我们编程吧 之 python 学习手册

Version 0.5

[TOC]

Python技巧

Python 缩写惯例

```
python import numpy as np import pandas as pd import matplotlib.pyplot as plt
```

Python IDE

Python（x，y）

GUI基于PyQt，曾经是功能最全也是最强大的，而且是Windows系统中科学免费Python发行版的不二选择.不过PythonXY里面的许多包为了兼容性的问题，无法使用最新的程序包。

对于科学计算要有的基本都有了：numpy，sicpy，matplotlib，spyder... 现在的版本是2.7.6.0。从安装到使用,时不时玩"崩溃"! 版本的稳定性远不如2.7.3.1。

WinPython

WinPython功能也是比较全的,软件包比较新，GUI基于PyQt，不过相对于Python（x，y），它主要是关注便携式安装体验：你可以把它装在u盘里面。

Enthought Python Distribution (EPD)

Enthought canopy是专门的Python集成编译环境，参考Enthought。 GUI基于wxpython,包含PySide,但不包括PyQt.WxPython使用起来是比较方便，但是远没有PyQt和PySide流行，需要使用PyQt的可以自己安装。Canopy有自己的集成开发环境（IDE），里面的代码智能提示和自动补全功能不比IPython差的！Canopy中还集成了Pyhton包的在线升级和管理系统，很是方便。由于是商业级别的，Canopy的性能和稳定性超强!也提供免费的free版本和学术版本（用于教育科研也是免费的）。以前叫EPD，现在改名叫Canopy。Canopy是第一个将Ipython升级到1.1.0的发行版；MatPlotLib已升级到1.3.1；NumPy 1.8.0；Scipy 1.2.0. 但是它主要是追求性能和稳定性，所以不能指望所有的安装包都是最新的，例如对于MinGW,Canopy是4.8.1，其它版本的发行版可都是4.7呀!basemap官方的也已换成1.0.7了,这对于那些还在被迫使用Grads和NCL的用户而言是个福音!如果你有学校邮箱的话，可以在Enthought的网站注册一下，选择学术+full的发行版本，会让你的工作如虎添翼的。

Anaconda

Anaconda是一个和Canopy类似的科学计算环境。自带的包管理器conda很强大。GUI基于PySide，所有的包基本上都是最新版，没有PyQt和wxpython等，容量适中，但该有的科学计算包都有：numpy，sicpy，matplotlib，spyder.....。

Linux系统里面,Anaconda安装、更新和删除都很方便,且所有的东西都只安装在一个目录中/home/username/anaconda/,这点比下面的Canopy要好得多。Anaconda目前提供Python 2.6.9,Python 2.7.X和Python 3.3.X三个系列发行包，这也是其他发行版所望尘莫及的。因此在各种操作系统中，无论是Linux，还是Windows，又或是Mac，都强烈推荐Anaconda！

在Anaconda中升级和安装都很方便,只是不像winPython那样提供图形环境,而是使用命令行:

Eclipse + PyDev

Python Tools for Visual Studio （仅限Windows用户）

PyCharm

正在使用，神器。

Spyder

Komodo IDE

Python库简介

运算

Numpy

NumPy系统是Python的一种开源的数值计算扩展。这种工具可用来存储和处理大型矩阵，比Python自身的嵌套列表（nested list structure)结构要高效的多（该结构也可以用来表示矩阵（matrix））。据说NumPy将Python相当于变成一种免费的更强大的MatLab系统。

- 快速高效的多维数组对象ndarray
- 对数组进行运算的函数
- 读写硬盘上基于数组的工具
- 线性代数、傅里叶变换以及随机数生成
- 用于将C、C++、Fortran代码集成到Python的工具

pandas

Python Data Analysis Library 或 **pandas** 是基于NumPy 的一种工具，该工具是为了解决数据分析任务而创建的。Pandas 纳入了大量库和一些标准的数据模型，提供了高效地操作大型数据集所需的工具。pandas提供了大量能使我们快速便捷地处理数据的函数和方法。你很快就会发现，它是使Python成为强大而高效的数据分析环境的重要因素之一。

