```
HOWTO zu den PEPs von Python

(C) 2016-2020 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>
              OSTC Open Source Training and Consulting GmbH
              http://www.ostc.de

$Id: python-pep.txt,v 1.8 2020/04/08 13:15:00 tsbirn Exp $

Dieses Dokument beschreibt die Python Enhancement Proposals (PEPs). Diese
dienen der Diskussion und Dokumentation des Python-Entwicklungsvorgangs und der
Python-Nutzung. Jeder PEP hat eine NUMMER (die sich nicht mehr Ã¤ndert, sobald
sie vergeben wurde), eine KATEGORIE, einen TYP und einen STATUS.

Doku --> http://www.python.org/dev/peps
         http://www.python.org/dev/peps/pep-0000
         http://www.python.org/dev/peps/pep-0008

Wichtige PEPs sind:

  +----+--------------------------+
  |  0 | Index aller PEPs         |
  |  1 | PEP-Zweck und -Leitfaden |
  |  7 | C-Programmierstil        |
  |  8 | Python-Programmierstil   |
  | 20 | The Zen of Python        |
  +----+--------------------------+

Die PEPs sind in folgende KATEGORIEN eingeteilt (K): --> http://www.python.org/dev/peps#id5

  +---------------------+-------------------------------------------------------+
  | Meta                | PEPs Ã¼ber PEPs und Prozesse                           |
  | Other Informational | Andere informelle PEPs                                |
  | Provisional         | VorlÃ¤ufig akzeptiert (Interface kann sich noch Ã¤ndern|
  | Accepted            | Akzeptiert (evtl. noch nicht Implementiert)           |
  | Open                | Offenn (in Diskussion)                                |
  | Finished            | Abgeschlossen (mit einem stabilen Interface)          |
  | Historical          | Historische Meta und informelle PEPs                  |
  | Deferred            | Verschoben (waren auf weitere PrÃ¼fung und Updates)    |
  | Abandoned           | Abgebrochen                                           |
  | Withdrawn           | ZurÃ¼ckgezogen                                         |
  | Rejected            | Abgelehnt                                             |
  +---------------------+-------------------------------------------------------+

Ein PEP kann folgenden TYP haben (T): --> http://www.python.org/dev/peps/#pep-types-key

  +---+-----------------+
  | I | Informational   |
  | P | Process (Meta)  |
  | S | Standards Track |
  +---+-----------------+

Ein PEP kann folgenden STATUS haben (S): --> https://www.python.org/dev/peps/#pep-status-key

  +---+-----------------------------------------+
  | A | Accepted (Standards Track only) or Active |
  | D | Deferred                                |
  | F | Final                                   |
  | P | Provisional                             |
  | R | Rejected                                |
  | S | Superseded                              |
  | W | Withdrawn                               |
  +---+-----------------------------------------+

Folgende PEPs sind relativ bekannt und wichtig:

+--+------+-----------------------------------------------------------------+-----+
|TS|    0 | Index of Python Enhancement Proposals (PEP Index)               | Ver |
+--+------+-----------------------------------------------------------------+-----+
|P |    1 | PEP Purpose and Guidelines                                      |     |
|P |    7 | Style Guide for C Code                                          |     |
|P |    8 | Style Guide for Python Code (Namenskonventionen, Indent)        |     |
|I |   20 | The Zen of Python (import this)                                 |     |
|  |  401 | BDFL Retirement (Benevolent Dicatator For Life)                 |     |
+--+------+-----------------------------------------------------------------+-----+
|I | 8000 | Python Language Governance Proposal Overview                    |     |
|PA| 8001 | Python Governance Voting Process                                |     |
|I | 8002 | Open Source Governance Survey                                   |     |
|IR| 8010 | The Technical Leader Governance Model                           |     |
|IR| 8011 | Python Governance Model Lead by Trio of Pythonistas             |     |
|IR| 8012 | The Community Governance Model                                  |     |
|IR| 8013 | The External Council Governance Model                           |     |
|IR| 8014 | The Commons Governance Model                                    |     |
```

