PYTHON 标准库

仅以此献给我挚爱的女朋友

目录

第一章　写此文档的目的

人总要有点追求
总要留点什么以供后人去瞻仰

作者: xiaohai，男
单位: 不正常人类研究中心
职位: 被研究对象

第二章　内建函数

abs(x):

返回 x 的绝对值。

all(iterable):

当可迭代变量 iterable 的所有元素为真时返回 True。

any(iterable):

当可迭代变量 iterable 的任何一个元素为真时返回 True。

ascii(object):

返回一个可打印的对象的字符串表示，当遇到非 ASCII 码时，输出\u,\x 或\U 等字符。

bin(x):

将一个 int 的整数转为二进制字符串。

class bool([x]):

返回一个 bool 值。

bytearray([source[, encoding[, errors]]]):

未知

bytes([source[, encoding[, errors]]]):

未知

callable(object):

检查 object 是否是可调用的，返回 True 或 False。

chr(i):

返回整数 i 对应的 ASCII 字符。

classmethod(function):

未知

compile(source, filename, mode, flags=0, dont_inherit=False, optimize=−1):

将 source 编译为代码或 AST 对象，代码对象可以被 exec() 或 eval() 执行。

complex([real[,imag]]):

生成一个复数对象

dir([object]):

无参数时，返回当前局部域的变量名列表，有参数时，则尝试返回该对象的有效属性列表。

divmod(a, b):

取两个数字 (非复数) 作为参数，并返回一对数，其中包含他们的商和余数。

enumerate(sequence, [start=0]):

将可循环序列 sequence 以 start 开始分别列出序列的数据下标和序列数据，即对一个可遍历的数据对象 (如列表、元组或字符串),enumerate 会将该数据对象组合为一个索引序列, 同时列出数据的下标和数据的内容。

eval(expression, globals=None, locals=None):

未知

exec(object[, globals[, locals]]):

未知

filter(function, iterable):

未知

float([x]):

未知

format(value[, format_spec]):

未知

frozenset([iterable]):

未知

getattr(object, name[, default]):

未知

globals():

未知

hasattr(object, name):

未知

hash(object):

未知

help([object]):

未知

hex(x):

把一个整数转换成十六进制表示的字符串。

id(object):

未知

input([prompt]):

如果存在 prompt 参数，prompt 将被写到标准输出，该函数从标准输入读取一行，并将其转为字符串。

int(x=0):

从 x 返回一个整形对象。

int(x, base=10):

根据每个给定的进制把一个字符串转换为一个整数。

isinstance(object, classinfo):

未知

issubclass(class, classinfo):

未知

iter(object[, sentinel]):

未知

len(s):

未知

list([iterable]):

未知

locals():

未知

map(function, iterable, ...):

未知

max(iterable, *[, key, default]):

未知

max(arg1, arg2, *args[, key]):

未知

memoryview(obj):

未知

min(iterable, *[, key, default]):

未知

min(arg1, arg2, *args[, key]):

未知

next(iterable[, default]):

未知

oct(x):

将一个 int 的整数转为八进制字符串。

open(file, mode='r', encoding=None, newline =None, closefd=True, opener=None):

未知

ord(c):

未知

pow(x, y[, z]):

未知

print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False):

未知

property(fget=None, fset=None, fdel=None, doc=None):

未知

range(stop):

未知

range(start, stop[, step]):

未知

repr(object):

未知

reversed(seq):

未知

round(number[, ndigits]):

未知

set(iterable):

未知

setattr(object, name, value):

未知

slice(stop):

未知

slice(start, stop[, step]):

未知

sorted(iterable, *, key=None, reverse=False):

未知

staticmethod(function):

未知

str(object=”):

未知

str(object=‘b’, encoding=‘utf-8’, errors=‘strict’):

未知

sun(iterable[, start]):

未知

super(type[, object-or-type]):

未知

typle([iterable]):

未知

type(object):

未知

type(name, bases, dict):

未知

vars([object]):

未知

zip(*iterable):

未知

___import___(name, globals=None, locals=None, fromlist=(), level=0):

未知

第三章　内建常量

False:

the false value of the bool type.

True:

the true value of the bool type.

None:

the sole value of the type NoneType.

第四章　内建类型

4.1 Truth Value Testing

4.2 Boolean Operations–and, or, not

4.3 Comparisons

4.4 Numeric Types–int, float, complex

4.5 Iterator Types

4.6 Sequence Types–list, tuple, range

4.6.1 Common sequence operations

操作	结果
x in s	True if an item of s is equal to x, else False
x not in s	False if an item of s is equal to x, else True
s + t	the concatenation os s and t
s*n or n*s	equivalent to adding s to iteself n times
s[i]	ith item of s, origin 0
s[i:j]	slice of s from i to j
s[i:j:k]	slice of s from i to j with step k
len(s)	length of s
min(s)	smallest item of s
max(s)	largest item of s
s.index(x[, i[, j]])	index of the first occurrence of x in s
s.count(x)	total number of occurrences of x in s

4.6.2 不可变序列类型

4.6.3 改变序列类型

s[i] = x	item i of s is replaced by x
s[i:j] = t	slice of s from i to j is replaced by the contents of the iterable t
del s[i:j]	same as s[i:j] = []
s[i:j:k] = t	the elements of s[i:j:k] are replaced by those of t
del s[i:j:k]	removes the elements of s[i:j:k] from the list
s.append(x)	appends x to the end of the sequence
s.clear()	removes all items from s (same as del s[:])
s.copy()	creates a shallow copy of s (same as s[:])
s.extend(t) or s += t	extends s with the contents of t
s *= n	updates s with its contents repeated n times
s.insert(i, x)	inserts x into s at the index given by i
s.pop([i])	retrieves the item at i and also removes it from s
s.remove(x)	remove the first item from s where s[i] == x
s.reverse()	reverses the items of s in place

4.6.4 Lists

list([iterable]):

创建列表的方式：

1. 使用一对中括号。
2. 一对中括号包含的逗号分割的元素。
3. 列表解析 [x for x in iterable]。
4. 使用 list() 函数。

[].sort(*, key=None, reverse=False):

对列表进行原地排序。

4.6.5 Tuples

tuple([iterable]):

