Python 学习笔记

ecom-sf-crm-qa 郭月月

目录

第一章 Python 快速入门

1.1 运行 python

1、在命令行输入 python 运行 python 解释器。

2、双击.py 文件

3、建立 bat 文件

文件中写入，如：python -i *.py。双击后自动进入 python 解释器。

4、在解释器中通过 execfile(“*.py”)函数运行文件程序。

5、Ctrl+Z 在交互模式下退出解释器。

6、使用函数退出程序：

```
import sys
sys.exit()
```

7、引发异常退出程序

```
raise SystemExit
```

1.2 变量和表达式

变量名只是各种数据和对象的引用！同一变量名可以代表不同形式的值。

可在同一行写多个语句，语句之间要用“;”分割。

使用缩进块来表示程序逻辑。同一层次的语句必须有相同的缩进空白。

可以使用格式字符串，如：%d,%f,%s，来决定特定类型的数据格式。

1.3 条件语句

某个子句不需要任何操作，则使用 pass 语句。

可以使用 or，and 和 not 关键字建立条件表达式。

通过 elif 可以检验多重条件。

1.4 文件输入/输出

open()

readline()

>>

write()

1.5 字符串

单引号、双引号、三引号（’ 、’ ’ 或”” ”）。

单引号和双引号只能创建单行字符。

切片运算符 s[i:j]。

+可以连接字符串。

str()函数，repr()函数可将其他类型数据转换成字符串。向后的引号（`）是 repr()函数的快捷版。

Python 字符是一个字符的字符串！

1.6 列表和元组

len()

append()

insert()

“+”运算符链接列表

readlines()

[float (s) for s in a]

max()、min()

sys.argv

某些情况下，即使没有圆括号，python 仍然可以根据上下文认出这是一个元组，但不建议这样做。

定义只有单个元素的元组必须在最后加上一个逗号，如 a=(1,)

元组支持大多数列表的操作，但是一个关键的不同是不能在元组创建之后修改它的内容。

1.7 循环

range()和 xrange()。

xrange()函数只有在需要时才临时通过计算提供值，可以大大节省内存。

1.8 字典

只要是不可修改对象，都可以作为字典的关键字。

```
has_key():
If a.has_key(“username”):
    username = a[“username”]
else:
    username = “unknown user”
```

上面的代码等价于下面的代码：

```
username = a.get(“username”, “unknown user”)
```

```
keys()
del
```

1.9 函数

def

函数返回元祖、列表

默认参数、关键字参数

要想在一个函数内部改变一个全局变量的值，需在函数中使用 global 语句声明该变量。

1.10 类

类中的每个方法的第一个参数总是引用类实例对象，习惯上使用 self 这个名字代表该参数。

特殊方法

1.11 异常

try:、except

raise

1.12 模块

import

import…as…

导入名称空间：from…import *

内建函数 dir()

第二章 语法及代码约定

2.1 行结构/缩进

可以使用续行符“\”来将较长的行分成多行。但是，当定义一个三引号字符串、列表、元组或者字典的时候，不需要续行符来分隔语句。

缩进的空格（制表符）书目可以是任意的，但是整个块中的缩进必须一致。

单行注释使用（#）即可，对于多行注释，可以采用如下技巧：

```
"""
这是多行注释，用三个双引号
这是多行注释，用三个双引号
这是多行注释，用三个双引号
"""
```

2.2 标识符及保留字

以下划线开始或结束的标识符通常有特殊的意义。

保留字：

and	elif	global	or
assert	else	if	pass
break	except	import	print
class	exec	in	raise
continue	finally	is	return
def	for	lambda	try
del	from	not	while

2.3 数值/文字

Python 内建有四种数值类型：整数、长整数、浮点数、复数。

特殊字符：

标准特殊字符	
字符	描述
\	续行符
\\	反斜杠
\'	单引号
\"	双引号
\a	Bell(音箱发出叱的一声)
\b	退格符
\e	Escape
\0	Null(空值)
\n	换行符，等价于\x0a和\cJ
\v	垂直制表符，等价于\x0b和\cK
\t	水平制表符，等价于\x09和\cI
\r	回车符，等价于\x0d和\cM
\f	换页符，等价于\x0c和\cL
\OOO	八进制值(000-377)
\xhh	十六进制值(x00-xff)
\un	Unicode字符值，n是四个十六进制数字表示的Unicode字符

