```
Built-in Functions von Python                (C) 2015-2020 T.Birnthaler OSTC GmbH
=============================

In Python sind 69 Funktionen fest eingebaut ("built-in"), sie kÃ¶nnen
Ã¼berschrieben werden, sollten dies aber nicht unbedingt (da sonst z.B. die
Funktion "print()" oder Datentypen wie "list" nicht mehr funktionieren).

Doku --> http://docs.python.org/3/library/functions.html
         http://docs.python.org/3/library/builtins.html

Diese GENERISCHEN Funktionen sind auf mehrere verschiedene Datentypen (manchmal
sogar auf alle) anwendbar, daher sind sie keinem Datentyp zugeordnet.

Eine Liste dieser Funktionen erhÃ¤lt man folgendermaÃM-^_en:

  pydoc builtins        # Auf der Kommandozeile
  import builtins       # Im Python-Skript
  dir(builtins)         # Im Python-Skript

Zugriff auf die in Python direkt eingebauten Objekte (Funktionen, Konstanten,
...) ist Ã¼ber das Modul "builtins" grundsÃ¤tzlich immer mÃ¶glich:

  import builtins
  builtins.open(...)    # Statt open(...)

+-------------------------+-----------------------------------------------------+
| Funktion                | Beschreibung                                        |
+-------------------------+-----------------------------------------------------+
| id(OBJ)                 | Identity von Objekt                                 |
| type(OBJ)               | Datentyp von Objekt                                 |
| dir(OBJ)                | Elemente von (lokalem) Namensraum/Objektattribute   |
| hash(OBJ)               | Hashwert eines Objekts erstellen --> __hash__()     |
| help(OBJ)               | Hilfe zu Objekt ausgeben --> __doc__                |
+-------------------------+-----------------------------------------------------+
| abs(N)                  | Absolutwert der Zahl N                              |
| divmod(A,B)             | Ergibt (A // B, A % B)                              |
| round(N,DIGITS)         | Runden auf DIGITS Nachkommastellen (oder Ganzzahl)  |
| pow(X,Y,Z)              | X hoch Y (modulo Z)                                 |
+-------------------------+-----------------------------------------------------+
| all(ITER)               | True falls alle Elemente True (oder ITER leer)      |
| any(ITER)               | False falls alle Elemente False (oder ITER leer)    |
| max(ITER)               | Maximaler Wert der Elemente (< Vergleich nÃ¶tig)     |
| min(ITER)               | Minimaler Wert der Elemente (< Vergleich nÃ¶tig)     |
| sum(ITER,START)         | Summe der Elemente                                  |
+-------------------------+-----------------------------------------------------+
| repr(OBJ)               | Datentyp-Konvertierung nach "str" (maschinenlesbar) |
| ascii(OBJ)              | repr() + non-ASCII Zeichen escaped (\x \u \U)       |
+-------------------------+-----------------------------------------------------+
| chr(N)                  | Zeichen mit Unicode N --> Gegenteil von ord()       |
| ord(STR)                | Unicode des 1. Zeichens --> Gegenteil von chr()     |
+-------------------------+-----------------------------------------------------+
| bin(N)                  | Zahl N im BinÃ¤rformat "0b..." zurÃ¼ckgeben           |
| hex(N)                  | Zahl N im Hexadezimalformat "0x..." zurÃ¼ckgeben     |
| oct(N)                  | Zahl N im Oktalformat "0o..." zurÃ¼ckgeben           |
+-------------------------+-----------------------------------------------------+
| bool(VAL)               | Datentyp-Konvertierung nach "bool"                  |