创建元组的方式：

1. 使用圆括号。

2. 使用逗号进行分割：a, 或者 (a,)。

3. 圆括号包围的逗号分割的元素：a, b, c 或者 (a, b, c)。

4. 使用 tuple() 函数。

4.6.6 Ranges

range(stop):
range(start, stop[, step]):
创建 range() 的方式。

1. start 是起始参数。

2. stop 是中之参数，但是不包含 stop 的数值。

3. step 是步长参数。

4.7 Text Sequence Type–str

4.7.1 String Methods

str.capitalize():
把字符串的第一个字母大写

str.casefold():
可识别更多的对象将其输出为小写，类似 str.lower() 函数。

str.center(width[, fillchar]):
返回一个原字符串剧中，并使用空格填充至长度 width 的新字符串。

str.count(sub[, start[, end]]):
返回 sub 在 str 中出现的次数，如果 start 和 end 指定，则返回指定范围内 sub 的出现的次数。

str.encode(encoding="utf-8", errors="strict"):
以 encoding 指定的格式编码 str。

str.endswith(suffix[, start[, end]]):

检查字符串是否以 suffix 结束，如果 start 和 end 指定范围则检查指定的范围内是否以 suffix 结束，如果是，返回 True, 否则返回 False。

str.expandtabs(tabsize=8):

把字符串 str 中的 tab 符号转换为空格，默认的宽度为 8。

str.find(sub[, start[, end]]):

检查 sub 是否包含在 str 中，如果 start 和 end 指定范围，则检查是否包含在指定的范围内。如果是则返回开始的索引值，否则返回-1。

str.format(*args, **kwargs):

未知

str.format_map(mapping):

未知

str.index(sub[, start[, end]]):

和 find() 类似，只不过如果没找到报 ValueError 类型的错误。

str.isalnum():

如果 str 至少有一个字符并且所有字符都是字母或者数字则返回 True，否则返回 False。

str.isalpha():

如果 str 至少有一个字符并且所有字符都是字母则返回 true，否则返回 False。

str.isdecimal():

如果 str 至少有一个字符并且所有字符都是十进制数字则返回 true，否则返回 False。

str.isdigit():

如果 str 至少有一个字符并且所有字符都是数字则返回 true，否则返回 False。

str.isidentifier():

未知

str.islower():

如果 str 中包含至少一个区分大小写的字符，并且所有的字符都是小写，则返回 True，否则返回 False。

str.isnumeric():

如果 str 中只包含数字字符，则返回 True，否则返回 False。

str.isprintable():

如果 str 中的字符都是可打印的或者为空，则返回 True，否则返回 False。

str.isspace():

如果 str 中只包含空格，则返回 True，否则返回 False。

str.istitle():

如果 str 是标题化的，则返回 True，否则返回 False。

4.8 Binary Sequence Types

4.9 Set Types

4.10 Mapping Types

4.11 Context Manager Types

4.12 Other Build-in Types

4.13 Special Attributes

第五章　内建异常

在 Python 中，所有的异常类型都派生于 BaseException 类。In a try statement with an except clause that mentions a particular class, that clause also handles any exception classes derived from that class (but not exception classes from which it is derived). Two exception classes that are not never equivalent, even if they have the same name.

The built-in exceptions listed below can be generated by the interpreter or built-in functions. Except where mentioned, they have an "associated value" indicating the detailed cause of the error. This may be a string or a tuple of several items of information (e.g., an error code and a string explaining the code). The associated value is usually passed as arguments to the exception class's constructor.

User code can raise built-in exceptions. This can be used to test an exception handler or to report an error condition "just like" the situation in which the interpreter raises the same exception; but beware that there is nothing to prevent user code from raising an inappropriate error.

The built-in exception classes can be subclassed to define new exceptions; programmers are encouraged to derive new exceptions from the Exception class or one of its subclasses, and not from BaseException. More information on defining exceptions is available in the Python Tutorial under User-defined Exceptions.

5.0.1　基类

下面的几个异常类型几乎作为其它异常类型的基类。

exception BaseException:

所有异常类的积累。

excetion Exception:

未知

exception ArithmeticError:

未知

exception BufferError:

未知

exception LookupError:

未知

5.0.2 具体类

exception AssertionError:

断言失败抛出的异常类型。

exception AttributeError:

Raised when an attribute reference (see Attribute references) or assignment fails. (When an object does not support attribute references or attribute assignments at all, TypeError is raised.)

exception EOFError:

Raised when the input() function hits an end-of-file condition (EOF) without reading any data. (N.B.: the io.IOBase.read() and io.IOBase.readline() methods return an empty string when they hit EOF.)

exception FloatingPointError:

Raised when a floating point operation fails. This exception is always defined, but can only be raised when Python is configured with the –with-fpectl option, or the WANT_SIGFPE_HANDLER symbol is defined in the pyconfig.h file.

exception GeneratorExit:

Raised when a generator or coroutine is closed; see generator.close() and coroutine.close(). It directly inherits from BaseException instead of Exception since it is technically not an error.

exception ImportError:

Raised when the import statement has troubles trying to load a module. Also raised when the “from list”in from ...import has a name that cannot be found.

exception ModuleNotFoundError:

A subclass of ImportError which is raised by import when a module could not be located. It is also raised when None is found in sys.modules.

第六章　文本处理服务

6.1　string

6.2　re

该模块提供了正则表达式的处理操作。Both patterns and strings to be searched can be Unicode strings as well as 8-bit strings. However, Unicode strings and 8-bit strings cannot be mixed: that is, you cannot match a Unicode string with a byte pattern or vice-versa; similarly,when asking for a substitution, the replacement string must be of the same type as both the pattern and the search string.

Regular expressions use the backslash character '\') to indicate special forms or to allow special characters to be used without invoking their special meaning. This collides with Python' s usage of the same character for the same purpose in string literals; for example, to match a literal backslash, one might have to write '\\\\' as the pattern string, because the regular expression must be \\, and each backslash must be expressed as \\ inside a regular Python string literal.

The solution is to use Python's raw string notation for regular expression patterns; backslashes are not handled in any special way in a string literal prefixed with 'r'. So r"\n" is a two-character string containing '\' and 'n', while "\n' is a one-character string containing a newline. Usually patterns will be expressed in Python code using this raw string notation.

