# Python数据科学库入门

# Python数据科学基础第一部分-Numpy

# 1.Numpy

## 1.1Numpy简介

Numpy（Numerical Python）

Numpy：提供了一个在Python中做科学计算的基础库，重在数值计算，主要用于多维数组（矩阵）处理的库。用来存储和处理大型矩阵，比Python自身的嵌套列表结构要高效的多。本身是由C语言开发，是个很基础的扩展，Python其余的科学计算扩展大部分都是以此为基础。

1.高性能科学计算和数据分析的基础包

2.ndarray，多维数组（矩阵），具有矢量运算能力，快速、节省空间

3.矩阵运算，无需循环，可完成类似Matlab中的矢量运算

4.线性代数、随机数生成

5.import numpy as np

Scipy

Scipy ：基于Numpy提供了一个在Python中做科学计算的工具集，专为科学和工程设计的Python工具包。主要应用于统计、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、稀疏矩阵等，在数学系或者工程系相对用的多一些，和数据处理的关系不大， 我们知道即可，这里不做讲解。

1.在NumPy库的基础上增加了众多的数学、科学及工程常用的库函数

2.线性代数、常微分方程求解、信号处理、图像处理

3.一般的数据处理numpy已经够用

4.import scipy as sp

参考学习资料：

Python、NumPy和SciPy介绍：http://cs231n.github.io/python-numpy-tutorial

NumPy和SciPy快速入门：https://docs.scipy.org/doc/numpy-dev/user/quickstart.html

## 1.2Numpy创建随机数

#（补充）Numpy随机数生成方式<注重学习方法>

#-\*-coding:utf8-\*-

import numpy as np

#1.给定上下限范围内选取整数

x=np.random.randint(0,10,size=(1,10)) #size指定1行10列

y=np.random.randint(0,10,7) #0-10之间的7个数

#2.产生[0,1)中均匀分布的样本值

z=np.random.uniform(-1, 5, size = (3, 4)) # 'size='可省略，（0,1）之间的均匀分布

#3.rand产生均匀分布的样本值,3行4列

t = np.random.rand(3, 4) #产生0-1之间的均匀分布的样本值

t = np.random.rand(3, 4) #产生0-1之间的均匀分布的样本值

#4.产生二项分布的样本值

n, p = 10, .5 # number of trials, probability of each trial试验次数，每次试验的概率

s = np.random.binomial(n, p, 1000) #产生二项分布的样本值

#5.产生高斯分布的样本值

k=np.random.normal(0,0,1,10) #参数顺序：1.均值 2.标准差 3.生成多少样本

#6.产生卡方分布的样本值

s = np.random.chisquare(2,size=(2,3)) #2为自由度

## 1.3Ndarray创建多维数组

ndarray 多维数组(N Dimension Array)

NumPy数组是一个多维的数组对象（矩阵），称为ndarray，具有矢量算术运算能力和复杂的广播能力，并具有执行速度快和节省空间的特点。

注意：ndarray的下标从0开始，且数组里的所有元素必须是相同类型（同构数据多维容器）

ndarray拥有的属性

1.ndim属性：维度个数

2.shape属性：维度大小

3.dtype属性：数据类型

注：size 数组元素的个数

### 1.3.1随机数创建矩阵

1. 实例代码：

# 导入numpy，别名np

import numpy as np

# 生成指定维度大小（3行4列）的随机多维浮点型数据（二维），rand固定区间0.0 ~ 1.0

arr = np.random.rand(3, 4)

print(arr)

print(type(arr))

# 生成指定维度大小（3行4列）的随机多维整型数据（二维），randint()可以指定区间（-1, 5）

arr = np.random.randint(-1, 5, size = (3, 4)) # 'size='可省略

print(arr)

print(type(arr))

# 生成指定维度大小（3行4列）的随机多维浮点型数据（二维），uniform()可以指定区间（-1, 5）产生-1到5之间均匀分布的样本值

arr = np.random.uniform(-1, 5, size = (3, 4)) # 'size='可省略

print(arr)

print(type(arr))

print('维度个数: ', arr.ndim)

print('维度大小: ', arr.shape)

print('数据类型: ', arr.dtype)

运行结果：

[[ 0.09371338 0.06273976 0.22748452 0.49557778]

[ 0.30840042 0.35659161 0.54995724 0.018144 ]

[ 0.94551493 0.70916088 0.58877255 0.90435672]]

<class 'numpy.ndarray'>

[[ 1 3 0 1]

[ 1 4 4 3]

[ 2 0 -1 -1]]

<class 'numpy.ndarray'>

[[ 2.25275308 1.67484038 -0.03161878 -0.44635706]

[ 1.35459097 1.66294159 2.47419548 -0.51144655]

[ 1.43987571 4.71505054 4.33634358 2.48202309]]

<class 'numpy.ndarray'>

维度个数: 2

维度大小: (3, 4)

