## 伪代码

## 面向对象

<https://www.jianshu.com/p/7a5b0043b035>

举个最简单点的例子来区分 面向过程和面向对象

有一天你想吃鱼香肉丝了，怎么办呢？你有两个选择

1、自己买材料，肉，鱼香肉丝调料，蒜苔，胡萝卜等等然后切菜切肉，开炒，盛到盘子里。

2、去饭店，张开嘴：老板！来一份鱼香肉丝！

看出来区别了吗？这就是1是面向过程，2是面向对象。

<https://www.zhihu.com/question/31021366/answer/761614647>

面向过程 (Procedure Oriented) 是一种 以过程为中心 的编程思想。这些都是以什么正在发生为主要目标进行编程，不同于面向对象的是谁在受影响。与面向对象明显的不同就是 封装、继承、类。

## Python 3.8.8 （官方中文）文档，及标准库

<https://docs.python.org/zh-cn/3.8/>

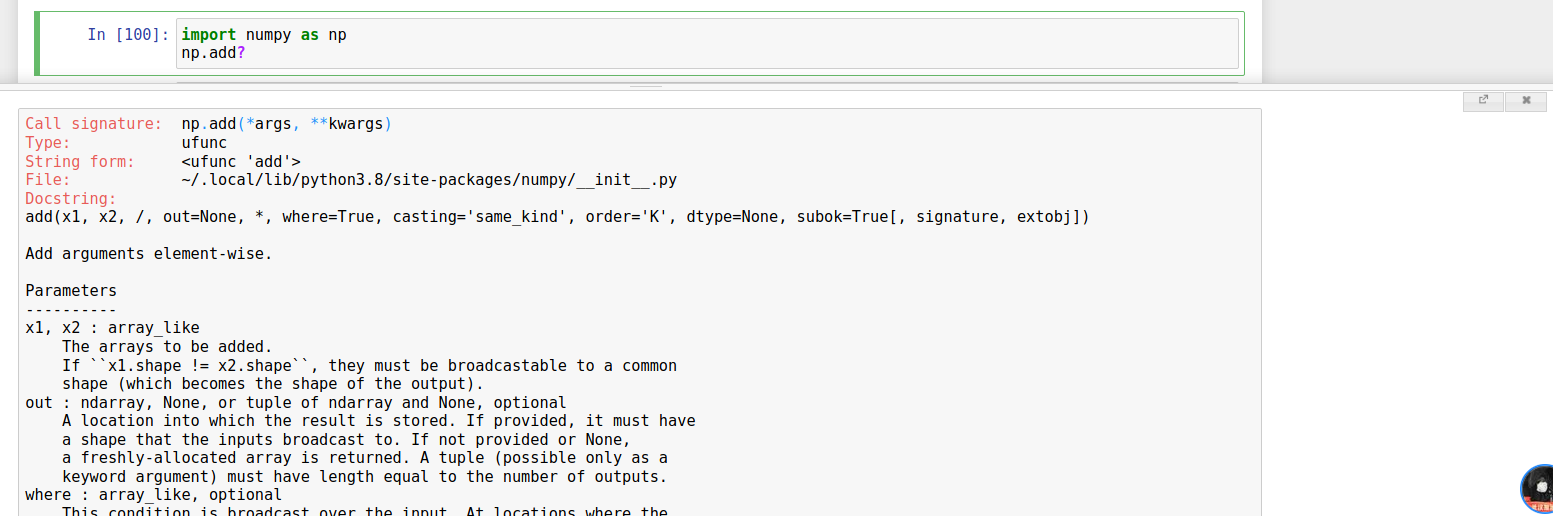
<https://docs.python.org/zh-cn/3.8/library/index.html>