正则表达式的基本语法：

1. '.': 在缺省的模式下，该符号匹配除了换行符外 ('\n') 的任何字符。

2. '^': 匹配字符串的起始部分。

3. '$': 匹配字符串的终止部分。

4. '*': 匹配 0 次或者多次前面出现的正则表达式。

5. '+': 匹配 1 次或者多次前面出现的正则表达式。

6. '?': 匹配 0 次或者 1 次前面出现的正则表达式。
7. '{m}': 匹配 m 次前面出现的正则表达式。
8. '{m, n}': 匹配 m~ 次前面出现的正则表达式。
9. [...]: 匹配来自字符集的任意单一字符。
10. '|':A|B 匹配 A，B 之间的任意一个。
11. (…): 匹配封闭的正则表达式，然后另存为子组。
12. (?…): 未知。
13. (?aiLmsux): 在正则表达式中嵌入一个或者多个特殊"标记"参数。
14. (?:…): 表示一个匹配但不用保存的分组。
15. (?imsx-imsx:…): 未知。
16. (?P<name>…): 像一个仅由 name 标记而不是数字 ID 标识的正则分组匹配。

6.3 difflib

6.4 textwrap

6.5 unicodedata

6.6 stringprep

6.7 readline

6.8 rlcompleter

第七章　二进制数据服务

第八章　数据类型

第九章　数值和数学

9.1　numbers

9.2　math–Mathematical functions

9.2.1　Number-theoretic and representation functions

math.ceil(x):

返回大于等于 x 的最小整数。

math.copysign(x, y):

返回与 y 同号的 x 的值。

math.fabs(x):

返回 x 的绝对值。

math.factorial(x):

返回 x 的阶乘。

math.floor(x):

返回小于等于 x 的最大整数。

math.fmod(x, y):

返回 x 对 y 取模的余数，fmod 类似%，但是产生的结果可能与% 不同，因为前者以 y 来决定余数的符号，后者以 x 来决定余数的符号。

math.frexp(x):

返回一个 2 元组分别是底数以及一个指数，也就是 $x = m * 2^n$。

math.fsun(iterable):

返回 x 序列值的和。

math.gcd(a, b):

返回 a 与 b 的最大公约数。

math.isclose(a, b, *, rel_tol=1e-09, abs_tol=0.0):

如果 a 和 b 是如此的接近 (小于 rel_tol 的值)，则返回 True，否则返回 Flase。

math.isfinite(x):

如果 x 是数字，且是有限的，则返回 True，否则返回 Flase。

math.isinf(x):

如果 x 是正无穷或者负无穷，则返回 True，否则返回 Flase。

math.isnan(x):

如果 x 不是数字，则返回 True，否则返回 Flase。

math.ldexp(x, i):

返回 x*(2^i)。

math.modf(x):

返回 x 的分数部分和整数部分。

math.trunc(x):

返回 x 的整数部分，等同于 int()。

9.2.2 Power and logarithmic functions

math.exp(x):

返回 e^x。

math.expm1(x):

返回 $e^x - 1$。

math.log(x[, base]):

返回以 base 为底数的对数。

math.loglp(x):

返回 $log_e(1+x)$。

math.log2(x):

返回 math.log(x, 2)。

math.log10(x):

返回 math.log(x, 10)。

math.pow(x, y):

返回 x^y。

math.sqrt(x):

返回 x 的均方根。

9.2.3 三角函数

math.acos(x):

返回 x 的反余弦。

math.asin(x):

返回 x 的反正弦。

math.atan(x):

返回 x 的反正切。

math.atan2(y, x):

返回 y/x 的反正切。

math.cos(x):

返回 x 的余弦。

math.hypot(x, y):

返回 (x, y) 距离原点的欧几里得距离。

math.sin(x):

返回 x 的正弦。

math.tan(x):
返回的正切。

9.2.4 角度变换

math.degrees(x):
将 x 由弧度转换为角度。

math.radians(x):
将 x 由角度转换为弧度。

9.2.5 双曲函数

math.acosh(x):
返回 x 的反双曲余弦。

math.asinh(x):
返回 x 的反双曲正弦。

math.atanh(x):
返回 x 的反双曲正切。

math.cosh(x):
返回 x 的双曲余弦。

math.sinh(x):
返回 x 的双曲正弦。

math.tanh(x):
返回 x 的双曲正切。

9.2.6 特殊函数

math.erf(x):

未知

math.erfc(x):

未知

math.gamma(x):

返回 x 的 gamma 函数。

math.lgamma(x):

未知

9.2.7 常数

math.pi:

常数 π。

math.e:

自然常数。

math.inf:

正无穷的浮点数。

math.nan:

浮点型的非数字。

9.3 cmath-Mathematical functions for complex numbers

9.3.1 Conversions to and from polar coordinates

cmath.phase(x):

返回 x 的相位角，等同于 math.atan2(x.img, x.real)。

cmath.polar(x):

返回在极坐标系下 x 的值，返回的是一个对 (r, phi)，其中 r 是 x 的模，phi 是方位角。x.polar(x) 等同于 (abs(x), phase(x))。

cmath.rect(r, phi):

返回一个复数，其数值为 r*(math.cos(phi) + math.sin(phi)*1j)。

9.3.2 Power and logarithmic functions

cmath.exp(x):

返回 e^x。

cmath.log(x[, base]):

对于给定的底数 base，返回 x 的对数。

cmath.log10(x):

返回 cmath.log(x, 10)。

cmath.sqrt(x):

返回 x 的均方根。

9.3.3 三角函数

cmath.acos(x):

返回 x 的反余弦。

cmath.asin(x):

返回 x 的反正弦。

cmath.atan(x):

返回 x 的反正切。

cmath.cos(x):

返回 x 的余弦。

cmath.sin(x):

返回 x 的正弦。

cmath.tan(x):

返回 x 的正切。

9.3.4 双曲函数

cmath.acosh(x):

返回 x 的反双曲余弦。

cmath.asinh(x):

返回 x 的反双曲正弦。

cmath.atanh(x):

返回 x 的反双曲正切。

cmath.cosh(x):

返回 x 的双曲余弦。

cmath.sinh(x):

返回 x 的双曲正弦。

cmath.tanh(x):

返回 x 的双曲正切。

9.3.5 Classification functions

同 math 模块。

9.3.6 常数

同 math 模块。

9.4 decimal-Decimal fixed point and floating point arithmetic

9.5 fractions–Rational numbers

9.6 random–Generate pseudo-random numbers

9.6.1 Bookkeeping functions

random.seed(a=None, version=2):

设置随机数的起始种子，如果 a 省略，将会使用系统当前的时间作为种子。

random.getstate():

未知

random.setstate(state):

未知

random.getrandbits(k):

未知

9.6.2 Functions for integers

random.randrange(stop):

返回指定递增基数集合中的一个随机数，基数的缺省值为 1。

random.randrange(start, stop[, step]):

返回指定递增基数集合中的一个随机数，start 指定范围内的起始值，end 指定范围的结束值 (左闭右开)，step 指定递增基数。

random.randint(a, b):

返回 [a, b] 之间的一个随机整数，等同于 random.randrange(a, b+1)。

9.6.3 Functions for sequences

random.choice(seq):

