Python 標準ライブラリ概観

[Python][言語まとめ Python][Python サンプルコード][言語まとめ Python]
[Python ライブラリ]

- Python チュートリアル
  - http://www.python.jp/doc/release/tutorial/
- Python 標準ライブラリ
  - http://www.python.jp/doc/release/library/index.html#library-index
- グローバルモジュールインデクス
  - http://www.python.jp/doc/release/modindex.html
- Python ライブラリリファレンス
  - http://www.python.jp/doc/release/lib/contents.html
- 日本 Python ユーザ会（PyJUG）
  - http://www.python.jp/Zope/

OS インターフェース [os]

■カレント作業ディレクトリの取得

取得

```
>>> import os
>>> os.getcwd()
'C:\\Programs\\Python25'
```

設定

```
>>> os.chdir(r'c:\work\py')
>>> os.getcwd()
'c:\\work\\py'
```

■環境変数の参照

```
>>> print os.environ['PATH']
/usr/local/pgsql/bin:/usr/lib/oracle/xe/app/oracle/product/10.2.0/server/bin: ･･･
```

■環境変数の設定

```
>>> os.environ["PHRASE_ENV"] = "test"
>>> os.environ.get("PHRASE_ENV")
'test'
```

■現在動いている Python モジュールの 絶対パスを取得

```
import os
print os.path.abspath(os.path.dirname(__file__))
```

■ファイルの存在を調べる

```
>>>import os.path
>>>os.path.exists(r'/test.txt')
True
```

■ファイルの情報を取得

```
>>> import os
>>> r = os.stat(r'/test.txt')
>>> r
nt.stat_result(st_mode=33206, st_ino=0L, st_dev=0, st_nlink=0, st_uid=0, st_gid=0, st_size=4508L,
st_atime=1266996480L, st_mtime=1266452528L, st_ctime=1251076742L)
```

■ファイルのタイムスタンプを取得

```
>>> import os
>>> os.stat('test.txt').st_mtime
1457156166.75
```

■絶対パスからファイル名のみを取得

・os.path.basename

```
>>> import os
>>> os.path.basename('c/work/test.txt')
'test.txt'
```

ファイルのワイルドカード [glob]

```
>>> import glob
>>> glob.glob('*.exe')
['python.exe', 'pythonw.exe', 'w9xpopen.exe']
```

httpd のアクセスログの内容を出力

```
import re
import glob
import os

file_pattern = r'/var/log/httpd/access*'

for file_name in glob.glob(file_pattern):
    if os.path.isfile(file_name):
        f = open(file_name)
        for line in f:
            print line
```

ファイルの文字コード (codecs)

・文字コードを指定して、ファイルを開く

```
import codecs
fd = codecs.open(search_result_file, 'r', 'utf-8')
    for l in fd:
       print l
```

open(filename, mode='rb', encoding=None, errors='strict', buffering=1):

・errors

```
'strict': raise an exception in case of an encoding error
'replace': replace malformed data with a suitable replacement marker, such as '?' or '\ufffd'
'ignore': ignore malformed data and continue without further notice
```

```
'xmlcharrefreplace': replace with the appropriate XML character reference (for encoding only)
'backslashreplace': replace with backslashed escape sequences (for encoding only)
```

コマンドライン引数 [sys]

```
import sys

print sys.argv
```

結果

```
c:\work\py>python test.py 1 2 3
['test.py', '1', '2', '3']
```

コマンドラインオプション [getopt]

```
getopt(args, options[, long_options])
```

options はスクリプトで認識させたいオプション文字と、引数が必要な場 合にはコロン (":") をつけます。

```
import getopt, sys

def main():
    try:
        opts, args = getopt.getopt(sys.argv[1:],"f:c:")
    except getopt.GetoptError:
        usage()
        sys.exit(2)
    content_file = None
    code_set = None
    print opts
    for o, a in opts:
        if o == '-f':
            content_file = a
        if o == '-c':
            code_set = a
```

標準エラーを出力しプログラムを終了 [sys]

```
import sys

sys.stderr.write('System Standerd Error\n')
sys.exit()
```

正規表現を利用する [re]

http://www.python.jp/doc/2.5/lib/re-objects.html

http://www.python.jp/Zope/articles/tips/regex_howto/regex_howto_4

■正規表現による文字列の置換

・sub(pattern, repl, string, count=0, flags=0) を利用する

```
>>> s = r'AaBbCc'
>>> re.sub(r'[a-z]',r'',s)
'ABC'
```

■一致箇所を抜き出す

```
>>> import re
>>> re.findall(r'\b[0-9][a-z]+', '1a,2b,cc,dd,4e')
['1a', '2b', '4e']
```