2.4 运算符、分隔符及特殊符号

运算符：

```
+       -       *       **      //      /       %       <<      >>
&       |       ^
+=      -=      *=      **=     //=     /=      %=      <<=     >>=
&=      |=      ^=
~       <       >       <=      >=      ==      !=      <>
```

2.5 文档字符串

如果一个模块、类、函数体的第一个语句是未命名的一个字符串，则该字符串就自动成为该对象的文档字符串(DocStrings),即这个对象的__doc__属性(可写的!)。

第三章 类型和对象

3.1 术语

一切数据都是对象，每个对象都有三个基本属性，即标识、类型和值。

一个对象被创建之后，它的标识和类型就不能再被改变。

某些对象的值是可变的，成为可变对象；反之，成为不可变对象。

包含其他对象引用的对象成为容器。

方法、属性、“.”操作符

3.2 对象的标识与类型

内建函数 id()返回一个对象的标识，通常是对象在内存中的位置。

内建函数 type()返回一个对象的类型。对象的类型也是对象，并且这个对象具有唯一性！

==表示对象的值。

is 运算符用来比较两个对象的标识。isinstance(s,C)用于测试 s 是否是 C 或 C 的子类的实例。

3.3 引用计数与垃圾收集

一切对象都是引用计数的。

3.4 引用与副本

当运行语句 a=b 时，就创建类对象 b 的一个新引用 a。

对于不可变对象（数字或字符串等），改变对象的一个引用就会创建一个新的对象。

对于可变对象（列表或字典等），改变对象的一个引用就等于改变了该对象的所有的引用。

对象副本：浅复制、深复制。

标准库中的 copy 模块里有一个 deepcopy()函数可以实现深复制（创建一个新对象，并递归复制所有子对象）。

3.5 内建类型

1、None 类型

2、数值类型：不可变类型，包括整型、长整形、浮点类型、复数类型

3、序列类型：有序的

所有序列都支持的方法：

项目	描述
s [i]	返回序列s的元素i
s [i :j]	返回一个切片
len(s)	序列中元素的个数
min(s)	s 中的最小值
max(s)	s 中的最大值

可变序列适用的操作：

项目	描述
s [i] = v	给某个元素赋新值
s [i:j] = t	用 序列 t 中的所有元素替换掉 s 序列中的索引从 i 至 j 的元素。
del s[i]	删除序列 s 中索引为 i 的元素。
del s [i :j]	删除序列 s 中的索引从 i 至 j 的元素

列表的方法：

方法	描述
list(s)	把序列s转换为一个列表
s.append(x)	把一个元素添加到列表的结尾,相当于` s[len(s):] = [x]`
s.extend(t)	将链表 t 的所有元素添加到 s 的末尾来扩充列表 s，相当于 `s[len(s):] = t`
s.count(x)	返回值 x 在列表 s 中出现的次数
s.index(x)	返回列表s中第一个值为 x 的元素的索引值
s.insert(i,x)	在 s[i] 前插入一个元素 x
s.pop([i])	返回 s[i] 的值并将 s[i] 元素从列表中删除。如果 i 被省略，` s.pop()` 就对最后一个元素进行操作。
s.remove(x)	删除列表中值为 x 的第一个元素
s.reverse()	翻转 s 中的全部元素
s.sort([cmpfunc])	对列表 s 中的元素进行排序，cmpfunc 是一个可选的比较函数

