

(c) 2007 Michael Goerz <goerz@physik.fu-berlin.de>
<http://www.physik.fu-berlin.de/~goerz/>
 Information taken liberally from the python documentation and various other sources.
 You may freely modify and distribute this document.

1.1 Numbers

1.2 Sequences

(lists are mutable, tuples and strings are immutable)	
s=l=[1, "bla", [1+2J, 1.4], 4]	list creation
s=t=(1, "bla", [1+2J, 1.4], 4)	tuple creation
l=list(t); t=tuple(l)	list/tuple conversion
l=range(1000)	list of integers (0-999)
s=xrange(1000)	immut. xrange-sequence
i=iter(s); i.next()	iterator from sequence
s[2][0]	get list element (1+2J)
s[-2][-1]	get list element (1.4)
s1+s1	sequence concat
n*s1	repeat s1 n times
s[i:j]; s[i:]; s[:j]	slicing (i incl., j excl.)
s[i:j:k]	slice with stride k
s[::-2]; s[::-1]	every 2 nd Element / reverse s
x in s; x not in s	is x a member of s?
len(s)	number of elements
min(s); max(s)	min/max
l[i:j]=['a','b','c','d']	replace slice
l[i:i]=['a','b']	insert before position i
l.count(x)	number of occurrences of x
l.index(x)	first index of x, or error
l.append(x)	append x at end of l
x=l.pop()	pop off last element
l.extend(l2)	append l2 at end of l
l.insert(i,x)	insert x at pos. i
l.remove(x)	delete first x
l.reverse()	reverse l
l.sort(f)	sort using f (default f=cmp)
zip(s,t,...)	[(s[0],t[0]),...],...

d={'x':42, 'y':3.14, 'z':7}	dict creation
d['x']	get entry for 'x'
len(d)	number of keys
del(d['x'])	delete entry from dict
d.copy()	create shallow copy
d.has_key(k)	does key exist?
d.items()	list of all items
d.keys()	list of all keys
d.values()	list of all values
i=d.iteritems(); i.next()	iterator over items
i=d.iterkeys(); i.next()	iterator over keys
i=d.itervalues(); i.next()	iterator over values
d.get(k,x)	get entry for k, or return x
d.clear()	remove all items
d.setdefault(k,x)	return d[k] or set d[k]=x
d.popitem()	return and delete an item

<code>s=set(s); fs=frozenset(s)</code>	create set
<code>fs.issubset(t); s<=t</code>	all <i>s</i> in <i>t</i> ?
<code>fs.issuperset(t); s>=t</code>	all <i>t</i> in <i>s</i> ?
<code>fs.union(t); s t</code>	all elements from <i>s</i> and <i>t</i>
<code>fs.intersection(t); s&t</code>	elements both in <i>s</i> and <i>t</i>
<code>fs.difference(t); s-t</code>	all <i>s</i> not in <i>t</i>
<code>fs.symmetric_difference(t); s^t</code>	all either <i>s</i> or <i>t</i>
<code>fs.copy()</code>	shallow copy of <i>s</i>
<code>s.update(t); s =t</code>	add elements of <i>t</i>
<code>s.intersection_update(t); s&=t</code>	keep only what is also in <i>t</i>
<code>s.difference_update(t); s-=t</code>	remove elements of <i>t</i>
<code>s.symmetric_differ... (t); s^=t</code>	keep only symm. difference
<code>s.add(x)</code>	add <i>x</i> to <i>fs</i>
<code>s.remove(x); fs.discard(x);</code>	remove <i>x</i> (/ with exception)
<code>s.pop();</code>	return and remove any elem.
<code>s.clear();</code>	remove all elements

<code>"bla"; 'hello "world"'</code>	string (of bytes)
<code>"""bla"""', '''bla'''</code>	triple quotes for multiline
<code>\\\ \\\ \0</code>	cont., backslash, null char
<code>\N{id} \uhhhh \Uhhhhhhhhh</code>	unicode char
<code>\xhh \ooo</code>	hex, octal byte
<code>u"Ûnic\u00F8de"; u"F8"</code>	unicode string (of characters)
<code>r"C:\new\text.dat"; ur"\ÿ"</code>	raw string (unicode)
<code>str(3.14); str(42)</code>	string conversion
<code>"%s-%s-%s" % (42,3.14,[1,2,3])</code>	string formatting
<code>'\t'.join(seq)</code>	join sequences with separator
<code>s.decode('utf-8')</code>	latin-1 string to unicode string
<code>u.encode('utf-8')</code>	unicode string to utf-8 string
<code>chr(i), unichr(i)</code>	char from code point
<code>str(x)</code>	string from number/object

```
search and replace: find(s,b,e), rfind(s,b,e),
                    index(s,b,e), rindex(s,b,e), count(s,b,e),
                    endswith(s,b,e), startswith(s,b,e), replace(o,n,m)
formatting: capitalize, lower, upper, swapcase, title
splitting: partition(s), rpartition(s), split(s,m),
```