■グループ名を利用

英文字 + 連番 の連番部分を抜き出す

・?P< 名前 > でグループに名前付けを行うことができる

```
import re
ptn =  re.compile(r'[A-Z]+(?P<SEQ>[0-9]+)')
for line in lines:
    m = ptn.match(line)
    if m:
        print m.group('SEQ')
    else:
        print None
```

MatchObject

・http://www.python.jp/doc/2.5/lib/match-objects.html

もしグループが、複数回マッチしたパターンの一部に 含まれていれば、最後のマッチが返されます。

■正規表現による分割

・re.split を利用

```
>>> import re
>>> re.split('[ \t\n\.\,]', 'this is\ta\npen.')
['this', 'is', 'a', 'pen', '']
```

■コンパイル時のフラグ

フラグ	意味
DOTALL(S)	. が改行も含めて、全ての文字とマッチするように指定する
IGNORECASE(I)	大文字小文字を区別しない
LOCALE(L)	ロケールを考慮してマッチングを行う
MULTILINE(M)	複数行にマッチング。これは ^ と $ に影響する
VERBOSE(X)	冗長な正規表現（もっと、きれいで分かりやすくまとめられる表現）を有効にする。

数学 [math]

・Python 数学

```
>>> import math
>>> pow(2,2)
4
```

URL による任意のリソースへのアクセス [urllib]

■日本語を URL エンコーディングする

例

・マップ型オブジェクト、または 2 つの要素をもったタプルからなるシーケンスを、"URL にエンコードされた (url-encoded)" に変換して、urlopen() のオプション引数 data に適した形式にする。

```
import urllib

#KEY-VALUE のペアのシーケンス
p = [
    ('key1',u'日本語'.encode('utf-8')),
    ('key2',u'エンコード'.encode('utf-8')),
]
print urllib.urlencode(p)
```

結果

```
key1=%E6%97%A5%E6%9C%AC%E8%AA%9E&key2=%E3%82%A8%E3%83%B3%E3%82%B3%E3%83%BC%E3%83%89]
```

インターネットアクセス [urllib2]

■指定された URL を読む

```
>>> import urllib2
>>> f = urllib2.urlopen('http://typea.info')
>>> print f.read(100)
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

■プロキシサーバーを経由

```
proxy_support = urllib2.ProxyHandler({"http" : "http://proxyhost:8080"})
opener = urllib2.build_opener(proxy_support)
urllib2.install_opener(opener)

url = r'http://hogehoge'
f = urllib2.urlopen(url_idx)
for l in f:
    print l
```

■ CGI にデータを送信し結果を得る

```
>>> import urllib2
>>> req = urllib2.Request(url='http://typea.info/tips/wiki.cgi',data='action=RSS')
>>> f = urllib2.urlopen(req)
>>> print f.read()
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rdf:RDF
 xmlns="http://purl.org/rss/1.0/"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xml:lang="ja"
>
<channel rdf:about="http://typea.info/tips/wiki.cgi?action=LIST">
                :
```

■ POST リクエストを送信する

例

・urllib2.urlopen の第 2 引数に、application/x-www-form-urlencoded 形式のデータを設定
・application/x-www-form-urlencoded 形式のデータは、urllib.urlencode にマップ型か 2 タプルのシーケンスを渡すことで得られる。

```
param_map = {'appid':self.aid,
             'sentence':sentence,
             'filter':'9',
            }
url = r'http://developer.yahoo.co.jp/webapi/jlp/ma/v1/parse.html'
data =  urllib.urlencode(param_map)
f = urllib2.urlopen(url, data)
```

■ CGI のサポート (cgi)

・http://www.python.jp/doc/release/lib/module-cgi.html

■ HTML タグのエスケープ

```
>>> from xml.sax.saxutils import escape
>>> escape("<div>")
'&lt;div&gt;'
```

日付と時刻 [date,datetime,time]

■基本

```
>>> from datetime import date
>>> now = date.today()
>>> now
datetime.date(2009, 6, 19)
>>> birthday = date(1971, 9, 30)
>>> birthday
datetime.date(1971, 9, 30)
>>> age = now - birthday
>>> age.days
13777
```

■タイムスタンプ

・ISO 8601

```
>>> from datetime import datetime
>>> datetime.now().isoformat()
'2009-07-29T16:44:56.631000'
>>> datetime.utcnow().isoformat()
'2009-07-29T07:45:05.227000'
```

■タイムスタンプから、datetime オブジェクトを作成

```
>>> from datetime import datetime
>>> ts = os.stat(r'/test.txt').st_ctime
>>> datetime.fromtimestamp(ts).isoformat()
'2009-08-24T10:19:02.118895'
```

■時間の演算 (timedelta)

・timedelta を利用する

属性
days
seconds
microseconds