## 具体函数帮助文档

import numpy as np

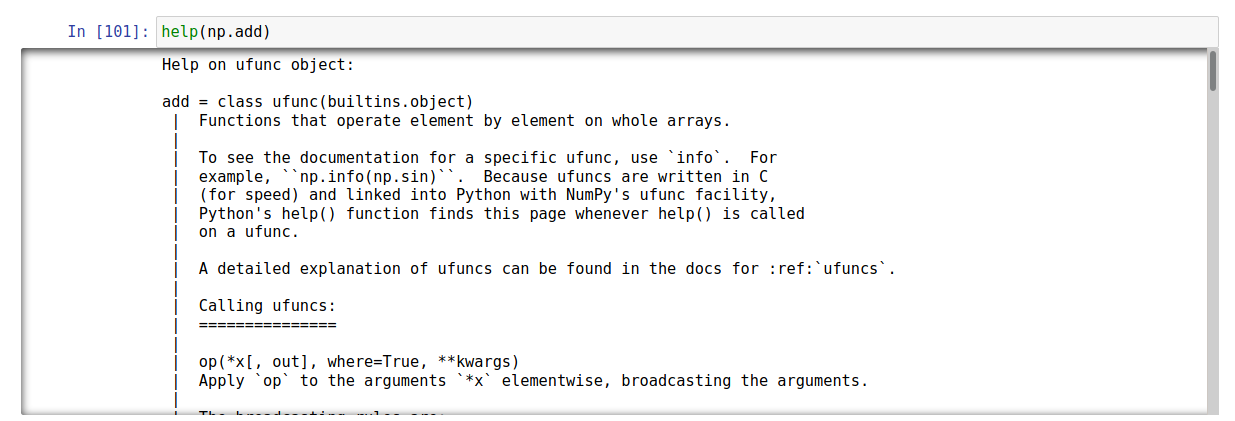
np.add?

界面弹出文档



help(np.add)

交互界面出文档



## 部分源码中方法只有pass占位符

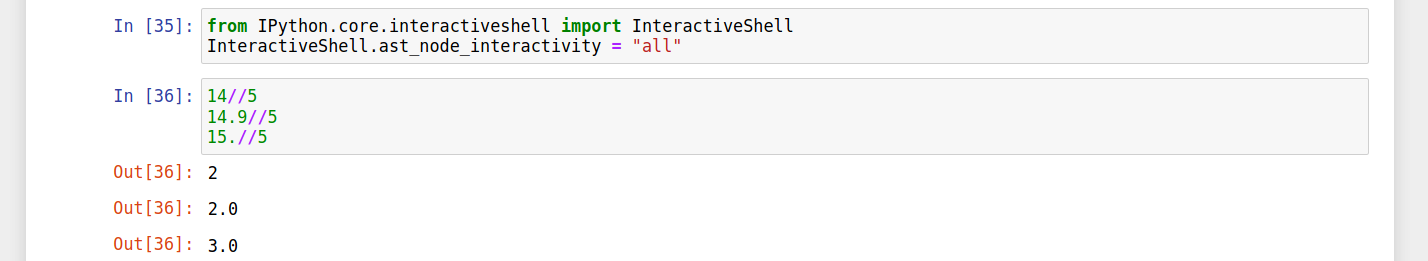
函数的具体实现不用python编写，而是由例如C这种高效语法编写，在包中只用一个空方法占位，调用的时候是调用C语言实现的方法。python中有一些需要大量运算的内置函数是用C或者C++写的。<https://blog.csdn.net/pansaky/article/details/89242404>

## Python2.7与3.8

Py2中：7/5=，7./5=1.4，出于省内存考虑；

Py3已无需（内存有钱即可买到），7.5=1.4；向下取整的功能由“//”保留。

<https://www.runoob.com/python3/python3-basic-operators.html#ysf5>



print a, b ——> print(a,b)

## Python 标准库

### glob，查找文件名函数

<https://docs.python.org/zh-cn/3.8/library/glob.html>

注意：绝对路径 (如 /usr/src/Python-1.5/Makefile)和相对路径 (如 ../../Tools/\*/\*.gif)。

### pickle，二进制读写

<http://c.biancheng.net/view/5736.html>

甚至可以直接与变量对接输出输入：

sim\_data = [event\_rate, zs\_detected, masses, rhos\_detected, dl\_zs]

pickle.dump(sim\_data, open(filename, 'wb'))

event\_rate, zs\_detected, masses, rhos\_detected, dl\_zs=pickle.load(open(filename,'rb'))