- create set
- all s in t ?
- all t in s ?
- all elements from s and t
- elements both in s and t
- all s not in t
- all either s or t

- string (of bytes)
- triple quotes for multiline
- cont., backslash, null char
- unicode char
- hex, octal byte
- unicode string (of characters)
- raw string (unicode)
- string conversion
- string formatting
- join sequences with separator
- latin-1 string to unicode string
- unicode string to utf-8 string
- char from code point
- string from number/object

<h2>2 Basic Syntax</h2>	
if expr: statements	conditional
elif expr: statements	
else: statements	
if a is b : ...	object identity
if a == 1	value identity
while expr: statements	while loop
else: statements	run else on normal exit
while True: ... if cond: break	do... while equivalent
for target in iter: statements	for loop
else: statements	
for key,value in d.items():...	multiple identifiers
break, continue	end loop / jump to next
print "hello world",	print without newline
[expr for x in seq lc]	list comprehension
lc = for x in seq / if expr	with lc-clauses
pass	empty statement
def f(params): statements	function definition
def f(x, y=0): return x+y	optional parameter
def f(*a1, **a2): statements	additional list of unnamed,
	dict of named paramters
def f(): f.variable = 1 ...	function attribute
return expression	return from function
yield expression	make function a generator
f(1,1), f(2), f(y=3, x=4)	function calls
global v	bind to global variable
def make_adder_2(a):	closure
def add(b): return a+b	
return add	
lambda x: x+a	lambda expression

```
compile(string,filename,kind)
eval(expr,globals,locals)
exec code in gldict, lcdict
execfile(file,globals,locals)
raw_input(prompt)
input(prompt)
```

```
compile string into code object
evaluate expression
compile and execute code
execute file
input from stdin
input and evaluate
```

```
assert expression
class MyExcept(Exception): ...
raise MyExcept , data
```

```
debug assertion
define user exception
raise user exception
```

5 System Interaction

```
sys.path
sys.platform
sys.stdout, stdin, stderr
sys.argv[1:]
os.system(cmd)
os.startfile(f)
os.popen(cmd, r|w, bufsize)
os.popen2(cmd, bufsize, b|t)
os.popen3(cmd, bufsize, b|t)
os.environ['VAR']; os.putenv[]
glob.glob('*.*txt')
```

Filesystem Operations

os module: access, chdir, chmod, chroot, getcwd, getenv, listdir, mkdir, remove, unlink, removedirs, rename, rmdir, getatime, getmtime, getsize, cmp, cmpfiles, dircmp, copy, copy2, copyfile, copyfileobj, copymode, copystat, copytree, rmtree, pipe

os.path module: abspath, altsep, basename, commonprefix, curdir, defpath, dirname, exists, expanduser, expandvar, extsep, get[acm]time, getsize, isabs, isdir, isfile, islink, ismout, join, lexists, normcase, normpath, os, pardir, pathsep, realpath, samefile, sameopenfile, samestat, sep, split, splitdrive, splitext, stat, walk

command line argument parsing:

```
restlist, opts = \
    getopt.getopt(sys.argv[1:], \
        "s:oh", \
        ["spam=", "other", "help"])
for o, a in opts:
    if o in ("-s", "--lol"): spam = a
    if o in ("-h", "--help"): show_help()
```

6 Input/Output

```
f=codecs.open(if,"rb","utf-8")
file = open(infilename, "wb")
codecs.EncodedFile(...)
r, w, a, r+
rb, wb, ab, r+b
file.read(N)
file.readline()
file.readlines()
file.write(string)
file.writelines(list)
file.close()
file.tell()
file.seek(offset, whence)
os.truncate(size)
os.tmpfile()
pickle.dump(x, file)
x = pickle.load(file)
```

```
module search path
operating system
standard input/output/error
command line parameters
system call
open file with assoc. program
open pipe (file object)
(stdin,stdout) fileobjects
(stdin,stdout,stderr)
read/write environment vars
wildcard search
```

```
open file with encoding
open file without encoding
wrap file into encoding
read, write, append, random
modes without eol conversion
N bytes ( entire file if no N )
the next linestring
list of linestring
write string to file
write list of linestrings
close file
current file position
jump to file position
limit output to size
open anon temporary file
make object persistent
load object from file
```