4、字符串类型：标准字符串、unicode 字符串

字符串方法：

方法	描述
s.capitalize()	第一个字母变大写
s.count(sub [,start [,end]])	子串sub出现的次数
s.encode([encoding [,errors]])	改变字符串的编码
s.startswith(prefix [,start [,end]])	检查字符串的开头是否为prefix
s.endswith(suffix [,start [,end]])	检查字符串的结尾是否是suffix
s.expandtabs([tabsize])	将制表符转换为一定数量的空格
s.find(sub [,start [,end]])	返回子串 sub 首次出现的位置或者 -1
s.rfind(sub [,start [,end]])	返回子串 sub 末次出现的位置或者 -1
s.index(sub [,start [,end]])	返回子串 sub 首次出现的位置或者引起
异常	
s.rindex(sub [,start [,end]])	返回子串 sub 末次出现的位置或者引发
异常	
s.isalnum()	字符是否都为字母或数字
s.isalpha()	字符是否都为字母
s.isdigit()	字符是否都为数字
s.islower()	字符是否都为小写
s.isspace()	字符是否都为空白
s.istitle()	检查字符是否为标题格式(每个单词的第一
个字母大写)	
s.isupper()	字符是否都为大写
s.join(t)	用 s 连接 t 中的所有字符串
s.center(width)	在长度为 width 范围内将字符串置中
s.ljust(width)	在宽度为 width 内左对齐
s.rjust(width)	在宽度为 width 内右对齐
s.lower()	s 中所有字符小写
s.upper()	s 中所有字符大写
s.replace(old , new [,maxreplace])	将子串 old 替换为 new
s.lstrip()	删去字符串s开头的空白
s.rstrip()	删去字符串s末尾的空白
s.strip()	删去字符串s开头和末尾的空白
s.split([sep [,maxsplit]])	将字符串 s 分割成一个字符串列表，其中
sep 为分隔符，maxsplit是最大分割次数	
s.splitlines([keepends])	将字符串按行分割为一个字符串列表，若
keepends为1，则保留换行符'\n'	
s.swapcase()	串内字符大写变小写，小写变大写，没有
大小写的不变	
s.title()	s 转换为标题格式(每个单词的第一个字母
大写)	
s.translate(table [,deletechars])	使用字符转换表转换一个字符串

5、XRangeType 类型：xrange()、tolist()

6、缓冲区类型

7、映射类型：无序的、可变类型

映射对象的方法和操作：

项目	描述
len(m)	返回m中的条目个数
m[k]	返回关键字k索引的元素
m[k] = x	设置关键字k索引的值为x
del m[k]	删除一个元素
m.clear()	删除所有元素
m.copy()	返回m的一个浅拷贝
m.has_key(k)	若 m 中存在 key k 返回True,否则返回False
m.items()	返回包含所有关键字和对应值(key ,value)的列表
m.keys()	返回由所有关键字组成的列表
m.update(b)	将字典b中的所有对象加入m
m.values()	返回一个包含m中所有对应值的列表
m.get(k[,v])	返回m[k]，若m[k]不存在时，返回 v
m.setdefault(k[,v])	返回m[k]，若m[k]不存在时，返回 v 并设置m[k] = v
m.popitem()	从 m 中随机删除一个元素，并以元组的形式返回其关键字和值

8、可调用类型

用户自定义函数 f 有如下属性

属性	描述
f.__module__	函数定义所在的模块名
f.__doc__ 或 f.func_doc	文档字符串
f.__name__ 或 f.func_name	函数名 (从2.4版开始该属性由只读变为可写)
f.__dict__ 或 f.func_dict	支持任意函数属性的函数名字空间
f.func_code	(函数编译后产生的)字节码
f.func_defaults	包含所有默认参数的元组
f.func_globals	函数所在模块的全局名称空间的字典(只读)
f.func_closure	None or a tuple of cells that contain bindings for the function's free variables. Read-only

用户自定义函数对象支持任意属性。

非绑定方法、绑定方法

内建方法的可用属性：

属性	方法
b.__doc__	文档字符串
b.__name__	函数/方法名
b.__self__	方法所绑定的实例(未绑定时，返回None)
b.__members__	方法的属性名(返回列表)

9、模块类型：

模块对象拥有一下属性：

属性	描述
m.__dict__	保存模块名字空间的字典
m.__doc__	模块的文档字符串
m.__name__	模块名字
m.__file__	模块的文件名
m.__path__	当一个模块通过一个包被引用时，__path__是包的名字

10、类类型：

class 对象定义的属性：

属性	描述
c.__dict__	类 c 的名字空间
c.__doc__	类 c 的文档字符串
c.__name__	类 c 的名字
c.__module__	类 c 的定义所在的模块
c.__bases__	类 c 的所有父类（这是一个元组）

