Python Tutorial

Release 2.7

March Liu<march.liu@gmail.com>

CONTENTS

CHAPTER

ONE

WHETTING YOUR APPETITE 开胃菜

•

•

•

CHAPTER

TWO

USING THE PYTHON INTERPRETER 使用 PYTHON 解释器

2.1 Invoking the Interpreter 调用解释器

/usr/local/bin/python
/usr/local/bin

/usr/local/bin/python /usr/local/bin

python

/usr/local/python

/usr/local/python
C:\Python27

C:\Python27

set path=%path%;C:\\python27

Control-D Control-Z

quit()

Ctrl+D Ctrl+Z
quit()

^P

^p

python -c command [arg] ...
-*c*

python -c command [arg] ...

python -m module
[arg] ...

python -m module [arg]...

python file python <file
input() raw_input()

python file python <file input()
raw_input()

-*i*

-*i*

2.1.1 Argument Passing 参数传递

sys.argv
sys.argv[0]
'-' sys.argv[0] '-' -*c*
sys.argv[0] '-c' -*m* sys.argv[0]

`-c` *-m*
`sys.argv`

`sys.argv`
`sys.argv[0]` `'-'`
`sys.argv[0]` `'-'` *-c* `sys.argv[0]` `'-c'` *-m*
`sys.agv[0]` *-m*
`sys.argv`

2.1.2 Interactive Mode 交互模式

`>>>`
`...`

`>>>` `...`

```
python
Python 2.7 (#1, Feb 28 2010, 00:02:06)
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

`if`

```
>>> the_world_is_flat = 1
>>> if the_world_is_flat:
...     print "Be careful not to fall off!"
...
Be careful not to fall off!
```

2.2 The Interpreter and Its Environment 解释器及其环境

2.2.1 Error Handling 错误处理

`except` `try`

`try` `except`

KeyboardInterrupt try

KeyboardInterrupt try

2.2.2 Executable Python Scripts 执行 Python 脚本

```
#! /usr/bin/env python
```

PATH
#!
'\n' '\r\n'
'#'

PATH #!
'\n' '\r\n' '#'

```
$ chmod +x myscript.py
```

.py python.exe
.pyw

.py python.exe
.pyw

2.2.3 Source Code Encoding 源程序编码

#!

```
# -*- coding: encoding -*-
```

codecs

codecs

```
# -*- coding: iso-8859-15 -*-

currency = u"€"
print ord(currency)
```

UTF-8

Options/General/
Default Source Encoding/UTF-8

#!

UTF-8

Options/General/Default Source
Encoding/UTF-8

#!

2.2.4 The Interactive Startup File 交互式环境的启动文件

PYTHONSTARTUP
.profile

PYTHONSTARTUP
.profile

/dev/tty

sys.ps1 sys.ps2

/dev/tty

sys.ps1 sys.ps2

if os.path.isfile('.pythonrc.py'): execfile('.pythonrc.py')

if os.path.isfile('.pythonrc.py'): execfile('.pythonrc.py')

```
import os
filename = os.environ.
```

CHAPTER

THREE

AN INFORMAL INTRODUCTION TO PYTHON PYTHON 概要介绍

>>> ...

>>> `...`

#

physical line

```
# this is the first comment
SPAM = 1                 # and this is the second comment
                         # ... and now a third!
STRING = "# This is not a comment."
```

3.1 Using Python as a Calculator 将 Python 当做计算器

>>>

>>>

3.1.1 Numbers 数值

+ - * /

+ - * /

```
>>> 2+2
4
>>> # This is a comment
... 2+2
4
>>> 2+2  # and a comment on the same line as code
4
>>> (50-5*6)/4
5
>>> # Integer division returns the floor:
... 7/3
2
>>> 7/-3
-3
```

'='

'='

```
>>> width = 20
>>> height = 5*9
>>> width * height
900
```

```
>>> x = y = z = 0  # Zero x, y and z
>>> x
0
>>> y
0
>>> z
0
```

```
>>> # try to access an undefined variable
... n
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'n' is not defined
```

```
>>> 3 * 3.75 / 1.5
7.5
>>> 7.0 / 2
3.5
```

j J (real+imagj) complex(real, imag)

j J (real+imagj) complex(real, imag)

```
>>> 1j * 1J
(-1+0j)
>>> 1j * complex(0,1)
(-1+0j)
>>> 3+1j*3
(3+3j)
>>> (3+1j)*3
(9+3j)
>>> (1+2j)/(1+1j)
(1.5+0.5j)
```

z.real z.imag

z.real z.imag

```
>>> a=1.5+0.5j
>>> a.real
1.5
>>> a.imag
0.5
```

float() int() long() abs(z) z.real

float() int() long() abs(z) z.real

```
>>> a=3.0+4.0j
>>> float(a)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: can't convert complex to float; use abs(z)
>>> a.real
3.0
>>> a.imag
4.0
>>> abs(a)  # sqrt(a.real**2 + a.imag**2)
5.0
```

_

_

```
>>> tax = 12.5 / 100
>>> price = 100.50
>>> price * tax
12.5625
>>> price + _
113.0625
>>> round(_, 2)
113.06
```

3.1.2 Strings 字符串

```
>>> 'spam eggs'
'spam eggs'
>>> 'doesn\'t'
"doesn't"
>>> "doesn't"
"doesn't"
>>> '"Yes," he said.'
'"Yes," he said.'
>>> "\"Yes,\" he said."
'"Yes," he said.'
>>> '"Isn\'t," she said.'
'"Isn\'t," she said.'
```

```
hello = "This is a rather long string containing\n\
several lines of text just as you would do in C.\n\
    Note that whitespace at the beginning of the line is\
 significant."

print hello
```

\n

\n

```
This is a rather long string containing
several lines of text just as you would do in C.
    Note that whitespace at the beginning of the line is significant.
```

""" '''

""" '''

```
print """
Usage: thingy [OPTIONS]
     -h                        Display this usage message
     -H hostname               Hostname to connect to
"""
```

.. code-block:: text

\n

\n

```
hello = r"This is a rather long string containing\n\
several lines of text much as you would do in C."

print hello
```

```
This is a rather long string containing\n\
several lines of text much as you would do in C.
```

print

print

+ *

+ *

```
>>> word = 'Help' + 'A'
>>> word
'HelpA'
>>> '<' + word*5 + '>'
'<HelpAHelpAHelpAHelpAHelpA>'
```

word = 'Help' 'A'

word ='Help' 'A'

```
>>> 'str' 'ing'                   #  <-  This is ok
'string'
>>> 'str'.strip() + 'ing'   #  <-  This is ok
'string'
>>> 'str'.strip() 'ing'     #  <-  This is invalid
  File "<stdin>", line 1, in ?
    'str'.strip() 'ing'
                      ^
SyntaxError: invalid syntax
```

```
>>> word[4]
'A'
>>> word[0:2]
'He'
>>> word[2:4]
'lp'
```

```
>>> word[:2]    # The first two characters
'He'
>>> word[2:]    # Everything except the first two characters
'lpA'
```

```
>>> word[0] = 'x'
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object does not support item assignment
>>> word[:1] = 'Splat'
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object does not support slice assignment
```

```
>>> 'x' + word[1:]
'xelpA'
>>> 'Splat' + word[4]
'SplatA'
```

s[:i] + s[i:] s

s[:i] + s[i:] s

```
>>> word[:2] + word[2:]
'HelpA'
>>> word[:3] + word[3:]
'HelpA'
```

```
>>> word[1:100]
'elpA'
>>> word[10:]
''
>>> word[2:1]
''
```

```
>>> word[-1]     # The last character
'A'
>>> word[-2]     # The last-but-one character
'p'
>>> word[-2:]    # The last two characters
'pA'
>>> word[:-2]    # Everything except the last two characters
'Hel'
```

```
>>> word[-0]     # (since -0 equals 0)
'H'
```

```
>>> word[-100:]
'HelpA'
>>> word[-10]    # error
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
IndexError: string index out of range
```

```
 +---+---+---+---+---+
 | H | e | l | p | A |
 +---+---+---+---+---+
 0   1   2   3   4   5
-5  -4  -3  -2  -1
```

word[1:3]

word[1:3]

len()

len()

```
>>> s = 'supercalifragilisticexpialidocious'
>>> len(s)
34
```

str.format()

%

3.1.3 Unicode Strings Unicode 文本

i18n 'i' 'n'

i18n 'i' 'n'

```
>>> u'Hello World !'
u'Hello World !'
```

'u'

'u'

```
>>> u'Hello\u0020World !'
u'Hello World !'
```

\uXXXX

\uXXXX

```
>>> ur'Hello\u0020World !'
u'Hello World !'
>>> ur'Hello\\u0020World !'
u'Hello\\\\u0020World !'
```

unicode()

str()

unicode()

str()

```
>>> u"abc"
u'abc'
>>> str(u"abc")
```

```
'abc'
>>> u""
u'\xe4\xf6\xfc'
>>> str(u"")
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
UnicodeEncodeError: 'ascii' codec can't encode characters in position 0-2: ordinal not in range(128)
```

encode()

encode()

```
>>> u"".encode('utf-8')
'\xc3\xa4\xc3\xb6\xc3\xbc'
```

unicode()

unicode()

```
>>> unicode('\xc3\xa4\xc3\xb6\xc3\xbc', 'utf-8')
u'\xe4\xf6\xfc'
```

3.1.4 Lists 列表