### format格式化（输入输出常用）

<https://blog.csdn.net/it_python/article/details/81037078?dist_request_id=1328680.53330.16164038487795099&depth_1->



## numpy

官方中文库：<https://www.numpy.org.cn/>



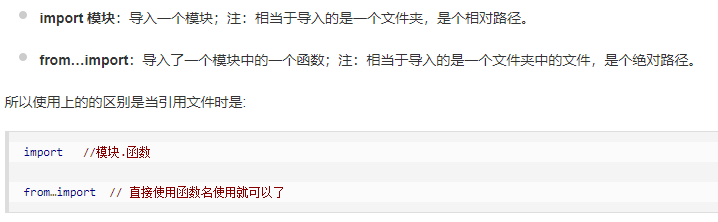
关于Numpy存在的必要性，方便于矩阵与大数据。最浅层，计算速度快，因为Numpy包底层是通过C++实现的。<https://zhuanlan.zhihu.com/p/140072485>

### import 和 from …import...的区别

import numpy as np

from scipy.integrate import quad

<https://www.runoob.com/note/26325>



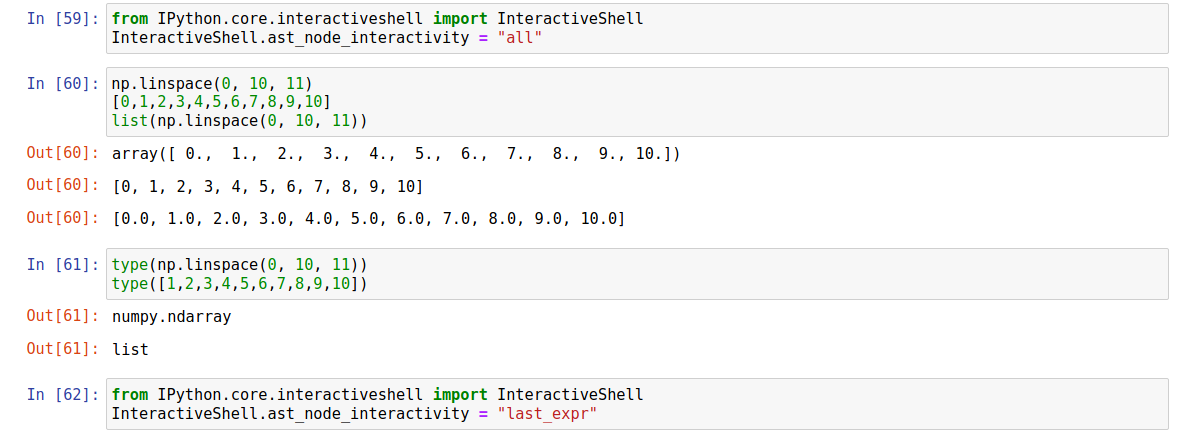
<https://blog.csdn.net/qq_38630100/article/details/80808757>

### numpy.ndarray数据类型

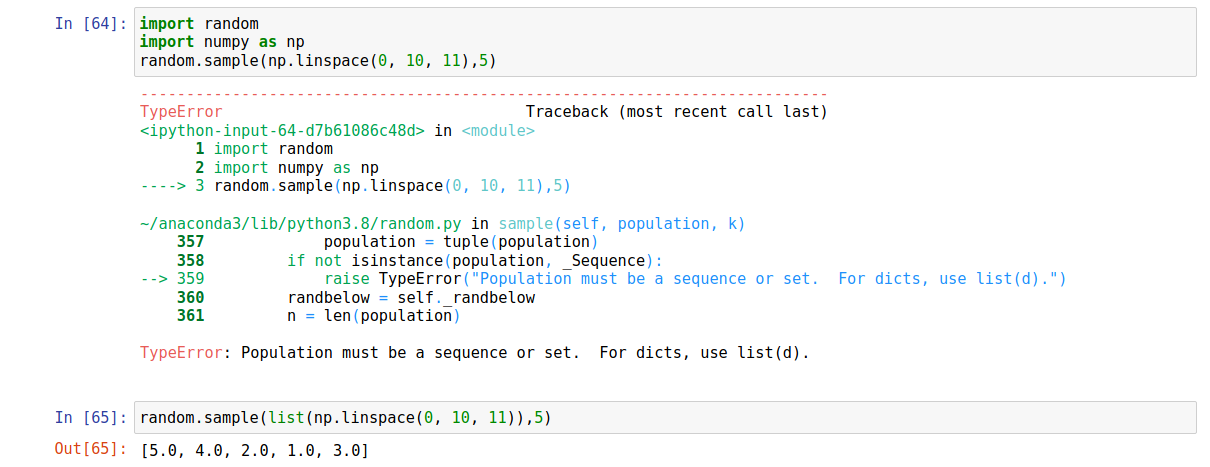
不是python自带数组的“list”型：相对于普通一维list，其矩阵的功能更多