11、类实例类型：

类实例有以下属性：

属性	描述
x.__dict__	实例 x 的名字空间
x.__class__	实例 x 所属的类

12、文件类型：open()

13、内部类型：调试对象、代码对象、frame 对象、切片对象、省略对象

3.6 特殊方法

内建的数据类型都有一些特殊方法，特殊方法的名字总是以两个下划线开头和结尾。

3.7 性能及内存占用

所有的 python 对象至少拥有一个整型引用计数、一个类型定义描述符及真实的数据表示这三部分。

第四章 运算符与表达式

4.1 数值操作

常规除 / 与地板除 //。

地板除会在任何时候将小数部分舍为 0，如 5.0 // 3.0 =1.0

取模操作%返回 x/y 的余数，如 7%4=3。对于浮点数，取模操作返回的是 x - int(x/y)*y。对于复数，取模操作返回 x - int((x/y).real)*y。 （负数参与取模操作的情况详见 csdn blog）

下列内建函数支持所有的数值类型。其中，divmod()函数返回一个包含商和余数的元组。round()函数总是返回一个四舍五入后的浮点数，并且 Python 的四舍五入规则不是银行家四舍五入规则，采用的是简单四舍五入。

函数	描述
abs(x)	绝对值
divmod(x ,y)	返回 (int(x / y), x % y)
pow(x ,y [,modulo])	返回 (x ** y) x % modulo
round(x ,[n])	四舍五入，n为小数点位数

Python 的比较运算可以连结在一起，如 x<y>z 表示 x<y and y >z。

4.2 序列运算

s*n 或 n*s，返回 s 的 n 次浅拷贝，n 为整数。

索引操作符 s[n]返回序列中的第 n 个对象（s[0]是第一个），如果 n 是负数，在求值之前，就先执行一次 n+=len(s)。

切片操作符 s[i:j]返回一个子序列。i 和 j 如果被省略，则默认值为序列的开始和结束，如果 i 和 j 超过了上界或下界，则自动取上界或下界。

序列可以使用<, >, <=, >=, == 和!=来进行比较。当比较两个序列的时候，将按顺序依次比较序列的元素，直到找到两个不同元素或者两个序列都没有多余元素为止。

4.3 字典的操作

字典的 key 可以是任意不可变对象，如字符串、数字、和元组。

```
d={}
d[1,2,3] = “foo”   #等价于 d[(1,2,3)] = “foo”
```

4.4 增量幅值语句

需要注意的是，增量幅值语句并不对对象进行原地修改，因此也不会改变对象的性质。x += y 语句创建了一个值为 x+y 的新对象，并将这个对象赋给 x。

4.5 属性(.)操作符

.

4.6 类型转换

下列内建函数提供了显式的类型转换操作：

函数	描述
int(x [,base])	将x转换为一个整数
long(x [,base])	将x转换为一个长整数
float(x)	将x转换到一个浮点数
complex(real [,imag])	创建一个复数
str(x)	将对象 x 转换为字符串
repr(x)	将对象 x 转换为表达式字符串
eval(str)	用来计算在字符串中的有效Python表达式,并返回一个对象
tuple(s)	将序列 s 转换为一个元组
list(s)	将序列 s 转换为一个列表
chr(x)	将一个整数转换为一个字符
unichr(x)	将一个整数转换为Unicode字符
ord(x)	将一个字符转换为它的整数值
hex(x)	将一个整数转换为一个十六进制字符串
oct(x)	将一个整数转换为一个八进制字符串

其中 repr(x)函数也可以写为 x。

str()函数和 repr()函数返回的结果是不同的。repr()函数取得对象的表达式字符串表示，通常可以使用 eval()函数来重新得到这个对象。而 str()产生一个对象的简洁格式表示。

4.7 Unicode 字符串

内建函数 unicode(s[,encoding[,errors]])可以把一个标准字符串转换为一个 Unicode 字符串。字符串方法 u.encode([encoding[,errors]])可以把一个 Unicode 字符串转换为一个标准字符串。

