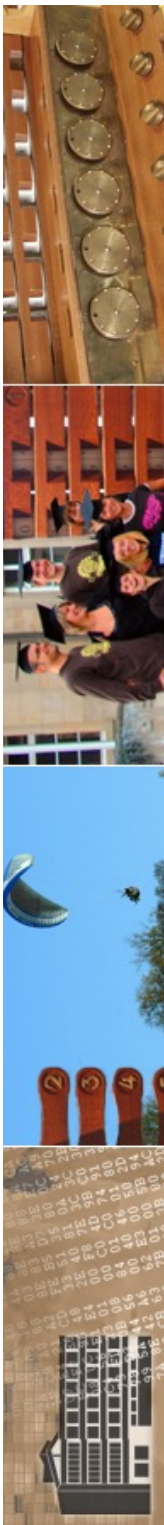
summe = 0
for i in list(range(1,101)):
    summe += i;
print ("Summation mit for:   ", summe,"\n")
[thomas@uni-vpn229 =>
```



```
[thomas@PetitMouton =>
[thomas@PetitMouton =>
[thomas@PetitMouton => python3 schleifen.py
Summation mit while: 5050

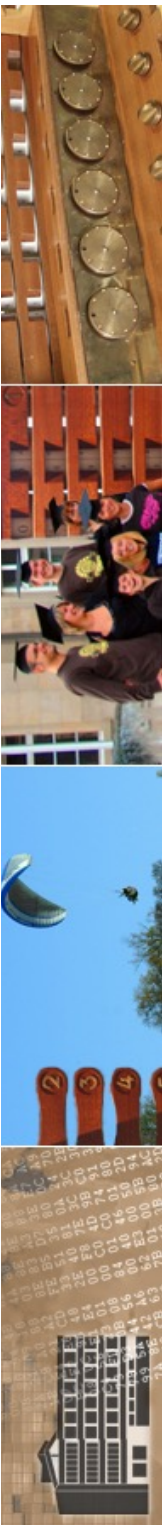
Summation mit for: 5050

[thomas@PetitMouton =>
[thomas@PetitMouton =>
```



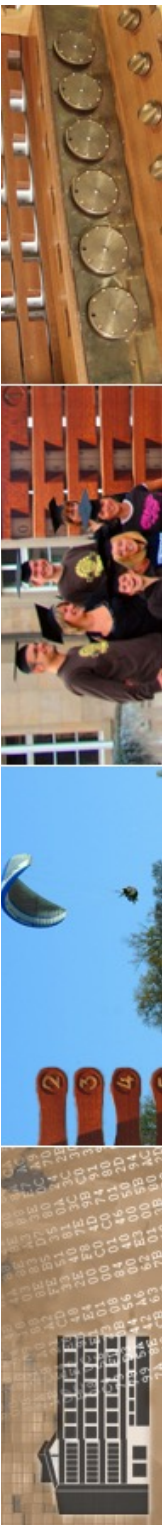
praktisch: range

- Erzeugen einer liste mit
 - `range (ende)`
 - `range (start, ende)`
 - `range (start, ende, step)`



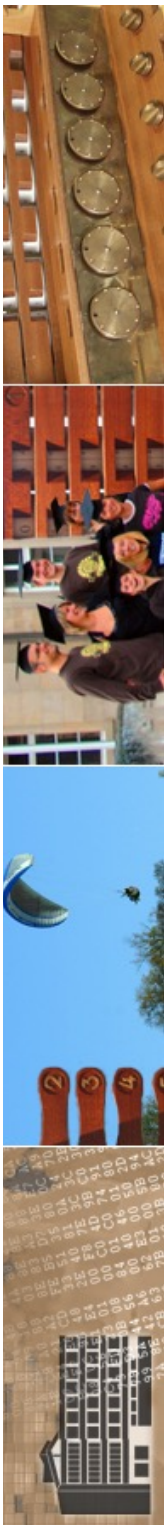
break und continue

- ...diese sind nun wieder “wie üblich” (Java):
 - break verlässt die Schleife
 - continue startet mit neuem Schleifendurchlauf



Methoden

- Funktionen werden - moderner als in PERL - folgendermaßen deklariert:
 - `def funktionsname (argument1, ...) :`
 `anweisungen`
 - im Gegensatz zu PERL und php muss eine Funktion vor der ersten Verwendung bereits definiert sein