IR	8015	Organization of the Python community		
	8016	The Steering Council Model		
	8100	January 2019 steering council election		
	8101	2020 Term steering council election		
SF	100	Python Unicode Integration		2.0
SF	261	Support for "wide" Uniode characters		2.2
SF	277	Unicode file name support for Windows NT		2.3
SF	414	Explicit Unicode Literal for Python 3.3		3.3
SF	528	Change Windows console encoding to UTF-8		3.6
SF	529	Change Windows filesystem encoding to UTF-8		3.6
SF	540	Add a new UTF-8 Mode		3.7
SF	3131	Supporting Non-ASCII Identifiers		3.0
SF	278	Universal Newline Support		2.3
SF	263	Defining Python Source Code Encodings (# coding: UTF-8)		2.3
SF	3120	Using UTF-8 as the default source encoding		3.0
I	12	Sample reStructuredText PEP Template		
IA	257	Docstring Conventions		
IA	287	reStructuredText Docstring Format		
SF	221	Import As		2.0
I	394	The "python" Command on Unix-Like Systems (Shee-Bang #!...)		
IF	396	Module Version Numbers		
SF	318	Decorators for Functions and Methods		2.4
SF	3129	Class Decorators		3.0
SA	614	Relaxing Grammer Restrictions On Decorators		3.9
SF	232	Function Attributes		2.1
SF	234	Iterators		2.1
SF	255	Simple Generators ("yield")		2.2
SF	289	Generator Expressions		2.4
SF	342	Coroutines via Enhanced Generators		2.5
SF	492	Coroutines with async and await Syntax		3.5
SF	525	Asynchronous Generators		3.6
SF	530	Asynchronous Comprehensions		3.6
ID	457	Syntax for Positional-only Parameters		?.?
SA	570	Positional-only Parameters		?.?
SF	3102	Keyword-only Arguments		3.0
SF	205	Weak References		2.1
SF	308	Conditional Expression (VAL1 if COND else VAL2)		?.?
SF	343	The "with" Statement (Context Manager)		2.5
SF	465	A dedicated infix operator for matrix multiplication (@)		3.5
SA	572	Assignment Expression (:= GvR burnout, Juli 2018)		3.8
SF	441	Improving Python ZIP Application Support !!!		?.?
SF	562	Customization module attr access (__getattr__ __dir__)		3.7
SF	567	Context Variables		3.7
SF	227	Statically Nested Scopes		2.1
SF	3104	Access to Names in Outer Scope (aka "nolocal")		3.0
SF	3105	Make print a function		3.0
SF	3155	Qualified name for classes and functions		3.3
SF	328	Imports: Multi-Line and Absolute/Relative		2.456
SF	366	Main module explicit relative imports	2.6	3.0
SF	451	A ModuleSpec Type for the Import System		3.4
SF	218	Adding a Built-In Set Object Type		2.2
SF	285	Adding a bool type		2.3
SF	353	Using ssize_t as the index type		2.5
SF	358	The "bytes" Object (ersetzt durch 3137)	2.6	3.0
SF	3137	Immutable Bytes and Mutable Buffer		3.0
SF	435	Adding an Enum type to the Python standard library		3.4
SF	412	Key-Sharing Dictionary		3.3/4
SF	237	Unifying Long Integers and Integers		2.2
SF	238	Changing the Division Operator //		2.2
SF	327	Decimal data type		2.4
SF	378	Format Specifier for Thoudsands Separator	2.7	3.1
SF	485	A Function for testing approximate equality isclose()		3.5
SF	515	Underscores in Numeric Literals		3.6
SF	3127	Integer Literal Support and Syntax		3.0
SF	3141	A Type Hierarchy for Numbers		--- <-+
SF	3135	New Super (super())		3.0 |
SF	442	Safe Object Finalization !!!		3.4 |
S	544	Protocols: Structural subtyping (static duck typing)		3.7 |
SF	252	Making Types Look More Like Classes		2.2 |