Pandas中的数据结构

- Series：一维数组，与Numpy中的一维array类似。二者与Python基本的数据结构List也很相近，其区别是：List中的元素可以是不同的数据类型，而Array和Series中则只允许存储相同的数据类型，这样可以更有效的使用内存，提高运算效率。
- Time- Series：以时间为索引的Series。
- DataFrame：二维的表格型数据结构。很多功能与R中的data.frame类似。可以将DataFrame理解为Series的容器。以下的内容主要以DataFrame为主。
- Panel ： 三维的数组，可以理解为DataFrame的容器。

SciPy

SciPy是一款方便、易于使用、专为科学和工程设计的Python工具包。

它包括

- 统计
- 优化
- 整合
- 线性代数模块
- 傅里叶变换
- 信号和图像处理
- 常微分方程求解器等等

绘图

matplotlib

matplotlib是最流行的用于绘制数据图表的Python库。非常适合创建出版物上用的图表。跟IPython结合的很好，提供了一种非常好用的交互式数据绘图环境，绘制的图表也是交互式的，你可以利用绘图窗口中的工具栏放大图表中的某个区域或对整个图表进行平移浏览。

Seaborn

statistical data visualization

机器学习

Scikit-learn

增强

ipython

ipython 是Python科学计算标准工具集的组成部分，是一个 python 的交互式 shell，比默认的python shell 好用得多，支持变量自动补全，自动缩进，支持 bash shell 命令，内置了许多很有用的功能和函数。

除了标准的基于终端的IPython shell外，该项目还提供了：

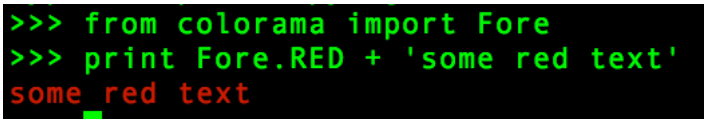
- 一个类似Mathematica的HTML笔记本
- 一个基于Qt框架的GUI控制台
- 用于交互式并行和分布式计算的基础架构

其他

colorama

既然你为日志设置了很好的进度条，何不让它们变得多彩起来！而且还可以当事情出现严重错误的时候还可以提醒自己。

colorama超级易于使用。只要弹出到你的脚本，添加任何你想要变色的文本即可。



progressbar

你知道那些你在一堆烂摊子中调用**main**的for循环执行print “still going…”脚本吗？那么你为什么不步骤化你的游戏并使用progressbar呢？

诚如其名，progressbar确实就是进度条（progress bar）。虽然这不是一个完全的数据科学中的具体活动，但它确实很好地改善了那些超长的运行脚本。

```python from progressbar import ProgressBar import time pbar = ProgressBar(maxval=10) for i in range(1, 11): pbar.update(i) time.sleep(1) pbar.finish()
```