#### .astype强制类型转换（np.ndarray.astype）

<https://numpy.org/doc/1.20/reference/generated/numpy.ndarray.astype.html?highlight=astype>



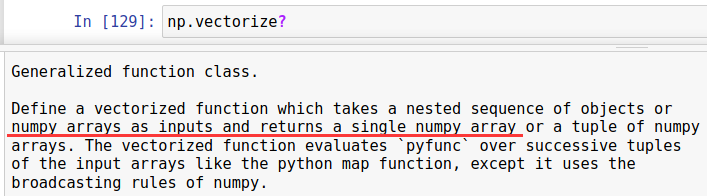
由于是numpy自己定义的数据类型，因而可能会在某些python库函数调用上出现问题：



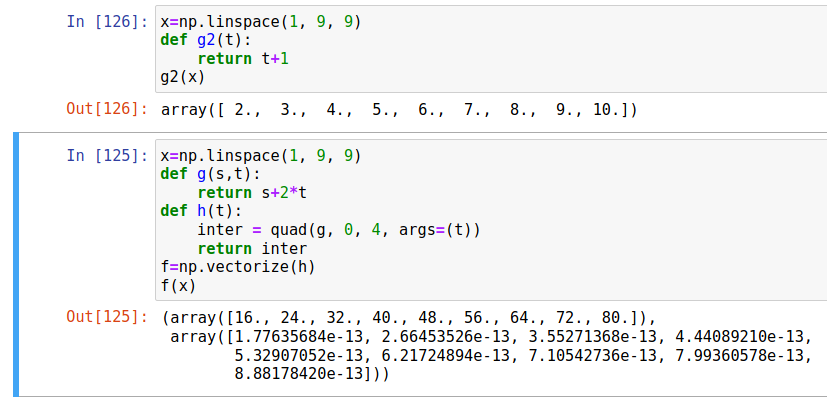
### 数组操作

#### 作为函数自变量

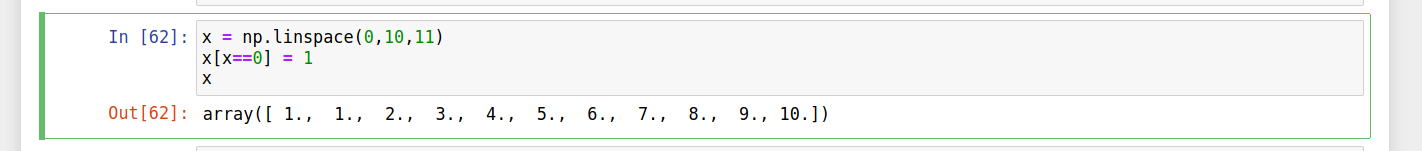
简单的函数可能可以，但更保险的是使用numpy的函数：np.vectorize()



复杂函数，如scipy.integrate中的quad，必须使用矢量化np.vectorize()



