Python 快速入门

张果

厦门大学　王亚南经济研究院

本版本：2017 年 2 月 13 日

目录

目录

什么是程序?

- 程序：输入 $\rightarrow$ 输出
- 程序 $=$ 数据结构 $+$ 算法
  - 数据结构：储存、组织数据
  - 算法：具体步骤
- 程序：实现需求，解决问题

为什么选择 Python?

- 易学易用，开发效率高
- 强大的胶水

如何选择 Python 版本?

- 2.x vs 3.x?
  - 为什么 3.x 无法完全取代 2.x? 没有真正解决 2.x 的问题。
  - 2.x 和 3.x 互相迁移很困难? 不是。
- 2.x: 选择 2.6 以上版本
- 3.x: 选择 3.4 以上版本
- 本教程使用 2.7.y(y>=9)

Python 开发环境

- Python 官方工具
- PyCharm：主流 IDE
- Anaconda：科学计算专用
- Jupyter：浏览器版 iPython
- Sublime Text 3：可拓展性极强的文本编辑器

目录

Python 解释器

```
1 >>> # 输出 0 到 9
2 >>> for i in range(10):
3 ...     print i,
4 ...
5 0 1 2 3 4 5 6 7 8 9
```

- >>> # 主提示符
- ... # 次提示符
- # # 注释

语句与表达式

- 语句： **print** 'Hello, world'
- 表达式： **abs**(-4)

帮助

```
1 >>> help(input)
2 Help on built-in function input in module
      __builtin__:
3
4 input(...)
5
6     input([prompt]) -> value
7
8     Equivalent to eval(raw_input(prompt)).
```

帮助

```
>>> help(print)
  File "<stdin>", line 1
    help(print)
         ^
SyntaxError: invalid syntax
```

操作符

- 算术操作符：+,-,*,/,//,%,**

```
1 >>> 1/2,1//2,1.0/2,1.0//2
2 (0, 0, 0.5, 0.0)
3 >>> 4**2
4 16
5 >>> 7%3
6 1
```

- 比较操作符：<,<=,>,>=,==,!=
- 逻辑操作符：**and**,**or**,**not**

变量与赋值

```
1 >>> counter = 0
2 >>> # 给变量 counter 赋值 0
```

目录

目录

输出

- 交互式解释器

```
>>> 'Hello, world'
'Hello, world'
>>> _  # 最后一个表达式的值
'Hello, world'
```

- print 语句

```
>>> print 'Hello, world'
Hello, world
```

输出

注释：

- **print**调用 Python 对象的__str__()方法;
- 交互式解释器调用__repr__()方法。

格式化输出

```
1 >>> print "%d, my favorate %s, but %f" %(1,'Python',1.1)
2 1, my favorate Python, but 1.100000
```

- %d: 整型
- %f: 浮点型
- %s: 字符串

重定向到日志

```
1 log = open('C:/test.log','a')
2 print >> log,'My favorate food'
3 print >> log,'My favorate drink'
4 log.close()
```

目录

input

```
1 >>> input('please␣input␣your␣favorate␣number:'
      )
2 please input your favorate number:7
3 7
4 >>> type(_)
5 <type 'int'>
```

raw_input

```
1 >>> raw_input('please input your favorate
      number:')
2 please input your favorate number:7
3 '7'
4 >>> type(_)
5 <type 'str'>
```

目录

目录

数字

- 整型
  - 有符号整型 (**int**): 1
  - 长整型 (**long**): 100000L,100000l
  - 布尔值 (**bool**): True，False
- 浮点型 (**float**): 1.1
- 复数 (**complex**): 1+1j,1+1J

数字

```
1  >>> type(1)
2  <type 'int'>
3  >>> type(100000L)
4  <type 'long'>
5  >>> type(True)
6  <type 'bool'>
7  >>> type(1.1)
8  <type 'float'>
9  >>> type(1+1j)
10 <type 'complex'>
```

目录

序列 (senquence)

- 列表 (**list**): [1,3,4]
- 元组 (**tuple**): (1,3,4,5)
- 字符串 (**str**): 'Hello,world',"Hello,world"

序列 (senquence)

```
1 >>> type([1,3,4])
2 <type 'list'>
3 >>> type((1,3,4,5))
4 <type 'tuple'>
5 >>> type('hello,world')
6 <type 'str'>
```

序列 (senquence)

```
1  >>> # 创建序列
2  >>> aList = [1,2,3,4]
3  >>> aTuple = (1,2,3,4)
4  >>> aStr = 'hello,world'
5  >>> # 索引
6  >>> aList[0]
7  1
8  >>> aTuple[3]
9  4
10 >>> aStr[4]
11 'o'
```