从序列的元素中随机挑选一个元素。

random.choices(population, weights=None, *, cum_weights=None, k=1):

未知

random.shuffle(x[, random]):

将序列的所有元素随机排序。

random.sample(population, k):

从序列中随机挑选 k 个元素。

9.6.4 Real-valued distributions

random.random():

随机生成下一个实数，它在 [0, 1) 范围内。

random.uniform(a, b):

返回一个随机数，它在 [a, b] 分为内。

random.triangular(low, high, mode):

未知

random.betavariate(alpha, beta):

未知

random.expovariate(lambda):

未知

random.gammavariate(alpha, beta):

未知

random.gauss(mu, sigma):

未知

random.lognormvariate(mu, sigma):

未知

random.normalvariate(mu, sigma):

未知

random.vonmisesvariate(mu, kappa):

未知

random.paretovariate(alpha):

未知

random.weibullvariate(alpha, beta):

未知

9.7 statistics–Mathematical statistics functions

第十章　函数处理

第十一章　文件和目录

11.1　pathlib

11.2　os.path

os.path.abspath(path):

返回路径名 path 的规范化的绝对路径, 在大多数的 9 平台上, 该函数等同于 normpath(), 即 normpath(join(os.getcwd(), path))。

os.path.basename(path):

返回路径名 path 的主文档名。在 path 上调用 split() 函数，返回的二元组中的第二个元素就是主文档名。这和 Unix 的 basename 不同，对于‘/foo/bar/’,basename 返回‘bar’，而 basename() 函数返回空字符串。

os.path.commonpath(paths):

未知

os.path.commonprefix(list):

返回 list 中所有路径的最长通用前缀 (逐字符的进行比较)。如果 list 为空，返回空字符串 (‘’)。因为是逐字符比较，所以可能会返回无效的路径。

os.path.dirname(path):

返回路径名 path 的目录名。在 path 上调用 split(), 返回的二元组中的第一个元素就是目录名。

os.path.exists(path):

如果 path 引用一个存在的路径，返回 True。如果 path 引用一个断开的符号链接，返回 False。在某些平台上，尽管 path 物理存在，但是由于没有执行 os.stat() 的权限，该函数也会返回 False。

os.path.lexists(path):

如果 path 引用一个存在的路径，返回 True. 对于 path 引用一个断开的符号链接，也返回 True。等同于缺少 os.lstat() 的平台上的 exists()。

os.path.expanduser(path):

在 Unix 和 Windows 平台上，返回参数，参数中开头的 ~ 或者 ~user 被替换成 user 的主家目录。在 Unix 上，开头的 ~ 被替换成环境变量 HOME，如果它被设置的话；否则，通过内建模块 pwd 在密码目录查询当前用户的家目录。如果开头是 ~user，则直接在密码目录中查询 (user 的家目录)。

在 Windows 上，将使用 HOME 和 USERPROFILE，如果它们被设置的话；否则使用 HOMEPATH 和 HOMEDRIVE 的组合。如果开头是 ~user, 首先按上述方式得到 user 路径, 然后移除最后的目录部分。如果扩展失败或者参数 path 不是以 ~ 打头, 则直接返回参数 (path)。

os.path.expandvars(path):

返回参数, 其中的环境变量被扩展。参数中形如 $name 或者 $name 的部分被替换成环境变量 name 的值。如果格式不正确或者引用了不存在的变量，则不进行替换 (或者扩展)。在 Windows 上, 除了 $name 和 $name 之外还支持如%name% 这样的扩展。

os.path.getatime(path):

返回 path 的最后访问时间。返回的是从 Unix 纪元开始的跳秒数 (epoch，Unix 纪元，即 GMT 1970-01-01 00:00:00)(参见 time 模块)。如果文件不存在或者不可访问，返回 os.error。

os.path.getmtime(path):

返回 path 的最后修改时间。返回的是从 Unix 纪元开始的跳秒数 (参见 time 模块)。如果文件不存在或者不可访问，返回 os.error。

os.path.getctime(path):

返回系统的 ctime，在 Unix 这样的系统上，它是文件元数据最后修改时间 (或者可以说是文件状态最后修改时间)；在 Windows 这样的系统上，它是 path 的创建时间。返回的是从 Unix 纪元开始的跳秒数 (参见 time 模块)。如果文件不存在或者不可访问，返回 os.error。

os.path.getsize(path):

返回 path 的大小，以字节为单位。如果文件不存在或者不可访问，返回 os.error。

os.path.isabs(path):

如果 path 是绝对路径名，返回 True。在 Unix 上，这表示路径以/开始；在 Windows 上，这表示路径以\开始 (在去掉可能的盘符后，如 C:)。

os.path.isfile(path):

如果 path 是一个存在的普通文件，返回 True。它会跟随符号链接，所以对相同的路径，islink() 和 isfile() 可以同时为真。

os.path.isdir(path):

如果 path 是一个存在的目录，返回 True。它会跟随符号链接，所以对于相同的路径，islink() 和 isdir() 可以同时为真。

os.path.islink(path):

如果 path 引用的目录条目是个符号链接，返回 True。如果 Python 运行期不支持符号链接，则总是返回 False。

os.path.ismount(path):

如果路径名 path 是一个 mount point(挂载点)，返回 True。挂载点是文件系统中的一点，不同的文件系统在此被挂载。它检查 path 的父目录 path/.. 和 path 是否在不同的设备上，或者检查 path/.. 和 path 是否指向相同设备的相同 i-node,——这可以检查出 Unix 和 POSIX 变种下的挂载点。

os.path.join(path, *paths):

将一个或多个路径正确地连接起来。如果任何一个参数是绝对路径，那之前的参数就会被丢弃，然后连接继续。

os.path.normcase(path):

未知

os.path.normpath(path):

未知

os.path.realpath(path):

未知

os.path.relpath(path, start=os.curdir):

返回一个相对路径，该路径从当前目录算起或者从 start 指定的目录算起。

os.path.samefile(path1, path2):

如果 path1 和 path2 是相同的目录或者文件，则返回 True。

os.path.sameopenfile(fp1, fp2):

如果 fp1 和 fp2 是指向相同的文件，则返回 True，注意 fp1 和 fp2 都必须是打开的文件。

os.path.samestat(stat1, stat2):

如果 stat 元组 stat1 和 stat2 指向同一文件，则返回 True。

os.path.split(path):

把路径 path 分割成 (head, tail) 的元组，类似分割为 dirname 和 basename。

os.path.splitdrive(path):