其中，encoding 参数是一个用于指定编码规则的特定字符串，默认为’ascii’。

值	描述
'ascii'	7-bit ASCII
'latin-1' or 'iso-8859-1'	ISO 8859-1 Latin-1
'utf-8'	8-位可变长度编码
'utf-16'	16-位可变长度编码(可能是 little endian或 big endian)

'utf-16-le'	UTF-16, little-endian 编码
'utf-16-be'	UTF-16, big-endian 编码
'unicode-escape'	与Unicode文字 u"string" 相同
'raw-unicode-escape'	与原始 Unicode文字 ur"string"相同

errors 参数是一个决定当有字符无法转换时如何处理错误的特定字符串，默认为’strict’。

值	描述
'strict'	编码错误时引起一个UnicodeError异常
'ignore'	忽略不可转换字符
'replace'	将不可转换字符用U+FFFD替代(Unicode中的U+FFFD 是标准字符串中的'?')

4.8 布尔表达式

and, or, not

任何非零的数字或非空列表，元组，字典，都返回 True。零，None，以及空列表，元组，字典返回 False。布尔表达式从左至右进行计算，具有短路行为。

4.9 对象的比较和身份

相等运算符 x==y 检验 x 和 y 的值是否相等。

身份运算符 x is y 和 x is not y 检验两个对象在内存中是否指向同一个对象。

4.10 运算优先级

除了乘方(**)外的所有运算符都是从左至右进行运算的。

第五章 控制流

5.1 条件语句

If, else, elif, pass

5.2 循环

while, for, break, continue.。

Python 不提供 goto 语句。

可以在 while 和 for 循环结构中使用 else 语句。当循环正常完成后才运行 else 语句（for 或 while 循环），或者在循环条件不成立时立即运行（仅 while 循环），或者迭代序列为空时立即指向（仅 for 循环）。使用 break 语句跳出的话，else 语句将被忽略。

5.3 异常

raise 语句的格式是 raise exception[, value]。其中 exception 是异常类型，value 是特定描述。如 raise RuntimError, ‘Unrecoverable Error’

try, except, finally 处理异常，pass 语句可以用来忽略异常

```
try:
    f = open(‘foo’)
except IOError, e:    #第二个参数可省略，也可以同时处理多个异常，如 except (IOError, TypeError)
    print “Unable to open ‘foo’:”, e
except NameError:
    pass
except:
    print ‘An error occurred’    #省略异常名和值可以捕获所有异常
```

try 语句也支持 else 从句，必须放在最后一个 except 从句后，这块代码只

在 try 块中的语句没有引发异常时运行。

finally 语句并不用于捕获异常，如果没有引起异常，finally 块中的语句将在 try 块中的语句执行完毕后执行；~~如果有异常，控制将先传递到 finally 块中的第一条语句，在这块语句执行完毕后，异常再次被自动引发，交由异常处理语句处理。finally 语句和 except 语句不能在同一个 try 语句中出现。~~

经验证，上段似乎不对，finally 和 except 是可以同时出现的，finally 语句应该是最后不管有无异常均要执行的，如果有 else 从句并且没发生异常，else 从句也将先于 finally 语句执行。

内建异常类型：

异常	描述
Exception	所有内建异常
SystemExit	由sys.exit()产生
StandardError	除SystemExit外所有内建异常
ArithmeticError	所有运算异常
FloatingPointError	浮点数运算异常
OverflowError	数值溢出
ZeroDivisionError	被零除
AssertionError	assert语句引起的异常
AttributeError	属性名称不可用时引起
EnvironmentError	Python外部错误
IOError	I/O 或与文件有关的错误(输入/输出错误)
OSError	操作系统错误
WindowsError	Windows错误
EOFError	当到达一个文件的末尾时引起
ImportError	import语句失败
KeyboardInterrupt	键盘中断(通常是 Ctrl+C)
LookupError	索引或关键字错误
IndexError	超出序列的范围
KeyError	不存在的字典关键字
MemoryError	内存不足
NameError	寻找局部或全局变量时失败
UnboundLocalError	未绑定变量
RuntimeError	一般运行时错误
NotImplementedError	不可实现的特征
SyntaxError	语法错误
TabError	不一致的制表符使用 (由 -tt 选项产生)
IndentationError	缩进错误
SystemError	解释器致命错误
TypeError	给一个操作传递了一个不适当的类型
ValueError	值错误(不合适或丢失)
UnicodeError	Unicode编码错误