```
thomas@uni-vpn229 =>
thomas@uni-vpn229 => more fakultaet.py
# Grundlagen Internet-Technologien
#
# Beispiel: Methoden in Python

#####
def fakultaet(n) :
    f = 1
    while (n > 1) :
        f *= n
        n -= 1
    return f
#####

arg = int(input ("\n Bitte n eingeben: "));    ### Eingabe

fakultaet = fakultaet (arg)

print ("\n Die Fakultäet von n betraegt: ", fakultaet, "\n")
thomas@uni-vpn229 =>
```



```
thomas@uni-vpn229 =>
thomas@uni-vpn229 => python3 fakultaet.py
```

Bitte n eingeben: 5

Die Fakultät von n beträgt: 120

```
thomas@uni-vpn229 =>
thomas@uni-vpn229 =>
```

```
[thomas@uni-vpn229 =>
[thomas@uni-vpn229 => python3 fakultaet.py
```

Bitte n eingeben: 25

Die Fakultät von n beträgt: 15511210043330985984000000

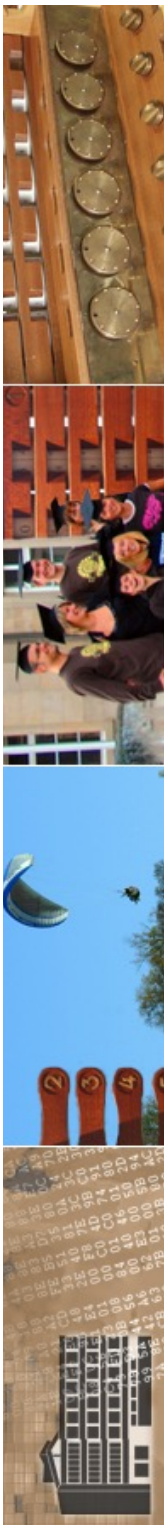
```
[thomas@uni-vpn229 =>
```

Methoden mit optionalen Parameter

- ein „Überladen“ durch Angabe eines Default-Parameters ist in Python möglich
 - `def funktionsname (argument1=arg1, ...)` :
 anweisungen
 - `def seiNett (arg1 = "Frau Bundeskanzler!")` :
 `print(" Guten Tag ", arg1)`

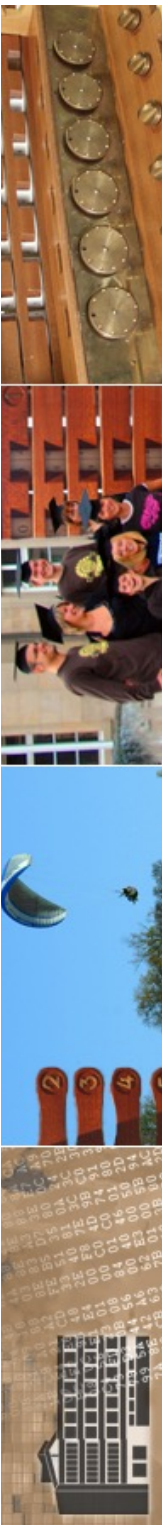
Namensräume in Python

- es sind drei verschiedene Namensräume vorhanden:
 - lokal: gültig innerhalb eines Blockes
 - Modulweit: gültig in einem Modul
 - Python-Systemvariablen: überall gültig



Namensräume

- `global` deklariert Variable als globale Variable (modulweit)
- `globals()` gibt Übersicht (als Hash) über die verwendeten globalen Variablen und ihre Belegung



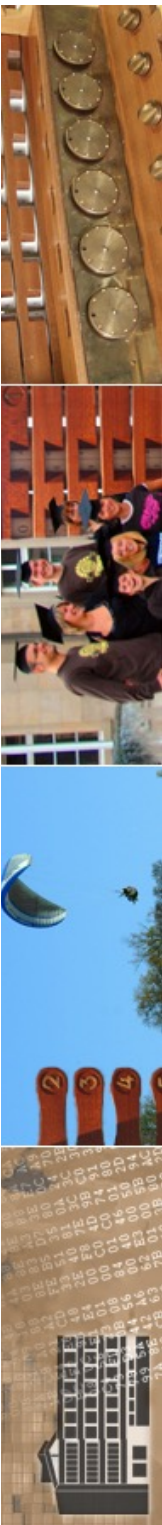


