……1……

try-except语句：

若语句可能发生异常中断，则设置try语句，把可能错误的语句放在try语句中，用except处理异常，每一个try后面至少有一个except。例如：

a=1 b=0

try:

c=a/b

print c //如果没有try则会报错：ZeroDivisionError

except ZeroDivisionError,e: //e代表异常的实例。

Print e.message

Print “done”

输出：integer division or modulo by zero

Done

则程序不会中断。

try-except-else：当没有异常发生时，else中的语句将会执行。

try-finally:

异常无论有没有发生，finally都会执行，若有异常，则在异常之前执行。

……2……

hasattr(object,name)：

判断object对象里面有无name属性或name 方法，返回bool值，注意name要用引号括起来。

例：

class test():

name=”ttt”

t=test

if hasattr(t,”name”):

print t.name

getattr(object,name[,default])：

获取对象的属性或方法，如果存在就打印出来，不存在则打印报错，如果是方法，则打印方法的地址，若在getattr后加()则可将方法运行后打印。

set(object,name,value):

给对象赋值，若对象不存在，则创建后赋值。

例：setattr(t,”age”,”33”)

**……3、正则表达式……**

^:匹配行或字符串起始位置

$:匹配……尾

\b：只匹配一个位置，常位于单词边界

\d：匹配数字

\w:字母、数字、下划线

\s:空格

（以上四种若字母大写，则意义相反）

. :除换行符以外任何字符

[abc123]:匹配包含括号内元素的字符

[^abc123]:匹配除了括号内以外的任意字符

+：重复1次或更多

\*：重复0次或更多

？：重复0或1次

{n}：重复n次

{n,m}：重复n到m次

{n,}：重复大于等于n次

\*？：重复任意次，但尽可能少重复（尽可能少匹配）

+？：重复1次或更多次，但尽可能少重复

??：重复0或1次

{n,m}?：重复n到m次，但尽可能少重复

{n,}?：重复n次以上，但尽可能少重复

(exp)：匹配exp

(?<name>exp)：匹配exp，并捕获文本到名称为name的组里

(?:exp)：匹配exp，但不捕获匹配的文本，也不给此分组分配组号

(?=exp)：匹配exp前面的位置

(?<=exp)：匹配exp后面位置

(?!exp)：匹配后面跟的不是exp的位置

(?<!exp)：匹配前面不是exp的位置

re.S 表示多行匹配，防止同一字符串有换行

group(0)匹配整个字符串，group(1)匹配第一个分组 即表达式第一个括号对应的匹配字符串，类型是字符串！！！

re.match(pattern,string,flags): /flags参数可省略。

第一个参数为正则表达式，第二个为要匹配的字符串，第三个为标志位，控制匹配方式如大小写等，若匹配成功则返回一个Match，否则返回None；

re.search(pattern,string，flags):

在字符串内查找匹配，找到第一个匹配就返回，若没有匹配则返回None；

re.findall(pattern,string):

可获取字符串中的所有匹配的字符串组成的元祖，抽取时用[]；

re.split(pattern，String):

分割字符串，将字符串String按正则表达式分割；

re.sub(pattern,rep,string,count,flags) /后两个为可选参数

pattern为正则表达式，rep为要替换为字符串，string为要被处理的字符串。

4 下划线

1. 函数：

……前带一个\_，表明这是一个私有函数，只用于标明。

……前后各带两个\_，表明特殊函数，如\_\_class\_\_，\_\_init\_\_。

1. 变量：

……大写加下划线：不会发生改变的全局变量。

……前带一个\_加小写：私有变量,表明是需要保护的变量，外部类可访问。

5 self

Python中的self类似于C中的this，传递当前类的实例。

6 \_\_init\_\_函数

\_\_init\_\_函数为初始化函数，每个对象必须正确初始化后才能正常工作，该函数可以有多种参数形式。定义形式为：

class T:

def \_\_init\_\_(self,par1,par2,….) //把self放在第一位置；

self.part1=par1 //self代表调用时的当前实例类

self.part2=par2

t=T(pa1,pa2) //创建对象同时初始化

print t.part1,t.part2 //输出: pa1,pa2

7 urllib

urllib2可接受一个request对象，并以此来设置一个URL的headers。

urllib2模块定义的函数和类用来获取URL：

urllib2.urlopen用来打开一个网址，url参数可以是字符串或一个request对象：

以下两个等价：

1、import urllib2

response=urllib2.urlopen(‘http://baidu.com)

Html=response.read()

2、import urllib2

request=urllib2.Request(‘http://baidu.com’)

response=urllib2.urlopen(request) /urlopen会返回一个response对象。

html=response.read() /read为response的一个方法 可返回读取到的内容

urllib2的urlopen有三个参数：urlopen(url,data,timeout):

 url:网址

data:表示访问url时传送的数据，比如用户名和密码，默认为空。

 timeout:设置超时时间，默认为socket.\_GLOBAL\_DEFAULT\_TIMEOUT

**response**：该对象有一个read方法，可返回获得的网页内容，若不加read，则会打印出对该对象的描述。

动态传递数据（网页是动态网页）的两种方法：

1、POST方式：不会在网址上显示出所有参数：

values={"username":"xxx","password":"xxxxx"}

#利用encode将字典编码

data=urllib.urlencode(values) url='http://xxx.com'