#### 内含布尔运算与逻辑表达式

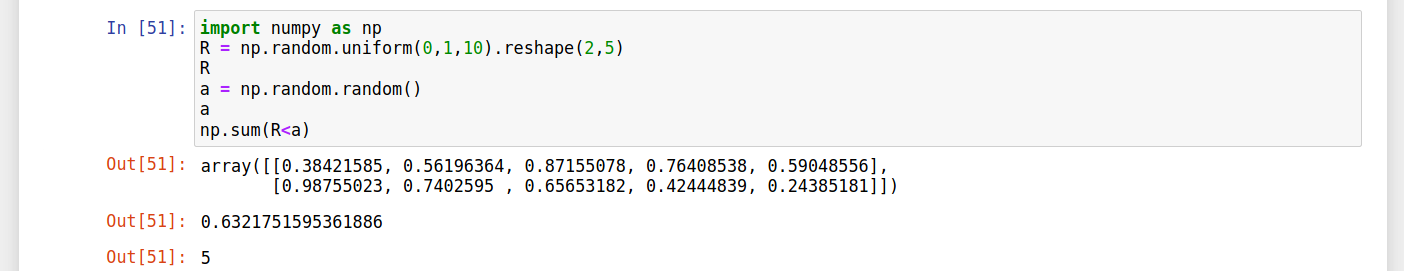


#### sum(R数组 < a数)

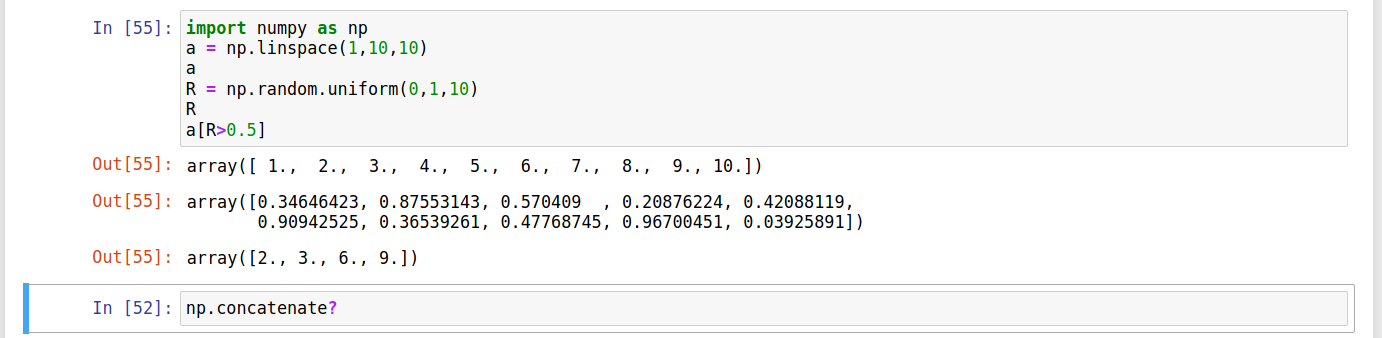
通常求和用法参阅官网：

<https://numpy.org/doc/1.20/reference/generated/numpy.sum.html?highlight=sum#numpy.sum>

np.sum(R数组 < a数)：计算数组R中 < 数a的总个数。即扩展数a为等大小数组，判断条件组成真值表数组（False = 0，True = 1），再对此真值表数组求和。