```
>>> from datetime import date
>>> from datetime import timedelta
>>> today = date.today()
>>> today
datetime.date(2010, 1, 25)
>>> new_dt = today - timedelta(days=10)
>>> new_dt
datetime.date(2010, 1, 15)
```

■日付の書式設定

```
>>> time.strftime('%Y/%m/%d', time.localtime())
'2010/02/24'
>>> datetime.today().strftime('%Y%m%d')
'20100224'
>>> datetime.fromtimestamp(os.stat(r'/test.txt').st_ctime).strftime('%Y%m%d')
```

'20090824'

・http://docs.python.jp/2/library/datetime.html

指定子	意味	備考
%a	ロケールの短縮された曜日名を表示します	
%A	ロケールの曜日名を表示します	
%b	ロケールの短縮された月名を表示します	
%B	ロケールの月名を表示します	
%c	ロケールの日時を適切な形式で表示します	
%d	月中の日にちを 10 進表記した文字列 [01,31] を表示します	
%f	マイクロ秒を 10 進表記した文字列 [000000,999999] を表示します (左側から 0 埋めされます)	(1)
%H	時 (24 時間表記) を 10 進表記した文字列 [00,23] を表示します	
%I	時 (12 時間表記) を 10 進表記した文字列 [01,12] を表示します	

%j	年中の日にちを 10 進表記した文字列 [001,366] を表示します	
%m	月を 10 進表記した文字列 [01,12] を表示します	
%M	分を 10 進表記した文字列 [00,59] を表示します	
%p	ロケールの AM もしくは PM を表示します	(2)
%S	秒を 10 進表記した文字列 [00,61] を表示します	(3)
%U	年中の週番号 (週の始まりは日曜日とする) を 10 進表記した文字列 [00,53] を表示します新年の最初の日曜日に先立つ日は 0 週に属するとします	(4)
%w	曜日を 10 進表記した文字列 [0(日曜日),6] を表示します	
%W	年中の週番号 (週の始まりは月曜日とする) を 10 進表記した文字列 [00,53] を表示します新年の最初の月曜日に先立つ日は 0 週に属するとします	(4)
%x	ロケールの日付を適切な形式で表示します	
%X	ロケールの時間を適切な形式で表示します	
%y	世紀なしの年 (下 2 桁) を 10 進表記した文字列 [00,99] を表示します	
%Y	世紀ありの年を 10 進表記した文字列を表示します	
%z	UTC オフセットを +HHMM もしくは -HHMM の形式で表示します (オブジェクトが naive であれば空文字列)	(5)
%Z	タイムゾーンの名前を表示します (オブジェクトが naive であれば空文字列)	
%%	文字 '%' を表示します	

1.strptime() メソッドと共に使われた場合、 %f 指定子は 1 桁から 6 桁の数字を受け付け、右側から 0 埋めされます。 %f は C 標準規格の書式セットに拡張されます。
2.strptime() メソッドと共に使われた場合、 %p 指定子は出力の時間フィールドのみに影響し、 %I 指定子が使われたかのように振る舞います。
3. 範囲は 0 から 61 で正しいです ; これはうるう秒と、 (極めて稀ですが) 2 秒のうるう秒を考慮してのことです。
4.strptime() メソッドと共に使われた場合、 %U と %W 指定子は、年と曜日が指定された場合の計算でのみ使われます。
5. 例えば、 utcoffset() が timedelta(hours=-3, minutes=-30) を返すとしたら、 %z は文字列、'-0330' で置き換えられます。

時間計算 [time]

■一定時間待つ

```
>>> import time
>>> time.sleep(2)
```

データ圧縮 [zlib]

```
>>> import zlib
>>> s = 'Common data archiving and compression formats are directly supported by
 modules including: zlib, gzip, bz2, zipfile, and tarfile. '
>>> len(s)
130
>>> t = zlib.compress(s)
>>> len(t)
111
>>> zlib.decompress(t)
'Common data archiving and compression formats are directly supported by modules
 including: zlib, gzip, bz2, zipfile, and tarfile. '
>>> zlib.crc32(s)
1515784898
```

パフォーマンス計測 [timeit]

・以下例は、ステートメントが、a * b 初期値 a=2、b=5 の処理時間を計測

```
>>> from timeit import Timer
>>> Timer('a * b', 'a=2;b=5').timeit();
0.12105141854635804
```

品質管理 [doctest]

・テストを各関数に記述し、開発中頻繁に実行するという高品質への開発アプローチがある
・doctest モジュールは、モジュールをスキャンし、プログラム上のドキュメント (docstrings) に埋め込まれたテストを検証する

・+1 した値を返す inc() 関数に、+2 した値を返すバグがある例