#构建request时传入两个参数，一般第二个参数可省略，需要传递参数时才加上。

request=urllib2.Request(url,data)

response=urllib2.urlopen(request)

print response.read()

2、GET方式：直接以链接形式访问，链接中包含所有参数：

values={}

values[username]= 'xxxx'

values[password]='xxxxxx'

data=urllib.urlencode(values)

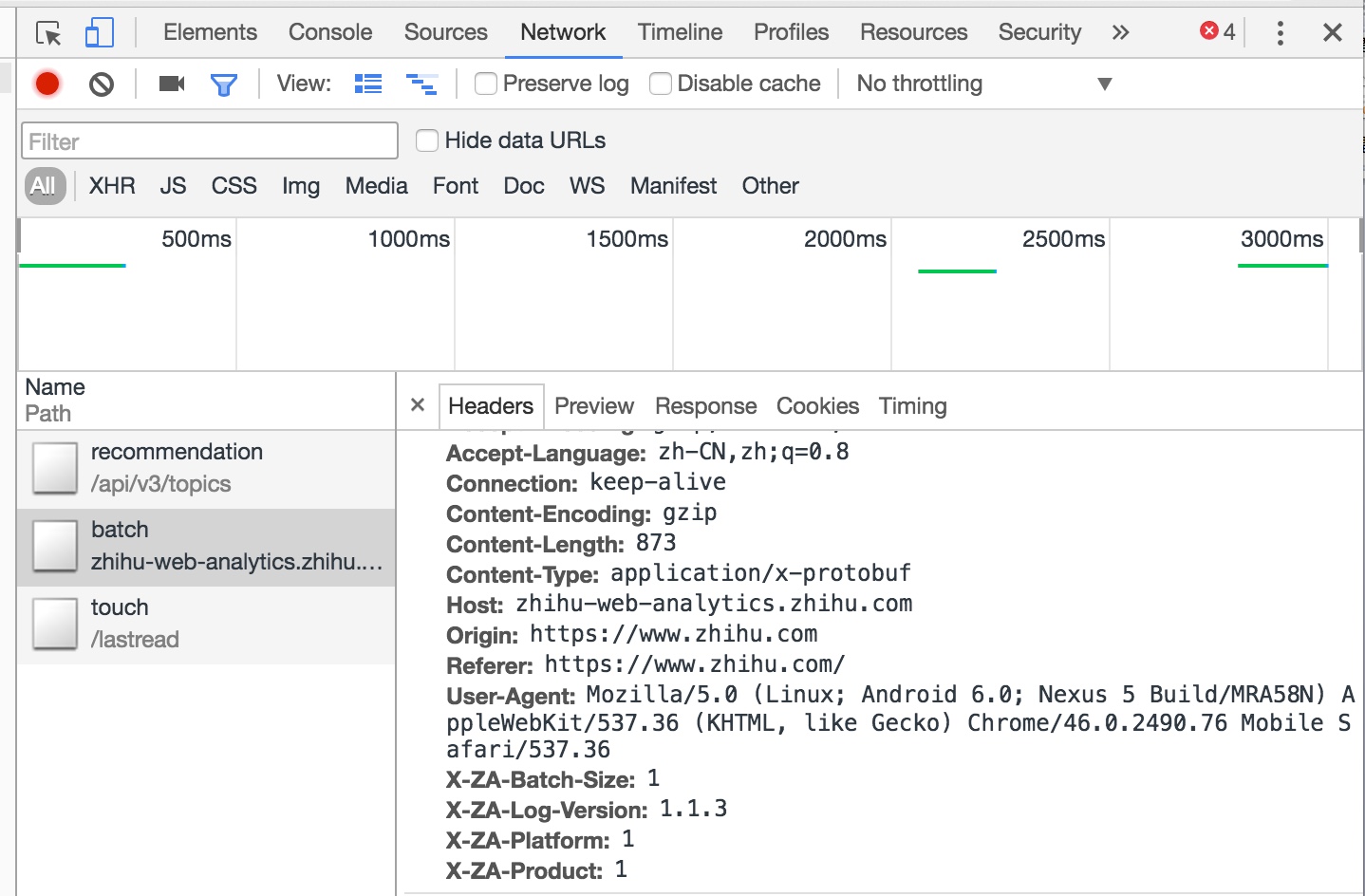
url='www.xxxxx.com'

geturl=url+”?”+data #其实就是把username和password一起加到了网址中

request=urllib2.Request(geturl)

response=urllib2.urlopen(request)

headers

若网站不同意上述方式直接访问，需要设置headers的属性，在chrome浏览器中，打开控制台，找到下方内容：

其中的user\_agent就是请求的身份，我们可以这样设置：

url=' https://www.zhihu.com/'

user\_agent='Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.76 Mobile Safari/537.36

'

#referer='https://www.zhihu.com/'

values={'username':'xxx','password':'xxxx'}

headers={'User-Agent':user\_agent}

#headers={'User-Agent':user\_agent,'Referer':referer}#服务器有防盗链是使用此句

data=urllib.urlencode(values)

request=urllib2.Request(url,data,headers) #构建request时，加入headers传送

response=urllib2.Response(request)

Timeout：

urlopen的第三个参数timeout，设置等待多久超时，解决某些网站响应过慢：

1、如果data为空，则要写明timeou：

response=urllib2.urlopen('http://www.baidu.com',timeout=10)

2、如果有data传入，则不必写明：

response=urllib2.urlopen('http://www.baidu.com',data,10)

8 循环

……split: A='I am Chinese'

B=A.split()

print A /输出['I', 'am', 'Chinese']，转换为了列表。

……total=0

B=[1,2,3,4]

for b in B:

total+=b

print total /输出10.