SF	253	Subtyping Built-in Types	2.2	
SF	520	Preserving Class Attribute Definition Order	3.6	
SF	3119	Introducing Abstract Base Classes (ABC)	---	<-+
I	482	Literature Overview for Type Hints		
IF	483	The Theory of Type Hints		
SP	484	Type Hints (Module "typing")	3.5	
SF	526	Syntax for Variable Annotations	3.6	
SA	557	Data Classes	3.7	
SA	560	Core support for typing module and generic types	3.7	
SA	561	Distributing and Packaging Type Information		
SA	563	Postponed Evaluation of Annotations	3.7	
S	585	Type Hinting Usability Conventions		
S	586	Literal Types	3.8	?
S	589	TypedDict: Type Hints for Dictionaries with a Fixed	3.8	?
SA	593	Flexible function and variable annotation	3.9	?
SA	613	Explicit Type Aliases		?
SF	3107	Function Annotations (Syntax)	3.0	
SF	3101	Advanced String Formatting	3.0	
SF	498	Literal String Interpolation (F"...")	3.6	
SF	3113	Removing of Tuple Parameter Unpacking (PY2)	3.0	
SF	448	Additional Unpacking Generalizations	3.5	
SF	3132	Extended Iterable Unpacking	3.0	
SF	553	Built-in breakpoint() function	3.7	
SF	282	A Logging System	2.3	
IF	248	Database API Spec 1.0	---	
IF	249	Database API Spec 2.0	---	
SF	305	CSV File API (RFC 4180: Common Format and MIME Type for CSV)	---	
IF	333	Web Server Gateway Interface 1.0 (WSGI)	---	
IF	3333	Web Server Gateway Interface 1.0.1 (WSGI)	---	
SF	236	Back to the __future__	1.2	
I	290	Code Migration and Modernization		
IF	291	Backward Compatibility for the Python 2 Standard Library	2.3	
	430	Migrating to Python 3 as the default online documentation		
	427	The Wheel Binary Package Format 1.0 (*.whl)		
	440	Version Identification and Dependency Specification		
	508	Dependency specification for Python Software Packages		
SF	307	Extensions to the pickle protocol		
SF	3153	Pickle protocol version 4	3.4	
SF	574	Pickle protocol 5 with out-of-band data	3.8	
IF	160	Python 1.6 Release Schedule	1.6	
IF	200	Python 2.0 Release Schedule	2.0	
IF	226	Python 2.1 Release Schedule	2.1	
IF	251	Python 2.2 Release Schedule	2.2	
IF	283	Python 2.3 Release Schedule	2.3	
IF	320	Python 2.4 Release Schedule	2.4	
IF	356	Python 2.5 Release Schedule	2.5	
IF	361	Python 2.6 und 3.0 Release Schedule	2.6	
I	373	Python 2.7 Release Schedule	2.7	
IF	404	Python 2.8 Un-release Schedule	2.8	
PF	3000	Python 3000 (3.0 Py3K)		
IF	375	Python 3.1 Release Schedule	3.1	
IF	392	Python 3.2 Release Schedule	3.2	
IF	398	Python 3.3 Release Schedule	3.3	
I	429	Python 3.4 Release Schedule	3.4	
I	478	Python 3.5 Release Schedule	3.5	
I	494	Python 3.6 Release Schedule	3.6	
I	537	Python 3.7 Release Schedule	3.7	
I	569	Python 3.8 Release Schedule	3.8	
ID	596	Python 3.9 Release Schedule	3.9	
IF	607	Reducing CPython's Feature Delivery Latency	3.9	

Noch offene PEPs

DR	554	Multiple Interpreters in the Stdlib (subinterpreters)	3.9
DR	603	Adding a frozenmap type to collections	3.9
DR	611	The one million limit	3.9

ZurÃ¼ckgewiesene/ZurÃ¼ckgezogene PEPs

SR	315	Enhanced While Loop	3.5
SR	275	Switching on Multiple Values	2.6
SR	3103	A Switch/Case Statement	3.0
SR	3136	Labeled break and continue	3.1
SD	3124	Overloading, Generic Functions, Interfaces, and Adaptation	---
SW	3146	Merging Unladen Swallow into CPython	3.3
SR	516	Build System abstraction for pip/conda/...	---
SP	517	A Build System independent format for source trees	---