```
>>> def inc(val):
...     """+1 して返す
...     >>> print inc(10)
...     11
...     """
...
...     return val + 2
...
>>> import doctest
>>> doctest.testmod()
**********************************************
```

```
File "__main__", line 3, in __main__.inc
Failed example:
    print inc(10)
Expected:
    11
Got:
    12
**********************************************
1 items had failures:
   1 of   1 in __main__.inc
***Test Failed*** 1 failures.
(1, 1)
```

出力書式

■ repr モジュール [repr]

・repr モジュールは 巨大で深くネストした内容を簡略表示する、repr() 関数のカスタム版を提供

repr() 関数使用

```
>>> import os
>>> repr(dir(os))
 "['F_OK',  'O_APPEND',  'O_BINARY',  'O_CREAT',  'O_EXCL',  'O_NOINHERIT',  'O_RANDOM',  'O_RDONLY',
'O_RDWR',  'O_SEQUENTIAL',  'O_SHORT_LIVED',  'O_TEMPORARY',  'O_TEXT',  'O_TRUNC',  'O_WRONLY',
'P_DETACH',  'P_NOWAIT',  'P_NOWAITO',  'P_OVERLAY',  'P_WAIT',  'R_OK',  'SEEK_CUR',  'SEEK_END',
'SEEK_SET', 'TMP_MAX',
 'UserDict',  'W_OK',  'X_OK',  '_Environ',  '__all__',  '__builtins__',  '__doc__',  '__file__',
'__name__',  '_copy_reg',  '_execvpe',  '_exists',  '_exit',  '_get_exports_list',  '_make_stat_result',
'_make_statvfs_result', '_pickle_stat_result', '_pickle_statvfs_result', 'abort', 'access', 'altsep',
'chdir','chmod','lossee','curdiir'    defppaath    'devnnuull    'dup'',,    'p222 ',    envviron,
'errnoo','rroorr','execll','xecclle''execcllp''execcllpe     'execcvv',execcvve''execcvvp''execcvvpe
'extsseep''fdoppeen''fstaatt',fsynncc',getccwwd''getccwwdu   'geteennv''getppiid''isatttty''lineessep
'listtddir 'lseeekk',lstaatt
  'maakkedirs  'mkdirr'',  ame'',,  'open',  'pardir','ath'',,  'pathsep'  pipe'',,  'popen',  ppenn22
',popeen33',popeen44',puteennv',read'',,  'remove',  'removedis',  'renammme','renamees',  'rmdir',  sp',,
'swnnnl','pawnnlle''spawnnvv',spawnnvve''starttffil,  'stat''',  'stat_float_times',  'stat_result',
'statvf
_esulltt', strerroor' sys',  ''syemmm', 'empnamm','imes'',, 'tmpfile','mpnamm'', mask'',, 'linkkk',
'nsetennv' urandoomm',utime'',, 'itpiiid','alk',,  'write']"
```

repr モジュール使用

```
>>> import os
>>> import repr
>>> repr.repr(dir(os))
"['F_OK', 'O_APPEND', 'O_BINARY', 'O_CREAT', 'O_EXCL', 'O_NOINHERIT', ...]"
```

■ pprint モジュール [pprint]

・pprint モジュール はビルトインおよびユーザ定義オブジェクトについて、インタプリタにより、もっと読みやすい出力を提供する。
・1 行で表示するには長すぎる場合、構造が明確になるように改行とインデントを挿入する。

ソース

```
import os
import pprint

# ディレクトリを走査し、階層構造を持ったリストを生成
def trav(path, fn=''):
```

```
  l = []
  l.append(fn)
  d = os.path.join(path,fn)
  if os.path.isdir(d):
    for f in os.listdir(d):
      l.append(trav(d,f))
  return l

l = trav('C:\\', 'Python25')

# 階層構造を持ったリストを出力
pprint.pprint(l, indent=4, width=80)
```

結果

```
[   'Python25',
    [   'DLLs',
        ['bz2.pyd'],
        ['py.ico'],
        ['pyc.ico'],
〜 略 〜
        ['_ssl.pyd'],
        ['_testcapi.pyd'],
        ['_tkinter.pyd']],
    [   'Doc',
        ['about.html'],
        ['acks.html'],
        [   'api',
            ['about.html'],
            ['abstract-buffer.html'],
            ['boolObjects.html'],
            ['buffer-structs.html'],
 〜 略 〜
```

■ textwrap **モジュール** [textwrap]

・textwrap モジュールは、文章の段落を画面幅に合わせるようにフォーマットする