```
>>> a = ['spam', 'eggs', 100, 1234]
>>> a
['spam', 'eggs', 100, 1234]
```

```
>>> a[0]
'spam'
>>> a[3]
1234
>>> a[-2]
100
>>> a[1:-1]
['eggs', 100]
>>> a[:2] + ['bacon', 2*2]
['spam', 'eggs', 'bacon', 4]
>>> 3*a[:3] + ['Boo!']
['spam', 'eggs', 100, 'spam', 'eggs', 100, 'spam', 'eggs', 100, 'Boo!']
```

```
>>> a[:]
['spam', 'eggs', 100, 1234]
```

```
>>> a
['spam', 'eggs', 100, 1234]
>>> a[2] = a[2] + 23
>>> a
['spam', 'eggs', 123, 1234]
```

```
>>> # Replace some items:
... a[0:2] = [1, 12]
>>> a
[1, 12, 123, 1234]
>>> # Remove some:
... a[0:2] = []
>>> a
[123, 1234]
>>> # Insert some:
... a[1:1] = ['bletch', 'xyzzy']
>>> a
[123, 'bletch', 'xyzzy', 1234]
>>> # Insert (a copy of) itself at the beginning
>>> a[:0] = a
>>> a
[123, 'bletch', 'xyzzy', 1234, 123, 'bletch', 'xyzzy', 1234]
>>> # Clear the list: replace all items with an empty list
>>> a[:] = []
>>> a
[]
```

len()

len()

```
>>> a = ['a', 'b', 'c', 'd']
>>> len(a)
4
```

```
>>> q = [2, 3]
>>> p = [1, q, 4]
>>> len(p)
3
>>> p[1]
[2, 3]
>>> p[1][0]
```

```
2
>>> p[1].append('xtra')     # See section 5.1
>>> p
[1, [2, 3, 'xtra'], 4]
>>> q
[2, 3, 'xtra']
```

`p[1]` `q`

`p[1]` `q`

3.2 First Steps Towards Programming 编程的第一步

```
>>> # Fibonacci series:
... # the sum of two elements defines the next
... a, b = 0, 1
>>> while b < 10:
...     print b
...     a, b = b, a+b
...
1
1
2
3
5
8
```

- `a` `b`

  `a` `b`

- `while` `b < 10`

  `<` `>` `==` `<=` `>=` `!=`

  `b < 10` `while`

<

> == <= >= !=

-

- print

print

```
>>> i = 256*256
>>> print 'The value of i is', i
The value of i is 65536
```

```
>>> a, b = 0, 1
>>> while b < 1000:
...     print b,
...     a, b = b, a+b
...
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

CHAPTER

FOUR

MORE CONTROL FLOW TOOLS 深入流程控制

while

while

4.1 **if** Statements **if** 語句

if

```
>>> x = int(raw_input("Please enter an integer: "))
Please enter an integer: 42
>>> if x < 0:
...      x = 0
...      print 'Negative changed to zero'
... elif x == 0:
...      print 'Zero'
... elif x == 1:
...      print 'Single'
... else:
...      print 'More'
...
More
```

elif else elif
if elif elif
switch case

4.2 **for** Statements **for** 语句

for

for

```
>>> # Measure some strings:
... a = ['cat', 'window', 'defenestrate']
>>> for x in a:
...     print x, len(x)
...
cat 3
window 6
defenestrate 12
```

```
>>> for x in a[:]: # make a slice copy of the entire list
...    if len(x) > 6: a.insert(0, x)
...
>>> a
['defenestrate', 'cat', 'window', 'defenestrate']
```

4.3 The **range()** Function **range()** 函数

`range()`

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

`range(10)`

```
>>> range(5, 10)
[5, 6, 7, 8, 9]
>>> range(0, 10, 3)
[0, 3, 6, 9]
>>> range(-10, -100, -30)
[-10, -40, -70]
```

`range()` `len()`

```
>>> a = ['Mary', 'had', 'a', 'little', 'lamb']
>>> for i in range(len(a)):
...     print i, a[i]
...
0 Mary
1 had
2 a
3 little
4 lamb
```

enumerate()

enumerate()

4.4 **break** and **continue** Statements, and **else** Clauses on Loops break 和 continue 语句，以及 循环中的 else 子句

break for while

continue

continue

else
for while
break

else for while
break

```
>>> for n in range(2, 10):
...     for x in range(2, n):
...         if n % x == 0:
...             print n, 'equals', x, '*', n/x
...             break
...     else:
...         # loop fell through without finding a factor
...         print n, 'is a prime number'
...
2 is a prime number
3 is a prime number
4 equals 2 * 2
5 is a prime number
6 equals 2 * 3
7 is a prime number
8 equals 2 * 4
9 equals 3 * 3
```

4.5 pass Statements pass 语句

pass

```
>>> while True:
...     pass  # Busy-wait for keyboard interrupt (Ctrl+C)
...
```

```
>>> class MyEmptyClass:
...     pass
...
```

pass

pass

pass

pass

```
>>> def initlog(*args):
...     pass   # Remember to implement this!
...
```

4.6 Defining Functions 定义函数

```
>>> def fib(n):    # write Fibonacci series up to n
...     """Print a Fibonacci series up to n."""
...     a, b = 0, 1
...     while a < n:
...         print a,
...         a, b = b, a+b
...
>>> # Now call the function we just defined:
... fib(2000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

def

def

global

global

```
>>> fib
<function fib at 10042ed0>
>>> f = fib
>>> f(100)
0 1 1 2 3 5 8 13 21 34 55 89
```

fib

return

None

None

print

fib

return None

```
>>> fib(0)
>>> print fib(0)
None
```

```
>>> def fib2(n): # return Fibonacci series up to n
...     """Return a list containing the Fibonacci series up to n."""
...     result = []
...     a, b = 0, 1
...     while a < n:
...         result.append(a)    # see below
...         a, b = b, a+b
...     return result
...
>>> f100 = fib2(100)    # call it
>>> f100                # write the result
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

- return return None None
  return return None None

- result.append(a) result obj.methodname obj methodname
  append()
  result = result + [a]
  result.append(b) result obj.methodename obj
  methodename
  append()

result = result + [b]

4.7 More on Defining Functions 深入函数定义

4.7.1 Default Argument Values 参数默认值

```
def ask_ok(prompt, retries=4, complaint='Yes or no, please!'):
    while True:
        ok = raw_input(prompt)
        if ok in ('y', 'ye', 'yes'):
            return True
        if ok in ('n', 'no', 'nop', 'nope'):
            return False
        retries = retries - 1
        if retries < 0:
            raise IOError('refusenik user')
        print complaint
```

- ask_ok('Do you really want to quit?')
- ask_ok('OK to overwrite the file?', 2)
- ask_ok('OK to overwrite the file?', 2, 'Come on, only yes or no!')

in

in

```
i = 5

def f(arg=i):
    print arg

i = 6
f()
```

5

```
def f(a, L=[]):
    L.append(a)
    return L

print f(1)
print f(2)
print f(3)
```

```
[1]
[1, 2]
[1, 2, 3]
```

```
def f(a, L=None):
    if L is None:
        L = []
    L.append(a)
    return L
```

4.7.2 Keyword Arguments 关键字参数

keyword = value

keyword = value

```
def parrot(voltage, state='a stiff', action='voom', type='Norwegian Blue'):
    print "-- This parrot wouldn't", action,
    print "if you put", voltage, "volts through it."
    print "-- Lovely plumage, the", type
    print "-- It's", state, "!"
```

```
parrot(1000)
parrot(action = 'VOOOOOM', voltage = 1000000)
parrot('a thousand', state = 'pushing up the daisies')
parrot('a million', 'bereft of life', 'jump')
```

```
parrot()                     # required argument missing
parrot(voltage=5.0, 'dead')  # non-keyword argument following keyword
parrot(110, voltage=220)     # duplicate value for argument
parrot(actor='John Cleese')  # unknown keyword
```

```
>>> def function(a):
...     pass
...
>>> function(0, a=0)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: function() got multiple values for keyword argument 'a'
```

`**name`

`*name`

`*name` `**name`

`**name` `*name`

`*name` `**name`

```
def cheeseshop(kind, *arguments, **keywords):
    print "-- Do you have any", kind, "?"
    print "-- I'm sorry, we're all out of", kind
    for arg in arguments: print arg
    print "-" * 40
    keys = keywords.keys()
    keys.sort()
    for kw in keys: print kw, ":", keywords[kw]
```

```
cheeseshop("Limburger", "It's very runny, sir.",
           "It's really very, VERY runny, sir.",
           shopkeeper='Michael Palin',
           client="John Cleese",
           sketch="Cheese Shop Sketch")
```

```
-- Do you have any Limburger ?
-- I'm sorry, we're all out of Limburger
It's very runny, sir.
It's really very, VERY runny, sir.
----------------------------------------
client : John Cleese
shopkeeper : Michael Palin
sketch : Cheese Shop Sketch
```

`sort()` `keywords`

`sort()`

4.7.3 Arbitrary Argument Lists 可变参数列表

```
def write_multiple_items(file, separator, *args):
    file.write(separator.join(args))
```

4.7.4 Unpacking Argument Lists 参数列表的分拆

range()

*

range()

*

```
>>> range(3, 6)             # normal call with separate arguments
[3, 4, 5]
>>> args = [3, 6]
>>> range(*args)            # call with arguments unpacked from a list
[3, 4, 5]
```

