内建函数

1. abs(x)　　（绝对值）

功能：返回一个数的绝对值。参数可以是普通的整数，长整数或者浮点数。如果参数是个复数，返回它的模。

2. all(iterable)　　（所有元素为真）

功能：如果 iterable 的所有元素为真（或者 iterable 为空）， 返回 True。

3. any(iterable)　　（任一元素为真）

功能：如果 iterable 的任一元素为真，返回 True。如果 iterable 为空，返回 False。

4. bin(x)　　（转化成二进制）

将一个整数转化成一个二进制字符串。结果是一个合法的 Python 表达式。如果 x 不是一个 Python int 对象，它必须定义一个返回整数的__index__()方法。

5. bool([x])

将一个值转化成布尔值，使用标准的真值测试例程。bool 也是一个类，它是 int 的子类。bool 不能被继承。它唯一的实例就是 False 和 True。

6. bytearray([source[, encoding[, errors]]])　　（字节数组）

返回一个新的字节数组。bytearray 类型是一个可变的整数序列，整数范围为 0 <= x < 256（即字节）。

7. callable(object)　　（可调用、返回真）

如果 object 参数可调用，返回 True；否则返回 False。如果返回真，对其调用仍有可能失败；但是如果返回假，对 object 的调用总是失败。注意类是可调用的（对类调用返回一个新实例）；如果类实例有__call__()方法，则它们也是可调用的。

8. chr(i)　　（返回单字符字符串）

返回一个单字符字符串，字符的 ASCII 码为整数 i。例如，chr(97)返回字符串'a'。 它是 ord()的逆运算。

9. classmethod(function)　　（类方法）

将 function 包装成类方法。类方法接受类作为隐式的第一个参数，就像实例方法接受实例作为隐式的第一个参数一样。声明一个类方法，使用这样的惯例：

```
class C(object):
    @classmethod
    def f(cls, arg1, arg2, ...):
        ...
```

10. cmp(x, y)　　（比较）

比较两个对象 x 和 y，根据结果返回一个整数。如果 x < y，返回负数；如果 x == y，返回 0；如果 x > y，返回正数。

11. delattr(object, name)　　（删除属性）

这个函数和 setattr()有关。参数是一个对象和一个字符串。字符串必须是对象的某个属性的名字。只要对象允许，这个函数删除该名字对应的属性。例如，delattr(x, 'foobar')等同于 del x.foobar。

12. dict(**kwarg)　　（字典）

dict(mapping, **kwarg)

dict(iterable, **kwarg)

创建一个新字典。dict 对象就是字典类。

13. dir([object]) （本地作用域内的名字列表）

如果没有参数，返回当前本地作用域内的名字列表。如果有参数，尝试返回参数所指明对象的合法属性的列表。

14. enumerate(sequence, start=0) （枚举对象）

返回一个枚举对象。sequence 必须是个序列、迭代器 iterator、或者支持迭代的对象。

15. execfile(filename[, globals[, locals]]) （执行文件中的语句）

该函数类似于 exec 语句，它解析执行储存在文件中的 Python 语句。它不同于 import 语句的地方在于它不使用模块管理——它无条件的读入文件且不会创建一个新模块。

注意：exec()用来执行储存在字符串或文件中的 Python 语句。

下面看一个简单的例子：

```
>>> exec 'print "Hello World"'
Hello World
```

16. filter(function, iterable) （过滤）

用于过滤序列，把传入的函数依次作用于每个元素，如果函数返回值为真，则保留该元素，否则删除。

17. float([x]) （转化成浮点数）

将字符串或者数字转化成浮点数。如果参数是字符串，它必须包含小数或者浮点数（可以有符号），周围可以有空白。参数也可以是[+|-]nan 或者[+|-]inf。其它情况下，参数可以是原始 / 长整数或者浮点数，（以 Python 的浮点数精度）返回具有相同值的浮点数。如果没有参数，返回 0.0。

```
>>> '{0} {1}:{2}'.format('hello', '1', '7')
'hello 1:7'
```

18. format(value[, format_spec]) （格式化、%、差不多）

将 value 转化成“格式化”的表现形式，格式由 format_spec 控制。这个函数和%函数差不多，都是用 value 值按要求插入前面的内容。

19. getattr(object, name[, default]) （获取属性值）

获取 object 的属性值。name 必须是个字符串。如果字符串是对象某个属性的名字，则返回该属性的值。例如，getattr(x, 'foobar')等同于 x.foobar。

20. Globals() （全局变量的字典）

返回当前模块中的全局变量的字典。当前模块指定义的模块而不是调用的模块它。

21. hasattr(object, name) （属性是否存在）

判断对象 object 的属性（name 表示）是否存在。如果属性（name 表示）存在，则返回 True，否则返回 False。参数 object 是一个对象，参数 name 是一个属性的字符串表示。

22. hash(object) （获取 hash）

返回对象的 hash（哈希/散列）值（如果有的话）。hash 值是整数。用于在字典查找时快速比较字典的键。相同的数值有相同的 hash（尽管它们有不同的类型，比如 1 和

1.0）。

23. help([object]) （帮助）

调用内置的帮助系统。（这个函数主要用于交互式使用。）如果没有参数，在解释器的控制台启动交互式帮助系统。如果参数是个字符串，该字符串被当作模块名，函数名，类名，方法名，关键字或者文档主题而被查询，在控制台上打印帮助页面。如果参数是其它某种对象，生成关于对象的帮助页面。

24. id(object) （获取 id）

返回对象的 id。这是一个整数（或长整数），保证在对象的生命期内唯一且不变。生命期不重叠的两个对象可以有相同的 id()值。

25. isinstance(object, classinfo) （是实例，或子类的实例）

如果参数 object 是参数 classinfo 的一个实例，或者是子类的实例，返回真。如果 classinfo 是类型对象（新式类）而 object 是该类型对象；或者是其子类（直接的，间接的，或者 virtual），返回真。如果 object 不是给定类型的类实例或者对象，该函数总是返回假。如果 classinfo 既不是类对象，也不是类型对象，它可以是类 / 类型对象的元组，或者递归包含这样的元组（不接受其它的序列类型）。如果 classinfo 不是类，类型，类 / 类型的元组，抛出 TypeError 异常。