或者：

total=sum(b for b in B) /sum为列表的一种函数，

print total.

9 读写文件

……写：

f=open('xutao','w') /文件名xutao

f.write('12345abc\n') /写入字符串，如果是数字，则需要强制转换为字符串

f.close() /关闭文件，否则文件为空

……读：

F=open('xutao')

for g in F:

print g /输出12345abc

……删除：

import os /提前导入os包

os.remove('xutao')

10 数据类型

|  |  |
| --- | --- |
| 整数 | -100 |
| 浮点数 | 3.1416 |
| 字符串 | 'hello' |
| 列表 | [1, 1.2, 'hello'] |
| 字典 | {'dogs': 5, 'pigs': 3} |
| Numpy数组 | array([1, 2, 3]) |
| 类型 | 例子 |
| 长整型 | 1000000000000L |
| 布尔型 | True, False |
| 元组 | ('ring', 1000) |
| 集合 | {1, 2, 3} |
| Pandas类型 | DataFrame, Series |
| 自定义 | Object Oriented Classes |

11列表推导式

values = [10, 21, 4, 7, 12]

squares = []

**for** x **in** values:

squares.append(x\*\*2) //生成列表

**print** squares

或者：

values = [10, 21, 4, 7, 12]

squares = [x\*\*2 **for** x **in** values]

**print** squares

12 模块和包

……模块：以.py结尾的文件可以被当做一个模块，用import加载并执行，加载时，会执行一遍模块中的所有内容，要使用其中变量，要用：xx.变量名，如果需要重新导入模块，可以用reload（模块名）来加载，若果直接再用import导入并不会执行。删除时：

import os

os.remove('xx.py')

导入方法或变量：

from xx import xxx

若要导入所有变量和方法：

from xx import \*

……包：

假设我们有这样的一个文件夹：

foo/

* \_\_init\_\_.py
* bar.py (defines func)
* baz.py (defines zap)

这意味着 foo 是一个包，我们可以这样导入其中的内容：

**from** **foo.bar** **import** func

**from** **foo.baz** **import** zap

12 Numpy

……array:

若a=[1,2,3,4],若想要每个元素加一，直接a+1报错，利用Numpy中的array则可以：

a=np.array(a)

print a+1 #输出2，3，4，5

或两个数组相加：

b=[2,3,4,5]

b=np.array(b)

print a+b /输出3，5，7，9

print a\*b /输出2，6，12，20

print a\*\*b /输出1，8，。。。。以b中元素为幂。

……提取数组元素：

提取前两个：a[:2]，提取后两个：a[-2:]

……形状：

a.shape /查看array形状，输出（6，）或者（6L，）

……重置形状：

a.shape=2,3

print a / 输出：[[1 2 3] [4 5 6]]，其实就是将一维行向量转换成2\*3的数组了,转换后还可以进行a+a,a\*a的操作，还是对应元素的加减乘除。

reshape()方法会返回一个新数组，原数组不变。

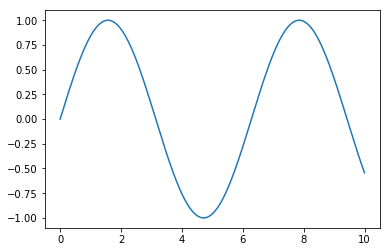
……画出图像：

import matplotlib.pyplot as pl

a=np.linspace(0,10,100) /0到10，共100个数

b=np.sin(a) /b中元素是a中元素的正弦

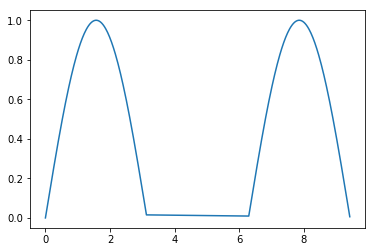
pl.plot(a,b) /以a为横轴，b为纵轴画出正弦图像。



……筛选图像：

c=b>=0 /选则非负值，输出的c为一组bool值。

pl.plot(a[c],b[c])

￼

……numpy数组属性：

a=array([1,2,3,4])

a.dtype /数据类型，比如int32或int64

a.itemsize /每个元素占的字节数，比如4或8

a.size /元素数，4

a.nbytes /总字节数

n.ndim /维数，1

a.fill(x) /将数组a全部填充为相同的x整数值

……数组切片：

a[x1:x2] /表示从a[x1]到a[x2-1]的元素

a[x1:] /表示从a[x1]到最后的数字

a[x1:-x2] /表示从a[x1]到从最后开始x2的元素

a[-x1:x2] /表示从a[x1+1]到a[x2-1]的元素

a[::2] /表示a[0]、a[2]、a[4].....