**

**

```
>>> def parrot(voltage, state='a stiff', action='voom'):
...     print "-- This parrot wouldn't", action,
...     print "if you put", voltage, "volts through it.",
...     print "E's", state, "!"
...
>>> d = {"voltage": "four million", "state": "bleedin' demised", "action": "VOOM"}
>>> parrot(**d)
-- This parrot wouldn't VOOM if you put four million volts through it. E's bleedin' demised !
```

4.7.5 Lambda Forms Lambda 形式

lambda

lambda a, b: a+b

lambda

lambda a, b: a+b

```
>>> def make_incrementor(n):
...     return lambda x: x + n
...
>>> f = make_incrementor(42)
>>> f(0)
42
>>> f(1)
43
```

4.7.6 Documentation Strings 文档字符串

```
>>> def my_function():
...     """Do nothing, but document it.
...
...     No, really, it doesn't do anything.
...     """
...     pass
...
```

```
>>> print my_function.__doc__
Do nothing, but document it.

    No, really, it doesn't do anything.
```

4.8 Intermezzo: Coding Style 插曲：编码风格

-
-
-
-
-
- `a = f(1, 2) + g(3, 4)`

`a = f(1, 2) + g(3, 4)`

- CamelCase lower_case_with_underscores self

  `` `` _ _ self

-

CHAPTER

FIVE

DATA STRUCTURES 数据结构

5.1 More on Lists 深入列表

list.**append**( )

a[len(a):] = [x]

a[len(a):] = [x]

list.**extend**( )

a[len(a):] =
L

a[len(a):] = L

list.**insert**( )

a.insert(0, x) a.insert(len(a),
x) a.append(x)

a.insert(0,
x) a.insert(len(a), x) a.append(x)

list.**remove**( )

list.**pop**( )

a.pop()

a.pop()

list.index( )

list.count( )

list.sort()

list.reverse()

```
>>> a = [66.25, 333, 333, 1, 1234.5]
>>> print a.count(333), a.count(66.25), a.count('x')
2 1 0
>>> a.insert(2, -1)
>>> a.append(333)
>>> a
[66.25, 333, -1, 333, 1, 1234.5, 333]
>>> a.index(333)
1
>>> a.remove(333)
>>> a
[66.25, -1, 333, 1, 1234.5, 333]
>>> a.reverse()
>>> a
[333, 1234.5, 1, 333, -1, 66.25]
>>> a.sort()
>>> a
[-1, 1, 66.25, 333, 333, 1234.5]
```

5.1.1 Using Lists as Stacks 把链表当作堆栈使用

append() pop()

append() pop()

```
>>> stack = [3, 4, 5]
>>> stack.append(6)
>>> stack.append(7)
>>> stack
[3, 4, 5, 6, 7]
>>> stack.pop()
7
>>> stack
[3, 4, 5, 6]
>>> stack.pop()
6
>>> stack.pop()
5
>>> stack
[3, 4]
```

5.1.2 Using Lists as Queues 把链表当作队列使用

collections.deque

collections.deque

```
>>> from collections import deque
>>> queue = deque(["Eric", "John", "Michael"])
>>> queue.append("Terry")           # Terry arrives
>>> queue.append("Graham")          # Graham arrives
>>> queue.popleft()                 # The first to arrive now leaves
'Eric'
>>> queue.popleft()                 # The second to arrive now leaves
'John'
>>> queue                           # Remaining queue in order of arrival
deque(['Michael', 'Terry', 'Graham'])
```

5.1.3 Functional Programming Tools 函数式编程工具

filter() map() reduce()

filter() map` :func:`reduce()

filter(function, sequence) function(item) string tuple list

filter(function, sequence) function(item) string tuple list

```
>>> def f(x): return x % 2 != 0 and x % 3 != 0
...
>>> filter(f, range(2, 25))
[5, 7, 11, 13, 17, 19, 23]
```

map(function, sequence) function(item)

map(function, sequence) function(item)

```
>>> def cube(x): return x*x*x
...
>>> map(cube, range(1, 11))
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

None

None

```
>>> seq = range(8)
>>> def add(x, y): return x+y
...
>>> map(add, seq, seq)
[0, 2, 4, 6, 8, 10, 12, 14]
```

reduce(function, sequence)

reduce(func, sequence)

```
>>> def add(x,y): return x+y
...
>>> reduce(add, range(1, 11))
55
```

```
>>> def sum(seq):
...     def add(x,y): return x+y
...     return reduce(add, seq, 0)
...
>>> sum(range(1, 11))
55
```

```
>>> sum([])
0
```

sum()
sum(sequence)
sum() sum(sequence)

5.1.4 List Comprehensions 列表推导式

map()
filter() lambda

for for if
for if

map() filter() lambda
for
for if for if

```
>>> freshfruit = ['  banana', '  loganberry ', 'passion fruit  ']
>>> [weapon.strip() for weapon in freshfruit]
['banana', 'loganberry', 'passion fruit']
>>> vec = [2, 4, 6]
>>> [3*x for x in vec]
[6, 12, 18]
>>> [3*x for x in vec if x > 3]
[12, 18]
>>> [3*x for x in vec if x < 2]
[]
>>> [[x,x**2] for x in vec]
[[2, 4], [4, 16], [6, 36]]
>>> [x, x**2 for x in vec]  # error - parens required for tuples
  File "<stdin>", line 1, in ?
    [x, x**2 for x in vec]
               ^
SyntaxError: invalid syntax
>>> [(x, x**2) for x in vec]
[(2, 4), (4, 16), (6, 36)]
>>> vec1 = [2, 4, 6]
>>> vec2 = [4, 3, -9]
>>> [x*y for x in vec1 for y in vec2]
[8, 6, -18, 16, 12, -36, 24, 18, -54]
>>> [x+y for x in vec1 for y in vec2]
[6, 5, -7, 8, 7, -5, 10, 9, -3]
>>> [vec1[i]*vec2[i] for i in range(len(vec1))]
[8, 12, -54]
```

map()

map()

```
>>> [str(round(355/113.0, i)) for i in range(1,6)]
['3.1', '3.14', '3.142', '3.1416', '3.14159']
```

5.1.5 Nested List Comprehensions 嵌套的列表推导式

```
>>> mat = [
...         [1, 2, 3],
...         [4, 5, 6],
...         [7, 8, 9],
...       ]
```

```
>>> print [[row[i] for row in mat] for i in [0, 1, 2]]
[[1, 4, 7], [2, 5, 8], [3, 6, 9]]
```

```
for i in [0, 1, 2]:
    for row in mat:
        print row[i],
    print
```

zip()

zip()

```
>>> zip(*mat)
[(1, 4, 7), (2, 5, 8), (3, 6, 9)]
```

5.2 The **del** statement 删除语句

del
pop() del

del pop()
del

```
>>> a = [-1, 1, 66.25, 333, 333, 1234.5]
>>> del a[0]
>>> a
[1, 66.25, 333, 333, 1234.5]
>>> del a[2:4]
>>> a
[1, 66.25, 1234.5]
>>> del a[:]
>>> a
[]
```

del

del

```
>>> del a
```

a
del
a del

5.3 Tuples and Sequences 元组和序列

```
>>> t = 12345, 54321, 'hello!'
>>> t[0]
12345
>>> t
(12345, 54321, 'hello!')
>>> # Tuples may be nested:
... u = t, (1, 2, 3, 4, 5)
>>> u
((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
```

```
>>> empty = ()
>>> singleton = 'hello',    # <-- note trailing comma
>>> len(empty)
0
>>> len(singleton)
1
>>> singleton
('hello',)
```

t = 12345, 54321, 'hello!' 12345 54321 'hello!'

t = 12345, 54321, 'hello!' 12345 54321 'hello!'

```
>>> x, y, z = t
```

5.4 Sets 集合

```
>>> basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']
>>> fruit = set(basket)               # create a set without duplicates
>>> fruit
set(['orange', 'pear', 'apple', 'banana'])
>>> 'orange' in fruit                 # fast membership testing
True
>>> 'crabgrass' in fruit
False

>>> # Demonstrate set operations on unique letters from two words
...
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a                                  # unique letters in a
set(['a', 'r', 'b', 'c', 'd'])
>>> a - b                              # letters in a but not in b
set(['r', 'd', 'b'])
>>> a | b                              # letters in either a or b
set(['a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'])
>>> a & b                              # letters in both a and b
set(['a', 'c'])
>>> a ^ b                              # letters in a or b but not both
set(['r', 'd', 'b', 'm', 'z', 'l'])
```

5.5 Dictionaries 字典

append() extend()

associative memories associative arrays

append() extend()

{}

{}

del

del

keys()

sort()

in

keys()

sort() in

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
>>> tel.keys()
['guido', 'irv', 'jack']
>>> 'guido' in tel
True
```

dict()

```
>>> dict([('sape', 4139), ('guido', 4127), ('jack', 4098)])
{'sape': 4139, 'jack': 4098, 'guido': 4127}
>>> dict([(x, x**2) for x in (2, 4, 6)])     # use a list comprehension
{2: 4, 4: 16, 6: 36}
```

dict()

dict()

```
>>> dict(sape=4139, guido=4127, jack=4098)
{'sape': 4139, 'jack': 4098, 'guido': 4127}
```

