

Okt 15, 24 15:00

python-pep.txt

Page 1/5

HOWTO zu den PEPs von Python

(C) 2016-2021 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>  
 OSTC Open Source Training and Consulting GmbH  
<http://www.ostc.de>

\$Id: python-pep.txt,v 1.12 2021/07/28 08:29:35 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 ändert, sobald sie vergeben wurde) und einen TITEL, eine KATEGORIE, einen TYP und einen STATUS sowie den BESCHREIBUNGSTEXT.

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 über PEPs und Prozesse
Other Informational	Andere informelle PEPs
Provisional	Vorläufig akzeptiert (Interface kann sich noch ändern)
Accepted	Akzeptiert (evtl. noch nicht implementiert)
Open	Offen (in Diskussion)
Finished	Abgeschlossen (mit einem stabilen Interface)
Historical	Historische Meta und informelle PEPs
Deferred	Verschoben (waren auf weitere Prüfung und Updates)
Abandoned	Abgebrochen
Withdrawn	Zurückgezogen
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S	Superseded
W	Withdrawn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			Ver
	0	Index of Python enhancement proposals (PEP index)	
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)	
P	401	BDFL retirement (FLUFL Barry Warsaw, Aprilscherz 1.4.2009!)	
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	

Okt 15, 24 15:00		python-pep.txt	Page 2/5
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" unicode characters	2.2
SF	277	Unicode file name support for windows NT	2.3
SF	414	Explicit unicode literal for Python 3.3 (u"...")	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
SF	302	New import hooks (--> ersetzt durch import + importlib)	2.3
SF	328	Imports: multi-line and absolute/relative	2.456
SF	338	Executing modules as scripts	2.5
SF	366	Main module explicit relative imports	3.0 2.6
IF	396	Module version numbers	
SF	451	A moduleSpec type for the import system	3.4
SR	299	Special __main__() function in modules	2.3
SR	3122	Delineation of the main module	
I	394	The "python" command on Unix-Like systems (Shee-Bang #!...)	
SF	318	Decorators for functions and methods	2.4
SF	3129	Class decorators	3.0
SF	232	Function attributes	2.1
SF	234	Iterators (protocol)	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
SF	3148	futures - execute computations asynchronously	3.2
SF	3156	Asynchronous IO support rebooted: the "asyncio" module	---
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 (:= walrus op 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 "nonlocal")	3.0
SF	3105	Make print a function	3.0
SF	3155	Qualified name for classes and functions (__qualname__)	3.3
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 thousands separator	2.7 3.1

Okt 15, 24 15:00		python-pep.txt		Page 3/5	
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		3.7	
SA	563	Postponed evaluation of annotations		3.7	
S	585	Type hinting usability conventions		3.9	
S	586	Literal types		3.8	?
S	589	TypedDict: type hints for dictionaries with a fixed		3.8	?
SA	604	Allow writing union types as X   Y		3.10	
SA	613	Explicit type aliases		3.9	
SF	3107	Function annotations (syntax)		3.0	
SF	292	Simple string substitutions	Template formatting	2.4	
SF	3101	Advanced string formatting	formatting	3.0	
SF	461	Adding % formatting to bytes and bytearray	formatting	3.5	
SF	498	Literal string interpolation	F"... " formatting	3.6	
RI	502	String interpolation - extended discussion		3.6	
SD	536	Final grammar for literal string interpolation		3.7	
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			
FS	427	The wheel binary package format 1.0 (*.whl)			
DS	491	The wheel binary package format 1.9 (*.whl)			
	440	Version identification and dependency specification			
DS	582	Python local packages directory __pypackages__		3.8	
FS	576	Database of installed python distributions		3.2	
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	
SF	384	Defining a stable ABI		3.2	
SF	3147	PYC repository directories (__pycache__)		3.2	
SF	3149	ABI version tagged ".so" files		3.2	
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	

Okt 15, 24 15:00

python-pep.txt

Page 4/5

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
I	596	Python 3.9 release schedule	3.9
ID	619	Python 3.10 release schedule	3.10
IA	602	Annual release cycle for Python	3.9
IF	607	Reducing CPython's feature delivery latency	3.9
SF	495	Local time disambiguation	3.6
SF	3151	Reworking the exception hierarchy	3.3
SF	405	Python virtual environments	3.3
SF	486	Make the Python launcher aware of virtual environments	3.5
SF	659	Specializing adaptive interpreter	3.11
SA	617	New PEG parser for CPython	3.9

switch/case-Mehrfachverzweigung in 3.10

SS	622	Structural pattern matching --> 634	3.10
SA	634	Structural pattern matching: Specification	3.10
IF	635	Structural pattern matching: Motivation and rationale	3.10
IF	636	Structural pattern matching: Tutorial	3.10
SD	640	Unused variable syntax ( <u>_</u> )	3.10
SD	642	Explicit pattern syntax for structural pattern matching	3.10
SD	653	Precise Semantics for pattern matching	

Noch offene PEPs

SA	593	Flexible function and variable annotation	3.9	?
SA	614	Relaxing grammar restrictions on decorators	3.9	
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	

Zurückgewiesene/Zurückgezogene PEPs

SW	288	Generator attributes and exceptions (--> 343)	2.5
SW	310	Reliable acquisition/release pairs (--> 343)	2.4
SR	315	Resource-Release support for generators (--> 342)	2.4
SR	340	Anonymous block statements (--> 343)	---
SR	315	Enhanced while loop	3.5
SR	606	Python compatibility version	3.9
DR	387	Backwards compatibility policy	---
DR	497	A standard mechanism for backward compatibility	---
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	---