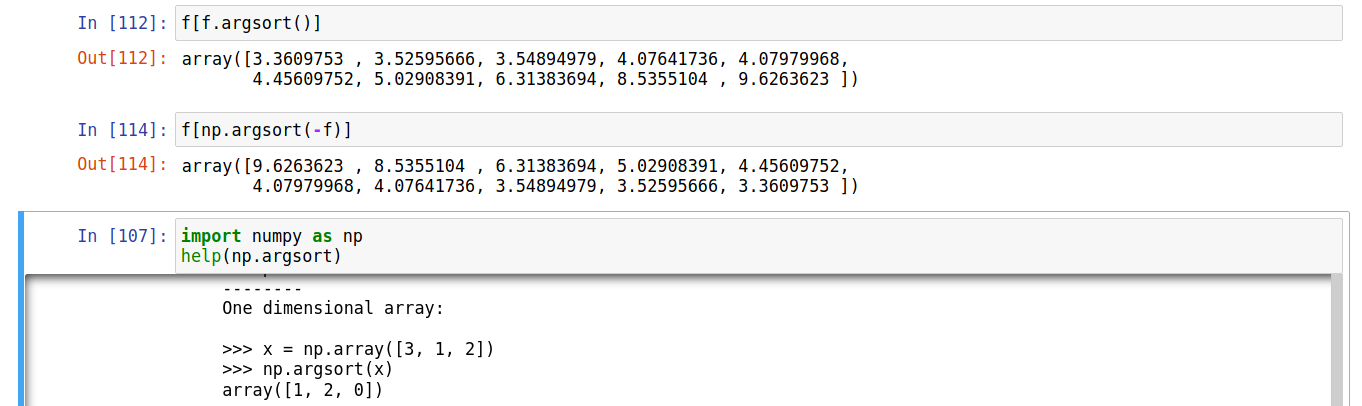
#### 按数组判别选择，及concatenate连接数组



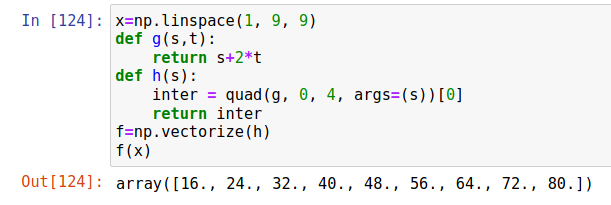
### argsort升序排列，返回索引指标

<https://blog.csdn.net/Python798/article/details/81138040>

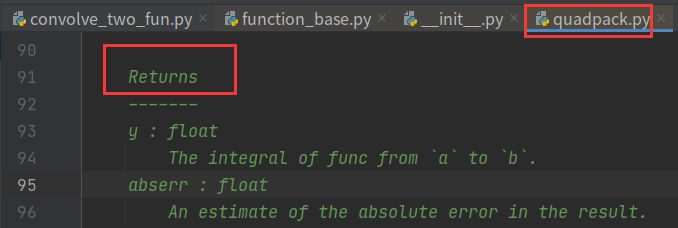
升序重排数组，的两种方式：f = f[np.argsort(f)]，或f = f[f.argsort()]。也可完成降序：



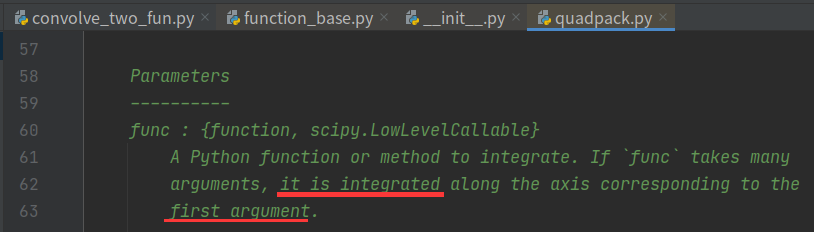
### quad积分



quad()返回一个数组，[积分结果，精度估计]

