Python 2.5 Reference Card

(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 Variable Types

1.1 Numbers

```
42 052 0x2A 42L 052L 0x2AL
0.2 .8 4. 1.e10 1.0e-7
z = 5.0 - 2.0J:
z = complex(real. imag)
z.real: z.imag
True: False
abs(n)
divmod(x, v)
hex(n)
oct(n)
ord(c)
round(x,n)
cmp(x,y)
coerce(x, y)
pow(x, y, z)
float("3.14")
int("42", base)
import math; import cmath
import random;
```

complex number complex number real and imag part of z constants for boolean values absolute value of n (x/v, x%v)create hex string create octal string unicode code point of char round x to n decimal places x < y: -1, x == y: 0, x > y: 1(x,y), make same type (x**y) % z float from string int from string more math functions

42 (dec, oct, hex, short/long)

floating point value

1.2 Sequences (lists are mutable, tuples and strings are immutable) s=1=[1, "bla", [1+2J, 1.4], 4]s=t=(1, "bla", [1+2J, 1.4], 4) l=list(t); t=tuple(l) l=range(1000) s=xrange(1000) i=iter(s); i.next() s[2][0] s[-2][-1] s1+s1 n*s1 s[i:j]; s[i:]; s[:j] s[i:j:k] s[::2]; s[::-1] x in s; x not in s len(s) min(s); max(s)l[i:i]=['a','b','c','d'] l[i:i]=['a','b'] 1.count(x) l.index(x) 1.append(x) x=1.pop()1.extend(12)

1.insert(i.x)

1.remove(x)

1.reverse()

zip(s,t,...)

1.sort(f)

random number generators list creation tuple creation list/tuple conversion list of integers (0-999) immut. xrange-sequence iterator from sequence get list element (1+2J) get list element (1.4) sequence concat repeat s1 n times slicing (i incl., j excl.) slice with stride k every 2nd Element / reverse s is x a member of s? number of elements min/max replace slice insert before position i number of occurances of x first index of x, or error append x at end of 1 pop off last element append 12 at end of 1 instert x at pos. i delete first x

[(s[0],t[0],...),..]

reverse 1

1.3 Dictionaries (Mappings)

```
d=\{ 'x':42, 'y':3.14, 'z':7 \}
d['x']
len(d)
del(d['x'])
d.copy()
d.has kev(k)
d.items()
d.keys()
d.values()
i=d.iteritems(): i.next()
i=d.iterkevs(): i.next()
i=d.itervalues(): i.next()
d.get(k.x)
d.clear()
d.setdefault(k,x)
d.popitem()
```

```
1.4 Sets
s=set(s); fs=frozenset(s)
fs.issubset(t): s<=t
fs.issuperset(t): s>=t
fs.union(t); s|t
fs.intersection(t); s&t
fs.difference(t); s-t
fs.symmetric difference(t);s^t
fs.copy()
s.update(t); s|=t
s.intersection update(t); s&=t keep only what is also in t
s.difference update(t); s-=t
s.symmetric differ...(t); s^=t
s.add(x)
s.remove(x); fs.discard(x);
s.pop():
s.clear():
```

1.5 Strings and Regular Expressions

```
"bla"; 'hello "world"'
"""bla""", '''bla'''
\ \\
\N{id} \uhhhh \Uhhhhhhhh
\xhh \ooo
u"Ünic\u00F8de"; u"\xF8"
r"C:\new\text.dat"; ur"\\Ü"
str(3.14); str(42)
"s-%s-%s" % (42,3.14,[1,2,3])
'\t'.join(seg)
s.decode('utf-8')
u.encode('utf-8')
chr(i). unichr(i)
str(x)
```

Other String Methods:

```
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
sort using f (default f = cmp) splitting: partition(s), rpartition(s), split(s,m),
                          rsplit(s.m). splitlines(ke)
```

dict creation get entry for 'x' number of kevs delete entry from dict create shallow copy does kev exist? list of all items list of all keys list of all values iterator over items iterator over keys iterator over values get entry for k, or return x remove all items return d[k] or set d[k]=x return and delete an item

create set alls int? 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 shallow copy of s add elements of t remove elements of t keep only symm, difference add x to fs remove x (/ with exception) return and remove any elem. remove all elements