数据类型: float64

### 1.3.2.ndarray的序列创建

1. np.array(collection)

collection 为 序列型对象(list)、嵌套序列对象(list of list)。

示例代码：

# list序列转换为 ndarray

list = range(10)

arr = np.array(list)

print(arr) # ndarray数据

print(arr.ndim) # 维度个数

print(arr.shape) # 维度大小

# list of list嵌套序列转换为ndarray

lis\_lis = [range(10), range(10)]

arr = np.array(lis\_lis)

print(arr) # ndarray数据

print(arr.ndim) # 维度个数

print(arr.shape) # 维度大小

运行结果：

# list序列转换为 ndarray

[0 1 2 3 4 5 6 7 8 9]

1

(10,)

# list of list嵌套序列转换为 ndarray

[[0 1 2 3 4 5 6 7 8 9]

[0 1 2 3 4 5 6 7 8 9]]

2

(2, 10)

### 1.3.3其他创建形式

2. np.zeros()

指定大小的全0数组。注意：第一个参数是元组，用来指定大小，如(3, 4)。

3. np.ones()

指定大小的全1数组。注意：第一个参数是元组，用来指定大小，如(3, 4)。

4. np.empty()

初始化数组，不是总是返回全0，有时返回的是未初始的随机值（内存里的随机值）。

实战：

# np.zeros

zeros\_arr = np.zeros((3, 4))

# np.ones

ones\_arr = np.ones((2, 3))

# np.empty

empty\_arr = np.empty((3, 3))

# np.empty 指定数据类型

empty\_int\_arr = np.empty((3, 3), int)

print('------zeros\_arr-------')

print(zeros\_arr)

#分别打印上述的值查看结果

### 1.3.4.arange创建

5. np.arange() 和 reshape()

arange() 类似 python 的 range() ，创建一个一维 ndarray 数组。

reshape() 将 重新调整数组的维数。

实战代码：

# np.arange()

arr = np.arange(15) # 15个元素的 一维数组

print(arr)

print(arr.reshape(3, 5)) # 3x5个元素的 二维数组

print(arr.reshape(1, 3, 5)) # 1x3x5个元素的 三维数组

6. np.arange() 和 random.shuffle()

random.shuffle() 将打乱数组序列（类似于洗牌）。

实战代码：

arr = np.arange(15)

print(arr)

np.random.shuffle(arr)

print(arr)

print(arr.reshape(3,5))

### 1.3.5.ndarray的数据类型

1. dtype参数

指定数组的数据类型，类型名+位数，如float64, int32

2. astype方法

转换数组的数据类型

实战：

# 初始化3行4列数组，数据类型为float64

zeros\_float\_arr = np.zeros((3, 4), dtype=np.float64)

print(zeros\_float\_arr)

print(zeros\_float\_arr.dtype) #float64

# astype转换数据类型，将已有的数组的数据类型转换为int32

zeros\_int\_arr = zeros\_float\_arr.astype(np.int32)

print(zeros\_int\_arr)

print(zeros\_int\_arr.dtype) #int32

## 2.1创建等差数列和等比数列方式

2.1补充等比数组和等差数组创建方式

● 先来看一个例子，我们让开始点为0，结束点为0，元素个数为10，看看输出结果。为什么是这样子？难道不都是0吗？

>>> a = np.logspace(0,0,10)

>>> a

array([ 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])

● 因为logspace中，开始点和结束点是10的幂，0代表10的0次方，9代表10的9次方。我们看下面的例子。

>>> a = np.logspace(0,9,10)

>>> a

array([ 1.00000000e+00, 1.00000000e+01, 1.00000000e+02,

1.00000000e+03, 1.00000000e+04, 1.00000000e+05,

1.00000000e+06, 1.00000000e+07, 1.00000000e+08,

1.00000000e+09])

>>> a = np.logspace(0,9,10)

>>> a

array([ 1.00000000e+00, 1.00000000e+01, 1.00000000e+02,

1.00000000e+03, 1.00000000e+04, 1.00000000e+05,

1.00000000e+06, 1.00000000e+07, 1.00000000e+08,

1.00000000e+09])

● 假如，我们想要改变基数，不让它以10为底数，我们可以改变base参数，将其设置为2试试。

>>> a = np.logspace(0,9,10,base=2)

>>> a

array([ 1., 2., 4., 8., 16., 32., 64., 128., 256., 512.])

numpy.linspace是用于创建一个一维数组，并且是等差数列构成的一维数组，它最常用的有三个参数。当然它不只有三个参数，我们通过例子来了解它是如何使用的：

● 首先，我们看一下第一个例子，用到三个参数，第一个参数表示起始点，第二个参数表示终止点，第三个参数表示数列的个数。

>>> a = np.linspace(1,10,10)

>>> a

array([ 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.])