26. issubclass(class, classinfo) （是子类，返回真）

如果 class 是 classinfo 的子类（直接的，间接的，或者 virtual) ，返回真。

27. iter(o[, sentinel]) （迭代对象）

返回一个 iterator 对象。

28. len(s) （长度）

返回对象的长度（元素的个数）。参数可以是序列（如字符串，字节，元组，列表或者范围）或者集合（如字典，集合或者固定集合）。

29. list([iterable]) （列表）

返回一个列表，其元素来自于 iterable（保持相同的值和顺序）。

30. Locals() （局部）

更新并返回表示当前局部变量的字典。当 locals 在函数块中而不是类块中被调用时，locals()返回自由变量。

31. map(function, iterable, ...) （应用、每个）

将 function 应用于 iterable 的每个元素，返回结果的列表。如果有额外的 iterable 参数，并行的从这些参数中取元素，并调用 function。如果一个参数比另外的要短，将以 None 扩展该参数元素。如果 function 是 None 使用特性函数；如果有多个参数，map()返回一元组列表，元组包含从各个参数中取得的对应的元素（某种变换操作）。iterable 参数可以是序列或者任意可迭代对象；结果总是列表。

32. min(iterable[, key]) （最小）

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

返回可迭代的对象中的最小的元素，或者返回 2 个或多个参数中的最小的参数。

33. next(iterator[, default]) （迭代、next）

通过调用 iterator 的 next()方法，得到它的下一个元素。如果有 default 参数，在迭代器迭代完所有元素之后返回该参数；否则抛出 StopIteration。

34. Object() （新的无特征的对象）

返回一个新的无特征的对象。object 是所有新式类的基类。它有对所有新式类的实例通用的方法。

35. oct(x)　　（八进制字符串）

将一个（任意尺寸）整数转化成一个八进制字符串。结果是一个合法的 Python 表达式。

36. open(name[, mode[, buffering]])　　（打开文件）

打开一个文件，返回一个 file 类型的对象，file 类型描述于 File Objects 章节。如果文件不能打开，抛出 IOError。当要打开一个文件，优先使用 open()，而不是直接调用 file 构造函数。

37. ord(c)

是 chr()函数（对于 8 位的 ASCII 字符串）或 unichr()函数（对于 Unicode 对象）的配对函数，它以一个字符（长度为 1 的字符串）作为参数，返回对应的 ASCII 数值，或者 Unicode 数值，如果所给的 Unicode 字符超出了你的 Python 定义范围，则会引发一个 TypeError 的异常。

38. pow(x, y[, z])　　（次幂）

返回 x 的 y 次幂; 如果 z 提供的时候，, 返回 x 的 y 次幂，然后对 z 取模。

39. print(*objects, sep=' ', end='\n', file=sys.stdout)　　（打印到文件流）

以 sep 分割，end 的值结尾，将对象打印到文件流中。sep, end 和 file，如果提供这三个参数的话，必须以键值的形式。

40. range(stop)　　（数列、0）

range(start, stop[, step])

这是一个创建包含数列的列表的通用函数。它最常用于 for 循环。参数必须为普通的整数。如果 step 参数省略，则默认为 1。如果 start 参数省略，则默认为 0。该函数的完整形式返回一个整数列表[start, start + step, start + 2 * step, ...]。如果 step 为正，则最后一个元素 start + i * step 最大且小于 stop；如果 step 为负，则最后一个元素 start + i * step 最小且大于 stop。step 必须不能为零（否则会引发 ValueError）。示例：

```
>>>
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(1, 11)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> range(0, 30, 5)
[0, 5, 10, 15, 20, 25]
>>> range(0, 10, 3)
[0, 3, 6, 9]
>>> range(0, -10, -1)
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
>>> range(0)
[]
>>> range(1, 0)
```

[]

41. raw_input([prompt])

如果有 prompt 参数，则将它输出到标准输出且不带换行。该函数然后从标准输入读取一行，将它转换成一个字符串（去掉一个末尾的换行符），然后返回它。当读到 EOF 时，则引发 EOFError。

42. reload(module)　　（重新加载）

重新加载之前已经引入过的模块。参数必须是一个模块对象，所以之前它必须成功导入过。

43. repr(object)　　（解释器可读取）

转化为解释器可读取的字符串。

44. reversed(seq)　　（反转）

反转一个序列对象，将其元素从后向前颠倒构建成一个新的迭代器。seq 必须是一个具有__reversed__() 方法或支持序列协议的对象（整数参数从 0 开始的__len__()方法和__getitem__() 方法）。

45. round(number[, ndigits])　　（浮点型近似值）

返回一个浮点型近似值，保留小数点后 ndigits 位。如果省略 ndigits，它默认为零。结果是一个浮点数。

46. set([iterable])

返回一个新的 set 对象，其元素可以从可选的 iterable 获得。set 是一个内建的类。关于该类的文档，请参阅 set 和集合类型 — set, frozenset。

47. setattr(object, name, value)　　（设置属性）

给对象的属性赋值，若属性不存在，先创建再赋值。

48. slice(stop)　　（切片对象）

slice(start, stop[, step])

本函数是实现切片对象，主要用在切片操作函数里的参数传递。　　（slice 的返回值本身是个切片）

```
#slice()

myslice = slice(5)
print(myslice)

l = list(range(10))
print(l[myslice])
```

输出：

```
slice(None, 5, None)
[0, 1, 2, 3, 4]
```

49. sorted(iterable[, cmp[, key[, reverse]]])　　（排序）

对元素进行排序，返回一个新的列表，原列表不变。

50. str(object='')

将值转化为适于人阅读的形式。

51. sum(iterable[, start])　　（相加）

将 start 以及 iterable 的元素从左向右相加并返回总和。start 默认为 0。iterable 的元素通常是数字，start 值不允许是一个字符串。

52. super(type[, object-or-type]) （父类）

使用 super 关键字初始化父类

53. tuple([iterable]) （转换为元组）

将列表转换为元组。iterable 可以是一个序列、支持迭代操作的容器或迭代器对象。如果 iterable 已经是一个元组，它将被原样返回。例如，tuple('abc')返回('a', 'b', 'c')，tuple([1, 2, 3])返回(1, 2, 3)。如果没有给出参数，则返回一个空的元组()。

54. type(object) （获取、类型）

type(name, bases, dict)

只有一个参数时，获取 object 的类型。返回值是一个类型对象。建议使用内建函数 isinstance()测试一个对象的类型。

55. unichr(i) （Unicode 字符）

返回 Unicode 码为整数 i 的 Unicode 字符。例如，unichr(97)返回字符串 u'a'。

56. xrange(stop) （rang()、对象）

xrange(start, stop[, step])