……二维数组：

引用：

a=np.array([[1,2,3],[4,5,6]])

a[0,0]=1

a[0]=[1,2,3]

切片：

a[0,1:] /输出第一行的第二个元素及后面的元素2，3

切片是引用，改变引用值，原值也会改变：

若b=a[0,1:]

b[0]=10 /重新赋值

则a变成[[1,10,3],[4,5,6]]

若让b copy()产生复制，则不会改变原值：

b=a[0,1:0].copy() /改变b的值不会改变a的值。

但是，若定义中a是列表，而非array，则不会变化。

……arange()产生数组:

a=np.arange(start,end,gap)

如a=np.arange(1,10,2),则a=[1，3，5,7,9] /若只有一个 参数，则默认从0到该参数

b=a[np.newaxis,:] /数组a产生一个新维度

b=b.squeeze() /去除b的长度为1的维度并返回一个新数组。

……randn和rand:

np.random.randn(2,4) //从标准正态分布中返回多个样本值

np.random.rand(2,4) /从[0,1]中返回随机样本。

……transpose()数组转置：

对于多维数组，利用transpose()可将其转置，如a为2\*3，b=a.transpose()就转置成3\*2

或者a.T也可以。

……concatenate()数组连接：

concatenate((a0,a1...),axis=0),要求各数组的形状相同，默认axis=0，即增加行，列不变，如x.shape()=(2,3),y.shape()=(2,3),

z=np.concatenate((x,y)),z.shape()=(4,3)/np.vstack()

z=np.concatenate((x,y),axis=1) /np.hstack()

z.shape()=(2,6)

……flatten:

b=a.flatten() /返回将数组a变成1维数组的数组，原数组不变。

……flat:

b=a.flat

print b[0] /flat返回了所有元祖组成的一个迭代器，修改b同时会影响a

c=a.ravel()

print c /返回一维数组，修改c同时影响a

……linspace：自动生成等间隔数组：

d=np.linspace(start,end,num) /start是第一个元素，end是最后一个，num是生成多少个元素，默认50个。

……ogrid,mgrid(start:end:step)：

x,y=np.ogrid(-1:1:0.5,-1:1:0.5)

print x /输出一列数组：[[-1. ] [-0.5] [ 0. ] [ 0.5]]，y为行数组

x,y=np.ogrid(-1:1:5j,-1:1:5j)

这样输出[[-1. -0.5 0. 0.5 1.]]，可以输出5个。

……r\_,c\_产生行向量或列向量：

np.r\_[0:1:.1]

输出行向量：([ 0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])

np.c\_[1:3:5j]

输出：([[ 1. ], /列向量

[ 1.5],

[ 2. ],

[ 2.5],

[ 3. ]])

……ones,zeros(shape,dtype=float64):

np.ones(3) /产生1\*3的全1的数组,zeros同理

np.ones(3)\*n /产生1\*3的全1的数组

np.ones([3,4]) /产生3\*4的全1的数组

……empty：

a=np.empty(n) /产生一个n个元素的数组，值为内存值，不确定

a.fill(x) /用x值填充数组

……identity(n,dtype=float64)：

np.identity(n) /产生一个n\*n的单位矩阵。

……花式索引：

b=[1,3,4]

print a[b] /输出a[1],a[3],a[4]即[3，7，9]

……where（array）语句：返回所有非零元素的索引：

a=array([2,0,1,4,6])

where(a>2) /输出array[3,4],返回符合条件的索引下标。

多维数组类似，返回行列索引。

……复数数组：

a=np.array([1+3j,4,5,2+4j]) /Python可以自动判别类型

a.real=[2,4,5,6] /实部可以重新定义,虚部也可以

print a /输出数组

print a.real /输出实部

print a.imag /输出虚部

print a.conj() /共轭复数

……ord函数：可以查看ASCII码值，print ord('a') /输出97

……数组类型：

若数组元素不一样，将dtype定为object即可：

a=np.array([1,'hello',[3,4],5],dtype=object) /若不加dtype这句，会报错。

print a\*2 /输出[2 'hellohello' [3, 4, 3, 4] 10]

利用dtype可以更改数组类型，比如int改为float。

……asarray:

a=np.array([1.4,2,0.5,4])

b=np.asarray(a,dtype=int)

当类型相同的时候，asarray 并不会产生新的对象，而是使用同一个引用.

……astype:

a=np.array([1.4,2,0.5,4])

b=a.astype(int) /astype不会改变原数组，返回一个新数组。

print b

print a

print a is b /a和b不一样，输出false

print a.sum() /求所有元素的和。

print a.sum(axis=0) /求每一列的和；axis=1，每一行的和。

print a.prod() /求所有元素的乘积

print a.prod(axis=0) /求每一列的积；axis=1则是每一行的积。

print a.min() /求最小值

print a.min(0) /求每一列的最小值

print a.min(1) /求每一行的最小值

print a.max() /求最大值，行列同min函数。

print a.argmin(0) /返回每一列最小值的位置；1：每一行。argmax同理。

print a.mean() /返回均值，参数0，1。

print np.average(a,axis=1,weights=[1,2,3,4]) /该函数支持加权平均。

print a.std() /标准差，参数0，1。

print a.var() /方差，参数0，1。

print a.clip(2,4) /限制范围，将小于2的变成2，大于4的变成4.

