

Spezielle Bezeichner (Identifizier) (C) 2017-2021 T.Birnthaler OSTC GmbH

Doku --> <http://docs.python.org/dev/peps/pep-0008>
http://docs.python.org/3/reference/lexical_analysis.html#identifiers-and-keywords
<http://www.pep8.org>

Folgende Namenskonventionen für Bezeichner (Identifizier) gelten in Python:

Name	Bedeutung	
<code>_*</code>	INTERNER Name (reserviert für Python selbst)	
<code>__*</code>	Privater Name in Klasse (mangled --> <code>__CLASS__*</code>)	
<code>_*</code>	Protected Name in Klasse (nur für Vererbung)	
<code>__*</code>	Interner Name in Modul (für modulinterne Zwecke) (von "from MODULE import *" nicht importiert) (Hilfsklasse, Hilfsfunktion, ...)	
<code>*_</code>	Schlüsselwort als Bezeichner (z.B. "if")	
<code>_</code>	Ergebnis der letzten Auswertung im interaktiven Interpreter Temporäre Variable in Schleife (for <code>_</code> in ...) "Wegwerfvariable" (syntaktisch notwendig, inhaltlich irrelevant) Internationalisierung (i18n, l10n, gettext)	
<code>self</code>	Objekt in normaler Methode einer Klasse	
<code>other</code>	2. Objekt in normaler 2-wertiger Methode einer Klasse	
<code>cls</code>	Klasse in Klassen-Methode einer Klasse	
<code>*args</code>	"Sammler" für Positions-Parameter in Funktion (--> Tupel)	
<code>**kwargs</code>	"Sammler" für Keyword-Parameter in Funktion (--> Dictionary)	
<code>NAME</code>	Konstante (GROSSB., Unterstrich, Ziffern) (Ausnahmen: <code>True</code> , <code>False</code> , <code>None</code> , <code>math.inf</code> , <code>math.nan</code>)	Subst/Adj
<code>Name</code>	Klassenname (CamelCase)	Substantiv
<code>Name</code>	Exceptionklasse (CamelCase)	Substantiv
<code>name</code>	Modulname (kleinb., KEIN Unterstrich, Ziffern)	Substantiv
<code>name</code>	Variablenname (kleinb., Unterstrich, Ziffern)	Subst/Adj
<code>name</code>	Funktionsname (kleinb., Unterstrich, Ziffern)	Verb
<code>name</code>	Objektname (kleinb., Unterstrich, Ziffern)	Subst/Adj

Qualifizierte Namen werden durch "." getrennt (der "." wird beim Zugriff auf das Dateisystem in den jeweiligen Verz.-Trenner "\" oder "/" umgesetzt).

```

MODUL.VARIABLE           # Zugriff auf Variable in Modul
MODUL.FUNKTION()         # Aufruf von Funktion in Modul
PAKET.SUBMODUL.VARIABLE  # Zugriff auf Variable in Paket-Untermodule
PAKET.SUBMODUL.FUNKTION() # Aufruf von Funktion in Paket-Untermodule
PFAD.ZU.MODUL.VARIABLE   # Zugriff auf Variable in Modul in Unterverzeichnis
PFAD.ZU.MODUL.FUNKTION() # Aufruf von Funktion in Modul in Unterverzeichnis

```

Komponenten von Klassen und Objekten werden ebenfalls durch "." im Namen angesprochen:

```

OBJEKT.ATTRIBUTE        # Zugriff auf Attribut von Objekt
OBJEKT.METHODE()        # Aufruf von Methode für Objekt
KLASSE.ATTRIBUTE        # Zugriff auf Klassen-Attribut
KLASSE.METHODE()        # Aufruf von Klassen-Methode oder statischer Methode
FUNKTION.ATTRIBUTE      # Zugriff auf Funktions-Attribut

```

Spezielle Attribute

Spezielle Attribute, werden teilweise von der built-in Funktion "dict()" nicht aufgelistet, wohl aber von der Funktion "dir()".

Attribut	Beschreibung
<code>__doc__</code>	Dokumentationsstring ("Docstring")
<code>__name__</code>	Name Klasse/Funktion/Methode/Descriptor/Generator-Instanz
<code>__qualname__</code>	Qualifizierter Name Klasse/.../Generator-Instanz
<code>__module__</code>	Modul in dem Name definiert ist
<code>__all__</code>	In <code>__init__.py</code> autom. zu lad. Subm. bei <code>from ... import *</code>
<code>__call__</code>	Codeobjekt einer Funktion
<code>__defaults__</code>	Tuple mit Defaultwerten für Position-Parameter von Funk.
<code>__kwdefaults__</code>	Dictionary mit Defaultwerten für Keyword-Parameter von F.
<code>__annotations__</code>	Dictionary mit "Annotations" für Param. + Rückgabewert
<code>__closure__</code>	Bindungen freier Variablen an Werte
<code>__mro__</code>	Tupel der Basisklassen für Methodenaufklärung (linear)
<code>mro()</code>	Liefert Ergebnis für <code>__mro__</code> bei Klassen-Instanziierung

Jul 02, 25 15:00

python-identifizier.txt

Page 2/2

<code>__subclasses__()</code>	Liste schwacher Referenzen zu direkten Unterklassen
<code>__class__</code>	Klasse zu der eine Instanz gehört
<code>__bases__</code>	Tupel der Basisklassen eines Klassenobjekts
<code>__dict__</code>	Speicher für dynamische Objektattribute (Name + Wert)
<code>__slots__</code>	Namen der statische Objektattribute
<code>__weakref__</code>	"Schwache" Referenzen
<code>__globals__</code>	Referenz zu Dictionary mit globalen Variablen
<code>__self__</code>	Objekt auf dem eine Methode arbeitet
<code>__func__</code>	Funktion die hinter einer Methode steckt (implementiert)

Spezielle Methoden