● 我经常用它创建一个元素全部是1的等差数列，或者你也可以让所有的元素为0。

>>> a = np.linspace(1,1,10)

>>> a

array([ 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])

● 我们看一下linspace创建的数组元素的数据格式，当然是浮点型。

>>> a.dtype

dtype('float64')

● 我们还可以使用参数endpoint来决定是否包含终止值，如果不设置这个参数，默认是True。

我们看一下这个例子：注意步长改变了。

>>> a = np.linspace(1,10,10,endpoint=False)

>>> a

array([ 1. , 1.9, 2.8, 3.7, 4.6, 5.5, 6.4, 7.3, 8.2, 9.1])

## 2.2Matrix创建矩阵（二维）

matrix是ndarray的子类，只能生成2维的矩阵，而ndarray(np.array可以生成多个维度的),并且matrix还可以用mat的别名方式生成矩阵：

Code：

import numpy as np

x1=np.mat("1 2;3 4")

print x1

x2=np.matrix("1 2;3 4")

print x2

x3=np.matrix("1,2;3,4")

print x3

x4=np.matrix([[1,2,3,4],[5,6,7,8]])

print x4

x5=np.matrix([[1,2,3,4],[5,6,7,8],[9,10,11,12]])

print x5

x6=np.mat([[1,2,3,4],[5,6,7,8],[9,10,11,12]])

print x6

#验证三维矩阵X是不能被mat或matrix矩阵转换的，这点需要注意

X = np.random.randint(0, 5, [3, 2, 2])

print(X)

print X.shape

#这个会报错ValueError: shape too large to be a matrix.

# Y=np.mat(X)

# print Y.shape

## 2.3Numpy内置函数

1.元素计算函数

np.ceil(): 向上最接近的整数，参数是 number 或 array

np.floor(): 向下最接近的整数，参数是 number 或 array

np.rint(): 四舍五入，参数是 number 或 array

np.isnan(): 判断元素是否为 NaN(Not a Number)，参数是 number 或 array

np.multiply(): 元素相乘，参数是 number 或 array

np.divide(): 元素相除，参数是 number 或 array

np.abs()：元素的绝对值，参数是 number 或 array

np.where(condition, x, y): 三元运算符，x if condition else y

测试代码：

# randn() 返回具有标准正态分布的序列。

arr = np.random.randn(2,3)

print(arr)

print(np.ceil(arr))

print(np.floor(arr))

print(np.rint(arr))

print(np.isnan(arr))

print(np.multiply(arr, arr))

print(np.divide(arr, arr))

print(np.where(arr > 0, 1, -1))

2.元素统计函数

np.mean(), np.sum()：所有元素的平均值，所有元素的和，参数是 number 或 array

np.max(), np.min()：所有元素的最大值，所有元素的最小值，参数是 number 或 array

np.std(), np.var()：所有元素的标准差，所有元素的方差，参数是 number 或 array

np.argmax(), np.argmin()：最大值的下标索引值，最小值的下标索引值，参数是 number 或 array

np.cumsum(), np.cumprod()：返回一个一维数组，每个元素都是之前所有元素的 累加和 和 累乘积，参数是 number 或 array

多维数组默认统计全部维度，axis参数可以按指定轴心统计，值为0则按列统计，值为1则按行统计。

测试代码：

arr = np.arange(12).reshape(3,4)

print(arr)

print(np.cumsum(arr)) # 返回一个一维数组，每个元素都是之前所有元素的 累加和

print(np.sum(arr)) # 所有元素的和

print(np.sum(arr, axis=0)) # 数组的按列统计和

print(np.sum(arr, axis=1)) # 数组的按行统计和

3.元素判断函数

np.any(): 至少有一个元素满足指定条件，返回True

np.all(): 所有的元素满足指定条件，返回True

测试代码：

arr = np.random.randn(2,3)

print(arr)

print(np.any(arr > 0))

print(np.all(arr > 0))

4.元素去重排序函数

np.unique():找到唯一值并返回排序结果，类似于Python的set集合

示例代码：

#[[1, 2, 1], [2, 3, 4]]这样声明也可以

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

print(arr)

print(np.unique(arr))

## 3.1Numpy线性代数和矩阵运算

3.1 Numpy线性代数和矩阵运算

import numpy as np

#矩阵乘法

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

y=np.array([[6,23],[-1,7],[8,9]])

print(x)

print(y)

print(x.dot(y))

print(np.dot(x,y))

#矩阵分解运算-矩阵求逆及转置、行列式求解

from numpy.linalg import inv,qr,svd,eig

X=np.random.randn(5,5)

mat=X.T.dot(X)

print(inv(mat)) #求解方阵的逆

mat.dot(inv(mat))

q,r=qr(mat)#计算矩阵的qr分解-特征值和特征向量分解

print(q,r)

print q.dot(r) #根据矩阵的特征值和特征向量得到原矩阵

#qr分解是求解一般的矩阵全部特征值最有效的方法，首先经过矩阵的正交相似变换为Hessen矩阵，再通过QR方法求特征值和特征向量，得到的是一个正规正交矩阵Q（m\*m）和一个上三角矩阵R（m\*m）

#eig方阵的特征值和特征向量

value,vector=eig(mat)#计算矩阵的qr分解-特征值和特征向量分解

print(value,vector)

补充：利用特征值和特征向量求解原矩阵

matrix1=np.matrix([[1,2,3], [4,5,6], [7,8,9]])

e,v=eig(matrix1)

# 根据特征值和特征向量得到原矩阵

y = v \* np.diag(e) \* np.linalg.inv(v)

# 根据特征值和特征向量得到原矩阵  
y = v.dot(np.dot(np.diag(e),np.linalg.inv(v)))

print y

print matrix1

#svd奇异值分解

U,sigma,VT=svd(X)

#使用A=mat([[1,2,3],[4,5,6]])也可测试

print("U",U)

print("sigma",sigma)

print("VT",VT)

#转换为原来的矩阵，一定记住sigma转化为单位矩阵

print U.dot(np.diag(sigma).dot(VT))

print np.allclose(matrix1, np.dot(U, np.dot(np.diag(sigma), VT)))

# Python数据科学基础第二部分-Pandas

# 2.Pandas

## 1.1Pandas介绍

Pandas

什么是Pandas?