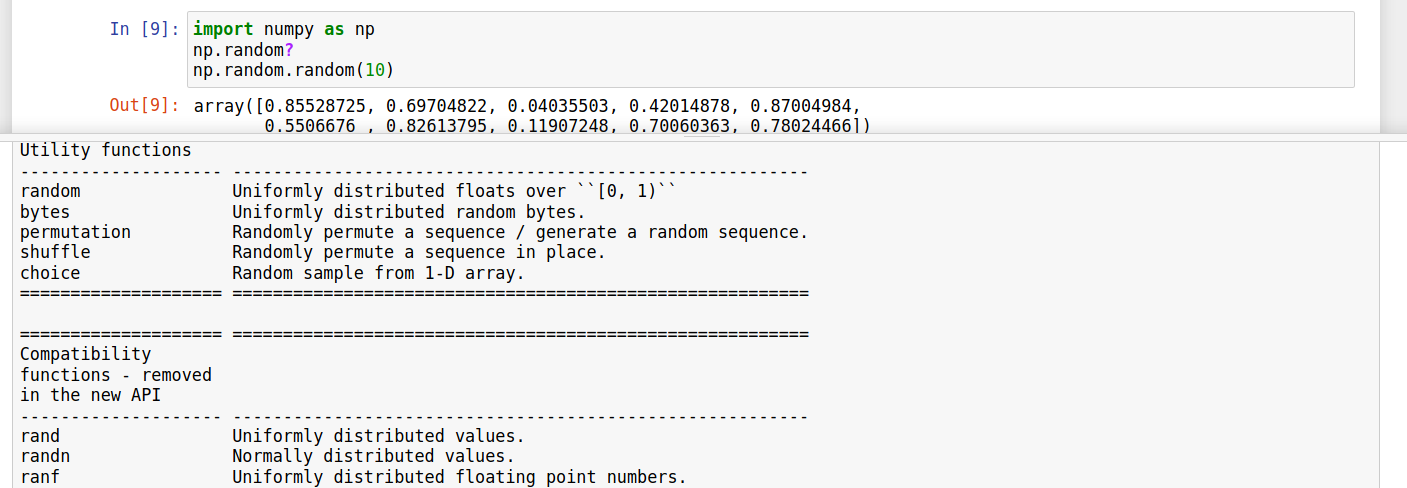


如果有多元参数，默认积分第一个。此处积掉g中的s，尽管h中也定义变量是s。



### random产生随机数

np.random.seed(42)，设置种子为42，以便后续随机数的重复验证。



### loadtxt，savetxt读入写出数据：

import numpy as np

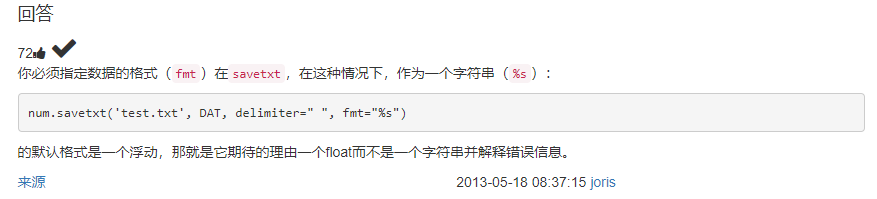
data = np.loadtxt('/home/chongzi/Documents/du\_data.txt',dtype=str)

data2 = data.reshape(int(len(data)/8),8)

np.savetxt('/home/chongzi/Documents/du\_data2.txt',data2,fmt="%s",delimiter="\t")

注意输入格式dtype=str、输出格式fmt="%s"，不做指定的话默认都是浮点数dtype=<class 'float'>、fmt='%.18e'。但显然txt是以字符串形式储存，“.dat”才是以浮点数形式。

<http://cn.voidcc.com/question/p-xyuxvblc-xm.html>



## corner图（contour等高线图）

<https://corner.readthedocs.io/en/latest/pages/quickstart.html>

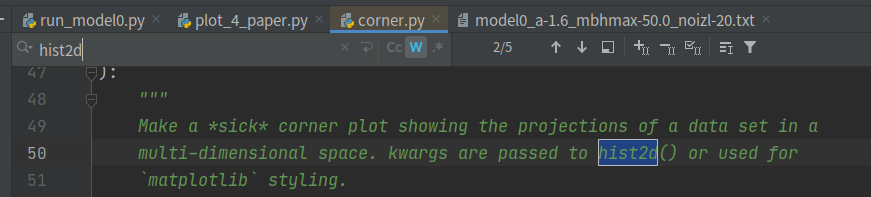
此包唯一面向用户的函数，只有corner.corner，



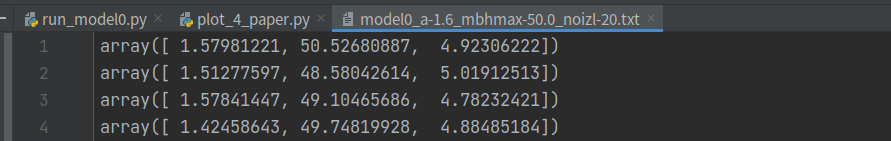
作用是，对（至少数千次）参数估计的结果，绘制其“相关性”图：

data = n\_sample行 \* dimension列。对n\_s次试验，取dim中任意两列参数估计结果，做2D（1D）平面散点图，进而可分析估计结果集中的趋势。数据点足够多（1000次以上），继续做1σ、2σ、3σ等高线图，即可看出各参数彼此制约的估计优良性。