print a.ptp() /计算最大与最小之差，参数0，1.

print a.round(decimals=n) /近似到n位数，无参数则四舍五入。

……排序：

print np.sort(a) /将数组a从小到大排列，维数不变。数组也不变

a.sort()

print a /这种方式sort后数组会改变为排序后的顺序。

print a.argsort() /输出从小到大排列的下标值。

如果是多维数组，sort(a)默认对每一行进行排序，sort(a,asix=0)对每一列进行排序。

a=np.linspace(0,1,5) /a=0,0.25,0.5,0.75,1

b=[0.4,0.1,0.3,0.6]

print np.searchsorted(a,b) /searchsorted函数，第一个参数必须是排序数组，第二个为数组，返回b中元素插入a中的位置，该例返回[2 1 2 3]数组。

……对角线：

a=np.array([11,12,13,21,22,23,31,32,33])

a.shape=3,3

print a.diagonal() /输出[11，22，33]

print a.diagonal(offset=1) /对角线右移1，输出[21，32];offset=-1,输出[12,23]

对角线花式索引：

i=[0,1,2]

print a[i,i] /输出[11，22，33]

a[i,i]=2 /修改对角线的值。

……choose：通过修改对应下标位置的值修改数组达到条件筛选：

i0 = np.array([[0,1,2],

[3,4,5],

[6,7,8]])

i2 = np.array([[20,21,22],

[23,24,25],

[26,27,28]])

C = np.array([[1,0,1],

[2,1,0],

[1,2,2]])

print np.choose(control, [i0, 10, i2]) /以C数组为基准，按照第二组参数[i0,10,i2]将对应下标位置修改为[i0,10,i2]的对应值：C00=1, [i0, 10, i2]1=10，所以C00=10，同理，C10=2，[i0, 10, i2]2=i2,而i210=23，故C10=23，以此类推输出为：

([[10,1, 10],

[23, 10,5],

[10, 27, 28]])

……选择筛选1：

i0=np.array([[1,11,0],

[12,2,10],

[4,2,32]])

print np.choose(i0<10,[i0,10]) /i0<10返回的是一组bool类型的数组，值为true或false，参数为(i0,10),为true的改为10，false的保持i0对应值不变，所以输出：

[[10 11 10] [12 10 10] [10 10 32]]

……选择筛选2：将数组小于10的改为10，大于15的改为15，其余不变。

i0=np.array([[1,11,0],[12,2,10],[4,2,32]])

l=i0<10 /若满足条件则为true即1，否则为0

m=i0>15

choice=l+2\*m /输出为数组，[[1 0 1] [0 1 0] [1 1 2]]，1表示小于10，2表示大于15，0表示介于10和15之间

print np.choose(choice,[i0,10,15]) /输出为[[10 11 10] [12 10 10] [10 10 15]]，以实现将数组小于10的改为10，大于15的改为15，其余不变。

……向量化函数，Vectorize():

例：定义一个函数：

**def** sinc(x):

**if** x == 0.0:

**return** 1.0

**else**:

w = np.pi \* x

**return** np.sin(w) / w /该函数只能作用于单值，无法作用于数组。

如a=np.array([1,2,3])，则sinc(a)则报错。

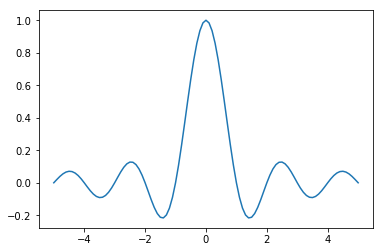
将sinc向量化：

vsinc=np.vectorize(sinc)

再定义数组：

x=np.linspace(-5,5,101)

plt.plot(x,vsinc(x)) /利用向量化后的函数画图：



……矩阵：

a = np.array([[1,2,4],

[2,5,3],

[7,8,9]])

A=np.mat(a) /利用mat()函数，将数组转换为矩阵。

np.linalg.inv(a) /逆矩阵。

利用分块创造新矩阵：

a = np.array([[1,2,4],

[2,5,3]])

b=np.array([[6,7],[8,9]])

print np.bmat('a,b;b,a') /利用bmat()函数和分块矩阵创造新矩阵

……矩阵和向量相乘：

a = np.array([[1,2,4],[2,5,3]])