该函数与 range()非常相似，但是它返回一个 xrange 对象而不是一个列表。

57. zip([iterable, ...]) （多、序、返、列）

zip 函数接受任意多个（包括 0 个和 1 个）序列作为参数，返回一个 tuple 列表。该函数返回一个以元组为元素的列表，其中第 i 个元组包含每个参数序列的第 i 个元素。返回的列表长度被截断为最短的参数序列的长度。当多个参数都具有相同的长度时，zip()类似于带有一个初始参数为 None 的 map()。只有一个序列参数时，它返回一个 1 元组的列表。没有参数时，它返回一个空的列表。

浮点数方法

1. float.as_integer_ratio() （分子分母）

返回一对整数，它们的比例准确地等于浮点数的原始值，且分母为正数。（也就是返回分子分母形式）

2. float.is_integer() （仅有整数，小数部分为 0）

如果浮点数实例仅有整数，小数部分为 0，则返回 True，否则返回 False：

3. float.hex() （十六进制）

返回浮点数的十六进制字符串表示形式。对于有限的浮点数，这种表示形式总是包括一个前导的 0x 和尾部 p 及指数。

4. float.fromhex(s) （浮点数）

类方法，返回十六进制字符串 s 表示的浮点数。字符串 s 可以有前导和尾随空白。

字符串方法

1. str.capitalize() （大写、小写）

将字符串的第一个字母变成大写，其他字母变小写。返回的是字符串的副本。

2. str.center(width[, fillchar]) （原字符串居中、空格填充）

返回一个原字符串居中，并使用空格填充至长度 width 的新字符串。默认填充字符为空格。

3. str.count(sub[, start[, end]]) （统计次数）

统计字符串里某个字符出现的次数。可选参数为在字符串搜索的开始与结束位置。

4. str.encode([encoding[, errors]]) （编码成指定）

将 unicode 形式的 str 编码成指定编码形式的字符串。errors 参数可以指定不同的错误处理方案。

5. str.endswith(suffix[, start[, end]]) （指定后缀）

判断字符串是否以指定后缀 suffix 结尾，如果以指定后缀结尾返回 True，否则返回 False。可选参数"start"与"end"为检索字符串的开始与结束位置。suffix 也可以是一个元组。可选的 start 表示从该位置开始测试。可选的 end 表示在该位置停止比较。

6. str.expandtabs([tabsize]) （转为空格）

把字符串中的 tab 符号('\t')转为空格，tab 符号('\t')默认的空格数是 8

7. str.find(sub[, start[, end]]) （查找、索引）

查找字符串中是否包含子字符串 sub ，如果指定 beg（开始） 和 end（结束） 范围，则检查是否包含在指定范围内，如果包含子字符串返回开始的索引值，否则返回-1。

8. str.rfind(sub[, start[, end]]) （最后一次出现的位置）

返回字符串最后一次出现的位置，如果没有匹配项则返回-1。

9. str.format(*args, **kwargs) （格式化）

执行字符串格式化操作。类似于 print 语句中的%。

10. str.index(sub[, start[, end]]) （查找、索引）

类似 find()，但未找到子字符串时引发 ValueError 。

11. str.rindex(sub[, start[, end]]) （最后出现的位置）

返回子字符串 sub 在字符串中最后出现的位置，如果没有匹配的字符串会报异常，你可以指定可选参数[beg:end]设置查找的区间。

12. str.isalnum() （数字、字母）

如果字符串中的所有字符都是数字或者字母，并且至少有一个字符，则返回 true，否则返回 false。

13. str.isalpha() （字母）

字符串至少有一个字符并且都是字母，则返回 true，否则返回 false。

14. str.isdigit() （数字）

如果在字符串中的所有字符都是数字并且至少一个字符，则返回 true。否则返回 false。

15. str.islower() （小写）

检测字符串是否由小写字母组成。

16. str.isspace() （空格）

检测字符串是否只由空格组成。

17. str.istitle()　　（首字母大写）

检测字符串中所有的单词拼写首字母是否为大写，且其他字母为小写。

18. str.isupper()　　（大写）

检测字符串中所有的字母是否都为大写。

19. str.join(iterable)　　（连接、生成）

将序列中的元素以指定的字符连接生成一个新的字符串。

20. str.ljust(width[, fillchar])　　（左对齐）

返回一个原字符串左对齐，并使用空格填充至指定长度的新字符串。如果指定的长度小于原字符串的长度则返回原字符串。

21. str.rjust(width[, fillchar])　　（右对齐）

返回一个原字符串右对齐,并使用空格填充至长度 width 的新字符串。如果指定的长度小于字符串的长度则返回原字符串。

22. str.swapcase()　　（大小写转换）

对字符串的大小写字母进行转换。返回字符串的一个拷贝，其中大写字符转换为小写，小写字符转换为大写。

23. str.lower()　　（大写转小写）

返转换字符串中所有大写字符为小写。

24. str.upper()　　（小写转大写）

将字符串中的小写字母转为大写字母。

25. str.strip([chars])　　（移除）

移除字符串头尾指定的字符（默认为空格）。返回字符串的副本。

26. str.lstrip([chars])　　（截掉左边）

截掉字符串左边的空格或指定字符。返回字符串的副本。

27. str.rstrip([chars])　　（删除末尾）

删除 string 字符串末尾的指定字符（默认为空格）。

28. str.partition(sep)　　（分割）

根据指定的分隔符将字符串进行分割，返回包含分隔符前面部分、分隔符本身和分隔符之后部分的 3 元组。如果找不到分隔符，返回包含字符串本身，跟着两个空字符串的 3 元组。

29. str.rpartition(sep)　　（从右边开始分割）

根据指定的分隔符将字符串进行分割，返回包含分隔符前面部分、分隔符本身和分隔符之后部分的 3 元组。如果找不到分隔符，返回包含字符串本身，跟着两个空字符串的 3 元组。该函数类似于 partition()函数，不过该函数是从右边开始分割。

30. str.replace(old, new[, count])　　（替换）

把字符串中的 old（旧字符串） 替换成 new(新字符串)，如果指定第三个参数 max，则替换不超过 max 次。

31. str.split([sep[, maxsplit]])　　（从左到右、切片）

通过指定分隔符从左至右对字符串进行切片，如果参数 maxsplit 有指定值，则仅分隔 num 个子字符串。

32. str.rsplit([sep[, maxsplit]]) （从右到左、切片）

通过指定分隔符从右至左对字符串进行切片，如果参数 maxsplit 有指定值，则仅分隔 num 个子字符串。

33. str.splitlines([keepends]) （行分隔、列表）

按照行('\r', '\r\n', \n')分隔，返回一个包含各行作为元素的列表，如果参数 keepends 为 False，不包含换行符，如果为 True，则保留换行符。

