面向物理学生的 Python 入门

Python 辅助科学计算与数据处理

吕铭, 李厚辰

2015 年 5 月 7 日

清华大学物理系学生科协

准备工作

- Python 的安装
- 打开命令行工具 (cmd, PowerShell, Terminal)
- 测试: *python [path]test.py*
- 准备好 Python shell:
  在命令行工具里输入 Python 或打开 IDLE
- Let's Start

目录

Python 是什么

Python 是什么

- ~~蟒蛇~~
- 面向对象 (~~C++~~)
- 通常解释执行 (~~编译器~~ 解释器)
- 脚本语言 / 高级动态编程语言
- 1989, Guido van Rossum
- ~~Python 2.7~~; Python 3
- .py; .pyc; .pyo; .pyd
- The zen of Python

```
import this
```

为什么 Python

Bruce Eckel:

> *Life is short, you need Python!*

Linus Torvalds:

> *Talk is cheap. Show me the code.*

基础语法

第一个程序 Hello World

Hello World

```
1 print("Hello World!")
```

使用 IDLE 运行; 使用 IDLE 新建文件, F5 运行

好像太简单了?

第一个程序 Hello World (一个更丰富的版本)

Hello World (2)

```
#!/usr/bin/env python
# Filename: helloWorld.py
# Author:   Lyu Ming

if __name__ == '__main__':
    print("Hello World!")
    print("""Hello
World""")
```

Hello World 的解释

- “#” 表示注释, 从此处到末尾
- 引号表示字符串.
  - 单引号 “'”: 内部的单引号要转义
  - 双引号 “"”: 内部的单引号是普通字符
  - 三引号 “"""” 或 “'''”: 内部的换行是普通字符
- “__name__” 是调用者名字的内部变量

- 用缩进标识代码块

表达式求值

表达式求值

```
1  #!/usr/bin/env python
2
3  if __name__ == '__main__':
4      expr = input(" 输入表达式: ")
5      print(eval(expr))
```

表达式求值的解释

- “input” 和 “eval” 的用法
- 数操作符: `+` `-` `*` `/` `//` `%` `**`
- 比较运算符: `<` `<=` `>` `>=` `==` `!=`
- 逻辑变量: “True”, “False”
- 逻辑操作符: “and”, “or”, “not”

- 循环控制与变量赋值!

Python 的数据类型

- 逻辑变量: “True”, “False”
- 数值: (长) 整型, 浮点数, 复数
- 字符串
- 列表 (list), 元组 (tuple) 和字典 (dict)

列表

```
>>> aList = [1, 2, 3, 'four']
>>> aList[0]
1
>>> aList[2:]
[3, 'four']
>>> aList[:3]
[1, 2, 3]
>>> aList[-1]
4
>>> aList[1] = 5
>>> aList
[1, 5, 3, 'four']
```

元组

```
>>> aList = (1, 2, 3, 4)
>>> aList[0]
1
>>> aList[2:]
(3, 4)
>>> aList[1] = 5
```

Traceback (innermost last):

.....

TypeError: object doesn't support item assignment

字典

```
>>> aDict = {'THU': 1, 'PKU': 2}
>>> aDict['ZJU'] = '?'
>>> aDict
{'THU': 1, 'ZJU': '?', 'PKU': 2}
>>> aDict.keys()
['THU', 'PKU', 'ZJU']
```

字符串

- 没有字符! 都是字符串!!
- 单引号, 双引号, 三引号
- 字符串的格式化:

```
>>> "The answer: %d"%42
'The answer: 42'
>>> "Too %s, too %s, sometimes %s"%\
    ("young", "simple", "naive")
'Too young, too simple, sometimes naive'
```

Python 的数据类型

- 逻辑变量: “True”, “False”
- 数值: (长) 整型, 浮点数, 复数
- 字符串
- 列表 (list), 元组 (tuple) 和字典 (dict)

- 试试各种运算符如何作用于这些类型

Python 的赋值

- = 等号赋值
  - Python 中 a=b 表示的是引用
- 增量赋值 +=, -=, *= 等
- 多重赋值

```
x = y = z = 1
```