60%
|##|

```
# IPython


## <Tab>键自动补全

使用IPython最大的好处莫过于代码的自动补全了，万万记得经常使用。

![tab-complete](http://img.blog.csdn.net/20160217170148418)


## Introspection功能

对于任何一个变量，比如data，在它的前面或者后面加上问好？，即**?data或者data?**就可以将关于该变量的一些通用信息显示出来。

![Introspection](http://img.blog.csdn.net/20160217170320795)

## %run命令

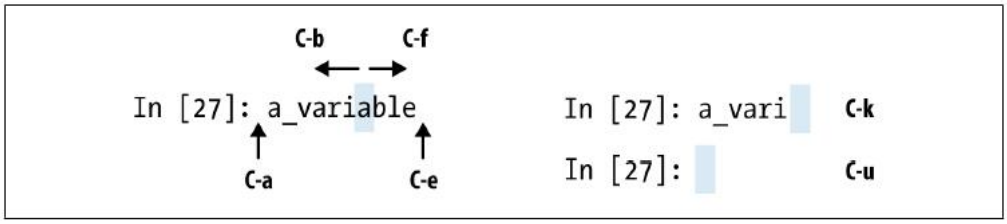
使用
```python
%run python_file.py
```

可以在IPython环境中运行脚本，并且脚本中的变量可以在IPython中直接访问。

如果希望脚本能够访问在交互式IPython命名空间中定义的变量，那就使用**%run -i python_file.py**。

键盘快捷键

比较有用的就这几个，熟悉吧。



魔术命令

IPython有一些特殊的命令，成为魔术命令Magic Command，他们能为常见的任务提供便利，魔术命令以%开始。

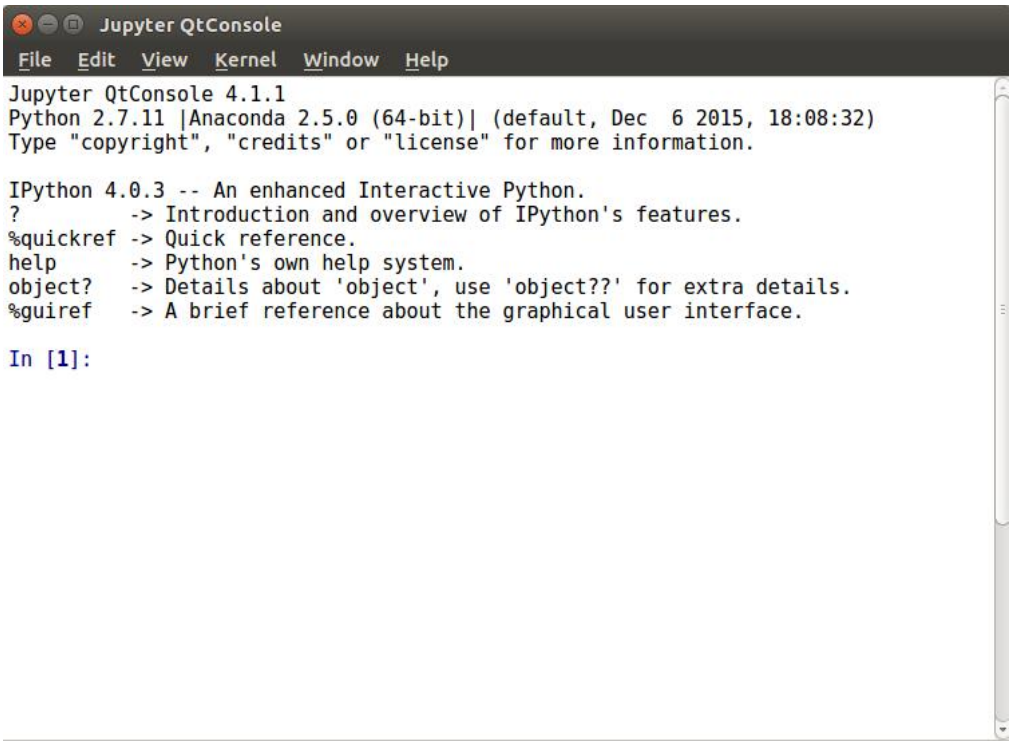
比如： 魔术命令 | 说明 ---------|------ %quickref | 显示IPython的快速参考 %magic | 显示所有魔术命令的说明 %debug | 从最新的异常跟踪底部进入交互式调试器 %hist | 打印命令的输入历史 %pdb | 在异常发生后自动进入调试器 %paste | 执行剪贴板中的python代码 %reset | 删除interactive命名空间中的全部变量及名称 %run script.py | 在IPython中执行一个Python脚本文件 %timeit stetement | 多次执行statement以计算系统平均执行时间 %who %who_ls %whos | 显示interactive命名空间中定义的变量、信息级别和冗余度 %xdel variable | 删除variable，并尝试清除其在IPython中的对象上的一切引用

基于Qt的GUI控制台

如果你已经安装了PyQt或者PySide，那么可以使用IPython团队开发的一个基于Qt框架的GUI控制台，

```
$ jupyter qtconsole
```

该控制台如下所示：
```



matplotlib集成与pylab模式

如果在标准的IPython shell中创建一个matplotlib绘图窗口，你会发现只有当你关闭绘图窗口后才能继续与shell交互，也就是说GUI的事件会占用Python会话的控制权，这就导致无法实现交互式的数据分析和可视化，因此IPython对各个GUI框架进行了专门的处理以使得能够与IPython配合地天衣无缝。

```
sh $ ipython --pylab
```