34. str.startswith(prefix[, start[, end]]) （检查、开头）

检查字符串是否是以指定子字符串开头，如果是则返回 True，否则返回 False。如果参数 beg 和 end 指定值，则在指定范围内检查。

35. str.title() （大写开头）

返回"标题化"的字符串,就是说所有单词都是以大写开始，其余字母均为小写

36. str.translate(table[, deletechars]) （转换）

根据参数 table 给出的表(包含 256 个字符)转换字符串的字符, 要过滤掉的字符放到 deletechars 参数中。

37. str.zfill(width) （指定长度、左对齐）

返回指定长度的字符串，字符串右对齐，左边填充 0。

可变序列类型操作

1. s[i] = x （下图是所有序列类型都适用的操作符）

   给 s[i]赋值。 （in、not in、len、min、max）

2. s[i:j] = t （s.index、s.count、加、乘、切片）

   从 s[i]到 s[j]都赋值为 t。

3. del s[i:j]

   删除 s[i]到 s[j]。

4. s[i:j:k] = t

   从 s[i]到 s[j]每隔 k-1 个元素重新赋值为 t。k 是步长。

5. del s[i:j:k]

   取出 s[i]到 s[j]间每隔 k-1 个元素的值，并删除。

6. s.append(x) （a、c、e、i、i、p、r、r、s）

   在序列末尾添加新的对象。

7. s.count(x)

   统计某个元素在序列中出现的次数。

8. s.extend(x) （追加、连接、扩展） （追、连、扩）

在序列末尾一次性追加另一个序列中的多个值（用新列表扩展原来的序列）。这个操作看起来很像连接操作，两者最主要区别在于：extend 方法修改了被扩展的序列。而原始的连接操作则不然，它会返回一个全新的列表。

9. s.index(x[, i[, j]])

   从序列中找出某个值第一个匹配项的索引位置。

```
x in s
x not in s
s + t
s * n, n * s
s[i]
s[i:j]
s[i:j:k]
len(s)
min(s)
max(s)
s.index(x)
s.count(x)
```

10. s.insert(i, x)

    将对象插入序列。

11. s.pop([i])

    移除序列中的一个元素（默认最后一个元素），并且返回该元素的值。

12. s.remove(x)

    移除序列中某个值的第一个匹配项。

13. s.reverse()

    反向序列中元素。

14. s.sort([cmp[, key[, reverse]]])

    对原序列进行排序。

字典操作

1. len(d)　　（个数）

   返回字典 d 中元素的个数。

2. d[key]

   返回字典 d 中键为 key 的元素。如果 key 不在映射中，则引发一个 KeyError。

3. d[key] = value

   设置 d[key]的值为 value。

4. del d[key]

   从 d 中删除 d[key] 。如果 key 不在映射中，则抛出 KeyError。

5. key in d

   如果 d 有一个键 key，则返回 True，否则返回 False。

6. key not in d

   相当于 not key in d。

7. iter(d)

   返回字典的键的一个迭代器。这是 iterkeys()的快捷方式。

8. clear()

   从字典中移除所有项。

9. copy()

   返回字典的一个浅拷贝。

10. fromkeys(seq[, value])　　（新字典）

    函数用于创建一个新字典，以序列 seq 中元素做字典的键，value 为字典所有键对应的初始值。

11. get(key[, default])　　（返回值）

    如果 key 在字典中，则返回 key 对应的值，否则返回 default。如果没有预置 default 的值，则它默认为 None，所以此方法永远不会引发 KeyError。

12. has_key(key) （是否在字典中存在）

测试 key 是否在字典中存在。has_key()已经被 key in d 弃用。

13. Items() （可遍历的键值对元组）

以列表返回可遍历的（键，值） 元组数组。

14. Iteritems() （键值对、迭代）

返回字典的键值对(key, value)的一个迭代器。请参阅 dict.items()注释。

15. Iterkeys() （键、迭代）

返回字典的键的一个迭代器。请参阅 dict.items()注释。

16. Itervalues() （值、迭代）

在字典的值返回一个迭代器。请参阅 dict.items()注释。

17. Keys() （键列表）

返回的字典的键列表的副本。请参阅 dict.items()注释。

18. len(dictview) （条数）

返回字典中的条目数。

19. pop(key[, default]) （删除、返回）

如果 key 在字典中，删除它并返回其值，否则返回 default。如果没有给出 default 且 key 不是在字典中，则引发一个 KeyError。

20. Popitem() （移除一个对）

从字典中移除并返回任意一个(key, value)对。

21. update([other]) （更新）

依据 other 更新词典的键/值对，覆盖现有的键。返回 None。

22. Values() （值的列表）

返回字典的值的列表的副本。请参阅 dict.items()注释。

23. Viewitems() （字典项）

返回字典项(key, value)对的一个新视图。

24. Viewkeys()

返回字典的键的新的视图。有关详细信息，请参阅以下文档的视图对象。

25. Viewvalues()

返回字典的值的新的视图。有关详细信息，请参阅以下文档的视图对象。

文件方法

1. file.close()

关闭该文件。

2. file.flush()

刷新缓冲区，即将缓冲区中的数据立刻写入文件，同时清空缓冲区，不需要是被动的等待输出缓冲区写入。一般情况下，文件关闭后会自动刷新缓冲区，但有时你需要在关闭前刷新它，这时就可以使用 flush() 方法。注 flush()不一定写入文件的数据到磁盘。在 flush()后

紧接着使用 os.fsync()来确保这种行为。

3. file.fileno() （描述符）

返回一个整型的文件描述符(file descriptor FD 整型)，可用于底层操作系统的 I/O 操作。

4. file.isatty() （连接、终端）

如果文件连接到一个终端设备返回 True，否则返回 False。

5. file.next() （下一行）

返回文件下一行。

6. file.read([size])

从文件读取指定的字节数，如果未给定或为负则读取所有。

7. file.readline([size])

从文件中读取一整行。结尾的换行符包含在字符串中 （当文件为非一整行结束时就可能不在）。[6]如果 size 参数存在且非负数，它是 （包括尾随换行符） 的最大字节数和可能返回不完整的行。当 size 不是 0 时，仅当遇到 EOF 时才立即返回空字符串。

8. file.seek(offset[, whence])

用于移动文件读取指针到指定位置。

9. file.tell()

返回文件的当前位置，类似 stdio 的 ftell()。

10. file.truncate([size]) （截取）

截取文件，截取的字节通过 size 指定，默认为当前文件位置。

11. file.write(str)