- 多元赋值

```
x, y = y, x
x, y = (1, 2)
```

Python 的条件和循环

- if, elif, else 语句
- while 语句和 for 语句
- break 语句和 continue 语句
- pass 语句

if 语句

用于控制

```
if expr:
    do_something
elif expr2:
    do_other
else:
    do_else
```

用于求值

```
smaller = x if x < y else y
```

while 和 for

while

```
while expr:
    to_repeat
```

for

```
for each_var in iterable:
    to_repeat
```

for 语句

for 循环的例子

```
1 s = ['a', 'b', 'c']
2 for char in s:
3     print(char)
4 for n in range(0, len(s)):
5     # 等价于 range(len(s))
6     print(n, s[n])
7 for n, char in enumerate(s):
8     print(n, char)
```

另一个 else 的例子

最大约数

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

if __name__ == '__main__':
    num = int(input(" 输入一个数: "))
    count = num // 2
    while count > 1:
        if num % count == 0 :
            print(" 最大的约数是 %d"%count)
            break;
        count -= 1
    else:
        print(" 这是个质数")
```

pass 语句

什么都不做

- 没有大括号, 在语法上需要语句的地方可以放 pass

```
if a == b:
    pass
else:
    pass
```

- 写代码时用来占位

列表的解析

使用一个 for 循环, 把所有的值放到一个列表中

```
>>> [ x**2 for x in range(5)]
[0, 1, 4, 9, 16]
>>> [ x**2 for x in range(8) if not x%2]
[0, 4, 16, 36]
```

文件读写

```
handle = open(file_name, access_mode)
do_something(handle)
handle.close()
```

或者如果你不想写 close

```
with open(file_name, access_mode) as f:
    do_something(f)
```

- access_mode 可以是 'r', 'w', 'a'. 其他标识包括 '+', 'b'

函数

```
def numbelow(n, step=1):
    ''' 按照 step 返回小于 n 整数列表'''
    return [x for x in range(0, n, step)]
```

调用时

```
>>> numbelow(5)
[0, 1, 2, 3, 4]
>>> numbelow(5,2)
[0, 2, 4]
>>> numbelow(5,step=3)
[0, 3]
>>> help(numbelow)
```

函数

- 传值还是传引用? (回忆一下 Python 变量的赋值)
- 多个返回值, 相当于返回一个元组 (tuple)

```
def bar():
    return 'a', 'b', 'c'
a, b, c = bar()
```

- 可变数量的参数
  - 把余下的当做一个元组

```
def linear_fit(x, *p):
    k, b = p
    return k*x + b
```

  - 把余下的当做一个字典 (func(x, **p))

匿名函数

```
>>> a = lambda x, y=2 : x+y
>>> a(3)
5
>>> a(0, 9)
9
```

类 (class)

Python 并不强求用面向对象的方式来编程.

```
class ClassName(object):
    """docstring for ClassName"""
    def __init__(self, arg):
        super(ClassName, self).__init__()
        self.arg = arg
    def func(self):
        """a function for class"""
```

- 编写时类成员使用 self.xxx 标识
- 非静态成员函数的第一个参数总是 self
- 使用 . 来调用类成员 (ClassName.func)

前面提到的一些类的成员举例

complex.imag[real]	复数的实部和虚部
list.append(obj)	向列表中添加一个对象 obj
list.extend(seq)	将序列添加到列表中
list.insert(index, obj)	在 list[index] 的位置插入 obj
list.pop(index=-1)	删除指定位置的对象
string.isdigit()[isalpha(), ...]	判断 string 的特征
string.lower()[upper()]	大小写转换
string.join(seq)	以 string 连接 seq 的所有元素

.....

模块与包管理

import

- 按照模块 (module) 来组织代码 (表现为 '.py' 文件)
- 将其他模块导入 (import) 到自己的模块中
- 标准库模块, 第三方模块, 自己定义的模块 [1]