一般在 Windows 下，返回驱动器名和路径组成的元组。

os.path.splitext(path):

分割路径，返回路径加文件名和文件扩展名的元组, 类似 (name, .py)。

os.path.splitunc(path):

把路径分割为加载点和文件。

11.3 fileinput

11.4 stat

11.5 filecmp

11.6 tempfile

11.7 glob

11.8 fnmatch

11.9 linecache

11.10 shutil

shutil 模块提供了较高级别的文件和目录的操作。尤其是函数提供了对文件的复制和删除。

11.10.1 Directory and files operations

shutil.copyfileobj(fsrc, fdst[, length]):

copy 文件内容到另一个文件，可以 copy 指定大小的内容，注意！在其中 fsrc，fdst 都是文件对象，都需要打开后才能进行复制操作。

shutil.copyfile(src, dst, *, follow_sysmlinks=True):

从源 src 复制到 dst 中去。当然前提是目标地址是具备可写权限。抛出的异常信息为 IOException. 如果当前的 dst 已存在的话就会被覆盖掉。

shutil.SameFileError:

如果 copyfile() 的 source 和 destination 相同，则导致该错误。

shutil.copymode(src, dst, *, follow_sysmlinks=True):

复制 src 的权限到 dst，也就是将 dst 的权限修改为和 src 一样，其中文件的内容，属主和所属组并不会受到任何的影响。

shutil.copystat(src, dst, *, follow_sysmlinks=True):

复制 src 的权限，最后的访问时间，修改时间和 flags 到 dst。

shutil.copy(src, dst, *, follow_sysmlinks=True):

复制文件 src 到 dst 文件或者目录，其中 src 和 dst 都应该是字符串。

shutil.copy2(src, dst, *, follow_sysmlinks=True):

在 copy 上的基础上再复制文件最后访问时间与修改时间也复制过来了。

shutil.ignore_patterns(*patterns):

未知

shutil.copytree(src, dst, sysmlinks=False, ignore=None,copy_function=copy2, ignore_dangling_sysmlinks=False):

递归的拷贝文件目录及其下面的所有文件。

shutil.rmtree(path, ignore_errors=False, onerror=None):

递归的删除文件。

shutil.move(src, dst, copy_function=copy2):

递归的移动文件或目录 (src) 到另一个位置 (dst)，并返回 destination。

shutil.disk_usage(path):

返回一个元组，元组的内容是盘符的总容量，已使用和剩余空间。

shutil.chown(path, user=None, group=None):

修改 path 的属主和所属组。

shutil.which(cmd, mode=os.F_OK, path=None):

返回可执行命令 cmd 的路径，同 linux 命令。

shutil.Error:

多文件操作引起的错误。

11.10.2 Archiving operations

shutil.make_archive(base_name, format[, root_dir[, base_dir[, verbose[, dry_run[, owner[, group[, logger]]]]]]]):

压缩打包文件并返回它的名字。

1. base_name：压缩打包后的文件名，包含路径。
2. format：文档的格式，可以是“zip”, “tar”, “gztar”, “bztar”, “xztar”。
3. root_dir：压缩完成后的的文件夹路径 (默认为当前路径)。
4. base_dir：要压缩的文档的路径 (默认为当前路径)。

shutil.get_archive_formats():
返回支持的文档的格式类型。

shutil.register_archive_format(name, function[, extra_args[, description]]):
未知

shutil.unregister_archive_format(name):
未知

shutil.unpack_archive(filename[, extra_dir[, format]]):
未知

shutil.register_unpack_format()name, extensions, function[, extra_args[, description]]:
未知

shutil.unregister_unpack_format(name):
未知

shutil.get_unpack_formats():
未知

11.10.3 Querying the size of the output terminal

shutil.get_terminal_size(fallback=(columns, lines)):
未知

11.11 macpath

第十二章　Data Persistence

第十三章　Data Compression and Archiving

第十四章　File Formats

第十五章　Cryptographic Services

第十六章　Generic Operating System Services

16.1　os–Miscellaneous operating system interfaces

该模块提供了方便的使用操作系统依赖的函数的方式。如果你想读或者写文件，你应该看 open()，如果你想操作路径，你应该看 os.path 模块，如果你想读取文件内的所有行，你应该看 fileinput 模块，如果你想操作临时文件或者目录，你应该看 tempfile 模块，对于较高级别的目录和文件处理，你应该看 shutil 模块。

os.error:

内建 OSError 错误的别名。

os.name:

导入依赖操作系统模块的名字。下面是目前被注册的名字 :“posix”, nt”, “java”。

16.1.1　File Names, Command Line Arguments, and Environment Variables

16.1.2　Process Parameters

os.ctermid():

返回进程控制终端的文件名。在 unix 中有效，请查看相关文档。

os.environ:

一个 mapping 对象表示环境。例如，environ[“HOME”] ，表示的你自己 home 文件夹的路径 (某些平台支持，windows 不支持) ，它与 C 中的 getenv(“HOME”) 一致。

这个 mapping 对象在 os 模块第一次导入时被创建, 一般在 python 启动时, 作为 site.py 处理过程的一部分。在这一次之后改变 environment 不影响 os.environ, 除非直接修改 os.environ。

注：putenv() 不会直接改变 os.environ，所以最好是修改 os.environ。

注：在一些平台上，包括 FreeBSD 和 Mac OS X，修改 environ 会导致内存泄露。参考 putenv() 的系统文档。

如果没有提供 putenv()，mapping 的修改版本传递给合适的创建过程函数，将导致子过程使用一个修改的 environment。

如果这个平台支持 unsetenv() 函数，你可以删除 mapping 中的项目。当从 os.environ 使用 pop() 或 clear() 删除一个项目时，unsetenv() 会自动被调用 (版本 2.6)。

os.environb:

environ 的 bytes 版本。

os.chdir(path):

用于改变当前的工作目录到 path 指定的目录。

os.fchdir(fd):

用于改变当前的工作目录到 fd 描述的目录。

os.getcwd():

返回一个用于表示当前目录的字符串。

os.fsencode(filename):

未知

os.fsdecode(filename):

未知

os.fspath(path):

返回 path 所代表的文件系统。

os.getenv(key, default=None):

返回 environment 变量 key 的值，返回的结果是个 str。

os.getenvb(key, default=None):

返回 environment 变量 key 的值，返回的结果是个 bytes。

os.get_exec_path(env=None):

未知

os.getegid():