即2-D histogram of samples，核心是corner.hst2d函数。



注意：如果数据是以np.array的形式储存，读入时需稍作转换：



filename = 'model0\_a-1.6\_mbhmax-50.0\_noizl-20.txt'

with open(filename) as f:

content = f.readlines()

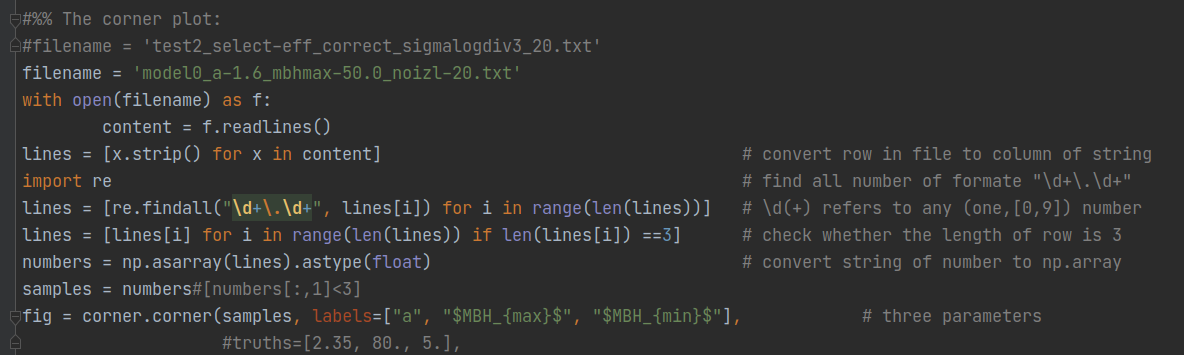
lines = [x.strip() for x in content]

import re

lines = [re.findall("\d+\.\d+", lines[i]) for i in range(len(lines))]

lines = [lines[i] for i in range(len(lines)) if len(lines[i]) ==3]

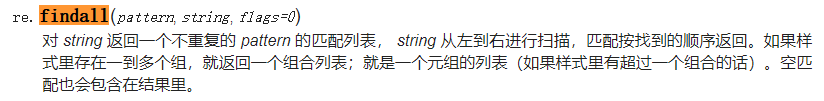
numbers = np.asarray(lines).astype(float)



## re 正则（regular）表达式操作与匹配：

<https://www.runoob.com/python/python-reg-expressions.html>

<https://docs.python.org/zh-cn/3.8/library/re.html>



## scipy

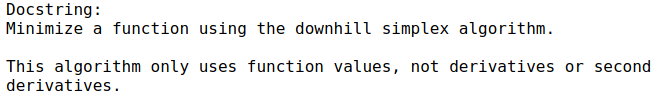
### optimize最优化，最佳（拟合）

教程<https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html>

from scipy.optimize import fmin

模块<https://docs.scipy.org/doc/scipy/reference/optimize.html#module-scipy.optimize>

利用Nelder-Mead 法或称下山单纯形法，计算函数最小值

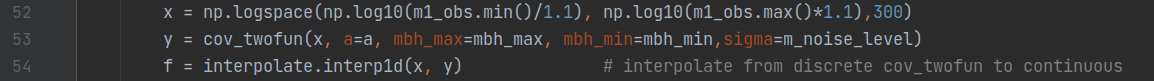




### interpolate向量插值为函数

教程<https://docs.scipy.org/doc/scipy/reference/tutorial/interpolate.html>

模块<https://docs.scipy.org/doc/scipy/reference/interpolate.html#module-scipy.interpolate>



## matplotlib

### 置信区间作图：

<https://matplotlib.org/stable/gallery/lines_bars_and_markers/fill_between_demo.html#sphx-glr-gallery-lines-bars-and-markers-fill-between-demo-py>

### 低配cornor图：

<https://matplotlib.org/stable/gallery/lines_bars_and_markers/scatter_hist.html#sphx-glr-gallery-lines-bars-and-markers-scatter-hist-py>