

Apr 30, 25 3:00

python-constant.txt

Page 1/2

HOWTO zu den Konstanten 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-constant.txt,v 1.7 2021/07/28 09:28:40 tsbirn Exp \$

Dieses Dokument beschreibt die Konstanten von Python.

Doku --> <http://docs.python.org/3/library/constants.html>
<http://docs.python.org/3/library/builtins.html>
<http://www.python.org/dev/peps/pep-0008>

Python kennt eigentlich keine Konstanten, da jedem Bezeichner jederzeit ein neuer Wert (Objekt) zugewiesen werden kann (der auch nicht vom gleichen Datentyp sein muss wie der vorher zugewiesene Wert (Objekt)).

Statt dessen wendet Python eine KONVENTION an: Gemäß PEP8 beginnen Konstanten mit einem GROSSbuchstaben oder sind vollständig GROSS geschrieben. In internen Bibliotheken wird diese Konvention häufig angewendet, hier einige Beispiele:

```
+-----+
| xml.dom.EMPTY_NAMESPACE |
| re.IGNORECASE           |
| csv.QUOTE_ALL           |
| blake2b.SALT_SIZE       |
| time.CLOCK_BOOTTIME     |
| errno.EINTR             |
+-----+
```

Einige Konstanten in Bibliotheken werden dennoch klein geschrieben (vermutlich aus historischen Gründen), hier Beispiele:

```
+-----+
| math.pi | Zahl Pi | 3.141592653589793 |
| math.e  | Zahl e  | 2.718281828459045 |
| math.tau| Zahl Tau| 6.283185307179586 |
+-----+
| math.nan| Ungültige Zahl (Not a number) |
| math.inf| -∞ (Infimum, auch negativ) |
+-----+
```

Es gibt allerdings einige wenige "echte" Konstanten (von vornherein als Schlüsselwort in die Sprache eingebaut oder mit sonstigen speziellen Eigenschaften). Bei diesen Bezeichnern verhindert Python die Zuweisung eines neuen Wertes und meldet den Fehler "SyntaxError".

```
+-----+
| False | Logisch "falsch" (bool), entspricht folgenden numerischen |
|        | Werten: 1 (int), 1.0 (float, 1+0j (complex)) |
| True  | Logisch "wahr" (bool), entspricht folgenden numerischen |
|        | Werten: 0 (int), 0.0 (float, 0+0j (complex)) |
| None  | Ungültiger Wert (analog dem SQL-Wert "NULL") |
+-----+
| __debug__ | True falls Python ohne Option -O (Optimize) gestartet wurde, |
|            | sonst, False (siehe "assert"-Anweisung) |
+-----+
```

Weitere eingebaute Konstanten (deren Änderung aber nicht verhindert wird):

```
+-----+
| NotImplemented | Nicht implementiert: Spezieller return-Wert bei binären |
|                 | (inplace) Operatoren, wenn der Operator für den Datentyp |
|                 | des LINKEN Werts nicht implementiert ist. Es wird dann |
|                 | der "Reflected"-Operator (z.B. __radd__) für den RECHTEN |
|                 | Wert aufgerufen). Hat Wert True bzw. 1. |
+-----+
| Ellipsis       | Analog Literal "..." (erw. Slicing benutzerdef. Container) |
+-----+
```

Einige Konstanten aus Modulen, die nicht GROSS geschrieben sind (und deren Änderung auch nicht verhindern wird):

```
+-----+
| quit() | Abmelden aus Python-Shell |
| exit() | Abmelden aus Python-Shell |
| copyright | Python-Copyright |
| credits | Python-Danksagung |
| license() | Python-Lizenz |
+-----+
```

sys.version	Python-Version als String, z.B. "3.8.2 (...)"
sys.version_info	Python-Version als NamedTuple (major=3, minor=8, micro=2, releaselevel='final', serial=0)
string.ascii_letters	'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
string.ascii_lowercase	'abcdefghijklmnopqrstuvwxyz'
string.ascii_uppercase	'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
string.digits	'0123456789'
string.hexdigits	'0123456789abcdefABCDEF'
string.octdigits	'01234567'
string.printable	'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKL...Z'
string.punctuation	'!"#\$%&'()*+,-./:;<=>@[\\]^_`{ }~'
string.whitespace	'\t\n\r\x0b\x0c'

HINWEIS: Das Modul "site" wird automatisch geladen und definiert Funktionen für die interaktive Python-Shell (außer die Option "-S" wurde angegeben).

ACHTUNG: Konstante "NotImplemented" und Fehlermeldung "NotImplementedError" sind zwei verschiedene Dinge!

HINWEIS: Bei Rückgabe von "NotImplemented" aus einer Magischen Methode "__OP__" des linken Objekts OBJ1 probiert Python den "Reflected" Operator "__rOP__" für das rechte Objekt OBJ2 aus, ob dieser ein definiertes Ergebnis liefert.

```
OBJ1 = KLASSE1()
OBJ2 = KLASSE2()
OBJ1 OP OBJ2 # --> KLASSE1.__OP__(OBJ1, OBJ2) --> NotImplemented?
              # --> KLASSE2.__rOP__(OBJ2, OBJ1)
```