5.4 定义新的异常

要定义一个新的异常，就创建一个父类为 exceptions.Exception 的新类。

5.5 断言和__debug__

assert test[, data]用于程序调试。实际上，语句在执行是会被翻译为下面的代码：

```
if __debug__:
    if not (test):
        raise AssertionError, data
```

__debug__是一个内建的只读值，除非解释器运行在最佳化模式（使用-O 或-OO 选项），否则它的值总是 True。

第六章 函数与函数编程

6.1 函数

“一切数据是对象，一切命名是引用”。函数默认参数的值在函数定义的时候就被决定，并且不会改变。因此，参数默认值是不可变对象和可变对象将会产生不同的结果！

如果最后一个参数名前有星号(*)，函数就可以接受可变数量的参数，这些不定数量的参数将作为一个元组传递给函数。

关键字参数：调用时显式地给每个形参绑定一个值，如 foo(x=3, y=22, w=’hello’)。传统的参数和关键字参数可以混合使用，前提是必须先给出传统参数，如 foo(‘hello’, 3, x=1)。

如果最后一个形参有**（双星号）前缀，则所有正常形参之外的其它关键字参数都将被放置在一个字典中传递给函数。*参数和**参数可以同时使用，这时，**参数必须位于参数表的最后。

函数可以有任意的属性，但是内建函数和类的方法是没有的。

6.2 参数传递和返回值

调用一个函数是，其参数是按引用传递的。因此，如果函数的实参是一个可变对象，则函数内对该对象的修改将会影响到函数之外。

6.3 作用域规则

globe 语句可以在函数内部声明使用一个或多个全局变量。

Python 允许嵌套的函数定义！

6.4 递归

函数 sys.getrecursionlimit()和 sys.setrecursionlimit()分别返回和设置允许的最大递归调用次数。默认值为 1000。

6.5 apply()函数

当要调用的函数参数已经存在与一个元组或字典中时，可以使用apply(func[, args[, kwargs]])函数间接地调用该函数。其中，args 是一个元组，kwargs 是一个字典。例如：

```
foo(3, “x”, name=’dave’, id=12345)
apply(foo, (3,”x”), {‘name’:’dave’, ‘id’:12345}) #与上面的效果一样

a=(3,”x”)
b= {‘name’:’dave’, ‘id’:12345}
foo(*a, **b)    #还可以用更简单的方式来调用!
```

6.6 lambda 操作符

lambda 语句用来创建一个匿名函数。——lambda 已经是过时的语句，即将被废除。

6.7 map(), zip(), reduce(), 和 filter()

6.8 列表内涵

Toggle line numbers

```
1 import math
2 a = [-3,5,2,-10,7,8]
3 b = 'abc'
4 c = [2*s for s in a]           # c = [-6,10,4,-20,14,16]
5 d = [s for s in a if s >= 0]   # d = [5,2,7,8]
6 e = [(x,y) for x in a          # e = [(5,'a'),(5,'b'),(5,'c'),
7            for y in b          #      (2,'a'),(2,'b'),(2,'c'),
8            if x > 0]           #      (7,'a'),(7,'b'),(7,'c'),
9                                #      (8,'a'),(8,'b'),(8,'c')]
10 f = [(1,2), (3,4), (5,6)]
11 g = [math.sqrt(x*x+y*y)       # f = [2.23606, 5.0, 7.81024]
12      for x,y in f]
```

```
13 h = reduce(lambda x,y: x+y,    # 平方根的和
14            [math.sqrt(x*x+y*y)
15             for x,y in f])
```

在一个列表内涵中定义的变量是与列表内涵本身具有同样的作用域，在列表内涵计算完成后会继续存在。

6.9 eval(), exec, execfile(),和 compile()