返回当前进程有效的 group 的 id。对应于当前进程的可执行文件的“set id”的 bit 位。在

unix 中有效，请查看相关文档。

os.geteuid():
返回当前进程有效的 user 的 id。在 unix 中有效，请查看相关文档。

os.getgid():
返回当前进程当前 group 的 id。

os.getgrouplist(user, group):
未知

os.getgroups():
返回当前进程支持的 groups 的 id 列表。

os.getlogin():
返回进程控制终端登陆用户的名字。在大多情况下它比使用 environment 变量 LOGNAME 来得到用户名，或使用 pwd.getpwuid(os.getuid())[0] 得到当前有效用户 id 的登陆名更为有效。

os.getpgid(pid):
返回 pid 进程的 group id. 如果 pid 为 0, 返回当前进程的 group id。

os.getpgrp():
返回当前进程组的 id。

os.getpid():
返回当前进程的 id。

os.getppid():
返回当前父进程的 id。

os.getpriority(which, who):
未知

os.PRIO_PROCESS
os.PRIO_PGRP
os.PRIO_USER

getpriority() 和 setpriority() 的参数。

os.getresuid():

返回一个元组 (ruid, euid, suid) 代表当前进程的 real, effective 和 saved user ids。

os.getresgid():

返回一个元组 (rgid, egid, sgid) 代表当前进程的 real, effective 和 saved group ids。

os.getuid():

返回当前进程的真正用户的 id。

os.initgroups(username, gid):

未知。

os.putenv(key, value):

将环境变量名字为 key 的值为 value。

os.setegid(egid):

设置当前进程的有效组 id。

os.seteuid(euid):

设置当前进程的有效用户 id。

os.setgid(gid):

设置当前进程组的 id.

os.setgroups(groups):

设置当前进程支持的 groupsid 列表。groups 必须是列表，每个元素必须是整数，这个操作只对超级用户组有效。

os.setpgrp():

未知

os.setpgid(pid, pgrp):

未知

os.setpriority(which, who, priority):

未知

os.setregid(rgid, egid):

未知

os.setresuid(ruid, euid, suid):

未知

os.setreuid(ruid, euid):

未知

os.getsid(pid):

未知

os.setsid():

未知

os.setuid(uid):

未知

os.strerror(code):

未知

os.supports_bytes_environ:

未知

os.umask(mask):

未知

os.uname():

未知

os.unsetenv(key):

未知

16.1.3 File Object Creation

os.fdopen(fd, *args, **kwargs):

未知

16.1.4 File Descriptor Operations

os.close(fd):

关闭文件描述符。

os.closerange(fd_low, fd_high):

未知

os.device_encoding(fd):

未知

os.dup(fd):

未知

os.dup2(fd, fd2, inheritable=True):

未知

os.fchmod(fd, mode):

未知

os.fchown(fd, uid, gid):

未知

os.fdatasync(fd):

未知

os.fpathconf(fd, name):

未知

os.fstat(fd):

未知

os.fstatvfs(fd):

未知

os.fsync(fd):

未知

os.ftruncate(fd, length):

未知

os.get_blocking(fd):

未知

os.isatty(fd):

未知

os.lockf(fd, cmd, len):

未知

os.F_LOCK:
os.F_TLOCK
os.F_ULOCK
os.F_TEST

未知

os.lseek(fd, pos, how):

未知

os.SEEK_SET
os.SEEK_CUR
os.SEEK_END

未知

os.open(path, flags, mode=0o777, *, dir_fd=None):

未知

os.O_RDONLY:
os.O_WRONLY:

os.O_RDWR:
os.O_APPEND:
os.O_CREAT:
os.O_EXCL:
os.O_TRUNC:
os.O_DSYNC:
os.O_RSYNC:
os.O_SYNC:
os.O_NDELAY:
os.O_NONBLOCK:
os.O_NOCTTY:
os.O_CLOEXEC:
os.O_BINATY:
os.O_NOINHERIT:
os.O_SHORT_LIVED:
os.O_TEMPORARY:
os.O_RANDOM:
os.O_SEQUENTIAL:
os.O_TEXT:
os.O_ASYNC:
os.O_DIRECT:
os.O_DIRECTORY:
os.O_NOFOLLOW:
os.O_NOATIME:
os.O_PATH:
os.O_TMPFILE:
os.O_SHLOCK:
os.O_EXLOCK:

未知

os.openpty():

未知

os.pipe():

未知

os.pipe2(flags):

未知

os.posix_fallocate(fd, offset, len):

未知

os.posix_fadvise(fd, offset, len, advice):

未知

os.POSIX_FADV_NORMAL:
os.POSIX_FADV_SEQUENTIAL:
os.POSIX_FADV_RANDOM:
os.POSIX_FADV_NOREUSE:
os.POSIX_FADV_WILLNEED:
os.POSIX_FADV_DONTNEED:

未知

os.pread(fd, buffersize, offset):

未知

os.pwrite(fd, str, offset):

未知

os.read(fd, n):

未知

os.sendfile(out, in, offset, count):
os.sendfile(out, in, offset, count, [headers,][trailers,]flags=0)

未知

os.set_blocking(fd, blocking):

未知

os.SF_NODISKIO:
os.SF_MNOWAIT:
os.SF_SYNC:

未知

os.read(fd, buffers):

未知

os.tcgetpgrp(fd):

未知

os.tcsetpgrp(fd, pg):

未知

os.ttyname(fd):

未知

os.write(fd, str):

未知

os.writev(fd, buffers):

未知

os.get_terminal_size(fd=STDOUT_FILENO):

未知

os.terminal_size:

未知

os.get_inheritable(fd):

未知

os.set_inheritable(fd, inheritable):

未知

os.get_handle_inheritable(handle):

未知

os.set_handle_inheritable(handle, inheritable):

未知

os.access(path, mode, *, dir_fd=None, effective_ids=False, follow_symlinks=True):

未知

os.F_OK: os.R_OK: os.W_OK: os.X_OK:

未知

os.chdir(path):

未知

os.chflags(path, flags, *, follows_symlinks=True):

未知

os.chmod(path, mode, *, dir_fd=None, follow_symlinks=True):

未知

os.chown(path, uid, gid, *, dir_fd=None, follow_symlinks=True):

未知

os.chroot(path):

未知

os.fchdir(fd):

未知

os.getcwd():

未知

os.getcwdb():

未知

os.lchflags(path, flags):

未知

os.lchmod(path, mode):

未知

os.lchown(path, uid, gid):

未知

os.link(src, dst, *, src_dir_fd=None, dst_dir_fd=None,
follow_symlinks=True):

未知

os.listdir(path='.'):

未知

os.lstat(path, *, dir_fd=None):