```
>>> import textwrap
>>> txt = """The pprint module offers more sophisticated control over printing
... both built-in and user defined objects in a way that is readable
... by the interpreter. When the result is longer than one line,
... the ``pretty printer'' adds line breaks and indentation to more
... clearly reveal data structure: """
>>>
>>> print textwrap.fill(txt, width=20)
The pprint module
offers more
sophisticated
control over
printing both built-
in and user defined
objects in a way
that is readable by
the interpreter.
When the result is
longer than one
line, the ``pretty
printer'' adds line
breaks and
indentation to more
clearly reveal data
structure:
```

■ locale module **モジュール** [locale]

・locale module モジュールは、文化圏特有の書式データベースにアクセスする
・地域による数値の区切書式を locale.format() 関数から直接指定できる
・RFC1766

locale を使用する

```
>>> import locale
>>> locale.setlocale(locale.LC_ALL, 'ja-JP')
'Japanese_Japan.932'
>>> x = 1234567.8
>>> locale.format('%d', x , grouping=True)
'1,234,567'
>>> conv = locale.localeconv()
>>> locale.format('%s%.*f', (conv['currency_symbol'], conv['frac_digits'], x), grouping=True)
'\\1,234,568'
```

変換 Map(locale.localeconv()) の内容

```
>>> import pprint
>>> pprint.pprint(conv)
{'currency_symbol': '\\',
 'decimal_point': '.',
 'frac_digits': 0,
 'grouping': [3, 0],
 'int_curr_symbol': 'JPY',
 'int_frac_digits': 2,
 'mon_decimal_point': '.',
 'mon_grouping': [3, 0],
 'mon_thousands_sep': ',',
 'n_cs_precedes': 1,
 'n_sep_by_space': 0,
 'n_sign_posn': 3,
 'negative_sign': '-',
 'p_cs_precedes': 1,
 'p_sep_by_space': 0,
 'p_sign_posn': 3,
 'positive_sign': '',
 'thousands_sep': ','}
```

テンプレート

■テンプレート [Template]

・string モジュールは、Template クラスを含んでいる。
・アプリケーションを変更することなしにカスタマイズできる
・プレイスホルダーの書式は、 $ + Python の識別子
・{} で囲むことで、スペースで区切られていなくても良い
・$$ で、$ をエスケープする

```
>>> from string import Template
>>> t = Template('<input type="$element_name" value="hello${element_value}world!" />')
>>> t.substitute(element_name='text', element_value=' python ')
'<input type="text" value="hello python world!" />'
```

■ substitute メソッド と safe_substitute メソッド

・substitute メソッドはプレースホルダーと置き換える値がが提供されない場合 KeyError を引き起こす
・safe_substitute メソッドは、その場合、プレースホルダーをそのままにする。

```
>>> from string import Template
>>> t = Template('$field_a, $field_b')
>>> d = dict(field_a='FIELD_A')
>>> t.substitute(d)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
```

```
  File "C:\Python25\lib\string.py", line 170, in substitute
    return self.pattern.sub(convert, self.template)
  File "C:\Python25\lib\string.py", line 160, in convert
    val = mapping[named]
KeyError: 'field_b'
>>> t.safe_substitute(d)
'FIELD_A, $field_b'
```

■デリミタのカスタマイズ

・Templete のサブクラスでは、カスタムデリミタを使用できる。

```
>>> from string import Template
>>> class PctDlimTemplate(Template):
...     delimiter = '%'
...
>>> t = PctDlimTemplate('<input type="%element_name" value="hello%{element_value}world!" />')
>>> t.substitute(element_name='text', element_value=' python ')
'<input type="text" value="hello python world!" />'
```

シリアライズ [pickle]

■シリアライズする

Python オブジェクトをシリアライズする簡単な例

```
>>> d = { 'k1':123, 'k2':456, 'k3':789}
>>> d
{'k3': 789, 'k2': 456, 'k1': 123}
>>> import pickle
>>> f = open(r'c:\work\py\dat.txt', 'w')
>>> pickle.dump(d, f)
>>> f.close();
```

■デシリアライズする

```
>>> f = open(r'c:\work\py\dat.txt')
>>> e = pickle.load(f)
>>> e
{'k3': 789, 'k2': 456, 'k1': 123}
```

バイナリデータレコードレイアウトでの作業 [struct]

・struct モジュールは pack() 、unpack() 関数を可変長のバイナリレコードフォーマットの作業用に提供する