5.6 Looping Techniques 循环技巧

iteritems()

iteritems()

```
>>> knights = {'gallahad': 'the pure', 'robin': 'the brave'}
>>> for k, v in knights.iteritems():
...     print k, v
...
gallahad the pure
robin the brave
```

enumerate()

enumerate()

```
>>> for i, v in enumerate(['tic', 'tac', 'toe']):
...     print i, v
...
0 tic
1 tac
2 toe
```

zip()

```
>>> questions = ['name', 'quest', 'favorite color']
>>> answers = ['lancelot', 'the holy grail', 'blue']
>>> for q, a in zip(questions, answers):
...     print 'What is your {0}?  It is {1}.'.format(q, a)
...
What is your name?  It is lancelot.
What is your quest?  It is the holy grail.
What is your favorite color?  It is blue.
```

reversed()

reversed()

```
>>> for i in reversed(xrange(1,10,2)):
...     print i
...
9
7
5
3
1
```

sorted()

sorted()

```
>>> basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']
>>> for f in sorted(set(basket)):
...     print f
...
apple
banana
orange
pear
```

5.7 More on Conditions 深入条件控制

while if

while if

in not in

is is not

in not in is is not

a < b == c a b

b c

a < b == c a b b c

and or

not

not or

A and not B or C (A and (not B)) or C

and or not

or A and not B or

C (A and (notB)) or C

and or

A C B A and B and C C

and or

A C B A and B and C

```
>>> string1, string2, string3 = '', 'Trondheim', 'Hammer Dance'
>>> non_null = string1 or string2 or string3
>>> non_null
'Trondheim'
```

= ==

== =

5.8 Comparing Sequences and Other Types 比较序列和其它类型

```
(1, 2, 3)              < (1, 2, 4)
[1, 2, 3]              < [1, 2, 4]
'ABC' < 'C' < 'Pascal' < 'Python'
(1, 2, 3, 4)           < (1, 2, 4)
(1, 2)                 < (1, 2, -1)
(1, 2, 3)             == (1.0, 2.0, 3.0)
(1, 2, ('aa', 'ab'))   < (1, 2, ('abc', 'a'), 4)
```

CHAPTER

SIX

MODULES 模块

.py

__name__

fibo.py

.py

__name__

fibo.py

```
# Fibonacci numbers module

def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
        print b,
        a, b = b, a+b

def fib2(n): # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result
```

```
>>> import fibo
```

fibo

fibo

fibo fibo

```
>>> fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>> fibo.fib2(100)
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
>>> fibo.__name__
'fibo'
```

```
>>> fib = fibo.fib
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

6.1 More on Modules 深入模块

modname.itemname

modname.itemname

import

reload()

reload(modulename)

reload() reload(modulename)

import

import

import

```
>>> from fibo import fib, fib2
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

fibo

fibo

```
>>> from fibo import *
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

-

-

*

*

6.1.1 Executing modules as scripts 作为脚本来执行模块

```
python fibo.py <arguments>
```

__name__

"__main__"

__name__ "__main__"

```
if __name__ == "__main__":
    import sys
    fib(int(sys.argv[1]))
```

```
$ python fibo.py 50
1 1 2 3 5 8 13 21 34
```

```
>>> import fibo
>>>
```

6.1.2 The Module Search Path 模块搜索路径

spam spam.py
PYTHONPATH PATH
PYTHONPATH
.:/usr/local/lib/python

spam spam.py
PYTHONPATH PATH PYTHONPATH
.:/usr/local/lib/python
sys.path

PYTHONPATH

sys.path
PYTHONPATH

6.1.3 ``Compiled'' Python files “编译的” Python 文件

spam.pyc spam.py
spam
spam.py spam.pyc spam.pyc .pyc

spam.py
spam.pyc spam
spam.pyc spam.py spam.pyc

spam.pyc spam.py
spam.pyc

spam.pyc

spam.pyc

spam.pyc spam.py
spam.pyc spam.pyc
spam.pyc

- *-O*
  .pyo assert
  -O .pyc .py

  -O .pyo
  assert *-O*
  .pyc .py
- *-O* *-OO*

  __doc__
  .pyo

  -O *-OO*
  __doc__ .pyo
- .pyc .pyo
  .py .pyc .pyo

  .pyc .pyo .py .pyc .pyo
- .pyc .pyo

  .pyc .pyo

  .pyc .pyo

  .pyc .pyo
- spam.pyc spam.pyo *-O*
  spam.py

  spam.py spam.pyc spam.pyc
  -O spam.py
- compileall .pyc .pyo *-O*

  compileall .pyc .pyo .pyo

6.2 Standard Modules 标准模块

winreg
sys
sys.ps1 sys.ps2

sys

```
>>> import sys
>>> sys.ps1
'>>> '
>>> sys.ps2
'... '
>>> sys.ps1 = 'C> '
C> print 'Yuck!'
Yuck!
C>
```

sys.path PYTHONPATH
PYTHONPATH

sys.path
PYTHONPATH

```
>>> import sys
>>> sys.path.append('/ufs/guido/lib/python')
```

6.3 The `dir()` Function `dir()` 函数

dir()

dir()

```
>>> import fibo, sys
>>> dir(fibo)
['__name__', 'fib', 'fib2']
>>> dir(sys)
['__displayhook__', '__doc__', '__excepthook__', '__name__', '__stderr__',
 '__stdin__', '__stdout__', '_getframe', 'api_version', 'argv',
 'builtin_module_names', 'byteorder', 'callstats', 'copyright',
 'displayhook', 'exc_clear', 'exc_info', 'exc_type', 'excepthook',
 'exec_prefix', 'executable', 'exit', 'getdefaultencoding', 'getdlopenflags',
 'getrecursionlimit', 'getrefcount', 'hexversion', 'maxint', 'maxunicode',
 'meta_path', 'modules', 'path', 'path_hooks', 'path_importer_cache',
 'platform', 'prefix', 'ps1', 'ps2', 'setcheckinterval', 'setdlopenflags',
 'setprofile', 'setrecursionlimit', 'settrace', 'stderr', 'stdin', 'stdout',
 'version', 'version_info', 'warnoptions']
```

`dir()`

`dir()`

```
>>> a = [1, 2, 3, 4, 5]
>>> import fibo
>>> fib = fibo.fib
>>> dir()
['__builtins__', '__doc__', '__file__', '__name__', 'a', 'fib', 'fibo', 'sys']
```

`dir()`

`__builtin__`

`dir()` `__builtin__`

```
>>> import __builtin__
>>> dir(__builtin__)
['ArithmeticError', 'AssertionError', 'AttributeError', 'DeprecationWarning',
 'EOFError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False',
 'FloatingPointError', 'FutureWarning', 'IOError', 'ImportError',
 'IndentationError', 'IndexError', 'KeyError', 'KeyboardInterrupt',
 'LookupError', 'MemoryError', 'NameError', 'None', 'NotImplemented',
 'NotImplementedError', 'OSError', 'OverflowError',
 'PendingDeprecationWarning', 'ReferenceError', 'RuntimeError',
 'RuntimeWarning', 'StandardError', 'StopIteration', 'SyntaxError',
 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'True',
 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError',
 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError',
 'UserWarning', 'ValueError', 'Warning', 'WindowsError',
 'ZeroDivisionError', '_', '__debug__', '__doc__', '__import__',
 '__name__', 'abs', 'apply', 'basestring', 'bool', 'buffer',
 'callable', 'chr', 'classmethod', 'cmp', 'coerce', 'compile',
 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'divmod',
 'enumerate', 'eval', 'execfile', 'exit', 'file', 'filter', 'float',
 'frozenset', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex',
 'id', 'input', 'int', 'intern', 'isinstance', 'issubclass', 'iter',
 'len', 'license', 'list', 'locals', 'long', 'map', 'max', 'memoryview',
 'min', 'object', 'oct', 'open', 'ord', 'pow', 'property', 'quit', 'range',
 'raw_input', 'reduce', 'reload', 'repr', 'reversed', 'round', 'set',
 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super',
 'tuple', 'type', 'unichr', 'unicode', 'vars', 'xrange', 'zip']
```

6.4 Packages 包

A.B B A

A.B B
A

.wav .aiff .au

.wav .aiff .au

```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/                  Subpackage for sound effects
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/                  Subpackage for filters
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
```

sys.path

sys.path

__init__.py

string
__init__.py

__all__

__init__.py
string
__init__.py __all__

```
import sound.effects.echo
```

sound.effects.echo

Sound.Effects.echo

```
sound.effects.echo.echofilter(input, output, delay=0.7, atten=4)
```

```
from sound.effects import echo
```

echo

echo

```
echo.echofilter(input, output, delay=0.7, atten=4)
```

```
from sound.effects.echo import echofilter
```

echo echofilter()

echo echofilter()

```
echofilter(input, output, delay=0.7, atten=4)
```

from package import item

import
ImportError

from package import item

import item.subitem.subsubitem

import item.subitem.subsubitem

6.4.1 Importing * From a Package

from sound.effects import *

from sound.Effects import *

import __init__.py
__all__ from
package import *

sounds/
effects/__init__.py

import
from package import * __init__.py __all__

Sounds/
Effects/__init__.py

```
__all__ = ["echo", "surround", "reverse"]
```

from sound.effects import *
sound
from Sound.Effects import * sound

__all__ from sound.effects import *
sound.effects
sound.effects __init__.py

__init__.py
import

__all__ from Sound.Effects import * sound.effects
sound.effects __ __
__init__.py
import