A=np.mat(a) /A已是矩阵

b=np.array([[6,7,8]]) /b是数组也是向量。

print a\*b /普通乘法，对应元素相乘,a为n阶矩阵。

A.dot(b) /矩阵乘法

print A.I /I为逆矩阵。

print A\*\*2 /表示二连乘，前提是该矩阵为n\*n。

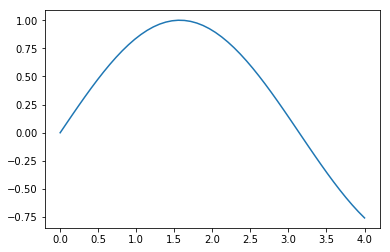
print np.linalg.det(A) /行列式的值。

13 Matplotlib:

……二维图：

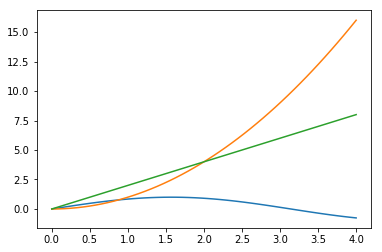
x=np.linspace(0,10,50) /生成0到10的50个数的数组。

plt.plot(x,sin(x)) /以x为横轴，sin(x)为纵轴画图：



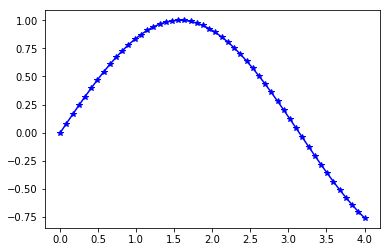
也可以在一幅图中作多条曲线：

plt.plot(x,sin(x),x,x\*\*2,x,x\*2): /参数之间相互用逗号隔开。



给线条以参数：

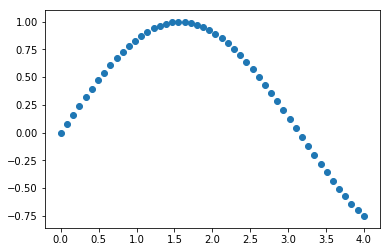
plt.plot(x,np.sin(x),'b-\*') /b代表蓝色，\*表示在每个点处的图案：



……scatter散点图：

方法一：plt.plot(x,np.sin(x),'bo')

方法二：plt.scatter(x,np.sin(x))



多图：

plt.plot(sin(x))

plt.figure()

plt.plot(cos(x)) /这样可以同时产生两幅图，而不是几条线在一幅图中。

hold:

plt.plot(sin(x))

plt.hold(False) /该命令可以将上一个命令的图关闭，只显示下面的图形。

plt.plot(cos(x))

……加图例：

plt.plot(x,label='sin') /在图中用label将每条线的标签表示出来。

plt.plot(y,label='cos')

plt.legend()

……加坐标轴和标题：

plt.xlabel=('xxx')

plt.ylabel=('yyy',fontsize='large') /默认小字体，可以改为大字体。

plt.title('title')

plt.grid() /在图形中添加网格。

14 结构化数组：

a=np.array([1,2,3,4],np.float32)

#创建自定义数据结构类型，然后用此结构来解释数组a：

my\_dtype=np.dtype([('mass','float32'),('vol','float32')])

print a.view(my\_dtype) /用自定义结构解释数组

b=np.array([(1,2),(3,4),(5,6),(7,8)],my\_dtype) /用该结构定义数组。

print b[0]['mass'] /输出某个元素的某个特征值。

也可以这样定义结构：

p\_dtype=np.dtype([('name','S10'),('age','int'),('wgt','float')])

people=np.empty((3,4),p\_dtype)

people['name'] = [['Brad', 'Jane', 'John', 'Fred'],

['Henry', 'George', 'Brain', 'Amy'],

['Ron', 'Susan', 'Jennife', 'Jill']]

people['age'] = [[33, 25, 47, 54],

[29, 61, 32, 27],

[19, 33, 18, 54]]

people['wgt'] = [[135., 105., 255., 140.],

[154., 202., 137., 187.],

[188., 135., 88., 145.]]

从文本中读取数据：

加入txt文档中有以下文件：

amy 11 38.2

john 10 40.3

bill 12 21.2

利用定义的结构类型和loadtxt读取该文件：p=np.loadtxt('www.txt',skiprows=0,dtype=p\_dtype)

print p

输出：[('amy', 11, 38.2) ('john', 10, 40.3) ('bill', 12, 21.2)]

文件读取完毕，利用os.remove('www.txt')删除文件。

15 datetime对象：

import datetime as dt

d1=dt.date(2017,8,23)

d2=dt.date(2018,8,23)

print d1.strftime('%A-%y-%m-%d') /按此格式输出：Wednesday-17-08-23

print d2-d1 /输出两个日期相差时间：365 days, 0:00:00

t2=dt.datetime.now() /当前日期时间

print t2+dt.timedelta(10) /当前日期再加十天。

DateTime格式字符表：

| **字符** | **含义** |
| --- | --- |
| %a | 星期英文缩写 |
| %A | 星期英文 |
| %w | 一星期的第几天，[0(sun),6] |
| %b | 月份英文缩写 |
| %B | 月份英文 |
| %d | 日期，[01,31] |
| %H | 小时，[00,23] |
| %I | 小时，[01,12] |
| %j | 一年的第几天，[001,366] |
| %m | 月份，[01,12] |
| %M | 分钟，[00,59] |
| %p | AM 和 PM |
| %S | 秒钟，[00,61] （大概是有闰秒的存在） |
| %U | 一年中的第几个星期，星期日为第一天，[00,53] |
| %W | 一年中的第几个星期，星期一为第一天，[00,53] |
| %y | 没有世纪的年份 |
| %Y | 完整的年份 |

15 函数：

map(function,squence[]): /返回一个数组序列，对squence中的每个元素用function求解。

匿名函数lambda用法:

如上例可以改写：map(lambda x:x\*\*2,sequence[]) /省去function的定义

全局变量：

在函数中修改全局变量需要加global：

x = 15

def newx():

global x /如果不加，x值不在函数中时不变。

x = 18

csv文件：

pandas的read\_csv:

import pandas as pa

d=pa.read\_csv('aaa.txt',index\_col=0') /若index\_col不定义，则输出会自动添加0开始的序号，index\_col=0表示按原状输出。

csv\_reader():

import csv

fp=open('aaa.txt')

f=csv.reader(fp)

for r in f:

print r /这种形式输出的都是字符串,不过可以定义数组来存储数据：

data=[]

with open('aaa.txt') as fp:

r=csv.reader(fp)

for row in r:

data.append([row[0],str(row[1]),str(row[2]),int(row[3])])

……稀疏矩阵：

from scipy.sparse import \*

A= coo\_matrix([[2,3,0],[0,0,3],[1,0,0]])

print A /输出下标和对应数值：

(0, 0) 2

(0, 1) 3

(1, 2) 3

(2, 0) 1

B=A.todense() /转换为普通矩阵。

B=np.array([1,0,-1])

print A.dot(B) /矩阵与向量乘法，输出[2,-3,1]

……coo\_matrix (data, (row, col)) 的元组来构建稀疏矩阵：

I=np.array([0,1,3,2])

J=np.array([0,2,1,1]) /I和J对应元素作为下标，V元素作为值构建。

V=np.array([9,8,8,6])

A=coo\_matrix((V,(I,J)),shape=(4,4)) /构建4\*4稀疏矩阵

row,col,val=sparse.find(A) /查找稀疏矩阵的非零值，返回行列下标和其值

print row,col,val /输出三个数组，分别代表行，列，元素值。

……minimize最小化函数；

from scipy.optimize import minimize

例：根据初试抛速度，求水平飞行最大距离：

def dist(theta,v0):

g=9.8

theta\_rad=3.14\*theta/180

return 2\*v0\*\*2/g\*np.sin(theta\_rad)\*np.cos(theta\_rad)

def neg(theta,v0):

return -1\*dist(theta,v0) /minimize求最小值，所以这里把dist转为复数

result=minimize(neg,40,args=(10,)) /minimize三个参数，第一个为函数，第二个为猜想值，默认将优化函数的第一个参数作为优化参数，即theta，将minimize的第三个参数给优化函数并开始优化。

16 概率计算相关：

直方图：

from scipy.stats import norm /导入正太分布

x\_norm=norm.rvs(size=500) /从正太分布产生500个随机点

h=plt.hist(x\_norm,normed=True,bins=20) /作直方图，出现频率为纵轴 划分区间20

x\_mean,x\_std=norm.fit(x\_norm) /该组数据下的最大似然估计值。

通过积分计算在某个区间的概率：

from scipy.integrate import trapz /导入积分函数

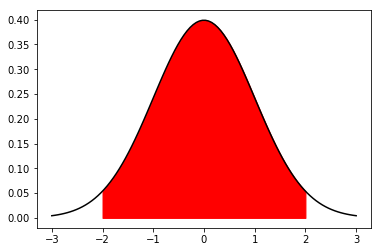
x1=np.linspace(-2,2,108)

p=trapz(norm.pdf(x1),x1) /默认正太分布的均值为0，标准差1，即标准正态分布

print p /输出在（-2，2）区间的概率大小。

plt.fill\_between(x1,norm.pdf(x1),color='red') /相交区间填充红色

plt.plot(x,norm.pdf(x),'k-') /作出在x范围即（-3，3）的正态图：



17 Pandas

……Serise:一维结构：pd.Series(data,index=index)

import pandas as pd

s=pd.Series([1,2,'d',2.3]) //无index

print s[2] //可通过下标引用

s=pd.Series(np.random.randn(5),index=['a','b','c','d','e'])//有index

d = {'a' : 0., 'b' : 1., 'c' : 2.}

pd.Series(d) //字典转换为Series

……DataFrame：二维结构：DataFrame(data,index(行),columns(列)):

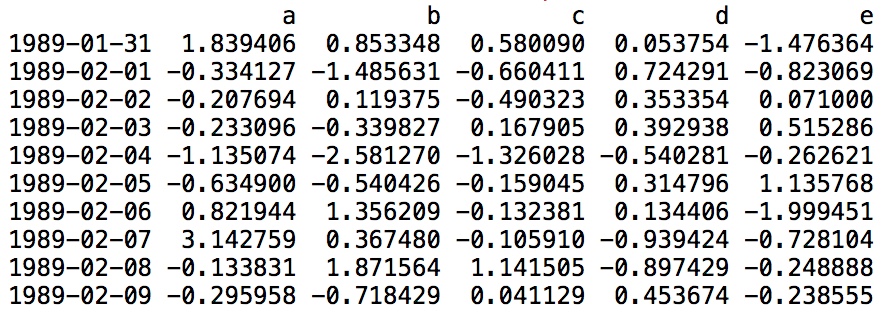
d={'1':pd.Series([1,2,3],index=['aa','bb','cc']),'2':pd.Series([4,5,5],index=['dd','bb','cc'])}

df=pd.DataFrame(d) //利用字典构造，若对应index数据无，则为NaN

先构造一维时间序列：

dates=pd.date\_range('19890131',periods=10) //时间序列，共10个时间，如果输出会输出19890131，19890201，。。19890209

df=pd.DataFrame(np.random.randn(10,5),index=dates,columns=list('abcde')) //创建DataFrame结构，这里选择三个参数，第一个参数返回标准正太分布中返回（10，5）个样本值，第二个参数表示各列名，第三个表示各行名。如果index和columns不选，则会从零开始。输出：



print df.head(n)/tail(n) //可以查看前n行或后n行的数据，默认为5.