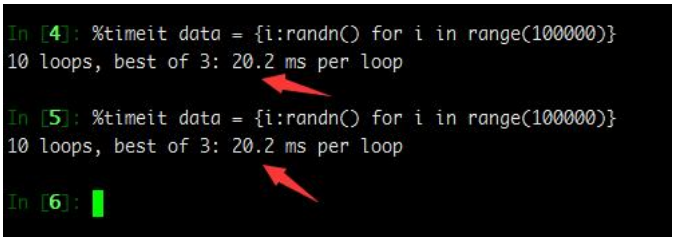
关于pylab选项的含义是：**使用默认的matplotlib后端，并载入matplotlib和numpy来更好地交互。**

shell命令

IPython中比较重要的一个特点是，可以执行系统的shell命令，只需要在前面加上一个！即可，如下所示：

统计花费时间

可以使用魔术命令%timeit 跟上语句来统计语句的执行时间。比如我们统计创建一个无序的含有100000个items的字典，看到大概需要花费20.2ms的时间。



调试

TBC

IPython HTML Notebook

NB是一种基于Web技术的交互式计算文档格式，是一种非常棒的交互式计算工具，还是科研和教学的一种理想媒介。

NB有一种基于JSON的文档格式.ipynb，使得我们可以轻松地分享代码、输出结果以及图片等。

IPython Notebook 既是一个交互计算平台，又是一个记录计算过程的「笔记本」。它由服务端和客户端两部分组成，其中服务端负责代码的解释与计算，而客户端负责与用户进行交互。服务端可以运行在本机也可以运行在远程服务器，包含负责运算的 IPython kernel (与 QT Console 的 kernel 相同) 以

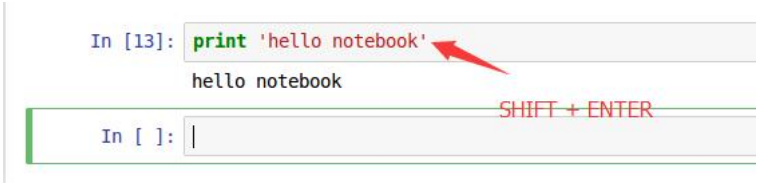
及一个 HTTP/S 服务器 (Tornado)。而客户端则是一个指向服务端地址的浏览器页面，负责接受用户的输入并负责渲染输出。

这个页面几乎涵括了 QT Console 具有的全部交互式功能，如代码高亮，自动补全，实时帮助，内嵌显示绘图结果等；其次，计算过程及结果可以方便地保存为多种格式，如默认的 JSON 格式，Python 脚本以及 PDF 等；此外，借助 Markdown 及 MathJax，用户可以在计算过程中插入详尽细致的描述，甚至以描述为主，计算为辅，从而将它当作科技类文章写作的工具。

正是由于 IPython Notebook 的灵活易用，方便传播等特点，它现已被用于可重复数据分析，课程教学，博客写作等众多领域。

每次运行按shift-enter

每次按下**shift-enter**就会执行代码。



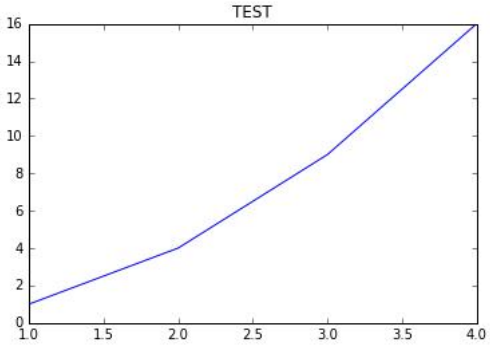
内嵌的魔术命令

```
%matplotlib inline
```

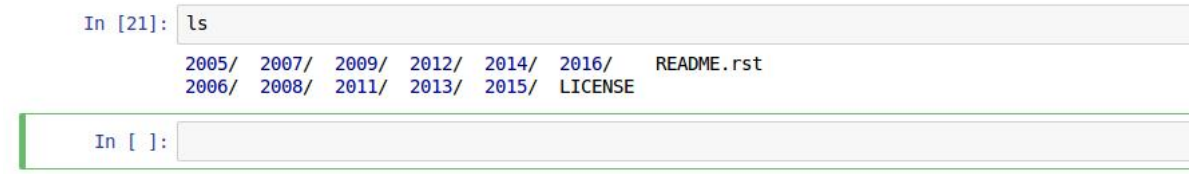
在Notebook中输入上述命令，就可以将matplotlib内嵌到Notebook内，如下图：

```
In [14]: %matplotlib inline

In [17]: plot([1,2,3,4],[1,4,9,16]);
         title('TEST');
```



可以直接运行shell命令



NumPy

NumPy（**Numerical Python**）系统是Python的一种开源的数值计算扩展。这种工具可用来存储和处理大型矩阵，比Python自身的嵌套列表（nested list structure)结构要高效的多。据说NumPy将Python相当于变成一种 **免费的更强大的** MatLab系统。