```
thomas@uni-vpn229 =>
thomas@uni-vpn229 => python3
Python 3.9.5 (v3.9.5:0a7dcdbdb13, May  3 2021, 13:17:02)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
>>> globals()
{'__name__': '__main__', '__doc__': None, '__package__': None, '__loader__':
<class '_frozen_importlib.BuiltinImporter'>, '__spec__': None, '__annotations__': {}, '__builtins__': <module 'builtins' (built-in)>}
```



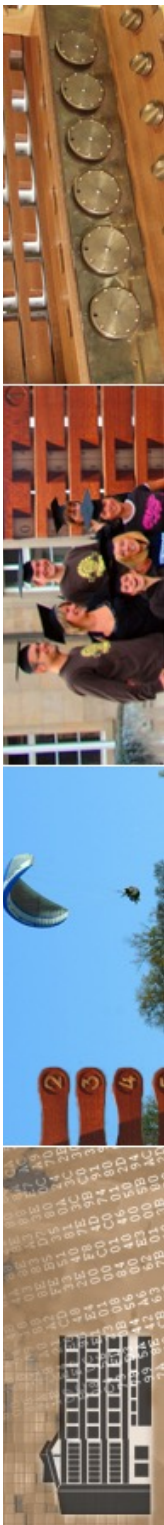
Dateien

- mittels
 - `open (datei, attribute)`
- wird eine Datei geöffnet
- Mögliche Attribute:
 - `r` : Lesen
 - `w` : Schreiben
 - `a` : Anfügen
 - `r+`
 - `w+`
 - `a+`



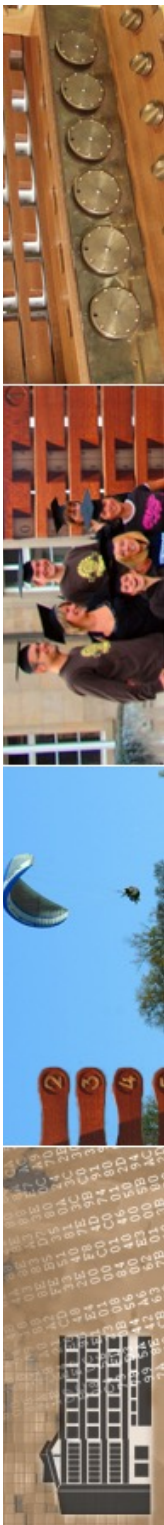
Dateien

- `open` gibt - wie üblich - einen File-Handler zurück:
- `f = open ('text.txt', 'r')`
- Methoden zum Lesen:
 - `f.readline()` : Liest eine Zeile
 - `f.readlines()` : Liest alle Zeilen (als Liste)
 - `f.read(n)` : Liest n Zeichen (ohne n ganze Datei)



Dateien

- Methoden zum Schreiben
 - `f.write (String)`
 - `f.writelines (ListeVonStrings)`
- Buffered Streams
 - Python verwendet prinzipiell buffered Streams
 - mittels `f.flush ()` wird Puffer geschrieben
- Datei schließen
 - `f.close ()` schließt Datei (und leert Buffer)



Fehlerbehandlung in Python

- Python verwendet wie Java **Exceptions**

- es gibt try - catch - Blöcke in Python
 - auch mit else (!) und finally kombinierbar

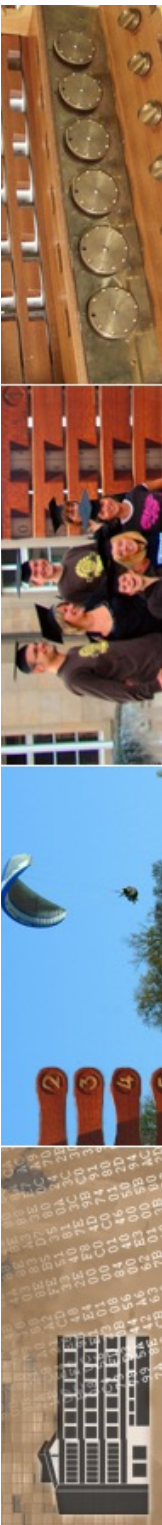
- Auslösen von Ausnahmen:

`raise exception , argument`



Objektorientierung

- Python ist - natürlich - objektorientiert
- OO in Python geht weiter als in PERL, php und insbesondere JavaScript:
 - Datenkapselung möglich
 - Vererbung möglich (sogar Mehrfachvererbung)
 - Konstruktor und Destruktor vorhanden





Beispiel Warenkorb