序列 (senquence)

```
12 >>> # 切片
13 >>> aList[0:2]
14 [1, 2]
15 >>> aTuple[:3]
16 (1, 2, 3)
17 >>> aStr[4:]
18 'o,world'
```

目录

集合和映射类型

- 集合 (**set**): {1,5,6}
- 字典 (**dict**): {'xiaoming':1,'xiaohong':2}

集合和映射类型

```
1 >>> type({1,5,6})
2 <type 'set'>
3 >>> type({'xiaoming':1,'xiaohong':2})
4 <type 'dict'>
```

字典

```
1 >>> aDict = {'x':1}  # 创建字典
2 >>> aDict['y']=2  # 添加键值对
3 >>> aDict
4 {'y': 2, 'x': 1}
5 >>> aDict['x']  # 索引字典
6 1
```

字典

```
1 >>> aDict.keys()  # 字典的键
2 ['y', 'x']
3 >>> aDict.values()  # 字典的值
4 [2, 1]
5 >>> aDict.items()  # 键值对
6 [('y', 2), ('x', 1)]
```

目录

目录

if

```
1 >>> a = 4
2 >>> if a<10:
3 ...     print 'Hello, world'
4 ...
5 Hello, world
```

注释：代码块**缩进**表达逻辑（一般是四个空格）

if-else

```
1 >>> a = 4
2 >>> if a<10:
3 ...     print 'Hello,world'
4 ... else:
5 ...     print 'Hello'
6 ...
7 Hello,world
```

if-elif-else

```
1 >>> a = 4
2 >>> if a<10:
3 ...     print 'Hello,world'
4 ... elif a<15:
5 ...     print 'Hello?'
6 ... else:
7 ...     print 'Hello'
8 ...
9 Hello,world
```

目录

迭代器循环

```
1 >>> for item in {'my','your','their','his','
      her'}:
2 ...     print item,
3 ...
4 their his my your her
```

注释：list,tuple,set,str 等都是可迭代对象

用 range 实现计数器循环

```
>>> for i in range(10):
...     print i,
...
0 1 2 3 4 5 6 7 8 9
```

enumerate

```
>>> foo = 'abcdefglijklmn'
>>> for i,ch in enumerate(foo):
...         print ch,'(%d)'%i,',',
...
a (0) , b (1) , c (2) , d (3) , e (4) , f (5)
    , g (6) , l (7) , i (8) , j (9) , k (10) ,
    l (11) , m (12) , n (13) ,
```

zip

```
1 >>> a = [1,2,3]
2 >>> b = (2,3,4)
3 >>> for i,j in zip(a,b):
4 ...     print i,j
5 ...
6 1 2
7 2 3
8 3 4
```

列表解析

```
>>> [x**2 for x in range(4)]
[0, 1, 4, 9]
>>> [x**2 for x in range(10) if ((x**2)%2)]
[1, 9, 25, 49, 81]
```

字典解析

```
1 >>> {x:x**2 for x in range(4)}
2 {0: 0, 1: 1, 2: 4, 3: 9}
```

集合解析

```
1 >>> {x**2 for x in range(4)}
2 set([0, 1, 4, 9])
```

生成器 (generator)

```
1 >>> (x**2 for x in range(4))
2 <generator object <genexpr> at 0
     x0000000002D25678>
3 >>> for i in _:
4 ...     print i,
5 ...
6 0 1 4 9
```

目录

计数器循环

```
1 >>> a = 0
2 >>> while a<10:
3 ...     print a,
4 ...     a = a + 1
```

无限循环

```
1 >>> a = 0
2 >>> while True:
3 ...     print a,
4 ...     a = a + 1
```

注意：慎用。一般在服务器程序等类似情况下中使用。

目录

break 和 continue

- **break**: 结束当前循环，跳转到下个语句
- **continue**；终止当前循环，忽略剩余语句，回到循环顶端

```
1 >>> a = 0
2 >>> while a<100:
3 ...     a = a + 1
4 ...     if not a%5:
5 ...         continue
6 ...     print a,
7 ...     if a == 97:
8 ...         break
```

break 和 continue

```
 9  ...
10  1  2  3  4  6  7  8  9  11  12  13  14  16  17  18  19  21  22
       23  24  26  27  28  29  31  32  33  34  36
11  37  38  39  41  42  43  44  46  47  48  49  51  52  53  54
       56  57  58  59  61  62  63  64  66  67  68  69
12  71  72  73  74  76  77  78  79  81  82  83  84  86  87  88
       89  91  92  93  94  96  97
```