```
import re
import numpy as np
from pylab import *
from scipy.optimize import curve_fit
```

[1] pylab 部分可能需要额外安装

Python 的包管理理

下载, 管理, 升级第三方包

- pip: Python 自带的包管理脚本.
  pip install xxxx
  pip uninstall xxxx
- Anaconda: Python 科学技术包的合集.
  conda install xxxx
  conda remove xxxx
- 手工下载

常用的模块

不再详细介绍, 感兴趣的可以自学

- 和系统交互: os, sys, subprocess
- 并行运算: threading, multiprocessing, multiprocessing.dump
- 保存 Python 类: pickle
- 网络: urlparse, urllib, urllib2
- .....
- 善用搜索引擎!
- 如果只是想开始, 可以试试 xlrd 用于读取 Excel
  `github.com/CareF/curriculum-calendar-THU`

使用模块的一个例子: 正则表达式

什么是正则表达式

- 正则表达式 (Regular Expression) : 使用单个字符串来描述, 匹配一系列匹配某个句法规则的字符串.
- 如 “`.*ing\b`” 可以匹配所有以 ing 结尾的单词.
- 在 Python 中, 正则表达式相关的函数和类在 re 中定义

字符	描述
\	下一个字符转义
^	字符串的起始位置
$	字符串的结束位置
*	匹配前一个子表达式零次或多次
+	匹配前一个子表达式一次或多次
?	匹配前一个子表达式零次或一次
.	匹配除换行之外的任何单个字符
(patt)	按照模式 patt 匹配子字符串
[xyz]	匹配字符集合
[^xyz]	排除型字符集合
...	.....

在 Python 里使用正则表达式

样例代码

Python 里的数学库: Numpy, Scipy, Matplotlib

Python 里的数学库: Numpy, Scipy, Matplotlib

- 开源数学图形库
- 基于 Python
- 类 Matlab 的接口
- 优缺点

Numpy

- Numpy 最主要的作用是实现了多维数组/矩阵
- numpy.array 和 list 很像, 但有不同的接口, 对于运算符有不同的响应
- 对于矩阵运算通过 C 代码优化, 其运行效率几乎都可以与编译过的等效 C 语言代码一样快
- 矩阵相关的运算

Numpy 中常用的函数

- import numpy as np
- np.arange([start,]stop, [step,]dtype=None)

```
>>> np.arange(3)
array([0, 1, 2])
```

- np.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None)

```
>>> np.linspace(2.0, 3.0, num=5)
array([ 2.  , 2.25, 2.5 , 2.75, 3.])
```

Numpy 中常用的函数

- np.array(*parameters*)
- np.loadtxt(fname, dtype=<type 'float'>, comments='#', delimiter=None,converters=None, skiprows=0, usecols=None, unpack=False, ndmin=0)

```
>>> fp = open('data.txt', 'r')
>>> data = np.loadtxt(fp, /
    delimter = '\t')
```

Scipy

- SciPy 是一个开源的 Python 算法库和数学工具包
- 包含的模块有:
  - 最优化 (scipy.optimize),
  - 线性代数 (scipy.linalg, scipy.sparse.*),
  - 积分与常微分方程 (scipy.integrate),
  - 插值 (scipy.interpolate),
  - 特殊函数 (scipy.special),
  - 快速傅里叶变换 (scipy.fftpack),
  - 信号处理和图像处理 (scipy.signal)

    等等
- `http://docs.scipy.org/doc/scipy`

Scipy 之 scipy.constant

`docs.scipy.org/doc/scipy/reference/constants.html`

包含了: pi, golden(golden_ratio), c (speed_of_light), mu_0, epsilon_0, h (Planck), hbar, G(gravitational_constant), g, e(elementary_charge), R(gas_constant), alpha(fine_structure), N_A(Avogadro), k(Boltzmann)

...

单位转换 (SI 单位, 非 SI 单位), **常数数据库** (值, 单位, 精度), **波长与频率**...

Scipy 之 scipy.optimize

非常丰富的算法包, 仅举两个例子:

- scipy.optimize.minimize
- scipy.optimize.curve_fit

Matplotlib 中常用的函数

- from pylab import *
- figure(number)
- plot(...)
- subplot(...)
- hist(...)
- bar(...)

iPython notebook

- 现又称 the Jupyter Notebook
- 交互式的计算环境
- 借助浏览器实现跨平台

- Try it! [2]

[2] ipython notebook %pylab; %matplotlib inline

The End...

愿 Python 成为大家学习生活的助力!

Acknowledge and Licence

感谢简洁雅致的 METROPOLIS 主题.

感谢 TUNA 围观和指导.