```
>>> import struct
>>> struct.pack('hhl',1,2,3)
'\x01\x00\x02\x00\x03\x00\x00\x00'
>>> struct.unpack('hhl','\x01\x00\x02\x00\x03\x00\x00\x00')
(1, 2, 3)
>>> struct.calcsize('hhl')
8
```

■書式

書式	C 言語の型	Python の型
x	pad byte	no value
c	char	string of length 1
b	signed char	integer

B	unsigned char	integer
h	short	integer
H	unsigned short	integer
i	int	integer
I	unsigned int	long
l	long	integer
L	unsigned long	long
q	long long	long
Q	unsigned long long	long
f	float	float
d	double	float
s	char[]	string
p	char[]	string
P	void *	integer

オブジェクトのコピー [copy]

■浅いコピー

・リストやディクショナリは、オブジェクトのリファレンスを格納するため、ネストしている場合など、元オブジェクトの変更の影響を受ける。
・copy() メソッドや、リストであれば、[:] 表記を使ってオブジェクトをコピーすることができる。この場合、元オブジェクトとは別のオブジェクトが作成される
・これは浅いコピーなので、トップレベルのオブジェクトしかコピーされない。

コピー例

```
>>> l1 = [1]                # 元オブジェクト
>>> l2 = [l1]               # 元オブジェクトをそのままセット
>>> l3 = [l1[:]]            # 元オブジェクトをコピーしてセット
>>> m1 = {1:l1, 2:l2, 3:l3}
>>> m1
{1: [1], 2: [[1]], 3: [[1]]}
>>> l1[0] = 9               # 元オブジェクトを 1 -> 9 に変更
>>> m1
{1: [9], 2: [[9]], 3: [[1]]}  # コピーしたものは影響を受けない
```

ネストしたオブジェクトはコピーされていない

```
>>> m1
{1: [9], 2: [[9]], 3: [[1]]}
>>> m2 = {4:m1.copy()}              # コピーした内容をディクショナリに格納
>>> l1[0] = 10                      # 大元の値を 9 -> 10 変更すると
>>> m2
{4: {1: [10], 2: [[10]], 3: [[1]]}}  # 反映されてしまう
```

■深いコピー

・深いコピーを行うには、copy モジュールの、deepcopy 関数を使用する。

```
>>> import copy
>>> m1
{1: [10], 2: [[10]], 3: [[1]]}
>>> m3 = {5:copy.deepcopy(m1)}
>>> m3
{5: {1: [10], 2: [[10]], 3: [[1]]}}
>>> l1[0] = 99                          # 大元オブジェクトを 10->99 に変更しても
>>> m3
{5: {1: [10], 2: [[10]], 3: [[1]]}}     # コピーした時点の値が保持されている
```

■リストのコピー

```
>>> l = range(10)
>>> k = l[:]
>>> k
```

マルチスレッド [threading]

■非同期にカウントアップ

ソース

```
#!python2.5
# -*- coding: utf-8 -*-
import threading

class AsyncCount(threading.Thread):
  def __init__(self, prefix):
    threading.Thread.__init__(self)
    self.prefix = prefix

  def run(self):
    for i in range(50):
      print '%s %d' % (self.prefix, i)

ac1 = AsyncCount('#Thread 1')
ac2 = AsyncCount('#Thread 2')
ac1.start()
ac2.start()

ac1.join()  #タスクの終了を待つ
print 'Thread 1 work done'
ac2.join()  #タスクの終了を待つ
print 'Thread 2 work done'
```

実行結果例

```
>python test.py
#Thread 1 0
#Thread 1 1
     略
#Thread 1 26
#Thread 1 27
#Thread 2 0
#Thread 2 1
#Thread 2 2
     略
#Thread 2 17
#Thread 2 18
#Thread 2 19 #Thread 1 28
#Thread 1 29
     略
#Thread 1 43
#Thread 1 44
#Thread 2 20
#Thread 2 21
```

```
     略
#Thread 2 34
#Thread 2 35

#Thread 1 45
     略
#Thread 1 49
Thread 1 work done
#Thread 2 36
#Thread 2 37
     略
#Thread 2 48
#Thread 2 49
Thread 2 work done
```

ロギング [logging]

■単純な例

- http://www.red-dove.com/python_logging.html

test.py

```
import logging

logging.warn("Hello")
logging.error("logging")
```

実行結果

```
>python test.py
WARNING:root:Hello
ERROR:root:logging
```

■出力は設定可能

mymodule.py

```
import logging
log = logging.getLogger('MyModule')
log.setLevel(logging.WARN)

def doSomething():
    log.debug("modlue's debugging log.")
    log.info("modlue's information log.")
    log.warn("modlue's warning log.")
    log.error("modlue's error log.")
```

myapp.py

```
import logging, mymodule

logging.basicConfig()

log = logging.getLogger("MyApp")
log.setLevel(logging.DEBUG)

log.debug("app's debugging log.")
log.info("app's information log.")
log.warn("app's warning log.")
log.error("app's error log.")
```