部分功能如下所示：

- 一个强大的N维数组对象Array
- 比较成熟的（广播）函数库
- 用于整合C/C++和Fortran代码的工具包
- 实用的线性代数、傅里叶变换和随机数生成函数等

ndarray - 一种多维数组对象

ndarray是一个通用的同构数据多维容器，其中的所有元素都是相同类型的，每个数组都有一个**shape**（表示各个维度大小的元组）和一个**dtype**（用于说

明数组数据类型的对象）。

```
In [10]: data
Out[10]:
array([[ 0.2,  0.4,  0.3,  1.2],
       [ 2.1,  2.3,  4.3,  5. ]])

In [11]: data.shape
Out[11]: (2, 4)

In [12]: data.dtype
Out[12]: dtype('float64')

In [13]:
```

创建ndarray

创建数组很简单，使用array函数即可，它接受一切序列型的对象（包括其他数组），然后产生一个新的含有传入数据的NumPy数组，比如将一个列表转换为数组的方法为：

```
In [15]: data1 = [i for i in range(10)]

In [16]: data1
Out[16]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [17]: arr1 = np.array(data1)

In [18]: arr1
Out[18]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [19]: data2 = [[1,2,3,4],[5,6,7,8]]

In [20]: arr2 = np.array(data2)

In [21]: arr2
Out[21]:
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])

In [22]:
```

除了array函数之外，也可以使用下面的函数来新建数组：

- zeros
- ones
- empty

```
In [22]: np.zeros(10)
Out[22]: array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.])

In [23]: np.zeros((10,4))
Out[23]:
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])

In [24]: np.ones((3,4))
Out[24]:
array([[ 1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.]])

In [25]: np.empty((2,3,4))
Out[25]:
array([[[  6.93804463e-310,   2.08989136e-316,   6.93804540e-310,
           6.93803448e-310],
        [  6.93804536e-310,   6.93803448e-310,   6.93804540e-310,
           6.93803448e-310],
        [  6.93804536e-310,   6.93803448e-310,   6.93804540e-310,
           6.93803448e-310]],

       [[  6.93804536e-310,   6.93803448e-310,   6.93804540e-310,
           6.93803448e-310],
        [  6.93804536e-310,   6.93803448e-310,   6.93804540e-310,
           6.93803448e-310],
        [  6.93804536e-310,   6.93803448e-310,   6.93804540e-310,
           6.93804244e-310]]])

In [26]:
```

这里需要注意的是，empty并不是返回零，而是一些未初始化的垃圾值。

也要注意range和arange的区别。

```
In [26]: range(10)
Out[26]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [27]: np.arange(10)
Out[27]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [28]:
```

ndarray的数据类型及转换

Python的数据类型大致分为：**浮点数、复数、整数、布尔值、字符串**等。

类型	代码	说明
int8,uint8	i1,u1	有符号和无符号的8位（1个字节）整型
int16,uint16	i2,u2	有符号和无符号的16位（2个字节）整型
int32,uint32	i4,u4	有符号和无符号的32位（4个字节）整型
int64,uint64	i8,u8	有符号和无符号的64位（8个字节）整型
float16	f2	半精度浮点数
float32	f4或f	标准的单精度浮点数，与C的float兼容
float64	f8或d	标准的双精度浮点苏，与C的double和Python的float兼容
float128	f16或g	扩展精度浮点数
complex64,complex128,complex256	c8,c16,c32	分备用两个32位，64位或128位浮点数表示的复数
bool	?	存储True和False值的布尔类型

类型的转换十分常见，比如使用**astype**函数

```
In [28]: arr2
Out[28]:
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])

In [29]: arr2.dtype
Out[29]: dtype('int64')

In [30]: arr2.astype(np.float64)
Out[30]:
array([[ 1.,  2.,  3.,  4.],
       [ 5.,  6.,  7.,  8.]])

In [31]: arr2.dtype
Out[31]: dtype('int64')

In [32]:
```

更多信息参考numpy官网。

SciPy

TBC

matplotlib

TBC

Chaco

TBC

mayavi

TBC

pandas

TBC

更多信息

Hi，XDJM们，更多信息欢迎移步我的github或微信公众号letsProgramming.