第七章 类和面向对象编程

7.1 class 语句

7.2 访问类属性

静态方法：可以直接被类或类实例调用。

```
class AClass(object):
    @staticmethod        #静态方法修饰符，表示下面的方法是一个静态方法
    def astatic(  ): print 'a static method'
anInstance = AClass(  )
AClass.astatic(  )                    # prints: a static method
anInstance.astatic(  )                # prints: a static method
```

类方法：可以通过类或类实例来调用的方法，该方法的第一个参数总是定义该方法的类对象（不是类实例对象！）。按照惯例，类方法的第一个形参被命名为 cls

```
class ABase(object):
    @classmethod         #类方法修饰符
    def aclassmet(cls): print 'a class method for', cls.__name__
class ADeriv(ABase): pass
bInstance = ABase(  )
dInstance = ADeriv(  )
ABase.aclassmet(  )               # prints: a class method for ABase
bInstance.aclassmet(  )           # prints: a class method for ABase
ADeriv.aclassmet(  )              # prints: a class method for ADeriv
dInstance.aclassmet(  )           # prints: a class method for ADeriv
```

静态方法 VS 类方法——类方法的隐含调用参数是类，类实例方法的隐含调用参数是类的实例，静态方法没有隐含调用参数。

7.3 类实例

7.4 引用计数和实例销毁

所有的实例都是引用计数的。

最好不好依赖__del__()函数来执行清除和关闭操作，可以定义一个 close()

方法来显式地调用。

如果一个实例拥有__del__()方法，则它永远不会被垃圾收集器回收！

使用 del 语句来删除对象的引用，如果这导致该对象引用计数变为零，就调用__del__()方法——del 语句并不直接调用__del__()方法。

7.5 继承

Python 支持多继承。

子类实例被创建时，基类的__init__()方法并不会被自动调用，也就是说子类必须显式地调用父类的__init__()方法来初始化。__del__()方法也时类似的。

7.6 多态

当使用 obj.methon()来访问一个方法时，方法的搜索顺序为：实例的 dict 属性，实例的类定义，基类。第一个被找到的方法被执行。

7.7 数据隐藏

默认情况下，所有属性都是公开的！想要添加“私有”的属性，只需在类中将需要隐藏的属性名字以两个下划线开头，如__Foo，这样，系统会自动生成一个新的名字_Classname__Foo。——实际上，只是换了一个名字，仍然不是真正私有的！

7.8 操作符重载

__add__(self, other) __sub__(self, other) __neg__(self) __radd__(self, other) __rsub__(self, other) __coerce__(self, other)（用于混合类型运算时，进行类型转换）

7.9 类，类型，和成员检测

isinstance(obj,cname)：测试 obj 对象是否是 cname 的实例。

issubclass(A,B)：测试 A 是否是 B 的子类。

isinstance()函数也可用于检查任意类型。

第八章 模块和包

8.1模块

导入一个模块的步骤：

1. 创建一个可以访问到模块中定义的函数和变量的名字空间；
2. 在新建的名字空间里执行源代码文件；（只在第一次导入时执行）
3. 创建一个名为源代码文件的对象，该对象引用模块的名字空间。

使用 from 语句可以将模块中的对象直接导入到当前的名字空间。

```
from socket import gethostname
from socket import *  #导入除下划线开头的其它所有对象到当前名字空间
```

不过，如果一个模块中定义有列表__all__,则 from module import * 语句只能导入__all__列表中存在的对象。

每个模块都有__name__属性，它是一个内容为模块名字的字符串。最顶层的模块名称是__main__。命令行或交互模式下程序都运行在__main__模块内部。

8.2 模块搜索路径

导入模块时，解释器会搜索 sys.path 列表。

8.2模块导入和汇编

可以被 import 语句导入的模块有以下四类：

1、使用 python 写的程序（.py 文件）

2、C 或 C++扩展（已编译为共享库或 DLL 文件）

3、包（内含多个模块）

4、内建模块（使用 C 编写并已链接到 python 解释器内）

8.4 重新导入模块

内建函数 reload()可以重新导入并运行更新后的模块代码。

8.5 包

创建一个名字为包名字的文件夹并在该文件夹下创建一个__inti__.py 文件就定义了一个包。

无论一个包的哪个部分被导入，在文件__inti__.py 中的代码都会被运行。这个文件的内容允许为空，不过通常情况下它用来存放包的初始化代码。

Python 导入一个包时，它定义了一个包含目录列表的特殊变量__path__，用于查找包的模块（类似于 sys.path 的作用）。可以在__init__.py 文件中访问__path__变量，其初始值只有一个元素，即包的目录。因此，可以做到一个模块属于一个包，却不位于这个包所在的目录或子目录下。