```
import sound.effects.echo
import sound.effects.surround
from sound.effects import *
```

echo surround
sound.effects from...import
__all__

echo surround from...import
sound.effects __all__

`import *`

`import *`

`from Package import specific_submodule`

`from Package import specific_submodule`

6.4.2 Intra-package References 包内引用

`surround`
`echo` `import`
`surround`
`import echo` `from echo import echofilter`

`import`

`echo`
`import` `surround`
`import echo` `from echo import echofilter`

`sound`

`sound.filters.vocoder` `echo` `sound.effects`
`from sound.effects import echo`

`sound`
`sound.filters.vocoder` `sound.effects` `echo` `from`
`Sound.Effects import echo`

`from module import name`

`surround`

`from module`
`import name`
`surround`

```
from . import echo
from .. import formats
from ..filters import equalizer
```

`"__main__"`

`"__main__"`

6.4.3 Packages in Multiple Directories 多重目录中的包

`__path__`

`__init__.py`

`__path__` `__init__.py`

CHAPTER

SEVEN

INPUT AND OUTPUT 输入和输出

7.1 Fancier Output Formatting 玩转输出格式

print
write()
sys.stdout

print
write() sys.stdout

string
str.format()

string
str.format()

repr() str()

repr() str()

str()
repr()
SyntaxError
str() repr()

str() repr()
SyntaxError str() repr()

```
>>> s = 'Hello, world.'
>>> str(s)
'Hello, world.'
>>> repr(s)
"'Hello, world.'"
>>> str(1.0/7.0)
'0.142857142857'
>>> repr(1.0/7.0)
'0.14285714285714285'
>>> x = 10 * 3.25
>>> y = 200 * 200
>>> s = 'The value of x is ' + repr(x) + ', and y is ' + repr(y) + '...'
>>> print s
The value of x is 32.5, and y is 40000...
>>> # The repr() of a string adds string quotes and backslashes:
... hello = 'hello, world\n'
>>> hellos = repr(hello)
>>> print hellos
'hello, world\n'
>>> # The argument to repr() may be any Python object:
... repr((x, y, ('spam', 'eggs')))
"(32.5, 40000, ('spam', 'eggs'))"
```

```
>>> for x in range(1, 11):
...     print repr(x).rjust(2), repr(x*x).rjust(3),
...     # Note trailing comma on previous line
...     print repr(x*x*x).rjust(4)
...
 1   1    1
 2   4    8
 3   9   27
 4  16   64
 5  25  125
 6  36  216
 7  49  343
 8  64  512
 9  81  729
10 100 1000

>>> for x in range(1,11):
...     print '{0:2d} {1:3d} {2:4d}'.format(x, x*x, x*x*x)
...
 1   1    1
 2   4    8
 3   9   27
 4  16   64
 5  25  125
 6  36  216
 7  49  343
 8  64  512
 9  81  729
10 100 1000
```

print

print

rjust()

ljust() center()

x.ljust(n)

[:n]

rjust()

ljust() center()

x.ljust(n)[:n]

zfill()

zfill()

```
>>> '12'.zfill(5)
'00012'
>>> '-3.14'.zfill(7)
'-003.14'
>>> '3.14159265359'.zfill(5)
'3.14159265359'
```

str.format()

str.format()

```
>>> print 'We are the {} who say "{}!"'.format('knights', 'Ni')
We are the knights who say "Ni!"
```

format()

format()

format() format()

```
>>> print '{0} and {1}'.format('spam', 'eggs')
spam and eggs
>>> print '{1} and {0}'.format('spam', 'eggs')
eggs and spam
```

format()

format()

```
>>> print 'This {food} is {adjective}.'.format(
...       food='spam', adjective='absolutely horrible')
This spam is absolutely horrible.
```

```
>>> print 'The story of {0}, {1}, and {other}.'.format('Bill', 'Manfred',
...                                                    other='Georg')
The story of Bill, Manfred, and Georg.
```

'!s'   str()   '!r'   repr()

'!s'   str()   '!r'   repr()

```
>>> import math
>>> print 'The value of PI is approximately {}.'.format(math.pi)
The value of PI is approximately 3.14159265359.
>>> print 'The value of PI is approximately {!r}.'.format(math.pi)
The value of PI is approximately 3.141592653589793.
```

':'

':'

```
>>> import math
>>> print 'The value of PI is approximately {0:.3f}.'.format(math.pi)
The value of PI is approximately 3.142.
```

':'

':'

```
>>> table = {'Sjoerd': 4127, 'Jack': 4098, 'Dcab': 7678}
>>> for name, phone in table.items():
...     print '{0:10} ==> {1:10d}'.format(name, phone)
...
Jack       ==>       4098
Dcab       ==>       7678
Sjoerd     ==>       4127
```

'[]'

```
>>> table = {'Sjoerd': 4127, 'Jack': 4098, 'Dcab': 8637678}
>>> print ('Jack: {0[Jack]:d}; Sjoerd: {0[Sjoerd]:d}; '
...        'Dcab: {0[Dcab]:d}'.format(table))
Jack: 4098; Sjoerd: 4127; Dcab: 8637678
```

```
>>> table = {'Sjoerd': 4127, 'Jack': 4098, 'Dcab': 8637678}
>>> print 'Jack: {Jack:d}; Sjoerd: {Sjoerd:d}; Dcab: {Dcab:d}'.format(**table)
Jack: 4098; Sjoerd: 4127; Dcab: 8637678
```

vars()

vars()

str.format()

str.format()

7.1.1 Old string formatting 旧式的字符串格式化

%
sprintf()

% sprintf()

```
>>> import math
>>> print 'The value of PI is approximately %5.3f.' % math.pi
The value of PI is approximately 3.142.
```

str.format() %
str.format()

str.format() %

7.2 Reading and Writing Files 读写文件

open() open(filename,
mode)

open() open(filename, mode)

```
>>> f = open('/tmp/workfile', 'w')
>>> print f
<open file '/tmp/workfile', mode 'w' at 80a0960>
```

'r'
'w'
'a'
'r+'
'r'

'r' 'w'
'a' 'r+'
'r'

'b'
'rb' 'wb' 'r+b'

JPEG EXE
'b'

'b' 'rb' 'wb' 'r+b'

'b'

7.2.1 Methods of File Objects 文件对象方法

f

f
f.read(size)

f.read() ""
f.read(size)

```
>>> f.read()
'This is the entire file.\n'
>>> f.read()
''
```

f.readline() \n
f.readline()
'\n'

f.readline() \n
f.readline()
'\n

```
>>> f.readline()
'This is the first line of the file.\n'
>>> f.readline()
'Second line of the file\n'
```

```
>>> f.readline()
''
```

f.readlines()

```
>>> f.readlines()
['This is the first line of the file.\n', 'Second line of the file\n']
```

```
>>> for line in f:
        print line,

This is the first line of the file.
Second line of the file
```

f.write(string) None

f.write(string) None

```
>>> f.write('This is a test\n')
```

```
>>> value = ('the answer', 42)
>>> s = str(value)
>>> f.write(s)
```

f.tell() f.seek(offset, from_what)

- -

-

f.tell()

f.seek(offset,from_what)

- -

-

```
>>> f = open('/tmp/workfile', 'r+')
>>> f.write('0123456789abcdef')
>>> f.seek(5)     # Go to the 6th byte in the file
>>> f.read(1)
'5'
>>> f.seek(-3, 2) # Go to the 3rd byte before the end
>>> f.read(1)
'd'
```

f.close()
f.close()

f.close() f.close()

```
>>> f.close()
>>> f.read()
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
ValueError: I/O operation on closed file
```

with

try finally

with
try finally

```
>>> with open('/tmp/workfile', 'r') as f:
...     read_data = f.read()
>>> f.closed
True
```

isatty() truncate()

isatty() truncate()

7.2.2 The **pickle** Module **pickle** 模块

read() int()
'123'

read()
int() '123'

pickle

pickle

x f

x f

```
pickle.dump(x, f)
```

f

f

```
x = pickle.load(f)
```

pickle

pickle

pickle

pickle

pickle

pickle

CHAPTER

EIGHT

ERRORS AND EXCEPTIONS 错误和异常

8.1 Syntax Errors 语法错误

```
>>> while True print 'Hello world'
  File "<stdin>", line 1, in ?
    while True print 'Hello world'
                   ^
SyntaxError: invalid syntax
```

print

':'

print

':'

8.2 Exceptions

```
>>> 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
ZeroDivisionError: integer division or modulo by zero
>>> 4 + spam*3
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
NameError: name 'spam' is not defined
>>> '2' + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: cannot concatenate 'str' and 'int' objects
```

ZeroDivisionError
NameError TypeError

ZeroDivisionError NameError`
:exc:`TypeError

8.3 Handling Exceptions 控制异常

Control-C
KeyboardInterrupt

Control-C
KeyboardInterrupt

```
>>> while True:
...     try:
...         x = int(raw_input("Please enter a number: "))
...         break
...     except ValueError:
```

```
...         print "Oops!  That was no valid number.  Try again..."
...
```

try

try

- try except

  try except

- try

  try

- except

  try

  except try

- try

```
except:
    print "Unexpected error:", sys.exc_info()[0]
    raise
```

try except

try except

```
for arg in sys.argv[1:]:
    try:
        f = open(arg, 'r')
    except IOError:
        print 'cannot open', arg
    else:
        print arg, 'has', len(f.readlines()), 'lines'
        f.close()
```

else try

try except

else try try except

instance.args

__str__()

.args

instance.args __str__()

.args