向文件中写入字符串。无返回值。

12. file.writelines(sequence)

向文件中写入一个字符串序列。序列可以是任何可迭代的对象产生的字符串，通常的字符串列表。没有返回值。（名称被用于匹配 readlines()； writelines()不会添加行分隔符。）

常用路径名操作

1. os.path.abspath(path) （绝对路径）

返回路径名 path 的规范化的绝对路径。在大多数平台，这等同于这样 normpath(join(os.getcwd(), path))调用 normpath()函数。

2. os.path.basename(path) （文件名）

返回路径名 path 的文件名。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（''）。

3. os.path.commonprefix(list) （共有的最长的路径）

返回 list(多个路径)中，所有 path 共有的最长的路径。。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。

4. os.path.dirname(path) （目录名）

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

5. os.path.exists(path) （存在）

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。

6. os.path.lexists(path) （存在的路径）

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

7. os.path.expanduser(path) （转换）

把 path 中包含的"~"和"~user"转换成用户目录

8. os.path.expandvars(path) （替换）

根据环境变量的值替换 path 中包含的”$name”和”${name}”

9. os.path.getatime(path) （最后访问）

返回 path 的最后访问时间。

10. os.path.getmtime(path) （最后修改）

返回 path 的最后修改时间。

11. os.path.getctime(path) （u 后修、w 创建）

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。

12. 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。

13. os.path.isabs(path) （绝对路径名）

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:）。

14. 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。

15. 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。

16. 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。

17. os.path.ismount(path) （挂载点）

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。

18. os.path.join(path1[, path2[, ...]]) （连接）

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（如果在 Windows 上，如果有盘符，盘符也会被丢弃）。返回的值是 path1、path2 等等的连接，每个非空部分除了最后一个后面只跟随一个目录分隔符（os.sep）。

19. os.path.normcase(path) （标准化路径名）

标准化路径名的大小写。在 Unix 和 Mac OS X 上，原样返回路径；在大小写不敏感的系统上，将路径名转换成小写。在 Windows 上，还会将斜线转换成反斜线。

20. os.path.normpath(path) （标准化路径名）

标准化路径名，合并多余的分隔符和上层引用，这样 A//B, A/B/, A/./B 和 A/foo/../B 都变成 A/B。该字符串操作可能改变包含符号链接的路径的含义。在 Windows 上，还会将斜线转换成反斜线。若要标准化大小写，请使用 normcase()。

21. os.path.realpath(path) （规范名字）

返回指定的文件名的规范名字，并消除路径中遇到的任何符号链接（如果操作系统支持的话）。

22. os.path.relpath(path[, start]) （相对路径）

返回自当前目录或者可选的 start 目录开始计算的 path 相对文件路径。这只是单纯的路径计算：不会访问文件系统以确认 path 或者 start 的存在性或者属性。

23. os.path.sameopenfile(fp1, fp2) （相同）

如果描述符 fp1 和 fp2 引用的是相同的文件，则返回 True。可用的平台：Unix。

24. os.path.samestat(stat1, stat2) （相同）

如果文件信息元组 stat1 和 stat2 引用的是相同的文件，则返回 True。这些结构可能是由 os.fstat()、os.lstat()或者 os.stat()返回。该函数底层使用 samefile() 和 sameopenfile()实现比较。可用的平台：Unix。

25. os.path.split(path) （分割）

将路径名 path 分割成(head, tail)，其中 tail 路径名的最后一部分，head 是其之前的所有部分。tail 部分永远不会包含斜线；如果 path 以斜线结束，tail 将为空。如果 path 中没有斜线，head 将为空。如果 path 为空，head 和 tail 两个都将为空。尾部的斜线会从 head 中去除掉，除非它是根目录（只包含一个或多个斜线的目录）。对于所有情况，join(head, tail)都将返回一个和 path 相同的位置（但是表示的字符串可能不一样）。

26. os.path.splitdrive(path) （分割 driver 对）

将 path 分割成一个(drive, tail)对，其中 drive 是一个驱动器描述符或者空字符串。在没有使用驱动器描述符的系统上，drive 将永远是空字符串。无论哪一种情况，drive + tail 将和 path 相同。

27. os.path.splitext(path) （分割 root 对）

将路径名 path 分割成一个(root, ext)对，使得 root + ext == path，ext 为空或者以点号开始并至多包含一个点。路径名前面的点号将被忽略；splitext('.cshrc') 返回 ('.cshrc', '')。

28. os.path.splitunc(path) （分割 unc 对）

将 path 分隔成一个(unc, rest)对，unc 是 UNC 挂载点（例如 r'\\hostmount'），rest 是路径剩下的部分（例如 r'\path\file.ext'）。对于包含驱动器字母的路径，unc 将永远是空字符串。可用的平台：Windows。

fileinput

说明：fileinput 模块可以对一个或多个文件中的内容进行迭代、遍历等操作。该模块的 input()函数有点类似文件 readlines()方法，区别在于前者是一个迭代对象，需要用 for 循环迭代，后者是一次性读取所有行。用 fileinput 对文件进行循环遍历，格式化输出，查找、替换等操作，非常方便。典型用法：　　（迭代、遍历）

```
import fileinput
for line in fileinput.input():
    process(line)
```

1. fileinput.input([files[, inplace[, backup[, bufsize[, mode[, openhook]]]]]])　　（创建实例）

创建 FileInput 类的一个实例。该实例将用作此模块的函数的全局状态，也会返回在迭代过程中使用。对此函数的参数将传递沿线到 FileInput 类的构造函数。

2. fileinput.filename()　　（当前文件名）

返回当前正在读取的文件的名称。在读第一行之前，则返回 None。

3. fileinput.fileno()　　（文件描述符）

返回整数为当前文件的"文件描述符"。当没有文件打开时 （第一行之前和文件之间），则返回-1。

4. fileinput.lineno()　　（行数）

返回当前已经读取的行的数量（或者序号）。在读第一行之前，则返回 0。已读取的最后一行的最后一个文件后，返回的行的行号。

5. fileinput.filelineno()　　（行号）

返回当前读取的行的行号。在读第一行之前，则返回 0。已读取的最后一行的最后一个文件后，返回该行在文件内的行号。

6. fileinput.isfirstline()　　（第一行）

如果当前行是文件的第一行，返回 true ； 否则返回假。

7. fileinput.isstdin()　　（最后一行从 stdin 读取）

如果最后一行从 stdin 读取，返回 true； 否则返回 false。

8. fileinput.nextfile()　　（下一个文件）

关闭当前文件，移动到下一个文件

9. fileinput.close()