- ```
class Warenkorb :
 def __init__ (self, inhalt=None) :
 self.inhalt = []
 if inhalt :
 self.inhalt.append (inhalt)
 def add (self, sache) :
 self.inhalt.append (sache)

 def show (self) :
 for element in self.inhalt :
 print (element)
```



```
[thomas@uni-vpn229 =>
[thomas@uni-vpn229 => more WarenKorb.py
Grundlagen Internet-Technologien
#
Beispiel: OO in Python
Basisklasse WarenKorb

class WarenKorb:

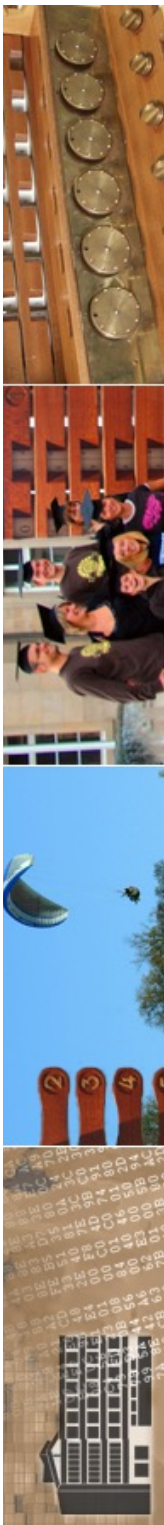
 ### Konstruktor ###
 def __init__ (self, content=None):
 self.inhalt = []
 if content:
 self.inhalt.append (content)

 def add (self, sache):
 self.inhalt.append (sache)

 def show (self):
 for element in self.inhalt:
 print (element)
[thomas@uni-vpn229 =>
```

# Instanzen

- Python hat kein »new«, es wird direkt der Konstruktor verwendet
  - `korb = Warenkorb ()`
- mit und ohne Argument (hier kein Überladen, da default-Parameter)





```
thomas@uni-vpn229 =>
thomas@uni-vpn229 => more korb.py
Grundlagen Internet-Technologien
#
Beispiel: OO in Python

from WarenKorb import *

unserKorb = WarenKorb ()

unserKorb.add ('Buch "Einfuehrung in Python3"')
unserKorb.add ('Apple iPhone 12 pro plus max')
unserKorb.add ('Buch "Egan Bernal: Leiden am Berg"')

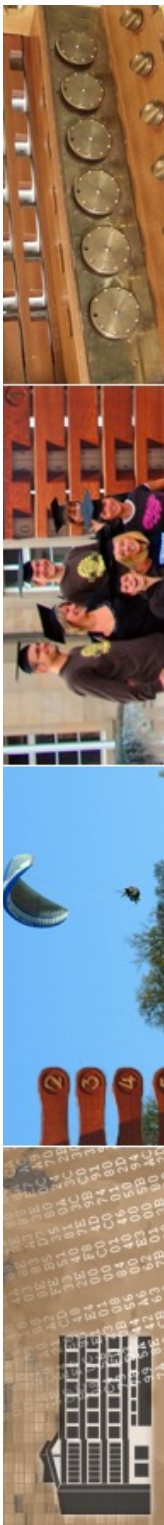
unserKorb.show()
thomas@uni-vpn229 =>
thomas@uni-vpn229 =>
```