```
mymodule.doSomething()
```

実行結果

```
>python myapp.py
DEBUG:MyApp:app's debugging log.
INFO:MyApp:app's information log.
WARNING:MyApp:app's warning log.
ERROR:MyApp:app's error log.
WARNING:MyModule:modlue's warning log.
ERROR:MyModule:modlue's error log.
```

XML Dom [xml.dom]

・構造化マークアップツール
  ・http://docs.python.jp/3.3/library/markup.html
・xml.dom -- 文書オブジェクトモデル (DOM) API
  ・http://docs.python.jp/3.3/library/xml.dom.html

■ RSS を解析する

```
import urllib2
import xml.dom.minidom

def get_text(node):
    ret = ''
    for child in node.childNodes:
        if child.nodeType in (child.TEXT_NODE, child.CDATA_SECTION_NODE):
            ret = ret + child.data
    return ret

url = r'http://typea.info/blg/glob/atom.xml'
dom = xml.dom.minidom.parse(urllib2.urlopen(url))

root = dom.documentElement
entries = root.getElementsByTagName('entry')

for entry in entries:
   for child in entry.childNodes:
       if child.nodeName == 'title':
           print get_text(child),
       elif child.nodeName == 'link':
           print child.getAttribute('href')
```

XML ElementTree [xml.etree]

・構造化マークアップツール
  ・http://pythonjp.sourceforge.jp/dev/library/markup.html
・The Python Standard Library(The Element Interface)
  ・
  http://www.python.org/doc/current/library/xml.etree.elementtree.html?highlight=xml.etree#the-element-interface
・xml.etree.ElementTree ― ElementTree XML API
  ・http://pythonjp.sourceforge.jp/dev/library/xml.etree.elementtree.html
・ElementTree オブジェクト
  ・http://www.python.jp/doc/release/lib/elementtree-elementtree-objects.html
・XML の論考 : Python における ElementTree の XML プロセス
  ・http://www.ibm.com/developerworks/jp/xml/library/x-matters28/index.html

■ XPath を利用して ElementTree を解析

・Amazon Web Service を解析する例

例

```
import urllib2
from xml.etree import ElementTree

def qn(tag):
    ''' xmlns を付加したタグ名を返す '''
    return ElementTree.QName(ns, tag).text

ns = r'http://webservices.amazon.com/AWSECommerceService/2005-10-05'

# xmlns が指定されている場合、タグを {xmlns の値 } タグ名 といちいち書く必要があるため、そのように展開したものを保持しておく
#               ./{http://webservices.amazon.com/AWSECommerceService/2005       -10       -05}ItemAttributes/{http://webservices.amazon.com/AWSECommerceService/2005-10-05}Title
q_items  = './/{0}'.format(qn('Item'))
q_title  = './{0}/{1}'.format(qn('ItemAttributes'), qn('Title'))
q_author = './{0}/{1}'.format(qn('ItemAttributes'), qn('Author'))
q_asin   = './{0}'.format(qn('ASIN'))
q_url    = './{0}'.format(qn('DetailPageURL'))
q_img    = './{0}/{1}'.format(qn('SmallImage'), qn('URL'))

# Amazon Product Advertise API リクエスト URL
   request      =      r'http://ecs.amazonaws.jp/onca/xml?AWSAccessKeyId=1498  TGK1  YPN1  JATPXXG2&AssociateTag=typea09  -22  &Keywords=%E6  %89  %8  B%E5  %A1  %9  A%E3  %80  %80  %E6  %B2  %BB%E8  %99%AB&Operation=ItemSearch&ResponseGroup=Large&SearchIndex=Books&Service=AWSECommerceService&Timestamp=2009-09-06T02%3A19
```

2.5系 (GAE/Python) だと、str.format が使えないため、q_items = './/%s' % qn('Item') とする必要あり。

結果

```
--------------------------------------------------------------------------------------------------------
TITLE : MW( ムウ ) (1) ( 小学館文庫 )
AUTHOR : 手塚 治虫
ASIN : 4091920047
 URL   :   http://www.amazon.co.jp/MW-%E3%83%A0%E3%82%A6-%E5%B0%8F%E5%AD%A6%E9%A4%A8%E6%96%87%E5%BA%AB-%E6%89%8B%E5%A1%9A-%E6%B2%BB%E8%99%AB/dp/4091920047%3FSubscriptionId%3D1498TGK1YPN1JATPXXG2%26tag%3Dtypea09-22%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D4091920047
IMG : http://ecx.images-amazon.com/images/I/21AWAK9R3WL._SL75_.jpg
--------------------------------------------------------------------------------------------------------
TITLE : ブッダ全 12 巻漫画文庫
AUTHOR : 手塚 治虫
ASIN : 4267890021
 URL  :  http://www.amazon.co.jp/%E3%83%96%E3%83%83%E3%83%80%E5%85%A812%E5%B7%BB%E6%BC%AB%E7%94%BB%E6%96%87%E5%BA%AB-%E6%89%8B%E5%A1%9A-%E6%B2%BB%E8%99%AB/dp/4267890021%3FSubscriptionId%3D1498TGK1YPN1JATPXXG2%26tag%3Dtypea09-22%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D4267890021
IMG : http://ecx.images-amazon.com/images/I/51QDB1Q447L._SL75_.jpg
            :
```

データベースの使用 [sqlite3]

- http://www.python.jp/doc/nightly/library/sqlite3.html
- http://docs.python.jp/2.7/library/sqlite3.html