关闭序列。

stat

说明：stat 模块描述了 os.stat(filename)返回的文件属性列表中各值的意义。我们可方便地根据 stat 模块存取 os.stat()中的值。　　（stat、属性）

1. stat.S_ISDIR(mode)　　（目录）

如果 mode 来自一个目录，返回非零。

2. stat.S_ISCHR(mode)　　（特、字、设）

   如果 mode 来自特殊的字符设备文件，返回非零。

3. stat.S_ISBLK(mode)　　（特、块）

   如果 mode 来自特殊的块设备文件，返回非零。

4. stat.S_ISREG(mode)　　（常规）

   如果 mode 来自一个常规的文件，返回非零。

5. stat.S_ISFIFO(mode)　　（FIFO）

   如果 mode 来自 FIFO（命名管道），返回非零。

6. stat.S_ISLNK(mode)¶　　（链接）

   如果 mode 来自一个符号链接，返回非零。

7. stat.S_ISSOCK(mode)　　（套接字）

   如果 mode 来自一个套接字，返回非零。

filecmp 模块

说明：filecmp 模块定义函数来比较文件和目录，与各种可选的时间/正确性权衡取舍。（比、文、目）

1. filecmp.cmp(f1, f2[, shallow])　　（比较）

比较文件 f1 和 f2，如果它们看上去相等则返回 True，否则返回 False。除非给出 shallow 并且为假，否则具有相同 os.stat()签名的文件被视为相等。比较过的文件不会使用该函数再次比较，除非它们的 os.stat()签名发生更改。

注意该函数没有调用外部的程序，因此给它带来可移植性和高效率。

2. filecmp.cmpfiles(dir1, dir2, common[, shallow])　　（比、目、反、三）

比较 dir1 和 dir2 两个目录中文件，文件的名字由 common 给出。返回三个文件名列表：match、mismatch、errors。mtach 包含匹配文件的列表，mismatch 包含不匹配文件的列表，errors 列出无法比较的文件的名称。如果文件在其中一个目录中不存在，用户没有足够的权限读取它们，或者某些其他原因不可比较，那么它们将在 errors 中列出。

tempfile 模块

1. tempfile.TemporaryFile([mode='w+b'[, bufsize=-1[, suffix=''[, prefix='tmp'[, dir=None]]]]])　　（对象）

该函数返回一个类文件对象(file-like)用于临时数据保存（实际上对应磁盘上的一个临时文件）。当文件对象被 close 或者被 del 的时候，临时文件将从磁盘上删除 。使用 mkstemp()创建了该文件。也就是这个不是创建文件，而是返回文件对象供应用调用，这个对象操作 mkstemp()函数创建的文件。

2. tempfile.NamedTemporaryFile([mode='w+b'[, bufsize=-1[, suffix=''[, prefix='tmp'[, dir=None[, delete=True]]]]]])　　（delete、删）

tempfile.NamedTemporaryFile 函数的行为与 tempfile.TemporaryFile 类似，只不过它多了一个 delete 参数，用于指定类文件对象 close 或者被 del 之后，是否也一同删除磁盘上的临时文件（当 delete = True 的时候，行为与 TemporaryFile 一样）。

3. tempfile.SpooledTemporaryFile([max_size=0[, mode='w+b'[, bufsize=-1[, suffix="[, prefix='tmp'[, dir=None]]]]]]) （一样、内存）

此函数操作完全像 TemporaryFile()一样，不同之处在于数据后台在内存中，直到文件的大小超过 max_size，或直到调用该文件的 fileno()方法，此时内容写入到磁盘和运营收益与 TemporaryFile()一样。此外，截断方法不接受大小参数。

4. tempfile.mkstemp([suffix="[, prefix='tmp'[, dir=None[, text=False]]]]) （创建文件）

   可能的最安全的方式创建一个临时文件。

5. tempfile.mkdtemp([suffix="[, prefix='tmp'[, dir=None]]]) （创建目录）

   可能的最安全的方式创建一个临时目录。

glob 模块

说明：glob 模块根据 Unix shell 使用的规则查找所有与指定模式匹配的路径名。（查找、路径）

1. glob.glob(pathname) （匹、路、列）

返回一个匹配 pathname 的路径名列表，列表可能为空，pathname 必须是一个包含路径信息的字符串。pathname 可以是绝对的（比如/usr/src/Python-1.5/Makefile）也可以是相对的（比如../../Tools/*/*.gif），且可以包含 shell 风格的通配符。损坏的符号链接也包含在结果中（和在 shell 中一样）。

2. glob.iglob(pathname) （迭、不）

   返回一个迭代器，它产生和 glob() 相同的值，但不会真正同时地保存它们。

3. 例如，假设有一个只包含以下文件的目录： 1.gif， 2.txt 和 card.gif。glob () 将产生以下结果。请注意路径的前导组件是如何保留的。

```
>>> import glob
>>> glob.glob('./[0-9].*')
['./1.gif', './2.txt']
>>> glob.glob('*.gif')
['1.gif', 'card.gif']
>>> glob.glob('?.gif')
['1.gif']
```

如果目录包含以 . 字符开头的文件，默认它们不会被匹配。例如，假设有一个包含 card.gif 文件和 .card.gif 文件的目录：

```
>>> import glob
>>> glob.glob('*.gif')
['card.gif']
>>> glob.glob('.c*')
['.card.gif']
```

pickle 模块

说明：pickle 模块实现了基本的数据序列和反序列化。通过 pickle 模块的序列化操作我们能够将程序中运行的对象信息保存到文件中去，永久存储；通过 pickle 模块的反序列化

操作，我们能够从文件中创建上一次程序保存的对象。（序、存、反、取）

1. pickle.dump(obj, file, [,protocol]) （存）

将对象 obj 保存到文件 file 中去。protocol 为序列化使用的协议版本，0：ASCII 协议，所序列化的对象使用可打印的 ASCII 码表示；1：老式的二进制协议；2：2.3 版本引入的新二进制协议，较以前的更高效。其中协议 0 和 1 兼容老版本的 python。protocol 默认值为 0。file：对象保存到的类文件对象。file 必须有 write()接口， file 可以是一个以'w'方式打开的文件或者一个 StringIO 对象或者其他任何实现 write()接口的对象。如果 protocol>=1，文件对象需要是二进制模式打开的。