```
>>> try:
...     raise Exception('spam', 'eggs')
... except Exception as inst:
...     print type(inst)     # the exception instance
...     print inst.args      # arguments stored in .args
...     print inst           # __str__ allows args to printed directly
...     x, y = inst          # __getitem__ allows args to be unpacked directly
...     print 'x =', x
...     print 'y =', y
...
<type 'exceptions.Exception'>
('spam', 'eggs')
('spam', 'eggs')
```

```
x = spam
y = eggs
```

```
>>> def this_fails():
...     x = 1/0
...
>>> try:
...     this_fails()
... except ZeroDivisionError as detail:
...     print 'Handling run-time error:', detail
...
Handling run-time error: integer division or modulo by zero
```

8.4 Raising Exceptions 抛出异常

raise

raise

```
>>> raise NameError('HiThere')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
NameError: HiThere
```

raise

Exception

raise Exception

raise

raise

```
>>> try:
...     raise NameError('HiThere')
... except NameError:
...     print 'An exception flew by!'
...     raise
...
An exception flew by!
Traceback (most recent call last):
  File "<stdin>", line 2, in ?
NameError: HiThere
```

8.5 User-defined Exceptions 用户自定义异常

Exception

Exception

```
>>> class MyError(Exception):
...     def __init__(self, value):
...         self.value = value
...     def __str__(self):
...         return repr(self.value)
...
>>> try:
...     raise MyError(2*2)
... except MyError as e:
...     print 'My exception occurred, value:', e.value
...
My exception occurred, value: 4
>>> raise MyError('oops!')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
__main__.MyError: 'oops!'
```

`__init__()` Exception

`__init__()`

```
class Error(Exception):
    """Base class for exceptions in this module."""
    pass

class InputError(Error):
    """Exception raised for errors in the input.

    Attributes:
        expr -- input expression in which the error occurred
        msg  -- explanation of the error
    """

    def __init__(self, expr, msg):
        self.expr = expr
        self.msg = msg
```

```
class TransitionError(Error):
    """Raised when an operation attempts a state transition that's not
    allowed.

    Attributes:
        prev -- state at beginning of transition
        next -- attempted new state
        msg  -- explanation of why the specific transition is not allowed
    """

    def __init__(self, prev, next, msg):
        self.prev = prev
        self.next = next
        self.msg = msg
```

8.6 Defining Clean-up Actions 定义清理行为

try

try

```
>>> try:
...     raise KeyboardInterrupt
... finally:
...     print 'Goodbye, world!'
...
Goodbye, world!
KeyboardInterrupt
```

try
try
except except else
finally finally
try break continue return
except finally try

try try
except except else finally
try break return finally
try except finally

```
>>> def divide(x, y):
...     try:
...         result = x / y
...     except ZeroDivisionError:
...         print "division by zero!"
...     else:
...         print "result is", result
...     finally:
...         print "executing finally clause"
...
>>> divide(2, 1)
result is 2
executing finally clause
>>> divide(2, 0)
division by zero!
executing finally clause
>>> divide("2", "1")
executing finally clause
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "<stdin>", line 3, in divide
TypeError: unsupported operand type(s) for /: 'str' and 'str'
```

finally TypeError
except finally

finally TypeError
except finally
finally

finally

8.7 Predefined Clean-up Actions 预定义清理行为

```
for line in open("myfile.txt"):
    print line
```

with

with

```python
with open("myfile.txt") as f:
    for line in f:
        print line
```

CHAPTER

NINE

CLASSES 类

9.1 A Word About Names and Objects 关于命名和对象的内容

9.2 Python Scopes and Namespaces Python 作用域和命名空间

abs()

maximize

abs()

maximize

z.real real z

modname.funcname modname

funcname

__dict__

__dict__

`z.real` `real`
`z` `modname.funcname` `modname`

`modname.the_answer = 42`
`del` `del modname.the_answer`
`the_answer` `modname`

`modname.the_answer = 42` `del` `del modname.the_answer` `modname`
`the_answer`

`__main__`
`__builtin__`

`__main__`
`__builtin__`

-
-

`__dict__` `__dict__`

-
-

`global`

`del x` `x`

`import`
`global`

`global`
`del x`
`x` `import`
`global`

9.3 A First Look at Classes 初识类

9.3.1 Class Definition Syntax 类定义语法

```
class ClassName:
    <statement-1>
    .
    .
    .
    <statement-N>
```

def

if

def

if

ClassName

ClassName

9.3.2 Class Objects 类对象

obj.name

```
class MyClass:
    """A simple example class"""
    i = 12345
    def f(self):
        return 'hello world'
```

MyClass.i MyClass.f

MyClass.i __doc__
"A simple example class"

MyClass.i MyClass.f
MyClass.i __doc__`
"A simple example class"

```
x = MyClass()
```

x

x

__init__()

__init__()

```
def __init__(self):
    self.data = []
```

__init__() __init__()

__init__() __init__`()

```
x = MyClass()
```

__init__()
__init__()

__init__() __init__`()

```
>>> class Complex:
...     def __init__(self, realpart, imagpart):
...         self.r = realpart
...         self.i = imagpart
...
>>> x = Complex(3.0, -4.5)
>>> x.r, x.i
(3.0, -4.5)
```

9.3.3 Instance Objects 实例对象

x MyClass
16

MyClass
16

```
x.counter = 1
while x.counter < 10:
    x.counter = x.counter * 2
print x.counter
del x.counter
```

x.f
MyClass.f x.i MyClass.i
x.f MyClass.f

MyClass.f x.i
MyClass.i x.f MyClass.f

9.3.4 Method Objects 方法对象

```
x.f()
```

MyClass 'hello world'
x.f

MyClass 'hello world' x.f

```
xf = x.f
while True:
    print xf()
```

hello world

x.f()

f()

x.f() f()

x.f()

MyClass.f(x)

x.f() MyClass.f(x)

9.4 Random Remarks 一些说明

self
self

self self

```python
# Function defined outside the class
def f1(self, x, y):
    return min(x, x+y)

class C:
    f = f1
    def g(self):
        return 'hello world'
    h = g
```

f g h C
C h g

f g h C C h
g

self

self

```
class Bag:
    def __init__(self):
        self.data = []
    def add(self, x):
        self.data.append(x)
    def addtwice(self, x):
        self.add(x)
        self.add(x)
```

object.__class__

object.__class__

9.5 Inheritance 继承

```
class DerivedClassName(BaseClassName):
    <statement-1>
    .
    .
    .
    <statement-N>
```

BaseClassName

BaseClassName

```
class DerivedClassName(modname.BaseClassName):
```

DerivedClassName()

DerivedClassName()

virtual

BaseClassName.methodname(self, arguments)
BaseClassName

BaseClassName.methodname(self, arguments)
BaseClassName

- isinstance() isinstance(obj, int) True obj.__class__ int int

  isinstance() isinstance(obj, int) obj.__class__ int int

- issubclass() issubclass(bool, int) True bool int issubclass(unicode, str) False unicode str basestring

  issubclass() issubclass(bool, int) True bool int issubclass(unicode, str) False unicode str basestring

9.5.1 Multiple Inheritance 多继承

```
class DerivedClassName(Base1, Base2, Base3):
    <statement-1>
    .
    .
    .
    <statement-N>
```

DerivedClassName Base1
Base1 Base2
DerivedClassName
Base1
Base2
Base2 Base3 Base1

Base1 Base1
Base2
Base1

Base1 Base3
Base1 Base2 Base1 Base1

super()

super()

object
object

object
object

9.6 Private Variables 私有变量

_spam

_spam

__spam
_classname__spam classname

__spam
_classname__spam classname

exec eval() execfile()
global
getattr() setattr() delattr() __dict__

exec eval() execfile()
global getattr()
setattr() delattr __dict__

9.7 Odds and Ends 补充

```
class Employee:
    pass

john = Employee() # Create an empty employee record

# Fill the fields of the record
john.name = 'John Doe'
john.dept = 'computer lab'
john.salary = 1000
```

read() readline()

read()
readline()

m.im_self

m() m.im_func

m.im_self m.im_func

9.8 Exceptions Are Classes Too 异常也是类

raise

raise

```
raise Class, instance

raise instance
```

instance Class

instance Class

```
raise instance.__class__, instance
```

except

except

```
class B:
    pass
class C(B):
    pass
class D(C):
    pass

for c in [B, C, D]:
    try:
        raise c()
    except D:
        print "D"
    except C:
        print "C"
    except B:
        print "B"
```

except B

execpt B

`str()`

`str()`

9.9 Iterators 迭代器

`for`

`for`

```
for element in [1, 2, 3]:
    print element
for element in (1, 2, 3):
    print element
for key in {'one':1, 'two':2}:
    print key
for char in "123":
    print char
for line in open("myfile.txt"):
    print line
```

`for` `iter()`
`next()`
`next()` `StopIteration`
`for`

`for`
`iter()` `next()`
`next()` `StopIteration` `for`

```
>>> s = 'abc'
>>> it = iter(s)
>>> it
<iterator object at 0x00A1DB50>
>>> it.next()
'a'
>>> it.next()
'b'
>>> it.next()
'c'
>>> it.next()