7 Standard Library (almost complete)

String Services: string, re, struct, difflib, StringIO, cStringIO, textwrap, codecs, unicodedata, stringprep, fpformat

Data Types: datetime, calendar, collections, heapq, bisect, array, sets, sched, mutex, Queue, weakref, UserDict, UserList, UserString, types, new, copy, pprint, repr

Numeric and Math Modules: math, cmath, decimal, random, itertools, functools, operator

Internet Data Handling: email, mailcap, mailbox, mhlib, mimetools, mimetypes, MimeWriter, mimify, multifile, rfc822, base64, binhex, binascii, quopri, uu

Structured Markup Processing Tools: HTMLParser, sgmlib, htmllib, htmlentitydefs, xml.parsers.expat, xml.dom.*, xml.sax.*, xml.etree.ElementTree

File Formats: csv, ConfigParser, robotparser, netrc, xdrlib

Crypto Services: hashlib, hmac, md5, sha

File/Directory Access: os.path, fileinput, stat, statvfs, filecmp, tempfile, glob, fnmatch, linecache, shutil, dircache

Compression: zlib, gzip, bz2, zipfile, tarfile

Persistence: pickle, cPickle, copy_reg, shelve, marshal, anydbm, whichdb, dbm, gdbm, dbhash, bsddb, dumbdbm, sqldb3

Generic OS services: os, time, optparse, getopt, logging, getpass, curses, platform, errno, ctypes

Optional OS services: select, thread, threading, dummy_thread, dummy_threading, mmap, readline, rlcompleter

Unix specific: posix, pwd, spwd, grp, crypt, dl, termios, tty, pty, fcntl, posixfile, resource, nis, syslog, commands

IPC/Networking: subprocess, socket, signal, popen2, asyncore, asynchat

Internet: webbrowser, cgi, scitb, wsgiref, urllib, httplib, ftplib, imaplib, nntplib, ...lib, smtpd, uuid, urlparse, SocketServer, ...Server,, cookielib, Cookie, xmlrpclib

Multimedia: audioop, imageop, aifc, sunau, wave, chunk, colorsys, rgbimg, imghdr, sndhdr, ossaudiodev

Tk: Tkinter, Tix, ScrolledText, turtle

Internationalization: gettext, locale

Program Frameworks: cmd, shlex

Development: pydoc, doctest, unittest, test

Runtime: sys, warnings, contextlib, atexit, traceback, qc, inspect, site, user, fpectl

Custom Interpreters: code, codeop

Restricted Execution: rexec, Bastion

Importing: imp, zipimport, pkgutil, modulefinder, runpy

Language: parser, symbol, token, keyword, tokenize, tabnanny, pycbr, py_compile, compileall, dis, pickletools, distutils

Windows: msilib, msvcrt, _winreq, winsound

Misc: formatter

3 Object Orientation and Modules

```
import module
from module import name1,name2
from __future__ import *
reload module
module.__all__
module.__name__
module.__dict__
__import__ ("name", glb, loc, fl)
class name (superclass,...):
    data = value
    def method(self,...): ...
    def __init__(self, x):
        Super.__init__(self)
        self.member = x
    def __del__(self): ...
__str__, __len__, __cmp__, __
__iter__(self): return self
__call__
__dict__
__getattr__(self, name),
__setattr__(self, name, value)
callable(object)
delattr(object, "name")
del(object)
dir(object)
getattr(object, "name", def)
hasattr(object, "name")
hash(object)
id(object)
isinstance(object,
classOrType)
issubclass(class1, class2)
iter(object, sentinel)
locals()
repr(object), str(object)
vars(object)
None
if __name__ == "__main__":
```

```
import module
load attr. into own namespace
activate all new features
reinitialize module
exported attributes
module name / "__main__"
module namespace
import module by name
class definition
shared class data
methods
constructor
call superclass constructor
per-instance data
destructor
some operator overloaders
use next method for iterator
call interceptor
instance-attribute dictionary
get an unknown attribute
set any attribute
1 if callable, 0 otherwise
delete name-attr. from object
unreference object/var
list of attr. assoc. with object
get name-attr. from object
check if object has attr.
return hash for object
unique integer (mem address)
check for type
```

```
class2 subclass of class1?
return iterator for object
dict of local vars of caller
return string-representation
return __dict__
the NULL object
make modul executable
```

```
Try-block
catch exception
multiple, with data
exception handling
pass up (re-raise) exception
if no exception occurred
in any case
```

4 Exception Handling

```
try: ...
except ExceptionName:
except (Ex1, ...), data:
    print data
    raise
else: ...
finally: ...
```