2. pickle.load(file) （取）

从 file 中读取一个字符串，并将它重构为原来的 python 对象。file:类文件对象，有 read()和 readline()接口。

注意：cpickle 提供了和 pickle 一样的接口，但 cpickle 用 C 语言实现，速度比 pickle 快得多。

logging.config 模块

1. logging.config.dictConfig(config) （获取）

从字典中获取日志配置。

2. logging.config.fileConfig(fname, defaults=None, disable_existing_loggers=True) （读取）

从名为 fname 的文件中读取 configparser 格式的日志配置。文件格式描述于配置文件格式。该函数可以从应用中多次调用，允许最终用户可以从各种预先配置好的配置中选择（如果开发者提供了展示配置并装载选择的配置的机制的话）。

3. logging.config.listen(port=DEFAULT_LOGGING_CONFIG_PORT) （监听）

在指定的端口起一个 socket 服务器，以监听新的配置。如果没有指定端口，使用模块默认的 DEFAULT_LOGGING_CONFIG_PORT。日志配置以文件形式发送，可以用 fileConfig()处理。返回 Thread 实例，可以在该线程上调用 start()来启动服务器，可以在合适的时候 join()该线程。调用 stopListening()以停止服务器。

4. logging.config.stopListening()

停止 listen()返回的服务器的监听。典型的在 listen()的返回值上调用 join()之前调用该函数。

Difflib

2. 相关类：

- difflib.SequenceMatcher 类：可以比较任何类型的序列对象的类。
- difflib.Differ 类：主要用来比较文本文件，可以一行一行地比较，产生出一个人能读懂的差异报告文本。每一行有差异时，采用下面的标记来标示出来：
  - ‘-’：表示在第一个序列里存在此行
  - ‘+’：表示在第二个序列里存在此行
  - ‘ ’：表示在第一个和第二个序列都存在
  - ‘？’：表示在两个序列都存在，但有差异，也就是变化的地方
- difflib.HtmlDiff 类：可以用此类来实现比较结果生成 HTML 的格式显示。

2. make_file(fromlines, tolines [, fromdesc][, todesc][, context][, numlines])

   实现参数 fromlines 和 tolines 进行比较，生成一个 HTML 文件并返回。

3. make_table(fromlines, tolines [, fromdesc][, todesc][, context][, numlines])

   比较两个不同文本序列生产一个 HTML 的表格返回。

4. difflib.context_diff(a, b[, fromfile][, tofile][, fromfiledate][, tofiledate][, n][, lineterm])

   两个文件进行比较后，输出简单文本标记方式。每一行进行比较，如果有改变就在前面添加一个感叹号，表示此行已经作了修改。

5. difflib.get_close_matches(word, possibilities[, n][, cutoff])

   从一个列表时查找最相似的字符串，通过列表返回。

6. difflib.ndiff(a, b[, linejunk][, charjunk])

   比较 a 和 b 序列，然后返回一份差别文本。

7. difflib.restore(sequence, which)

   给个 differ 对象，可以从其中获取到原来比较序列 1 或序列 2。

8. difflib.unified_diff(a, b[, fromfile][, tofile][, fromfiledate][, tofiledate][, n][, lineterm])

   比较两个序列，返回压缩的方式来表示差别的文本。

9. difflib.IS_LINE_JUNK(line)

   用来判断是否忽略的行内容。如果 line 是空白行或者只包括#，就会返回 True。

10. difflib.IS_CHARACTER_JUNK(ch)

    用来判断是否忽略的字符。如果 ch 是空格或 Tab 符，就返回 True。

A) SequenceMatch

1. set_seqs(a, b)

   设置要比较的两个序列。

2. set_seq1(a)

   设置第一个要比较的序列。要比较的第二个序列不会更改。

3. set_seq2(b)

   设置要比较的第二个序列。要比较的第一个序列不会更改。

4. find_longest_match(alo, ahi, blo, bhi)

   从[Alo:ahi]和[blo:bhi] b 中找到最长匹配块。

5. get_matching_blocks()

   返回描述匹配子序列的三元组列表。

6. get_opcodes()

   返回描述如何将 a 变成 b 的 5 元组列表。

7. get_grouped_opcodes([n])

   返回一个生成器组。

8. ratio()

   在 [0，1] 范围内返回一个浮点数作为序列的相似性度量。

9. quick_ratio()

   在 ratio()上相对较快地返回上限。

10. real_quick_ratio()

    在 ratio()上很快返回上限。

datetime

1. 对象：

- aware 对象具有关于应用规则和时间调整的充分的信息，例如时区和夏令时信息，来定位相对其他 aware 对象的位置。aware 对象用于表示时间的一个特定的时刻，它是明确无二的。
- naive 对象没有充分的信息来明确地相对其他日期/时间定位它自己。一个 naive 对象表示的是世界协调时（UTC）、本地时间还是其它时区的时间完全取决于程序，就像一个特定的数字表示的是米、英里还是质量一样。

A) timedelta

1. timedelta 对象表示一个时间段，即两个日期 (date) 或时间 (time) 之间的差。两个 date 或 datetime 对象相减时可以返回一个 timedelta 对象。 （时间段、差）

2. timedelta.total_seconds() （总秒数）

   返回 timedatelta 的总秒数。

B) date

3. date 对象表示理想化日历中的日期（年、月和日），即当前的公历可以在两个方向无限扩展。第一年的一月一日叫做第一天，第一年的一月二日叫做第二天，以此类推。

4. classmethod date.today() （当前日期）

   返回当前本地的日期。classmethod 表示该函数为类方法。下同。

5. classmethod date.fromtimestamp(timestamp) （POSIX timestamp）

   返回对应 POSIX timestamp 的本地日期，POSIX timestamp 就是 time.time()返回的那种。

6. classmethod date.fromordinal(ordinal) （公历日期）

   返回对应于公历序数的日期，其中第一年的一月一日为序数 1。

7. date.replace(year, month, day) （新的日期）

   依据关键字参数给出的新值，返回一个新的日期。例如，如果 d == date(2002, 12, 31)，那么 d.replace(day=26) == date(2002, 12, 26)。

8. date.timetuple() （struce_time）

   返回一个 time.struct_time，正如 time.localtime()返回的一样。小时、分钟和秒数为 0，DST 标记为 -1。d.timetuple() 等同于 time.struct_time((d.year, d.month, d.day, 0, 0, 0, d.weekday(), yday, -1))， 其中 yday = d.toordinal() - date(d.year, 1, 1).toordinal() + 1 是当前年份自一月一日第 1 天以来的天数。

9. date.toordinal() （公历日期序数）

   返回公历日期的序数，其中第 1 年的 1 月 1 日为第 1 天。对于任何 date 对象 d，date.fromordinal(d.toordinal()) == d.