##Pandas的名称来自于面板数据（panel data）和Python数据分析（data analysis）。

Pandas是一个强大的分析**结构化数据的工具集**，基于NumPy构建，提供了 高级数据结构 和 数据操作工具，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。

1.一个强大的分析和操作大型**结构化数据集**所需的工具集

2.基础是NumPy，提供了高性能矩阵的运算

3.提供了大量能够快速便捷地处理数据的函数和方法

4.应用于数据挖掘，数据分析

5.提供数据清洗功能

http://pandas.pydata.org

## 1.2Series

1.Pandas的数据结构

import pandas as pd

Pandas有两个最主要也是最重要的数据结构： Series 和 DataFrame

2.Series

简要介绍

1.Series是一种类似于一维数组的 对象，由一组数据（各种NumPy数据类型）以及一组与之对应的索引（数据标签）组成。

2..类似一维数组的对象

由数据和索引组成

索引(index)在左，数据(values)在右

索引是自动创建的

2.1. 通过list构建Series

ser\_obj = pd.Series(range(10))

示例代码：

# 通过list构建Series

ser\_obj = pd.Series(range(10, 20))

print(ser\_obj.head(3))

print(ser\_obj) #打印全部的9个值

print(type(ser\_obj))

2.2 获取数据和索引

ser\_obj.index 和 ser\_obj.values

示例代码：

# 获取数据

print(ser\_obj.values)

# 获取索引

print(ser\_obj.index)

运行结果：

[10 11 12 13 14 15 16 17 18 19]

RangeIndex(start=0, stop=10, step=1)

3. 通过索引获取数据

ser\_obj[idx]

示例代码：

#通过索引获取数据

print(ser\_obj[0])

print(ser\_obj[8])

运行结果：

10

18

4. 通过dict构建Series

示例代码：

# 通过dict构建Series

year\_data = {2001: 17.8, 2002: 20.1, 2003: 16.5}

ser\_obj2 = pd.Series(year\_data)

print(ser\_obj2.head())

print(ser\_obj2.index) #Int64Index([2001, 2002, 2003], dtype='int64')

S4=pd.Series(dict(zip([1,2,3],["a","b","c"])))  
 print(S4)

6.name属性

对象名：ser\_obj.name

对象索引名：ser\_obj.index.name

示例代码：

# name属性

ser\_obj2.name = 'temp'

ser\_obj2.index.name = 'year'

print(ser\_obj2.head())

举例：

obj=Series([4,7,-5,3])

print(obj)

print(obj.values)

print(obj.index)

obj2=Series([4,7,-5,3],index=['d','b','a','c'])

print(obj2)

obj2['a']

obj2['d']=6

obj2['c','a','b']

obj2[obj2>0]

obj2[obj2\*2]

np.exp(obj2)

print('b' in obj2)

sdata={1:'a',2:'b',3:'c'}

obj3=Series(sdata)

print(obj3)

states=[3,2,1]

obj4=Seroes(sdata,index=states)

print(obj4)

pd.isnull(obj4)

pd.notnull(obj4)

obj4.isnull()

#series会在自动对齐不同索引数据

print(obj3+obj4)

（补充：）计算series唯一值：

ubique 唯一值

value\_count 返回一个Series，其索引为唯一值，其值为频率，按计数值降序排序

obj=Series(['c','a','d','a','a','b','b','c','c'])

unique=obj.unique()

print(unique)

print(obj.value\_counts) #计算出现频率

## 1.3Pandas-DataFrame

DataFrame是一个表格型的数据结构，它含有一组有序的列，每列可以是不同类型的值。DataFrame既有行索引也有列索引，它可以被看做是由Series组成的字典（共用同一个索引），数据是以二维结构存放的。

1.类似多维数组/表格数据 (如，excel, R中的data.frame)

2.每列数据可以是不同的类型

3索引包括列索引和行索引

1. 通过ndarray构建DataFrame

示例代码：

import numpy as np

# 通过ndarray构建DataFrame

array = np.random.randn(5,4)

print(array)

df\_obj = pd.DataFrame(array)

print(df\_obj.head())

可以通过columns改变列名

可以通过index改变行的名字

2. 通过dict构建DataFrame

示例代码：

# 通过dict构建DataFrame

dict\_data = {'A': 1,

'B': pd.Timestamp('20170426'),

'C': pd.Series(1, index=list(range(4)),dtype='float32'),

'D': np.array([3] \* 4,dtype='int32'),

'E': ["Python","Java","C++","C"],

'F': 'ITCast' }

#print dict\_data

df\_obj2 = pd.DataFrame(dict\_data)

print(df\_obj2)