```
thomas@uni-vpn229 =>
thomas@uni-vpn229 =>
thomas@uni-vpn229 => python3 korb.py
Buch "Einfuehrung in Python3"
Apple iPhone 12 pro plus max
Buch "Egan Bernal: Leiden am Berg"
thomas@uni-vpn229 =>
thomas@uni-vpn229 =>
```



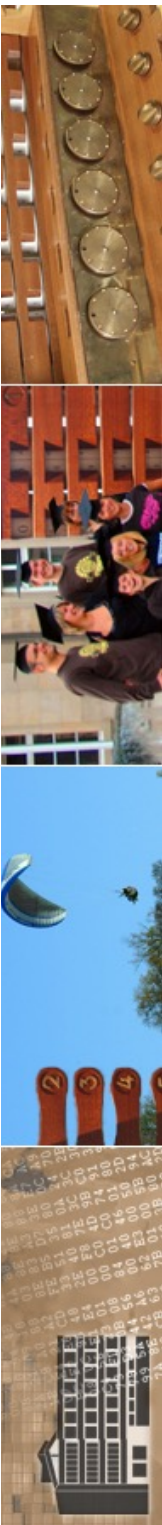
# wie findet Python die Klassen?

- anstelle des Klassenpfades (CLASSPATH) verwendet Python die Umgebungsvariable
  - **PYTHONPATH**
- zur Lokalisierung von Klassen
- durch das Einbinden wird Python-Bytecode persistent erzeugt und gespeichert: Wir erzeugen eine Datei **WarenKorb.pyc**



# die Referenz `self`

- `self` ist die Python-Version von »this«
- anstelle von `unserKorb.add(artikel)` auch:
- `WarenKorb.add (unserKorb, artikel)`

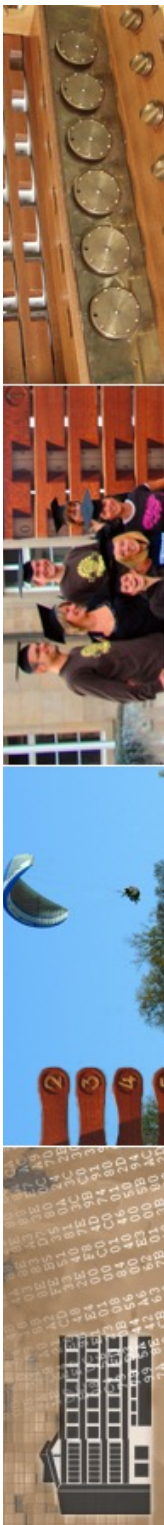


# Vererbung

- Python hat elementare Mechanismen zur Vererbung
- Syntax:
  - `class KlassenName (ElternKlasse1  
[, Elternklasse2,  
...]) :`
- auch Überschreiben von Methoden möglich

# Datenkapselung

- Python bietet auch eine *Datenkapselung*
- Syntax:
  - `__inhalt__`
- ist privates Attribut  
( 2 x `_` am Anfang, 1 x `_` am Ende)





```
thomas@uni-vpn229 =>
thomas@uni-vpn229 => more WarenKorb2.py
Grundlagen Internet-Technologien
#
Beispiel: 00 in Python

class WarenKorb2:

 ### Konstruktor ###
 def __init__ (self, content=None):
 self.__inhalt_ = []
 if content:
 self.__inhalt_.append (content)

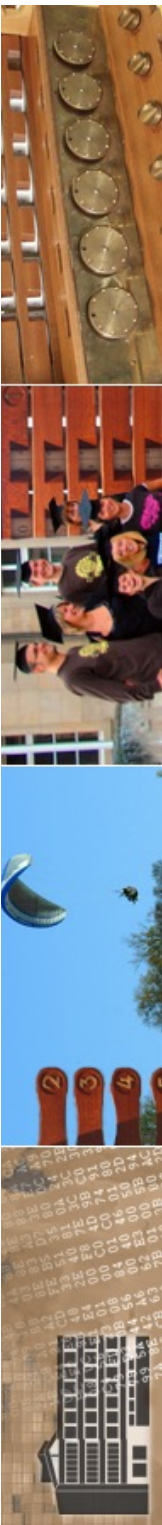
 def add (self, sache):
 self.__inhalt_.append (sache)

 def show (self):
 for element in self.__inhalt_:
 print (element)