Print df.index/columns/values //查看index、、的属性值。

Print df.describe() //查看各项统计数据

Print df.T //查看转置

print df.sort\_values(by='a') //按照a列的大小进行排列。

Print df.a/df[‘a’] //索引a列。

Print df[0:2] //选取0、1两行,或：

print df['1989-02-01':'1989-02-04']

print df.loc[:,['a','c','b']] //选择多列。

print df.loc['1989-01-31':'1989-02-03',['a','c','b']] //多行多列。

print df.loc['1989-01-31','a'] //定位某个值。

df['f']=['1','2','3','4','5','6','7','8','9','10'] //添加f列。

print df[df['f'].isin(['4','5'])] //过滤出f列的某些值

df.loc[dates[0]:dates[1],'e']=1 //更改某行某列的值。

对数据做些处理，使其中某些数据丢失，然后丢失所有缺失数据的行得到新数据：

print df1.dropna(how='any') //得到新数据。

df3=df1.fillna(value=5) //填充空缺数据，得到新数据。

print df3

print df.mean() //每一列的均值。mean(1)为每一行的均值。

print df.apply(lambda x:x.max()-x.min()) //每列的最大值-最小值。

s=pd.Series(np.random.randint(0,7,size=10)) //构造一维数据结构

print s.value\_counts() //统计数据分布

s.hist() //绘制直方图。

s.str.lower() //当某一列是字符串时，可以对其操作转换大小写。

d=[df[:2],df[3:5],df[6:]] //对多个pandas对象进行连接，注意格式！。

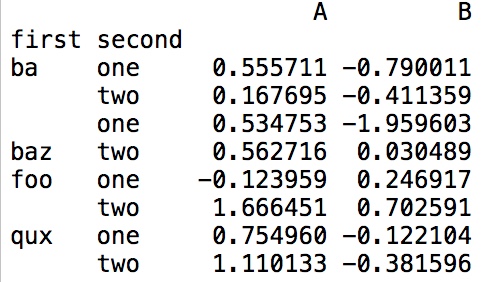
print pd.concat(d)

print df.groupby('A').sum() //对数据以A列数据进行分组，其他列以此求和。

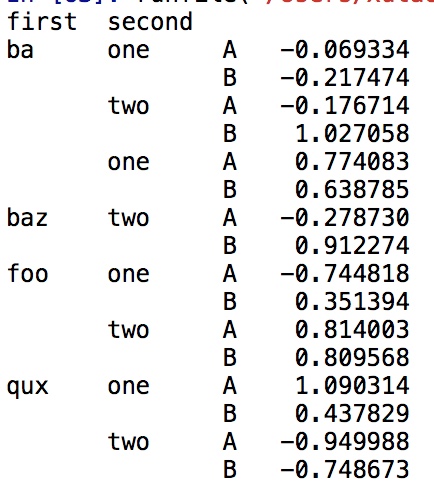
tuples = list(zip(\*[['ba', 'ba', 'ba', 'baz', 'foo', 'foo', 'qux', 'qux'], ['one', 'two', 'one', 'two','one', 'two', 'one', 'two']]))

index = pd.MultiIndex.from\_tuples(tuples, names=['first', 'second'])

df = pd.DataFrame(np.random.randn(8, 2), index=index, columns=['A', 'B']) //产生多index的DataFrame。



print df.stack() //转换为新的index：

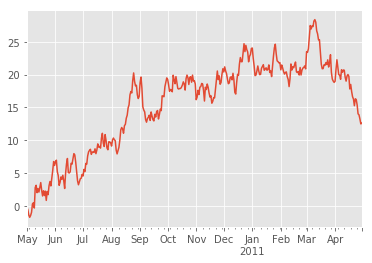


……Series绘图：

plt.style.use('ggplot')

ts=pd.Series(np.random.randn(365),index=pd.date\_range('5/1/2010',periods=365))

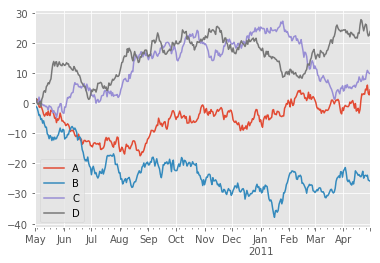
ts.cumsum().plot() //图像如下：



……DataFrame绘图：

ts=pd.DataFrame(np.random.randn(365,4),index=pd.date\_range('5/1/2010',periods=365),columns=['A','B','C','D'])

ts.cumsum().plot() //绘图如下：



del df['c'] //删除c列

dd=df.pop('d') //同样删除某列。

计时函数：

import time

start=time.time()

end=time.time()

dur=end-start //以秒为单位。