10. date.weekday() （一星期、第几天）

返回一星期中的第几天，其中星期一是 0，星期日是 6。例如，date(2002, 12, 4).weekday() == 2，是星期三。另请参阅 isoweekday()。

11. date.isoweekday()　　（一星期、第几天）

返回一星期中的第几天，其中星期一是 1，星期日是 7。 例如，date(2002, 12, 4).isoweekday() == 3，是星期三。另请参阅 weekday()、isocalendar()。

12. date.isoformat()　　（格式、日期）

返回以 ISO 8601 格式'YYYY-MM-DD'表示日期的字符串。例如，date(2002, 12, 4).isoformat() == '2002-12-04'。

13. date.__str__()

对于日期 d，str(d)等同于 d.isoformat()。

14. date.ctime()

返回一个表示日期的字符串，例如 date(2002, 12, 4).ctime() == 'Wed Dec 4 00:00:00 2002'。在原生的 ctime() 函数（time.ctime() 调用的函数，而不是 date.ctime() 调用的函数）遵循 C 标准的平台上 d.ctime() 等同于 time.ctime(time.mktime(d.timetuple())) 。

15. date.strftime(format)

返回一个表示日期的字符串，由显式的格式字符串控制。引用小时、分钟和秒的格式代码的值将为 0。完整的格式指令列表，请参阅 strftime() and strptime() Behavior 一节。

16. date.__format__(format)

等同于 date.strftime()。这使得在使用 str.format()时可以为 date 对象指定格式字符串。参见 strftime() and strptime() Behavior 一节。

C)　Time

1. time.altzone

返回格林威治西部的夏令时地区的偏移秒数。如果该地区在格林威治东部会返回负值（如西欧，包括英国）。对夏令时启用地区才能使用。

2. time.asctime([tupletime])

接受时间元组并返回一个可读的形式为"Tue Dec 11 18:07:14 2008"（2008 年 12 月 11 日 周二 18 时 07 分 14 秒）的 24 个字符的字符串。

3. time.clock( )　　（CPU 时间、耗时）

用以浮点数计算的秒数返回当前的 CPU 时间。用来衡量不同程序的耗时，比 time.time()更有用。

4. time.ctime([secs])

作用相当于 asctime(localtime(secs))，未给参数相当于 asctime()

5. time.gmtime([secs])　　（1970）

接收时间辍（1970 纪元后经过的浮点秒数）并返回格林威治天文时间下的时间元组 t。注：t.tm_isdst 始终为 0

6. time.localtime([secs])　　（当地、时间元组）

接收时间辍（1970 纪元后经过的浮点秒数）并返回当地时间下的时间元组 t（t.tm_isdst 可取 0 或 1，取决于当地当时是不是夏令时）。

7. time.mktime(tupletime)

接受时间元组并返回时间辍（1970 纪元后经过的浮点秒数）。

8. time.sleep(secs)

   推迟调用线程的运行，secs 指秒数。

9. time.strftime(fmt[,tupletime])

   接收以时间元组，并返回以可读字符串表示的当地时间，格式由 fmt 决定。

10. time.strptime(str,fmt='%a %b %d %H:%M:%S %Y')

    根据 fmt 的格式把一个时间字符串解析为时间元组。

11. time.time( )

    返回当前时间的时间戳（1970 纪元后经过的浮点秒数）。

12. time.tzset()

    根据环境变量 TZ 重新初始化时间相关设置。

D) datetime

17. datetime 对象包含 date 对象和 time 对象的所有信息。
18. classmethod datetime.today()　　（本地当前）

    返回本地当前的时间，tzinfo 为 None。

19. classmethod datetime.now([tz])　　（本地当前）

    返回本地当前的日期和时间。

20. classmethod datetime.utcnow()

    返回当前 UTC 日期和时间。

21. classmethod datetime.fromtimestamp(timestamp[, tz])

    返回与 POSIX timestamp 对应的本地日期和时间，POSIX timestamp 例如 time.time()返回的值。

22. datetime.date()　　（相同）

    返回具有相同年、月、日的 date 对象。

23. datetime.time()　　（相同）

    返回具有相同时、分、秒和微秒的 time 对象。tzinfo 为 None。

24. datetime.timetz()　　（相同）

    返回具有相同时、分、秒、微秒和时区属性的 time 对象。

25. datetime.weekday()　　（整数、星期几）

    返回一个整数，表示星期几；其中星期一为 0，星期日为 6。

26. datetime.isoweekday()　　（整数、星期几）

    返回一个整数，表示星期几；其中星期一为 1，星期日为 7。

通用日历相关函数

1. calendar.calendar(year,w=2,l=1,c=6)

   返回一个多行字符串格式的 year 年年历，3 个月一行，间隔距离为 c。 每日宽度间隔为 w 字符。每行长度为 21* W+18+2* C。l 是每星期行数。

2. calendar.firstweekday( ) （每周起始）

   返回当前每周起始日期的设置。默认情况下，首次载入 caendar 模块时返回 0，即星期一。

3. calendar.isleap(year) （闰年）

   是闰年返回 True，否则为 false。

4. calendar.leapdays(y1,y2) （闰年总数）

   返回在 Y1，Y2 两年之间的闰年总数。

5. calendar.month(year,month,w=2,l=1)

   返回一个多行字符串格式的 year 年 month 月日历，两行标题，一周一行。每日宽度间隔为 w 字符。每行的长度为 7* w+6。l 是每星期的行数。

6. calendar.monthcalendar(year,month)

   返回一个整数的单层嵌套列表。每个子列表装载代表一个星期的整数。Year 年 month 月外的日期都设为 0;范围内的日子都由该月第几日表示，从 1 开始。

7. calendar.monthrange(year,month)

   返回两个整数。第一个是该月的星期几的日期码，第二个是该月的日期码。日从 0（星期一）到 6（星期日）;月从 1 到 12。

8. calendar.prcal(year,w=2,l=1,c=6)

   相当于 print calendar.calendar(year,w,l,c).

9. calendar.prmonth(year,month,w=2,l=1)

   相当于 print calendar.calendar（year，w，l，c）。

10. calendar.setfirstweekday(weekday)

    设置每周的起始日期码。0（星期一）到 6（星期日）。

11. calendar.timegm(tupletime)

    和 time.gmtime 相反：接受一个时间元组形式，返回该时刻的时间辍（1970 纪元后经过的浮点秒数）。

12. calendar.weekday(year,month,day)

    返回给定日期的日期码。0（星期一）到 6（星期日）。月份为 1（一月） 到 12（12 月）。

13.