3. 通过列索引获取列数据（Series类型）

df\_obj[col\_idx] 或 df\_obj.col\_idx

示例代码：

# 通过列索引获取列数据

print(df\_obj2['A'])

print(type(df\_obj2['A']))

print(df\_obj2.A)

4. 增加列数据

df\_obj[new\_col\_idx] = data

类似Python的 dict添加key-value

示例代码：

# 增加列

df\_obj2['G'] = df\_obj2['D'] + 4

print(df\_obj2.head())

运行结果：

A B C D E F G

0 1.0 2017-01-02 1.0 3 Python ITCast 7

1 1.0 2017-01-02 1.0 3 Java ITCast 7

2 1.0 2017-01-02 1.0 3 C++ ITCast 7

3 1.0 2017-01-02 1.0 3 C ITCast 7

5. 删除列

del df\_obj[col\_idx]

示例代码：

# 删除列

del(df\_obj2['G'] )

print(df\_obj2.head())

运行结果：

A B C D E F

0 1.0 2017-01-02 1.0 3 Python ITCast

1 1.0 2017-01-02 1.0 3 Java ITCast

2 1.0 2017-01-02 1.0 3 C++ ITCast

3 1.0 2017-01-02 1.0 3 C ITCast

例子：obj2使用的是上面的数据

ix用法

方法：obj.ix[:,val] #选择单个列

obj.ix[val1,val2] #同时选取行和列

print(df\_obj2.ix['A']) #通过ix获取指定索引对应的行信息，ix表示索引字段

print(df\_obj2.ix[:,'A'])#选取所有航的指定'A'列的数据

实例代码

import pandas as pd  
import numpy as np  
df1=pd.DataFrame(np.random.randn(5,5))  
df2=pd.DataFrame(np.random.randn(5,5),columns=["one","two","three","four","five"],  
 index=["a","b","c","d","e"])  
print df2  
#属性信息  
print df2.size #25  
print df2.shape #(5, 5)  
print df2.ndim #2  
print df2.columns #Index([u'one', u'two', u'three', u'four', u'five'], dtype='object')  
print df2.index #Index([u'a', u'b', u'c', u'd', u'e'], dtype='object')  
#获取当前列  
print df2["one"]  
print df2.one  
#增加列  
df2["seven"]=["apple","pear","banana","orange","oooo"]  
print df2  
#删除列  
  
#指定行列中的元素  
print df2.ix[:,"one"]  
print df2.ix["a",:]  
print df2.ix["a","one"]  
#iloc和loc属性  
#iloc----根据下标索引进行访问，可以访问对应的行和列元素  
print df2.iloc[:3]  
print df2.iloc[[1,2],[1,2]]  
#loc----根据下标进行访问，可以访问对应行或列元素  
print df2.loc["a":"d"]  
print df2.loc["a",:]

总结：可以输入给DataFrame构造器数据的有如下类型：

（1）二维ndarry

（2）由数组、列表、元祖组成的字典

（3）由Series组成的字典

（4）由字典组成的字典

（5）另外一个DataFrame

案例补充：

import pandas as pd  
import numpy as np  
dfl = pd.DataFrame(np.random.randn(5,4), columns=list('ABCD'),  
 index=pd.date\_range('20130101',periods=5))  
  
print(dfl)  
# A B C D  
# 2013-01-01 1.075770 -0.109050 1.643563 -1.469388  
# 2013-01-02 0.357021 -0.674600 -1.776904 -0.968914  
# 2013-01-03 -1.294524 0.413738 0.276662 -0.472035  
# 2013-01-04 -0.013960 -0.362543 -0.006154 -0.923061  
# 2013-01-05 0.895717 0.805244 -1.206412 2.565646  
print(dfl.loc['20130102':'20130104'])  
print(dfl.loc['20130102':'20130104'])  
print(dfl.loc['20130102':'20130104',"A":"C"])  
print(dfl.iloc[2:4])  
print(dfl.iloc[2:4,0:3])  
print(dfl.loc[:, lambda df: ['A', 'B']])  
pd.read\_pickle()

#补充1：索引对象是不可以修改的：如

obj=Series(range(3),index=['a','b','c'])

index=obj.index

print(index)

print(index[1:])

index[1]='d' #会报错

这个用法会保证index对象在多个数据结构之间安全共享。

#补充2：pandas对象重新索引

obj=Series(range(3),index=['a','b','c'])

obj2=obj.reindex(['c','a','b','f'])

obj3=obj.reindex(['c','a','b','f'],fill\_value=0)

obj4=obj.reindex(['c','a','b','f'],method='ffill')

method可选：

#ffill或pad前向填充值

#bfill或backfill 后向填充值

#补充3：丢弃指定轴上的项,按照索引删除

obj=Series(range(3),index=['a','b','c'])

new\_obj=obj.drop('c')

print(new\_obj)

new\_obj2=obj.drop(['a','b'])

## 2.1Pandas的对齐运算

是数据清洗的重要过程，可以按索引对齐进行运算，如果没对齐的位置则补NaN，最后也可以填充NaN

Series的对齐运算

1. Series 按行、索引对齐

示例代码：

s1 = pd.Series(range(10, 20), index = range(10))

s2 = pd.Series(range(20, 25), index = range(5))

print('s1: ' )

print(s1)

print('s2: ')

print(s2)

