eingebaute Funktionen („Built-in Functions“)

Eine Frage ist immer, welche Methoden und Klassen enthält ein Paket? Was gibt es bereits an Funktionen und kann sehr einfach genutzt werden, ohne dass man das Rad neu erfinden muss? Auch was ist bereits vorhanden und muss bzw. muss nicht importiert werden

Über `dir` haben wir eine mächtige Funktion, die uns eine Liste der Attribute und Methoden von jedem Objekt zurückliefert.

- bei Klassen-Objekten wird eine Liste von Namen aller validen Attribute (und auch Basis-Attribute) zurück übergeben
- bei Modulen (Library-Objekten) wird eine Liste der Namen von allen Attributen, die in dem Modul vorhanden sind.
- wenn kein Parameter angegeben wird, wird eine Liste von Namen des aktuellen lokalen Geltungsbereichs/Namensbereich („local scope“) zurückgeliefert

Das letzte Verhalten machen wir uns zunutze.

```
>>> dir()
```

Als Rückmeldung erhalten wir:

['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__']

Mehr Informationen (Anmerkung: In Python 2.7 wurde noch bedeutend weniger bzw. anders ausgegeben) erhalten wir über die bereits enthaltenen Funktionen und Typen von Python mit

```
dir(__builtins__)
```

Als Ergebnis erhalten wir:

>>> dir(__builtins__)

['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False', 'FileExistsError', 'FileNotFoundError', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'ModuleNotFoundError', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError', 'RecursionError', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'ZeroDivisionError', '_', '__build_class__', '__debug__', '__doc__', '__import__', '__loader__', '__name__', '__package__', '__spec__', 'abs', 'all', 'any', 'ascii', 'bin', 'bool', 'breakpoint', 'bytearray', 'bytes', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval', 'exec', 'exit', 'filter', 'float', 'format',

'frozenset', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex', 'id', 'input', 'int', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'map', 'max', 'memoryview', 'min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'quit', 'range', 'repr', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple', 'type', 'vars', 'zip']

Mehr über die einzelnen Objekte zu erfahren nutzt man die Anweisung `help`

Möchte man also mehr über die print-Funktion erfahren, gibt man einfach ein:

```
help(print)
```

Als Ausgabe erfolgt:

>>> help(print)

Help on built-in function print in module builtins:

print(...)

print(value, ..., sep=' ', end='

', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.

Optional keyword arguments:

file: a file-like object (stream); defaults to the current sys.stdout.

sep: string inserted between values, default a space.

end: string appended after the last value, default a newline.

flush: whether to forcibly flush the stream.

Die erste Linie zeigt einem, aus welchem Modul diese Funktion ist – im Fall von `print` kommt diese Funktion aus bereits im System integrierten Bereich. Wir müssen also nichts in unser Python-Programm importieren, um diese Funktion nutzen zu können.

In den darauffolgenden Linien wird die Nutzungsweise beschreiben.

Man sieht bei dieser Funktion, dass weitere Parameter übergeben werden können (was auch Sinn ergibt, sonst wird nichts ausgegeben). Auch sind bereits Parameter mit Standardausgaben belegt wie beispielsweise der Separator `sep=' '` als Leerzeichen.

Hier können wir eingreifen, wenn gewünscht. Anstelle des Leerzeichens des Separators lassen wir einen Unterstrich ausgeben:

```
>>> print("Hallo", "Welt", "Wie", sep='_')
```

Als Ergebnis erhalten wir:

Hallo_Welt_Wie

Schauen wir uns eine weitere Funktion an: hex

>>> help(hex)

Help on built-in function hex in module builtins:

hex(number, /)

Return the hexadecimal representation of an integer.

>>> hex(12648430)

'0xc0ffee'

Wir haben wieder eine Funktion die „ab Werk“ sofort in Python zur Verfügung steht.

Es wird eine Ganzzahl (Integer) als Hexadezimale Zahl ausgegeben:

```
>>> hex(255)
'0xff'
```

Wir sehen, dass wir als Rückgabe einen String erhalten – daher wird der Rückgabewert in Anführungszeichen ausgegeben.

Wofür steht das „0x“? Das ist sehr einfach: das x steht für „heXadezimal“. Hätten wir einen binären Wert (z.B. 0110011) würde als Präfix „0b“ für „Binär“ ausgegeben. Für eine klare Abgrenzung zu einer dezimalen Zahl fangen hexadezimale und binäre Zahlen immer damit an – das wurde bereits in der ANSI Norm für C Compiler anno dazumal festgelegt.