| bytearray(SRC,ENC,ERR)  | Datentyp-Konvert. nach "bytearray" (mutable int seq)|
| bytes(SRC,ENC,ERR)      | Datentyp-Konvert. nach "bytes" (immutable int seq)  |
| complex(REAL,IMAG)      | Datentyp-Konvertierung nach "complex"               |
| dict(OBJ)               | Datentyp-Konvertierung nach "dict"                  |
| float(VAL)              | Datentyp-Konvertierung nach "float"                 |
| frozenset(ITER)         | Datentyp-Konvertierung nach "frozenset"             |
| int(OBJ)                | Datentyp-Konvertierung nach "int"                   |
| list(ITER)              | Datentyp-Konvertierung nach "list"                  |
| memoryview(OBJ)         | Datentyp-Konvertierung nach "memoryview"            |
| set(ITER)               | Datentyp-Konvertierung nach "set"                   |
| str(OBJ)                | Datentyp-Konvertierung nach "str"                   |
| tuple(ITER)             | Datentyp-Konvertierung nach "tuple"                 |
+-------------------------+-----------------------------------------------------+
| globals()               | Globale Variablen im aktuellen Namensraum (Modul)   |
| locals()                | Lokale Variablen im aktuellen Namensraum (Funktion) |
| vars()   vars(OBJ)      | Analog locals() bzw. OBJ.__dict__                   |
+-------------------------+-----------------------------------------------------+
| delattr(OBJ,ATTR)       | Attribut zu Objekt lÃ¶schen --> del OBJ.ATTR         |
| getattr(OBJ,ATTR,DFLT)  | Wert zu Attribut von Objekt holen                   |
| hasattr(OBJ,ATTR)       | True falls Objekt Attribut hat, sonst False         |
| setattr(OBJ,ATTR,VAL)   | Attribut in Objekt auf Wert setzen                  |
+-------------------------+-----------------------------------------------------+
| callable(OBJ)           | True falls Objekt aufrufbar ist (Func/Class/Object) |
| isinstance(OBJ,CLS)     | Objekt ist Instanz von Kl. oder einer ihrer Unterkl.|
| issubclass(CLS,CLS2)    | Klasse ist Unterklasse (auch von sich selbst)       |
| super(TYPE,OBJ/TYPE)    | Elternklasse oder Geschwisterklasse                 |
```

object()	Neues leeres Objekt (Basisklasse aller Klassen)
property(GET,SET,DEL,DOC)	Property-Attribut erzeugen
classmethod()	Decorator (Methode --> Klassenmethode --> C.f(cls))
staticmethod()	Decorator (Methode --> Statische M. --> C.f())
open(FILE,MODE="r",BUF=-1,ENC=None,ERR=None,NL=None,CLOSEFD=True,OPENER=None)	Datei Ã¶ffnen und Fileobjekt zurÃ¼ckgeben
input(PROMPT)	Terminal-Eingabe als "str" einlesen
print(LIST,SEP=" ",END="\n",FILE=sys.stdout,FLUSH=False)	Alle Elemente von LIST per str() ausgeben
format(VAL,FORMAT)	Formatierter Wert
len(SEQ)	Anzahl Elemente (Sequenz, Range, Collection)
reversed(SEQ)	Umgekehrten Iterator zurÃ¼ckgeben
sorted(ITER,KEY,REV)	Sortierte Folge zurÃ¼ckgeben
iter(OBJ,SENTINEL)	Iterator-Objekt zurÃ¼ckgeben --> __iter__()
next(ITER,DEFAULT)	NÃ¤chstes Element aus Iterator holen --> __next__()
range(START,STOP,STEP)	Zahlenliste erzeugen
slice(START,STOP,STEP)	Slice-Objekt erzeugen
enumerate(ITER,START)	Elemente mit aufsteigender Nummer versehen
filter(FUNC,ITER)	Elemente filtern gemÃ¤ÃM-^_ Funktion --> True
map(FUNC,ITER)	Elemente abbilden gemÃ¤ÃM-^_ Funktion
zip(*ITER)	VerschrÃ¤nkt Folgen
breakpoint(*, **)	Sprung in Debugger sys.breakpointhook() --> pdb.set_trace()
compile(SRC,FILE,MOD,FLG,DONTINHERIT=False,OPT=-1)	Kompil. SRC in Code oder AST Objekt --> exec/eval()
eval(STR,GLOB,LOCAL)	String als Python-Skript ausfÃ¼hren
exec(OBJ,GLOB,LOCAL)	String/Code-Objekt als Python-Skript ausfÃ¼hren
__import__(NAME,...)	Von "import" aufgerufen