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

```
padding: center(w.c). liust(w.c). lstrip(cs).
   rjust(w,c), rstrip(cs), strip(cs), zfill(w),
   expandtabs(ts)
checking: isalnum. isalpha. isdigit. islower. isspace.
   istitle. isupper
String Constants: import string
   digits, hexdigits, letters, lowercase, octdigits,
   printable, punctuation, uppercase, whitespace
Regexes: import re
```

```
r=re.compile(r'rx',re.ILMSUX)
(?P<id>...)
m=r.match(s.b.e)
re.match(r'(?iLmsux)rx'.s)
m=r.search(s,b,e)
l=r.split(s,ms)
l=r.findall(string)
s=r.sub(s,r,c)
(s.n) = r.subn(s.r.c)
s=re.escape(s)
m.start(g);m.span(g);m.end(g)
m.expand(s)
m.group(g); m.group("name")
m.groups()
m.groupdict()
```

comile 'rx' as regex named group full match direct regex usage partial match split and return list list of all matched groups replace c counts of s with r n is number of replacements escape all non-alphanumerics group-match delimiters replace \1 etc. with matches matched group no. g list of groups dict of named groups

2 Basic Syntax

```
if expr: statements
elif expr: statements
else: statements
if a is b : ...
if a == 1
while expr: statements
else: statements
while True: ... if cond: break
for target in iter: statements for loop
else: statements
for kev.value in d.items():...
break, continue
print "hello world",
[ expr for x in seq lc ]
  lc = for x in seg / if expr
def f(params): statements
def f(x, v=0): return x+v
def f(*a1, **a2): statements
def f(): f.variable = 1 ...
```

return expression yield expression f(1.1), f(2), f(v=3, x=4)global v def make adder 2(a): def add(b): return a+b

return add lambda x: x+a

compile(string,filename,kind)

conditional

object identity value identity while loop run else on normal exit do... while equivalent

multiple identifiers end loop / jump to next print without newline list comprehension with lc-clauses empty statement function definition optional parameter additional list of unnamed, dict of named paramters function attribute return from function make function a generator function calls bind to global variable closure

lambda expression compile string into code object

eval(expr,globals,locals) exec code in gldict, lcdict execfile(file,globals,locals) raw input(prompt) input(prompt)

evaluate expression compile and execute code execute file input from stdin input and evaluate

3 Object Orientation and Modules

```
import module as alias
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

4 Exception Handling

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

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

class MyExcept (Exception): ... define user exception raise MyExcept , data

5 System Interaction

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

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 rlcompleter read/write environment vars wildcard search

raise user exception

Filesystem Operations listdir. mkdir. remove. unlink. removedirs. rename.

rmdir, pipe, ...

shutil module: copy, copy2, copyfile, copyfileobj, copymode, copystat, copytree, rmtree

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, 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)
```

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

open file with encoding

7 Standard Library (almost complete)

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

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

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,

(stdin, stdout, stderr) Data Types: datetime, calendar, collections, heapq, bisect, array, sets, sched, mutex, Oueue, weakref, UserDict, UserList, UserString, types, new, copy, pprint, repr

os module: access, chdir, chmod, chroot, getcwd, getenv, 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, scmllib. 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 Compression: zlib, gzip, bz2, zipfile, tarfile

Persistence: pickle, cPickle, copy reg, shelve, marshal, anydbm, whichdb, dbm, gdbm, dbhash, bsddb, dumbdbm, sglite3

Unix specific: posix, pwd, spwd, grp, crypt, dl, termios, ttv. ptv. fcntl. posixfile. resource. nis. svslog. commands

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

Internet: webbrowser, cqi, 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: svs. 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, pyclbr, py compile, compileall, dis, pickletools. distutils

Windows: msilib, msvcrt, winreq, winsound

Misc: formatter