Siehe dazu einfach auch `help(bin)`

Durch die Schreibweise wird auch direkt im Python-Interpreter eine binäre bzw. Hexadezimale Eingabe in dezimaler Ausgabe zurückgegeben:

```
>>> 0xff
255
```

Liste von Modulen

Um alle weiteren Module angezeigt zu bekommen, einfach eingeben:

```
>>> help('modules')
```

Wir bekommen daraufhin eine sehr umfangreiche Ausgabe aller Module:

>>> help('modules')

Please wait a moment while I gather a list of all available modules...

```
__future__          _threading_local    getpass             rlcompleter
_abc                _tkinter            gettext             runpy
_ast                _tracemalloc        glob                sched
_asyncio            _uuid               grp                 secrets
_bisect             _warnings           gzip                select
_blake2             _weakref            hashlib             selectors
_bootlocale         _weakrefset         heapq               setuptools
_bz2                _xxtestfuzz         hmac                shelve
_codecs             abc                 html                shlex
_codecs_cn          aifc                http                shutil
_codecs_hk          antigravity         idlelib             signal
_codecs_iso2022     argparse            imaplib             site
_codecs_jp          array               imghdr              smtpd
_codecs_kr          ast                 imp                 smtplib
_codecs_tw          asynchat            importlib           sndhdr
_collections        asyncio             inspect             socket
```

```
_collections_abc    asyncore            io                  socketserver
_compat_pickle      atexit              ipaddress           sqlite3
_compression        audioop             itertools           sre_compile
_contextvars        base64              json                sre_constants
_crypt              bdb                 keyword             sre_parse
_csv                binascii            lib2to3             ssl
_ctypes             binhex              linecache           stat
_ctypes_test        bisect              locale              statistics
_curses             builtins            logging             string
_curses_panel       bz2                 lzma                stringprep
_datetime           cProfile            macpath             struct
_dbm                calendar            mailbox             subprocess
_decimal            cgi                 mailcap             sunau
_dummy_thread       cgitb               marshal             symbol
_elementtree        chunk               math                symtable
_functools          cmath               mimetypes           sys
_hashlib            cmd                 mmap                sysconfig
_heapq              code                modulefinder        syslog
_imp                codecs              multiprocessing     tabnanny
_io                 codeop              netrc               tarfile
_json               collections         nis                 telnetlib
_locale             colorsys            nntplib             tempfile
_lsprof             compileall          ntpath              termios
_lzma               concurrent          nturl2path          test
_markupbase         configparser        numbers             textwrap
_md5                contextlib          opcode              this
_multibytecodec     contextvars         operator            threading
_multiprocessing    copy                optparse            time
_opcode             copyreg             os                  timeit
_operator           crypt               parser              tkinter
_osx_support        csv                 pathlib             token
_pickle             ctypes              pdb                 tokenize
_posixsubprocess    curses              pickle              trace
_py_abc             dataclasses         pickletools         traceback
_pydecimal          datetime            pip                 tracemalloc
_pyio               dbm                 pipes               tty
_queue              decimal             pkg_resources       turtle
_random             difflib             pkgutil             turtledemo
_scproxy            dis                 platform            types
_sha1               distutils           plistlib            typing
_sha256             doctest             poplib              unicodedata
_sha3               dummy_threading     posix               unittest
_sha512             easy_install        posixpath           urllib
_signal             email               pprint              uu
_sitebuiltins       encodings           profile             uuid
_socket             ensurepip           pstats              venv
_sqlite3            enum                pty                 warnings
_sre                errno               pwd                 wave
_ssl                faulthandler        py_compile          weakref
_stat               fcntl               pyclbr              webbrowser
_string             filecmp             pydoc               wsgiref
_strptime           fileinput           pydoc_data          xdrlib
_struct             fnmatch             pyexpat             xml
_symtable           formatter           queue               xmlrpc
_sysconfigdata_m_darwin_darwin fractions          quopri              xxlimited
_testbuffer         ftplib              random              xxsubtype
_testcapi           functools           re                  zipapp
_testimportmultiple gc                  readline            zipfile
_testmultiphase     genericpath         reprlib             zipimport
_thread             getopt              resource            zlib
```

Enter any module name to get more help. Or, type "modules spam" to search

for modules whose name or summary contain the string "spam".

Um ein Modul nutzen zu können, müssen wir dieses im ersten Schritt importieren:

```
import math
```

Zur Kontrolle, ob der Import geklappt hat und uns dieses Modul zur Verfügung steht, können wir die Funktion `dir()` aufrufen.

```
dir()
```

Nun sollte das Modul zusätzlich erscheinen:

>>> dir()

['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__', 'math']

Um die nun durch „math“ möglichen Funktionen anzusehen, einfach auf diese `dir(math)` anwenden.

>>> dir(math)

['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']

Es werden bei dem Aufruf keine Anführungszeichen verwendet! Wir wollen das Modul nutzen und nicht einen Zeichenkette (string).

Um nun weitere Hilfen über help für die im Modul math vorhandenen Möglichkeiten zu bekommen muss immer das Modul mit angegeben werden. Ansonsten bekommen wir eine „Traceback“-Fehlermeldung.

```
>>> help(math.sin)
Help on built-in function sin in module math:
sin(x, /)
    Return the sine of x (measured in radians).
```

Testen wir die Sinusfunktion:

```
>>> sin(45)
Traceback (most recent call last):
  File "<pyshell#33>", line 1, in <module>
    sin(45)
NameError: name 'sin' is not defined
```

Wir erhalten als Fehler einen NameError.

Wir müssen also immer den Pfad zur Funktion „math“ angeben, um die Funktion nutzen zu können.

```
>>> math.sin(45)
0.8509035245341184
```