第九章 输入输出

9.1 读取参数及环境变量

sys.argv 这个列表中存放着命令行参数，其中第一个元素是程序的名字。

通过访问 os.environ 字典可以访问（操作系统的）环境变量。如，path = os.environ[“PATH”]。

改变环境变量可以直接设定 os.environ 变量或者使用 os.putenv()函数，如 os.putenv(“FOO”,” BAR”)。

9.2 文件

内建函数 open(name[, mode[, buffer]]打开或创建文件。mode 是打开模式，可选，默认为读方式。buffer 是缓冲区，可选。

mode 有如下几种：

‘r’:读模式

‘w’:写模式

‘a’:追加模式

‘b’:二进制模式

‘t’:文本模式（默认值）

‘+’:更新已有磁盘文件（读写模式，读操作之前只要刷新输出缓冲区就不会有问题）

‘U’:通用换行模式

文件方法如下：

方法	描述
f.read([n])	读取至多 n 字节
f.readline([n])	读取一行中的前 n 字符。如果 n 被省略，就读取整行
f.readlines()	读取所有的行并返回一个包含所有行的列表
f.xreadlines()	返回一个迭代器，每次迭代返回文件的一个新行
f.write(s)	将字符串 s 写入文件
f.writelines(l)	将列表 l 中的所有字符串写入文件
f.close()	结束文件
f.tell()	返回当前的文件指针
f.seek(offset [, where])	定位到一个新的文件位置
f.isatty()	如果 f 是一个交互式终端则返回 1
f.flush()	刷新输出缓冲区
f.truncate([size])	如果文件长于 size 就截短它至 size 大小
f.fileno()	返回一个整型的文件描述符
f.readinto(buffer ,nbytes)	读取 n 字节数据至一个 buffer 对象。

9.3 标准输入，标准输出和标准错误

sys.stdin, sys.stdout, sys.stderr 对象，可以被替换。

读取用户输入的方法：

sys.stdin.read([n]) 或者 raw_input(prompt)

键盘中断会引发 KeyboardInterrupt 异常。

9.4 print 语句

print 语句将一个或多个对象的字符串表示(str()函数)输出到 stdout 对象。多个对象时可以用逗号分隔，输出时将用一个空格连接起来，并在最后添加一个换行符。如不需要添加换行符，可以在语句最后添加一个逗号。

```
f = open(“output”,”w”)
print >>f, “Helloworld”     #将输出内容重定向到 f 文件对象中。
```

9.5 对象持久化

将一个对象内容保存到一个文件中，当再次需要该对象时通过读取这个文件重新生成该对象是很有用的。Python 提供的 pickle 和 shelve 模块可实现该功能。

pickle 模块的 dump 方法可以方便地把一个对象保存到一个文件，之后可以用 load 方法重新得到该对象。

shelve 模块也类似，不过它将对象数据保存在一个字典格式的文本数据库中。

9.6 Unicode I/O

第十章 执行环境

10.1 解释器选项及运行环境

-i　程序结束时进入交互模式

-t　若发现制表符和空格混合缩进则给出警告信息

-tt　若发现制表符和空格混合缩进则引发 TabError 异常

-U　Unicode 模式，所有字符串都被转换为 unicode

-c cmd　运行字符串 cmd

10.2 交互模式

进入交互模式时，首先显式版本信息，如果 PYTHONSTARTUP 环境变量设置来有效的脚本，python 就会运行这个脚本，最后显式交互提示符>>>。

10.3 运行 Python 程序

10.4 Site 配置文件

10.5 启用 Future 特性

10.6 程序终止

当两个对象相互引用时，在程序结束时，这两个对象将无法被销毁，造成内存泄漏——尽管 python 的垃圾回收机制能在运行时删除这些对象，但在程序结束时该机制不会被自动调用！