未知

os.mkdir(path, mode=0o777, *, dir_fd=None):

未知

os.makedirs(name, mode=0o777, exist_ok=False):

未知

os.mkfifo(path, mode=0o666, *, dir_fd=None):

未知

os.mknod(path, mode=0o666, device=0, *, dir_fd=None):

未知

os.major(device):

未知

os.minor(device):

未知

os.makedev(major, minor):

未知

os.pathconf(path, name):

未知

os.pathconf_names:

未知

os.readlink(path, *, dir_fd=None):

未知

os.remove(path, *, dir_fd=None):

未知

os.removedirs(name):

未知

os.rename(src, dst, *, src_dir_fd=None, dst_dir_fd=None):

未知

os.renames(old, new):

未知

os.replace(src, dst, *, src_dir_fd=None, dst_dir_fd=None):

未知

os.rmdir(path, *, dir_fd=None):

未知

os.scandir(path='.'):

未知

os.stat(path, *, dir_fd=None, follow_symlinks=True):

Get the status of a file descriptor. Perform the quivalent of a stat() system call on the given path. path may be specified as either a string or bytes —-directly or indirectly through the Pathlike interface —- or as an open file descriptor. Return a stat_result object.

```
>>> import os
>>> statinfo = os.stat('somefile.txt')
>>> statinfo
os.stat_result(st_mode = 33188, st_ino = 7876932, st_dev = 23456,
st_mlink = 1, st_uid = 501, st_gid = 501, st_size = 264, st_atime=123,
```

```
st_mtime=129, st_ctime=123)
>>> statinfo.st_size
264
```

* st_mode : file mode: file type and file mode bits.
* st_ino : Inode number.
* st_dev : Identifier of the device on which this file resides.
* st_nlink : Number of hard links.
* st_uid : User identifier of the file owner.
* st_gid : Group identifier of the file owner.
* st_size : Size of the file in bytes.
* st_atime : Time of most recent access expressed in seconds.
* st_mtime : Time of most recent content modification expressed in seconds.
* st_ctime : The time of most recent metadata change on Unix.
* st_atime_ns : Time of most recent access expressed in nanoseconds as an integer.
* st_mtime_ns : Time of most recent modification expressed in nanoseconds as an integer.
* st_ctime_ns : Time of most recent metadata change on Unix.
* st_blocks : Number of 512-byte blocks allocated for file.
* st_blksize : “Preferred” blocksize for efficient file system I/O. Writing to a file in smaller chunks may cause an inefficient read-modify-rewrite.
* st_rdev : Type of device if an inode device.
* st_flags : User defined flags for file.

os.stat_float_times([newvalue]):

未知

os.statvfs(path):

未知

os.supports_dir_fd:

未知

os.supports_effective_ids:

未知

os.supports_fd:

未知

os.supports_follow_symlinks:

未知

os.symlinks(src, dst, target_is_directory=False, *, dir_fd=None):

未知

os.sync():

未知

os.truncate(path, length):

未知

os.unlink(path, *, dir_fd=None):

未知

os.utime(path, times=None, *, [ns,]dir_fd=None, follow_symlinks=True):

未知

os.walk(top, topdown=True, onerror=None, followlinks=False):

未知

os.fwalk(top='.', topdown=True, onerror=None, *, follow_symlinks=False, dir_fd=None):

未知

os.getattr(path, attribute, *, follow_symlinks=True):

未知

os.listxattr(path=None, *, follow_symlinks=True):

未知

os.listattr(path=None, *, follow_symlinks=True):

未知

os.removexattr(path, attribute, *, follow_symlinks=True):

未知

os.setxattr(path, attribute, value, flags=0, *, follow_symlinks=True):

未知

os.XATTR_SIZE_MAX:

未知

os.XATTR_SIZE_CREATE:

未知

os.XATTR_SIZE_REPLACE:

未知

os.abort():

未知

os.execl(path, arg0, arg1, …):

未知

os.execle(path, arg0, arg1, …, env):

未知

os.execlp(file, arg0, arg1, …):

未知

os.execlpe(file, arg0, arg1, …, env):

未知

os.execv(path, args):

未知

os.execve(path, args, env):

未知

os.execvp(file, args):

未知

os.execvpe(file, args, env):

未知

os._exit(n):

未知

* os.EX_OK : Exit code that means no error occurred.
* os.EX_USAGE : Exit code that means the command was used incorrectly, such as when the wrong number of arguments are given.
* os.EX_DATAERR : Exit code that means the input data was incorrect.
* os.EX_NOTINPUT : Exit code that means an input file did not exist or was not readable.
* os.EX_NOUSER : Exit code that means a specified user did not exist.
* os.EX_NOHOST : Exit code that means a specified host did not exist.
* os.EX_UNAVAILABLE : Exit code that means that a required service is unavailable.
* os.EX_SOFTWARE : Exit code that means an internal software error was detected.
* os.EX_OSERR : Exit code that means an operating system error was detected, such as the inability to fork or create a pipe.
* os.EX_OSFILE : Exit code that means some system file did not exist, could not be opened, or had some other kind of error.
* os.EX_CANTCREATE : Exit code that means a user specified output file could not be created.
* os.EX_IOERR : Exit code that means that an error occurred while doing I/O on some file.

* os.EX_TEMPFAIL : Exit code that means a temporary failure occurred.
* os.EX_PROTOCOL : Exit code that means that a protocol exchange was illegal, invalid, or not understood.
* os.EX_NOPERM : Exit code that means that there were insufficient permissions to perform the operation (but not intended for file system problems).
* os.EX_CONFIG : Exit code that means that some kind of configuration error occurred.
* os.EX_NOTFOUND : Exit code that means something like “an entry was not found”.

os.folk():

创建子进程。

os.forkpty():

未知

os.kill(pid, sig):

对进程 pid 发送信号值 sig.

os.killpg(pgid, sig):

未知

os.nice(increment):

未知

os.plock(op):

未知

os.popen(cmd, mode=‘r’, buffering=-1):

未知

os.spawnl(mode, path, …):
os.spawnle(mode, path, …, env):
os.spawnlp(mode, file, …):
os.spawnlpe(mode, file, …, env):
os.spawnv(mode, path, args):
os.spawnve(mode, path, args, env):

os.spawnvp(mode, file, args)

os.spawnvpe(mode, file, args, env)

未知

第十七章　Concurrent Execution

第十八章　Interprocess Communication and Networking

第十九章　Internet Data Handling

第二十章　Structured Markup Processing Tools

第二十一章　Internet Protocols and Support

21.1　webbrower

21.2　cgi

21.3　cgitb

21.4　wsgiref

21.5　urllib

21.6　urllib.request

The urllib.request module defines functions and classes which help in opening URLs (mostly HTTP) in a complex world —-basic and digest authentication, redirections, cookies and more.