```
# -*- encoding: utf-8 -*-
import sqlite3

con = sqlite3.connect('/work/py/test.db')

#create database in RAM
#con = sqlite3.connect(':memory:')

con.execute("create table test (id  text, value text, note text)")

con.execute("insert into test values('1','aaa','01')")
```

```
con.execute("insert into test values('2','bbb','02')")
con.execute("insert into test values('3','ccc','01')")

con.commit()

p = ('01',)
c = con.cursor()
c.execute("select * from test where note=?", p)

for r in c:
  print r

con.close()
```

結果

```
(u'1', u'aaa', u'01')
(u'3', u'ccc', u'01')
```

■ INSERT 例

・タブ区切りのテキストファイルの内容をデータベースに投入
・integer primary key に None を渡すと auto number 扱い

```
conn = sqlite3.connect(r'c:\work\test.db')
c = conn.cursor()

create_ddl = """create table dict (
            _id integer primary key,
            keyword text,
            content text,
            level   integer
    )
    """
c.execute(create_ddl)

fd = open(r'c:\work\sorce.tsv', 'r')
for l in fd:
    cols = l.rsplit('\t')
    print c.execute('insert into ejdict values(?,?,?,?)',
                    (None,
                     unicode(cols[0]),
                     unicode(cols[1]),
                     0
                    )
conn.commit()
fd.close()
```

■ UPDATE 例

・更新ファイル (キーと更新値を持つ) を読み込んで、UPDATE、なければ INSERT する例
・rowcount は SELECT 時には使えない

```
conn = sqlite3.connect(r'C:\work\test.db')
c = conn.cursor()

ptn = re.compile(r'^(?P<LVL>[0-9])[ ](?P<WD>.+)')
fd = open(r'C:\work\update.txt', 'r')
for l in fd:
    m = ptn.match(l)
    if m:
        kw  = unicode(m.group('WD'))
        lvl = m.group('LVL')

        c.execute("update dict set level=? where keyword=?", (lvl, kw))
        if c.rowcount > 0:
            pass
        else:
            c.execute('insert into ejdict values(?,?,?,?)',
```

```
                    (None,
                     kw,
                     '',
                     lvl))
            print 'inserted {0}'.format(kw)
conn.commit()
```

■ SELECT 例

3 つの方法

・カーソルをイテレータ (iterator) として扱う
・カーソルの fetchone() メソッドを呼んで一致した内の一行を取得する
・fetchall() メソッドを呼んで一致した全ての行のリストとして受け取る

iterator の例

```
>>> c = conn.cursor()
>>> c.execute('select * from stocks order by price')
>>> for row in c:
...    print row
...
(u'2006-01-05', u'BUY', u'RHAT', 100, 35.14)
(u'2006-03-28', u'BUY', u'IBM', 1000, 45.0)
(u'2006-04-06', u'SELL', u'IBM', 500, 53.0)
(u'2006-04-05', u'BUY', u'MSOFT', 1000, 72.0)
```

名前によるカラムの取得

```
conn = sqlite3.connect(r'C:\work\test.db')
conn.row_factory = sqlite3.Row

c = conn.cursor()
c.execute("select * from dict")
for r in c:
    # conn.row_factory = sqlite3.Row を行うことで、名前によるカラムの取得が可能になる
    # print r[0], r[1], r[2].decode('utf-8'), r[3]
    print r["_id"], r["keyword"], r["content"].decode('utf-8'), r['level']
```

件数を取得

```
cur = conn.cursor()
r = cur.execute("select count(key) from harai")
print r.fetchone()[0]
```

UUID を使用する

・http://99blues.dyndns.org/blog/2010/07/uuid_generate/

```
>>> import uuid
>>> uuid.uuid4()
UUID('68c6d889-a64e-4e94-86f0-14012fc90364')
```

メールを送信する

・CentOS にメールサーバーを構築して Python からメールを送信する