Traceback (most recent call last):
  File "<stdin>", line 1, in ?
    it.next()
StopIteration
```

`__iter__()` `next()`
`next()` `__iter__()` `self`

__iter__()
next() next() __iter__()
self

```
class Reverse:
    "Iterator for looping over a sequence backwards"
    def __init__(self, data):
        self.data = data
        self.index = len(data)
    def __iter__(self):
        return self
    def next(self):
        if self.index == 0:
            raise StopIteration
        self.index = self.index - 1
        return self.data[self.index]

>>> for char in Reverse('spam'):
...     print char
...
m
a
p
s
```

9.10 Generators 生成器

yield next()

yield next()

```
def reverse(data):
    for index in range(len(data)-1, -1, -1):
        yield data[index]

>>> for char in reverse('golf'):
...     print char
...
f
l
o
g
```

__iter__() next()

__iter__()
next()

self.index self.data

self.index self.data

StopIteration

StopIteration

9.11 Generator Expressions 生成器表达式

```
>>> sum(i*i for i in range(10))                 # sum of squares
285

>>> xvec = [10, 20, 30]
>>> yvec = [7, 5, 3]
>>> sum(x*y for x,y in zip(xvec, yvec))         # dot product
260

>>> from math import pi, sin
>>> sine_table = dict((x, sin(x*pi/180)) for x in range(0, 91))

>>> unique_words = set(word  for line in page  for word in line.split())

>>> valedictorian = max((student.gpa, student.name) for student in graduates)

>>> data = 'golf'
>>> list(data[i] for i in range(len(data)-1,-1,-1))
['f', 'l', 'o', 'g']
```

CHAPTER

TEN

BRIEF TOUR OF THE STANDARD LIBRARY 标准库概览

10.1 Operating System Interface 操作系统接口

os

os

```
>>> import os
>>> os.system('time 0:02')
0
>>> os.getcwd()      # Return the current working directory
'C:\\Python26'
>>> os.chdir('/server/accesslogs')
```

import os from os import * os.open() open()

import os from os import * os.open() open() dir() help() os

os dir() help()

```
>>> import os
>>> dir(os)
<returns a list of all module functions>
>>> help(os)
<returns an extensive manual page created from the module's docstrings>
```

shutil

```
>>> import shutil
>>> shutil.copyfile('data.db', 'archive.db')
>>> shutil.move('/build/executables', 'installdir')
```

10.2 File Wildcards 文件通配符

glob

glob

```
>>> import glob
>>> glob.glob('*.py')
['primes.py', 'random.py', 'quote.py']
```

10.3 Command Line Arguments 命令行参数

sys
python demo.py one two three

sys

python demo.py one two three

```
>>> import sys
>>> print sys.argv
['demo.py', 'one', 'two', 'three']
```

getopt getopt()
argparse

getopt getopt() optparse

10.4 Error Output Redirection and Program Termination 错误输出重定向和程序终止

sys

sys

```
>>> sys.stderr.write('Warning, log file not found starting a new one\n')
Warning, log file not found starting a new one
```

sys.exit()

sys.exit()

10.5 String Pattern Matching 字符串正则匹配

re

re

```
>>> import re
>>> re.findall(r'\bf[a-z]*', 'which foot or hand fell fastest')
['foot', 'fell', 'fastest']
>>> re.sub(r'(\b[a-z]+) \1', r'\1', 'cat in the the hat')
'cat in the hat'
```

```
>>> 'tea for too'.replace('too', 'two')
'tea for two'
```

10.6 Mathematics 数学

math

math

```
>>> import math
>>> math.cos(math.pi / 4.0)
0.70710678118654757
>>> math.log(1024, 2)
10.0
```

random

random

```
>>> import random
>>> random.choice(['apple', 'pear', 'banana'])
'apple'
>>> random.sample(xrange(100), 10)   # sampling without replacement
[30, 83, 16, 4, 8, 81, 41, 50, 18, 33]
>>> random.random()    # random float
0.17970987693706186
>>> random.randrange(6)    # random integer chosen from range(6)
4
```

10.7 Internet Access 互联网访问

urllib2 smtplib

urls

urllib2 smtplib

```
>>> import urllib2
>>> for line in urllib2.urlopen('http://tycho.usno.navy.mil/cgi-bin/timer.pl'):
...     if 'EST' in line or 'EDT' in line:  # look for Eastern Time
...         print line

<BR>Nov. 25, 09:43:32 PM EST

>>> import smtplib
>>> server = smtplib.SMTP('localhost')
>>> server.sendmail('soothsayer@example.org', 'jcaesar@example.org',
... """To: jcaesar@example.org
... From: soothsayer@example.org
...
... Beware the Ides of March.
... """)
>>> server.quit()
```

10.8 Dates and Times 日期和时间

datetime

datetime

```
>>> # dates are easily constructed and formatted
>>> from datetime import date
>>> now = date.today()
>>> now
datetime.date(2003, 12, 2)
>>> now.strftime("%m-%d-%y. %d %b %Y is a %A on the %d day of %B.")
'12-02-03. 02 Dec 2003 is a Tuesday on the 02 day of December.'

>>> # dates support calendar arithmetic
>>> birthday = date(1964, 7, 31)
>>> age = now - birthday
>>> age.days
14368
```

10.9 Data Compression 数据压缩

zlib
gzip bz2 zipfile tarfile

zlib gzip bz2 zipfile tarfile

```
>>> import zlib
>>> s = 'witch which has which witches wrist watch'
>>> len(s)
41
>>> t = zlib.compress(s)
>>> len(t)
37
>>> zlib.decompress(t)
'witch which has which witches wrist watch'
>>> zlib.crc32(s)
226805979
```

10.10 Performance Measurement 性能度量

timeit

timeit

```
>>> from timeit import Timer
>>> Timer('t=a; a=b; b=t', 'a=1; b=2').timeit()
0.57535828626024577
>>> Timer('a,b = b,a', 'a=1; b=2').timeit()
0.54962537085770791
```

timeit profile pstats

timeit pstats

10.11 Quality Control 质量控制

doctest

doctest

```
def average(values):
    """Computes the arithmetic mean of a list of numbers.

    >>> print average([20, 30, 70])
    40.0
    """
    return sum(values, 0.0) / len(values)

import doctest
doctest.testmod()   # automatically validate the embedded tests
```

unittest doctest

unittest doctest

```
import unittest

class TestStatisticalFunctions(unittest.TestCase):

    def test_average(self):
        self.assertEqual(average([20, 30, 70]), 40.0)
        self.assertEqual(round(average([1, 5, 7]), 1), 4.3)
        self.assertRaises(ZeroDivisionError, average, [])
        self.assertRaises(TypeError, average, 20, 30, 70)

unittest.main() # Calling from the command line invokes all tests
```

10.12 Batteries Included 电池已备

- xmlrpclib SimpleXMLRPCServer

  xmlrpclib SimpleXMLRPCServer

- email smtplib poplib

  email smtplib poplib

- xml.dom xml.sax csv

  xml.dom xml.sax csv

- gettext locale codecs gettext locale codecs

CHAPTER

ELEVEN

BRIEF TOUR OF THE STANDARD LIBRARY -- PART II 标准库概览 II

11.1 Output Formatting 输出格式

repr repr()

repr repr()

```
>>> import repr
>>> repr.repr(set('supercalifragilisticexpialidocious'))
"set(['a', 'c', 'd', 'e', 'f', 'g', ...])"
```

pprint

pprint

```
>>> import pprint
>>> t = [[[['black', 'cyan'], 'white', ['green', 'red']], [['magenta',
...     'yellow'], 'blue']]]
...
>>> pprint.pprint(t, width=30)
[[[['black', 'cyan'],
   'white',
   ['green', 'red']],
  [['magenta', 'yellow'],
   'blue']]]
```

textwrap

textwrap

```
>>> import textwrap
>>> doc = """The wrap() method is just like fill() except that it returns
```

```
... a list of strings instead of one big string with newlines to separate
... the wrapped lines."""
...
>>> print textwrap.fill(doc, width=40)
The wrap() method is just like fill()
except that it returns a list of strings
instead of one big string with newlines
to separate the wrapped lines.
```

locale

locale

```
>>> import locale
>>> locale.setlocale(locale.LC_ALL, 'English_United States.1252')
'English_United States.1252'
>>> conv = locale.localeconv()          # get a mapping of conventions
>>> x = 1234567.8
>>> locale.format("%d", x, grouping=True)
'1,234,567'
>>> locale.format_string("%s%.*f", (conv['currency_symbol'],
...                      conv['frac_digits'], x), grouping=True)
'$1,234,567.80'
```

11.2 Templating 模板

string Template

string template

$

$$ $

$

$$ $

```
>>> from string import Template
>>> t = Template('${village}folk send $$10 to $cause.')
>>> t.substitute(village='Nottingham', cause='the ditch fund')
'Nottinghamfolk send $10 to the ditch fund.'
```

substitute() KeyError

safe_substitute()

substitute() KeyError
safe-substitute()

```
>>> t = Template('Return the $item to $owner.')
>>> d = dict(item='unladen swallow')
>>> t.substitute(d)
Traceback (most recent call last):
  . . .
KeyError: 'owner'
>>> t.safe_substitute(d)
'Return the unladen swallow to $owner.'
```

```
>>> import time, os.path
>>> photofiles = ['img_1074.jpg', 'img_1076.jpg', 'img_1077.jpg']
>>> class BatchRename(Template):
...     delimiter = '%'
>>> fmt = raw_input('Enter rename style (%d-date %n-seqnum %f-format):  ')
Enter rename style (%d-date %n-seqnum %f-format):  Ashley_%n%f