The urllib.request 模块包含以下函数:

urllib.request.urlopen(url, data=None, [timeout,]*, cafile=None, capath=None, cadefault=False, context=None)

:

打开 url，其中 url 可以是字符串也可以是 Request 对象。

. data : 是一个定义了需要发给 server 端数据的对象或者 None, 一个标准的 application/x-www-from-urlencoded 格式的 buffer.urllib.request 模块使用 HTTP/1.1 协议，并且在 HTTP 请求的报头中包含 Connection:close.

. timeout：设定连接的超时时间，如果没有设定的话就使用全局超时时间，仅对 HPPT, HPPTS，FTP 连接有效。

. context: 如果定义了,则必须是一个描述 SSL 选项的 ss.SSLContext 实例,查看 HPPTSConnection 相关资料获取详细描述。

. cafile 和 capath：参数定义了 HTTPS 请求中的可信任 CA 证书，cadefault 参数可以省略。

调用该方法返回一个像 context manager 一样工作且包含如下方法的对象 (http.client.HTTPSRespon
的实例)：

. gerurl()：返回一个资源索引 URL，决定是否需要重定向。

. info()：返回有关 HTTP 报头的元信息。

. getcode()：获取响应码通。

urllib.request.install_opener(opener):
安装一个 openerDirector 实例作为默认的 opener.

urllib.request.build_opener([handler, ...]):
返回一个 OpenerDirector 的实例。

urllib.request.pathname2url(path):
转换路径到 URL，默认为/，并不会生成一个完整的 URL。

urllib.request.url2pathname(path):
转换 the path component path from a percent-encoded URL to the local syntax for a path.

urllib.request.getproxies():
返回一个代理服务器调用的 map 的字典。

下面的这些类也是提供的：class urllib.request.Request(url, data=None, headers=,
origin_req_host=None, unverifiable=False, method=None):
未知

class urllib.request.OpenerDirector:
未知

class urllib.request.BaseHandler:

未知

class urllib.request.HTTPDefaultErrorHandler:

未知

class url.request.HTTPRedirecHandler:

未知

class urllib.request.HTTPCookieProcessor(cookiejar=None):

未知

class urllib.request.ProxyHandler(proxies=None):

未知

class urllib.request.HTTPPasswordMgr:

未知

class urllib.request.HTTPPasswordMgrWithDefaultReal:

未知

class urllib.request.AbstractBasicAuthHandler(password_mgr=None):

未知

class urllib.request.HTTPBasicAuthHandler(password_mgr=None):

未知

class urllib.request.ProxyBasicAuthHandler(password_mgr=None):

未知

class urllib.request.AbstractDigestAuthHandler(password_mgt=None):

未知

class urllib.request.HTTPDigestAuthHandler(password_mgr=None):

未知

class urllib.request.ProxyDigestAuthHandler(password_mgt=None):

未知

class urllib.request.HTTPHandler:

未知

class urllib.request.HTTPSHandler(debuglevel=0, context=None, check_hostname=None):

未知

class urllib.request.FileHandler:

未知

class urllib.request.DataHandler:

未知

class urllib.request.FTPHandler:

未知

class urllib.request.CacheFTPHandler:

未知

class urllib.request.UnknownHandler:

未知

class urllib.request.HTTPErrorProcessor:

未知

21.6.1 Request Objects

下面的方法描述了 Request 的共有方法，因此下面的所有的方法都可以在继承类中进行重载。

Request.full_url:

未知

Request.type:

未知

Request.host:

未知

Request.origin_req_host:

未知

Request.selector:

未知

Request.data:

未知

Request.unverifiable:

未知

Request.method:

未知

Request.get_method():

未知

Request.add_header(key, val):

未知

Request.add_unredirected_header(key, header):

未知

Request.has_header(header):

未知

Request.remove_header(header):

未知

Request.get_full_url():

未知

Request.set_proxy(host, type):

未知

Request.get_header(header_name, default=None):

未知

Request.header_items():

未知

21.6.2 OPenerDirector Objects

OpenerDirector instance 有下面的方法。

OpenerDirector.add_Handler(Handler):

未知

OpenerDirector.open(url, data=None[, timeout]):

未知

OpenerDirector.error(proto, *args):

未知

21.6.3 BaseHandler Objects

Basehandler objects provide a couple of methods that are directly useful, and others that are meant to be used by derived classes. These are intended for direct use:

BaseHandler.add_parent(director):

未知

BaseHandler.close():

未知

21.7 urllib.response

the urllib.response module defines functions and classes which define a minimal file like interface, including read() and readline(). The typical reponse object is an addinfourl instance, which defines an info() method and that returns headers and a geturl() method that returns the url. Functions defined by this module are used internally by the urllib.request module.

21.8 urllib.parse

This module defines a standard interface to break Uniform Response Locator(URL) strings up in components (addressing scheme, network location, path etc.) to combine the components back into a URL string, and to convert a "relative URL" to an absolute URL given a "base URL".

The module has been designed to match the internet RFC on Relative Uniform Resource Locator. It supports the following scheme: file, ftp, gopher, hdl, http, https, imap, mailto, mms, news, nntp, prospero, rsync, rtsp, rtspu, sftp, shttp, sip, sips, snews, svn, svn+ssh, telnet, wais, ws, wss.

21.9 urllib.error

21.10 urllib.robotparser

21.11 http

21.12 http.client

21.13 ftplib

21.14 poplib

21.15 imaplib

21.16 nntplib

21.17 smtplib

21.18 smtpd

21.19 telnetlib

21.20 uuid

21.21 socketserver

21.22 http.server

21.23 http.cookies

21.24 http.cookiejar

21.25 xmlrpc

21.26 xmlrpc.client

21.27 xmlrpc.server

第二十二章　Multimedia Services

第二十三章　Internationalization

第二十四章　Program Frameworks

第二十五章　Graphical User Interfaces with Tk

第二十六章　Development Tools

第二十七章　Debugging and Profiling

第二十八章　Software Packaging and Distribution

第二十九章　Python Runtime Services

第三十章　Custom Python Interpreters

第三十一章　Importing Modules

第三十二章　Python Language Services

第三十三章　Miscellaneous Services

第三十四章　MS Windows Specific Services

第三十五章　Unix Specific Services

第三十六章　Superseded Modules