SchlÃ¼sselworte von Python (C) 2016-2021 T.Birnthaler OSTC GmbH
==========

Folgende 35 Worte sind SchlÃ¼sselworte (keywords) in Python, ihre Bedeutung ist FIX und kann nicht geÃ¤ndert werden (d.h. sie sind insbesondere nicht als Bezeichner von Variablen verwendbar).

--> http://docs.python.org/3/reference/lexical_analysis.html#keywords
http://docs.python.org/3/reference/lexical_analysis.html#identifiers-and-keywords

Funktionen zur Information Ã¼ber Keywords:

```
help("keywords")
import keyword
keyword.iskeyword("for")   # --> True
keyword.kwlist()           # --> ['False', 'None', ..., 'with', 'yield']
```

Soll ein SchlÃ¼sselwort trotzdem als Name einer Variablen/Funktion/Klasse/... genutzt werden, dann als Konvention einen Unterstrich anhÃ¤ngen (z.B. "if_").

True	Konstante "logisch wahr" (entspricht in Rechenausdruck dem Wert 1)	
False	Konstante "logisch falsch" (entspricht in Rechenausdruck dem Wert 0)	
None	Konstante "unmÃ¶glicher Wert" (fÃ¼hrt im Rechenausdruck zu Fehler)	
and	Operator "logisch UND" (&& gibt es nicht, & ist bitweise UND)	
or	Operator "logisch ODER" (|| gibt es nicht, | ist bitweise ODER)	
not	Operator "logisch NICHT" (! gibt es nicht, ~ ist bitweise NICHT)	
class	Klasse definieren	
def	Funktion definieren	
lambda	Unbenannte Funktion definieren (nur ein Statement: Rechenausdruck)	
global	Globale Variable in Funktion	
nonlocal	Nichtlokale Variable in eingeschachtelter Funktion	
return	RÃ¼cksprung aus Funktion	
yield	Generator-RÃ¼ckgabe (erhÃ¤lt Zustand im Gegensatz zu "return")	
for	Schleife Beginn	
while	Schleife Beginn	
break	Umgebende Schleife abbrechen	
continue	Umgebende Schleife an Anfang springen (nÃ¤chster Durchlauf)	
if	Fallunterscheidung Beginn	
elif	Fallunterscheidung Mehrfachverzweigung	
else	Fallunterscheidung (auch Schleife, Ausnahmebehandlung, Kontext Manager)	
try	Ausnahmebehandlung Beginn	
except	Ausnahmebehandlung Fehlerbehandlung	
finally	Ausnahmebehandlung Abschluss (optional)	
raise	Ausnahme werfen	
import	Modul-Import	
from	Teil von import, raise, ...	
as	Teil von import, except, with, ...	
in	Objekt-Suche (Elem in Sequenz, Key in Dict, String in String, ...)	
is	Vergleich (ObjektidentitÃ¤t, KlassenzugehÃ¶rigkeit)	
assert	Zusicherung prÃ¼fen (und ggf. AssertionError auslÃ¶sen)	
del	Name (+ Referenz auf Objekt) lÃ¶schen (Objekt selbst nicht unbedingt)	
pass	Leere Anweisung (syntaktisch notwendig fÃ¼r "leeren Block")	
with	Kontext Manager	
async	Koroutinen (PEP 492)	PY3.5
await	Koroutinen (PEP 492)	PY3.5
switch	Structural Pattern Matching (PEP 622,634,635,636,642,653)	PY3.10
case	Structural Pattern Matching (PEP 622,634,635,636,642,653)	PY3.10
_	Structural Pattern Matching (PEP 622,634,635,636,642,653)	PY3.10

HINWEIS: In Python 3.10 wurden "switch", "case" und "_" als "Soft Keywords" hinzugefÃ¼gt, d.h. sie zÃ¤hlen nur im Kontext "Pattern Matching" als SchlÃ¼sselworte, um bereits bestehenden Code, der diese Namen verwendet, nicht ungÃ¼ltig zu machen. Mit dem seit Python 3.9 eingesetzten PEG-Parser ist das kein Problem, das Modul "keyword" enthÃ¤lt seit Python 3.9 die Members "issoftkeyword()" und "swkwlist" zusÃ¤tzlich zu "iskeyword()" und "kwlist".

HINWEIS: Analog wurden in Python 3.5 und 3.6 die Namen "async" und "await" zunÃ¤chst als "Soft Keywords" hinzugefÃ¼gt (damals durch einen Parser-Trick), um bestehenden Code nicht ungÃ¼ltig zu machen. Erst seit Python 3.7 zÃ¤hlen sie als

echte SchlÃ¼sselworte.

HINWEIS: Neue SchlÃ¼sselworte werden extrem selten zu Python hinzugefÃ¼gt,
(gestrichen werden SchlÃ¼sselworte gar nicht). Bevor so etwas passiert, wird auf
jeden Fall ein PEP erstellt, es findet eine Diskussion und eine
Abstimmung/Entscheidung sowie die Festlegung der Python-Version statt, ab der
dieses SchlÃ¼sselwort verfÃ¼gbar sein wird. D.h. man hat mehrere Jahre Zeit, um
seinen Code daran anzupassen.