2.Series的对齐运算

示例代码：

# Series 对齐运算

s1 + s2 #5-9索引的值为Nan

DataFrame的对齐运算

1. DataFrame按行、列索引对齐

示例代码：

df1 = pd.DataFrame(np.ones((2,2)), columns = ['a', 'b'])

df2 = pd.DataFrame(np.ones((3,3)), columns = ['a', 'b', 'c'])

print('df1: ')

print(df1)

print('')

print('df2: ')

print(df2)

2.DataFrame的对齐运算

示例代码：

# DataFrame对齐操作

df1 + df2

运行结果：

a b c

0 2.0 2.0 NaN

1 2.0 2.0 NaN

2 NaN NaN NaN

填充未对齐的数据进行运算

1. fill\_value

使用add, sub, div, mul的同时，

通过fill\_value指定填充值，未对齐的数据将和填充值做运算

示例代码：

print(s1)

print(s2)

s1.add(s2, fill\_value = -1)

print(df1)

print(df2)

df1.sub(df2, fill\_value = 2.)

## 3.1Pandas的函数应用

处理缺失数据

示例代码：

df\_data = pd.DataFrame([np.random.randn(3), [1., 2., np.nan],

[np.nan, 4., np.nan], [1., 2., 3.]])

print(df\_data.head())

运行结果：

0 1 2

0 -0.281885 -0.786572 0.487126

1 1.000000 2.000000 NaN

2 NaN 4.000000 NaN

3 1.000000 2.000000 3.000000

1. 判断是否存在缺失值：isnull()

示例代码：

# isnull

print(df\_data.isnull())

运行结果：

0 1 2

0 False False False

1 False False True

2 True False True

3 False False False

2. 丢弃缺失数据：dropna()

根据axis轴方向，丢弃包含NaN的行或列。 示例代码：

# dropna

print(df\_data.dropna())

print(df\_data.dropna(axis=1))

运行结果：

0 1 2

0 -0.281885 -0.786572 0.487126

3 1.000000 2.000000 3.000000

1

0 -0.786572

1 2.000000

2 4.000000

3 2.000000

3. 填充缺失数据：fillna()

示例代码：

# fillna

print(df\_data.fillna(-100.))

运行结果：

0 1 2

0 -0.281885 -0.786572 0.487126

1 1.000000 2.000000 -100.000000

2 -100.000000 4.000000 -100.000000

3 1.000000 2.000000 3.000000

*# drop操作- Drop specified labels from rows or columns.***import** pandas **as** pd  
**import** numpy **as** np  
print(np.random.randn(3))  
df1=pd.DataFrame([np.random.randn(3),[np.nan,2,2],[1,np.nan,3]],  
 columns=(**"A"**,**"B"**,**"C"**),index=(**"a"**,**"b"**,**"c"**))  
print(df1.drop([**"A"**],axis=1))  
*# print(df1.drop([0]))*print(df1.drop([**"a"**]))

（补充）

applymappply 和 applymap

1. 可直接使用NumPy的函数

示例代码：

# Numpy ufunc 函数

df = pd.DataFrame(np.random.randn(5,4) - 1)

print(df)

print(np.abs(df))

2. 通过apply将函数应用到列或行上

示例代码：

# 使用apply应用行或列数据

#f = lambda x : x.max()

print(df.apply(lambda x : x.max()))

注意指定轴的方向，默认axis=0，方向是列

示例代码：

# 指定轴方向，axis=1，方向是行

print(df.apply(lambda x : x.max(), axis=1))

3. 通过applymap将函数应用到每个数据上

示例代码：

# 使用applymap应用到每个数据

f2 = lambda x : '%.2f' % x

print(df.applymap(f2))

排序（补充）

1. 索引排序

sort\_index()

排序默认使用升序排序，ascending=False 为降序排序

示例代码：

# Series

s4 = pd.Series(range(10, 15), index = np.random.randint(5, size=5))

print(s4)

# 索引排序

s4.sort\_index() # 0 0 1 3 3

对DataFrame操作时注意轴方向

示例代码：

# DataFrame

df4 = pd.DataFrame(np.random.randn(3, 5),

index=np.random.randint(3, size=3),

columns=np.random.randint(5, size=5))

print(df4)

df4\_isort = df4.sort\_index(axis=1, ascending=False)

print(df4\_isort) # 4 2 1 1 0

2. 按值排序

sort\_values(by='column name')

根据某个唯一的列名进行排序，如果有其他相同列名则报错。

示例代码：

# 按值排序

df4\_vsort = df4.sort\_values(by=0, ascending=False)

print(df4\_vsort)

## 3.2层次索引

8.层级索引（hierarchical indexing）

8.1下面创建一个Series， 在输入索引Index时，输入了由两个子list组成的list，第一个子list是外层索引，第二个list是内层索引。

示例代码：

import pandas as pd

import numpy as np

ser\_obj = pd.Series(np.random.randn(12),index=[

['a', 'a', 'a', 'b', 'b', 'b', 'c', 'c', 'c', 'd', 'd', 'd'],

[0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2]