>>> t = BatchRename(fmt)
>>> date = time.strftime('%d%b%y')
>>> for i, filename in enumerate(photofiles):
...     base, ext = os.path.splitext(filename)
...     newname = t.substitute(d=date, n=i, f=ext)
...     print '{0} --> {1}'.format(filename, newname)

img_1074.jpg --> Ashley_0.jpg
img_1076.jpg --> Ashley_1.jpg
img_1077.jpg --> Ashley_2.jpg
```

11.3 Working with Binary Data Record Layouts 使用二进制记录层

struct pack() unpack()

zipfile "H" "I"
"<"

struct pack() unpack()
"H" "L" "<"

```
import struct

data = open('myfile.zip', 'rb').read()
start = 0
```

```
for i in range(3):                      # show the first 3 file headers
    start += 14
    fields = struct.unpack('<IIIHH', data[start:start+16])
    crc32, comp_size, uncomp_size, filenamesize, extra_size = fields

    start += 16
    filename = data[start:start+filenamesize]
    start += filenamesize
    extra = data[start:start+extra_size]
    print filename, hex(crc32), comp_size, uncomp_size

    start += extra_size + comp_size     # skip to the next header
```

11.4 Multi-threading 多线程

threading

threading

```
import threading, zipfile

class AsyncZip(threading.Thread):
    def __init__(self, infile, outfile):
        threading.Thread.__init__(self)
        self.infile = infile
        self.outfile = outfile
    def run(self):
        f = zipfile.ZipFile(self.outfile, 'w', zipfile.ZIP_DEFLATED)
        f.write(self.infile)
        f.close()
        print 'Finished background zip of: ', self.infile

background = AsyncZip('mydata.txt', 'myarchive.zip')
background.start()
print 'The main program continues to run in foreground.'

background.join()    # Wait for the background task to finish
print 'Main program waited until background was done.'
```

Queue
Queue.Queue

Queue
Queue.Queue

11.5 Logging 日志

logging
sys.stderr

logging sys.stderr

```
import logging
logging.debug('Debugging information')
logging.info('Informational message')
logging.warning('Warning:config file %s not found', 'server.conf')
logging.error('Error occurred')
logging.critical('Critical error -- shutting down')
```

```
WARNING:root:Warning:config file server.conf not found
ERROR:root:Error occurred
CRITICAL:root:Critical error -- shutting down
```

DEBUG
INFO WARNING ERROR CRITICAL

DEBUG INFO
WARNING ERROR CRITICAL

11.6 Weak References 弱引用

weakref

weakref

```
>>> import weakref, gc
>>> class A:
...     def __init__(self, value):
...             self.value = value
...     def __repr__(self):
...             return str(self.value)
...
>>> a = A(10)                   # create a reference
>>> d = weakref.WeakValueDictionary()
>>> d['primary'] = a            # does not create a reference
>>> d['primary']                # fetch the object if it is still alive
10
>>> del a                       # remove the one reference
>>> gc.collect()                # run garbage collection right away
0
>>> d['primary']                # entry was automatically removed
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    d['primary']                # entry was automatically removed
  File "C:/python26/lib/weakref.py", line 46, in __getitem__
    o = self.data[key]()
KeyError: 'primary'
```

11.7 Tools for Working with Lists 列表工具

array array()

"H"

array array "H"

```
>>> from array import array
>>> a = array('H', [4000, 10, 700, 22222])
>>> sum(a)
26932
>>> a[1:3]
array('H', [10, 700])
```

collections deque()

collections deque()

```
>>> from collections import deque
>>> d = deque(["task1", "task2", "task3"])
>>> d.append("task4")
>>> print "Handling", d.popleft()
Handling task1
```

```
unsearched = deque([starting_node])
def breadth_first_search(unsearched):
    node = unsearched.popleft()
    for m in gen_moves(node):
        if is_goal(m):
            return m
        unsearched.append(m)
```

bisect

bisect

```
>>> import bisect
>>> scores = [(100, 'perl'), (200, 'tcl'), (400, 'lua'), (500, 'python')]
>>> bisect.insort(scores, (300, 'ruby'))
>>> scores
[(100, 'perl'), (200, 'tcl'), (300, 'ruby'), (400, 'lua'), (500, 'python')]
```

heapq

heapq

```
>>> from heapq import heapify, heappop, heappush
>>> data = [1, 3, 5, 7, 9, 2, 4, 6, 8, 0]
>>> heapify(data)                      # rearrange the list into heap order
>>> heappush(data, -5)                 # add a new entry
>>> [heappop(data) for i in range(3)]  # fetch the three smallest entries
[-5, 0, 1]
```

11.8 Decimal Floating Point Arithmetic 十进制浮点数算法

decimal Decimal float

decimal Decimal float

-

-

-
-
-

```
>>> from decimal import *
>>> x = Decimal('0.70') * Decimal('1.05')
>>> x
Decimal('0.7350')
>>> x.quantize(Decimal('0.01'))  # round to nearest cent
Decimal('0.74')
>>> round(.70 * 1.05, 2)         # same calculation with floats
0.73
```

Decimal

Decimal

Decimal

Decimal

```
>>> Decimal('1.00') % Decimal('.10')
Decimal('0.00')
>>> 1.00 % 0.10
0.09999999999999995

>>> sum([Decimal('0.1')]*10) == Decimal('1.0')
True
>>> sum([0.1]*10) == 1.0
False
```

decimal

decimal

```
>>> getcontext().prec = 36
>>> Decimal(1) / Decimal(7)
Decimal('0.142857142857142857142857142857142857')
```

CHAPTER

TWELVE

WHAT NOW? 接下来?

•

•

•

•

•

-

-

comp.lang.python

< >
Misc/

comp.lang.python

< >
Misc/

CHAPTER

THIRTEEN

INTERACTIVE INPUT EDITING AND HISTORY SUBSTITUTION

13.1 Line Editing 行编辑

C-A
C-E C-B C-F
C-D C-K
C-Y C-
underscore

C-A C-B
C-F C-D C-K
C-Y C-underscore

13.2 History Substitution 历史回溯

C-P C-N

Return C-R
C-S

C-P C-N
C-R

13.3 Key Bindings 快捷键绑定

~/.inputrc
~/.inputrc

```
key-name: function-name
```

```
"string": function-name
```

```
set option-name value
```

```
# I prefer vi-style editing:
set editing-mode vi

# Edit using a single line:
set horizontal-scroll-mode On

# Rebind some keys:
Meta-h: backward-kill-word
"\C-u": universal-argument
"\C-x\C-r": re-read-init-file
```

Tab Tab

Tab Tab

Tab: complete

~/.inputrc
Tab

~/.inputrc Tab

```
import rlcompleter, readline
readline.parse_and_bind('tab: complete')
```

Tab Tab

string.a
'.'
__getattr__()

Tab Tab
string.a '.'
__getattr__()

os

os

```
# Add auto-completion and a stored history file of commands to your Python
# interactive interpreter. Requires Python 2.0+, readline. Autocomplete is
# bound to the Esc key by default (you can change it - see readline docs).
#
# Store the file in ~/.pystartup, and set an environment variable to point
# to it:  "export PYTHONSTARTUP=/home/user/.pystartup" in bash.
#
# Note that PYTHONSTARTUP does *not* expand "~", so you have to put in the
# full path to your home directory.

import atexit
import os
import readline
import rlcompleter

historyPath = os.path.expanduser("~/.pyhistory")

def save_history(historyPath=historyPath):
```

PYTHONSTARTUP

PYTHONSTARTUP

```
    import readline
    readline.write_history_file(historyPath)

if os.path.exists(historyPath):
    readline.read_history_file(historyPath)

atexit.register(save_history)
del os, atexit, readline, rlcompleter, save_history, historyPath
```

13.4 Alternatives to the Interactive Interpreter 其它交互式解释器

CHAPTER

FOURTEEN

FLOATING POINT ARITHMETIC: ISSUES AND LIMITATIONS 浮点数算法：争议和限制

`0.125`

`0.001`

`0.3`

`0.33`

`0.333`

```
0.0001100110011001100110011001100110011001100110011...
```

```
>>> 0.1
0.10000000000000001
```

```
>>> 0.1
0.1000000000000000055511151231257827021181583404541015625
```

`repr()`

`repr(float)`

`repr()` `repr(float)`

`0.10000000000000001`

`repr(float)`
`eval(repr(x)) == x`

`repr(float)` `eval(repr(x)) == x`

`str()` `eval(str(x))`

`str()` `eval(str(x))`

```
>>> print str(0.1)
0.1
```

```
>>> 0.1
0.10000000000000001
```

`round()`

```
>>> round(0.1, 1)
0.10000000000000001
```

```
>>> sum = 0.0
>>> for i in range(10):
...     sum += 0.1
...
>>> sum
0.9999999999999999
```

< >

str()
str.format()

str() str.format()

14.1 Representation Error 表达错误

```
>>> 0.1
0.10000000000000001
```

```
1 / 10 ~= J / (2**N)
```

```
J ~= 2**N / 10
```

>= 2**52 < 2**53

>= 2**52 < 2**53

```
>>> 2**52
4503599627370496L
>>> 2**53
9007199254740992L
```

```
72057594037[redacted]7793L

>>> q, r =
```