目录

pass 语句

- 功能：不做任何事情
  - 开发和调试
  - 异常处理

```
1 >>> for i in range(10):
2 ...     pass
3 ...
4 >>>
```

目录

目录

写入文件

```
1 >>> import os
2 >>> os.getcwd()  # get current working
      dictionary
3 >>> f = open('test.txt','w')
4 >>> f.write('Hello,world')
5 >>> f.close()
```

读取文件

```
1 >> import os
2 >> os.getcwd()
3 >> f = open('test.txt','r')
4 >> f.read()
5 >> f.close()
```

目录

os 模块: 文件处理

- os.remove(fname) # 删除文件
- os.rename(oldname,newname) # 重命名/移动文件
- os.walk(path) # 生成一个目录树下所有文件

os 模块: 目录和文件夹

- os.chdir(path) # 改变当前工作目录
- os. listdir (path) # 列出指定目录的文件名
- os.getcwd() # 返回当前工作目录
- os.mkdir(path) # 创建目录

os.path 模块: 路径名分隔

- os.path.basename(fname) # 去掉目录路径，返回文件名
- os.path.dirname(fname) # 去掉文件名，返回目录路径
- os.path.join(dirname,basename) # 组合目录名和文件名
- os.path.split(fname) # 拆分目录名和文件名
- os.path.splitext(path) # 拆分路径名和扩展名

os.path 模块: 路径名信息

- os.path. exists (path) # 指定路径（文件 or 目录）
- os.path. isfile (fname) 指定路径是否存在且为一个文件
- os.path. isdir (dirname) 指定路径是否存在且为一个目录

目录

try-except 语句

```
1 >>> try:
2 ...     open('test.txt')
3 ... except IOError:
4 ...     print 'No such file'
5 ...
6 No such file
```

目录

目录

调用函数

- 帮助：dir, help
- 工厂函数：int, float, str, dict, tuple, list, set, file
- 长度：len
- 类型：type
- 排序：sort, reserved
- 数学：max, min, abs

目录

定义函数

```
1 >>> def test(arg):
2 ...     return arg*2-4
3 ...
4 >>> test(4)
5 4
```

目录

函数的参数 *

- 默认参数
- 可变参数
- 关键词参数

参考资料：廖雪峰 Python2.7 教程：函数的参数

目录

目录

定义类 *

```
1 >>> class Test(object):
2 ...     def __init__(self,myname):
3 ...         self.name = myname
4 ...     def __str__(self):
5 ...         return self.name*2
6 ...     def __repr__(self):
7 ...         return self.name*3
8 ...
9 >>>
```

目录

```
1 >>> Test('Bob')
2 BobBobBob
3 >>> print Test('Bob')
4 BobBob
```

目录

目录

导入和使用模块

```
1 >>> import string
2 >>> dir(string)
3 ['Formatter', 'Template', '_TemplateMetaclass'
      , '__builtins__', '__doc__', '__fi
4 le__', '__name__', '__package__', '_float', '
      _idmap', '_idmapL', '_int', '_long'
5 , '_multimap', '_re', 'ascii_letters', '
      ascii_lowercase', 'ascii_uppercase', 'at
6 of', 'atof_error', 'atoi', 'atoi_error', 'atol
      ', 'atol_error', 'capitalize', 'ca
```

导入和使用模块

```
7  pwords', 'center', 'count', 'digits', '
       expandtabs', 'find', 'hexdigits', 'index'
8  , 'index_error', 'join', 'joinfields', '
       letters', 'ljust', 'lower', 'lowercase',
9   'lstrip', 'maketrans', 'octdigits', '
        printable', 'punctuation', 'replace', 'rfi
10 nd', 'rindex', 'rjust', 'rsplit', 'rstrip', '
       split', 'splitfields', 'strip', 'sw
11 apcase', 'translate', 'upper', 'uppercase', '
       whitespace', 'zfill']
12 >>> string.uppercase
```

导入和使用模块

```
13  'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
14 >>> string.lowercase
15  'abcdefghijklmnopqrstuvwxyz'
16 >>> string.whitespace
17  '\t\n\x0b\x0c\r '
18 >>> string.upper
19 <function upper at 0x0000000002BB8278>
20 >>> string.upper('agesrag')
21  'AGESRAG'
```

目录

下载第三方模块

- 在命令行输入:
  - pip install <package>
  - easy_insall <package>
- 常见问题:
  - pip 不能使用
  - C 扩展库不能安装

目录