])

print(ser\_obj)

运行结果：

a 0 0.099174

1 -0.310414

2 -0.558047

b 0 1.742445

1 1.152924

2 -0.725332

c 0 -0.150638

1 0.251660

2 0.063387

d 0 1.080605

1 0.567547

2 -0.154148

dtype: float64

8.2MultiIndex索引对象

打印这个Series的索引类型，显示是MultiIndex

直接将索引打印出来，可以看到有levels,和labels两个信息。lavels表示两个层级中分别有那些标签，labels是每个位置分别是什么标签。

示例代码：

print(type(ser\_obj.index))

print(ser\_obj.index)

运行结果：

<class 'pandas.indexes.multi.MultiIndex'>

MultiIndex(levels=[['a', 'b', 'c', 'd'], [0, 1, 2]],

labels=[[0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3], [0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2]])

8.3选取子集

根据索引获取数据。因为现在有两层索引，当通过外层索引获取数据的时候，可以直接利用外层索引的标签来获取。

当要通过内层索引获取数据的时候，在list中传入两个元素，前者是表示要选取的外层索引，后者表示要选取的内层索引。

1. 外层选取：

ser\_obj['outer\_label']

示例代码：

# 外层选取

print(ser\_obj['c'])

运行结果：

0 -1.362096

1 1.558091

2 -0.452313

dtype: float64

2. 内层选取：

ser\_obj[:, 'inner\_label']

示例代码：

# 内层选取

print(ser\_obj[:, 2])

运行结果：

a 0.826662

b 0.015426

c -0.452313

d -0.051063

dtype: float64

常用于分组操作、透视表的生成等

8.4交换分层顺序

1. swaplevel()

.swaplevel( )交换内层与外层索引。

示例代码：

print(ser\_obj.swaplevel())

运行结果：

0 a 0.099174

1 a -0.310414

2 a -0.558047

0 b 1.742445

1 b 1.152924

2 b -0.725332

0 c -0.150638

1 c 0.251660

2 c 0.063387

0 d 1.080605

1 d 0.567547

2 d -0.154148

dtype: float64

2.交换并排序分层

sortlevel()

.sortlevel( )先对外层索引进行排序，再对内层索引进行排序，默认是升序。

示例代码：

# 交换并排序分层

print(ser\_obj.swaplevel().sortlevel())

运行结果：

0 a 0.099174

b 1.742445

c -0.150638

d 1.080605

1 a -0.310414

b 1.152924

c 0.251660

d 0.567547

2 a -0.558047

b -0.725332

c 0.063387

d -0.154148

dtype: float64

## 4.1Pandas统计计算和描述

4.1示例代码：

import numpy as np

import pandas as pd

df\_obj = pd.DataFrame(np.random.randn(5,4), columns = ['a', 'b', 'c', 'd'])

print(df\_obj)

运行结果：

a b c d

0 1.469682 1.948965 1.373124 -0.564129

1 -1.466670 -0.494591 0.467787 -2.007771

2 1.368750 0.532142 0.487862 -1.130825

3 -0.758540 -0.479684 1.239135 1.073077

4 -0.007470 0.997034 2.669219 0.742070

4.2常用的统计计算

sum, mean, max, min…

axis=0 按列统计，axis=1按行统计

skipna 排除缺失值， 默认为True

示例代码：

df\_obj.sum()

df\_obj.max()

df\_obj.min(axis=1, skipna=False)#按行填充，NA值自动排除

运行结果：

a 0.605751

b 2.503866

c 6.237127

d -1.887578

dtype: float64

a 1.469682

b 1.948965

c 2.669219

d 1.073077

dtype: float64

0 -0.564129

1 -2.007771

2 -1.130825

3 -0.758540

4 -0.007470

dtype: float64

4.3常用的统计描述

describe 产生多个统计数据

示例代码：

print(df\_obj.describe())

运行结果：

a b c d

count 5.000000 5.000000 5.000000 5.000000

mean 0.180305 0.106488 0.244978 0.178046

std 0.641945 0.454340 1.064356 1.144416

min -0.677175 -0.490278 -1.164928 -1.574556

25% -0.064069 -0.182920 -0.464013 -0.089962

50% 0.231722 0.127846 0.355859 0.190482

75% 0.318854 0.463377 1.169750 0.983663

max 1.092195 0.614413 1.328220 1.380601

4.4Query等函数用法参考：

<http://pandas.pydata.org/pandas-docs/version/0.18/api.html>

from numpy.random import randn  
from pandas import DataFrame  
df = DataFrame(randn(10, 2), columns=list('ab'))#【“a”,"b"】  
print df.query('a > b')  
print df[df.a > df.b]