thomas@uni-vpn229 =>
```

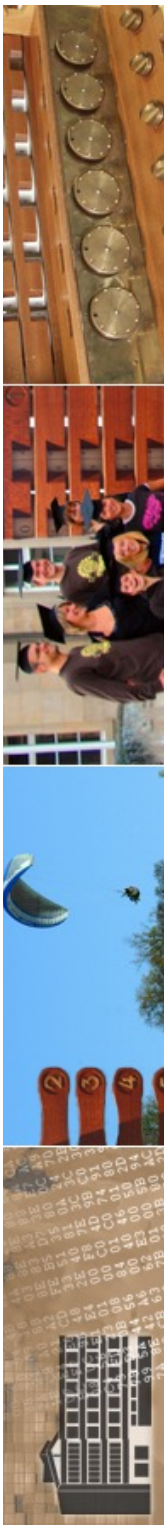
# einige spezielle OO-Methoden

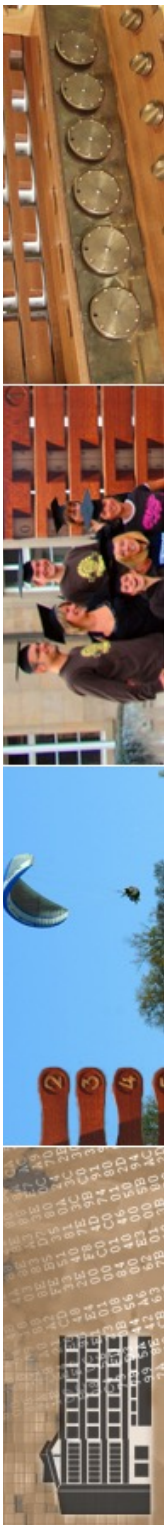
- `__init__ (self, args)` : Konstruktor
- `__del__ (self)` : Destruktor
- `__str__ (self)` : toString



# weiteres Beispiel: Buch

- Klasse Buch
- Instanz in Buchhandlung
- Klasse BuchPublisher erweitert Buch um weitere Attribute
- Instanz in Buchhandlung2





```
thomas@uni-vpn229 => more Book.py
Grundlagen Internet-Technologien
#
Programmierung mit Python
#
Beispiel fuer Objektorientierung
Klasse Buch: Konstruktor, gekapselte Attribute, Stringumwandlung
```

```
class Buch:
```

```
 ### Konstruktor ###
```

```
 def __init__(self, titel, nname, vname):
```

```
 self.__titel_ = titel
```

```
 self.__nname_ = nname
```

```
 self.__vname_ = vname
```

```
 ### get-Methoden fuer private Attribute ###
```

```
 def getAutor(self):
```

```
 autor = self.__vname_ + " " + self.__nname_
```

```
 return autor
```

```
 def getTitel(self):
```

```
 return self.__titel_
```

```
 def getVerlag(self):
```

```
 return self.__verlag_
```

```
 ### __str__-Methode ###
```

```
 def __str__(self):
```

```
 return self.getAutor() + ": " + self.__titel_
```

```
thomas@uni-vpn229 =>
```





```
[thomas@uni-vpn229 => more Buchhandlung.py
Grundlagen Internet-Technologien
#
Einfuehrung in Python
Instanz der Klasse Buch
```

```
from Book import Buch
```

```
meinNeuesBuch = Buch("Kompendium der Web-Programmierung", \
 "Walter", "Thomas")
```

```
print (meinNeuesBuch)
```

```
[thomas@uni-vpn229 =>
```

```
[thomas@uni-vpn229 =>
```

```
[thomas@uni-vpn229 => python3 Buchhandlung.py
Thomas Walter: Kompendium der Web-Programmierung
```

```
[thomas@uni-vpn229 =>
```



```
[thomas@uni-vpn229 => more BookPublisher.py
Grundlagen Internet-Technologien
#
Programmierung mit Python
#
Beispiel fuer Vererbung
Klasse Buch: Konstruktor, zwei zusaetzliche gekapselte Attribute,
Stringumwandlung

from Book import *

class BuchPublisher (Buch):

 ### Konstruktor ###
 def __init__(self, titel, nname, vname,verlag="Springer",jahr=2021):
 Buch.__init__(self,titel,nname,vname)
 self.__verlag_ = verlag
 self.__jahr_ = jahr

 ### get-Methoden fuer private Attribute ###
 def getVerlag(self):
 return self.__verlag_

 def getJahr(self):
 return self.__jahr_

 ### __str__-Methode ###
 def __str__(self):
 return Buch.__str__(self) + " (" + \
 self.getVerlag() + ", " + str(self.getJahr()) + ")"

[thomas@uni-vpn229 =>
```

# ...und nun...

- haben wir die universelle Scriptsprache Python grundlegend kennen gelernt
  - Prinzip, Variablen, Kontrollstrukturen und mehr



- als nächstes:

Wir verwenden Python,  
*um effizient*  
*CGI-Programme*  
*für serverseitige Web-*  
*Applikationen zu schreiben*