描述和汇总的方法汇总：(大家自行测试这些函数)

count 非Nan数量

describe 针对个列汇总统计

min和max 最大最小值

argmin、argmax 计算最大值或最小值对应的索引位置

quantile 计算样本的分位数（0-1）

mean 均值

median 中位数

mad 平均绝对离差

var 样本方差

std 样本的标准差

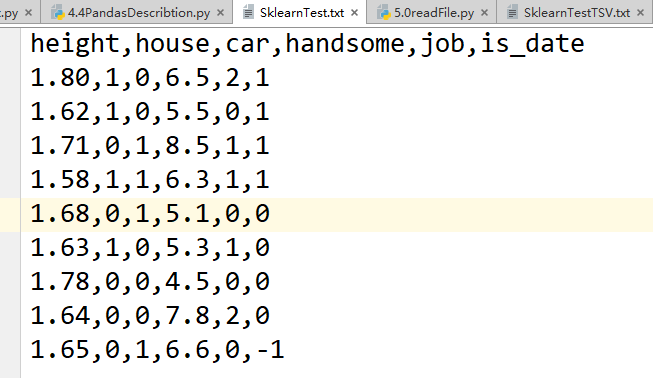
skew 样本值的偏度----正态分布的偏度为0

kurt 样本值的峰度------正态分布的峰度为3

cumsum样本值的累计和

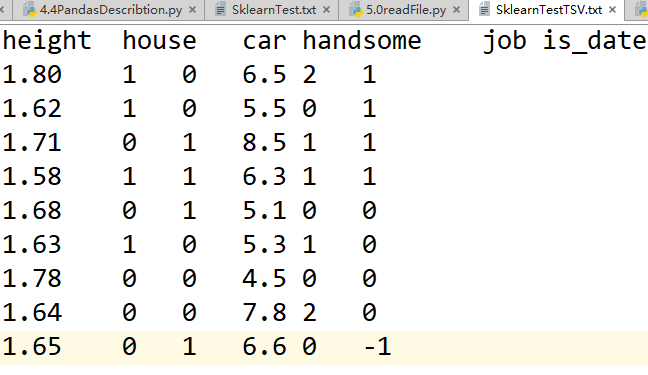
## 5.1读取文件

### 5.1.1读取CSV文件



import pandas as pd  
content=pd.read\_csv("SklearnTest.txt", sep=',')  
print content  
print content.index #RangeIndex(start=0, stop=9, step=1)  
print content.columns #Index([u'height', u'house', u'car', u'handsome', u'job', u'is\_date'], dtype='object')  
print content.info()  
# RangeIndex: 9 entries, 0 to 8  
# Data columns (total 6 columns):  
# height 9 non-null float64  
# house 9 non-null int64  
# car 9 non-null int64  
# handsome 9 non-null float64  
# job 9 non-null int64  
# is\_date 9 non-null int64  
# dtypes: float64(2), int64(4)  
# memory usage: 504.0 bytes  
print content.size #54  
print content.shape #(9, 6)  
print content.ndim #2  
print content.memory\_usage()  
  
content1=content.query("is\_date!=-1")  
print content1.drop("is\_date", axis=1)  
print content1  
# height house car handsome job is\_date  
# 0 1.80 1 0 6.5 2 1  
# 1 1.62 1 0 5.5 0 1  
# 2 1.71 0 1 8.5 1 1  
# 3 1.58 1 1 6.3 1 1  
# 4 1.68 0 1 5.1 0 0  
# 5 1.63 1 0 5.3 1 0  
# 6 1.78 0 0 4.5 0 0  
# 7 1.64 0 0 7.8 2 0  
# 8 1.65 0 1 6.6 0 -1

### 5.1.1读取TSV文件



import pandas as pd  
content=pd.read\_csv("SklearnTestTSV.txt", sep='\t')  
print content  
print content.index #RangeIndex(start=0, stop=9, step=1)  
print content.columns #Index([u'height', u'house', u'car', u'handsome', u'job', u'is\_date'], dtype='object')  
print content.info()  
# RangeIndex: 9 entries, 0 to 8  
# Data columns (total 6 columns):  
# height 9 non-null float64  
# house 9 non-null int64  
# car 9 non-null int64  
# handsome 9 non-null float64  
# job 9 non-null int64  
# is\_date 9 non-null int64  
# dtypes: float64(2), int64(4)  
# memory usage: 504.0 bytes  
print content.size #54  
print content.shape #(9, 6)  
print content.ndim #2  
print content.memory\_usage()  
#数据预处理  
content1=content.query("is\_date!=-1")  
print content1.drop("is\_date", axis=1)  
print content1  
# height house car handsome job is\_date  
# 0 1.80 1 0 6.5 2 1  
# 1 1.62 1 0 5.5 0 1  
# 2 1.71 0 1 8.5 1 1  
# 3 1.58 1 1 6.3 1 1  
# 4 1.68 0 1 5.1 0 0  
# 5 1.63 1 0 5.3 1 0  
# 6 1.78 0 0 4.5 0 0  
# 7 1.64 0 0 7.8 2 0  
# 8 1.65 0 1 6.6 0 -1

# Python数据科学基础第三部分-绘图基础

# 3.Matplotlib

## 1.1-Matplotlib&Seaborn数据可视化库

Matplotlib 是一个 Python 的 2D绘图库，通过 Matplotlib，开发者可以仅需要几行代码，便可以生成绘图，直方图，功率谱，条形图，错误图，散点图等。

http://matplotlib.org

whl文件下载地址（pypi）：https://pypi.python.org/pypi/matplotlib/

特点：

1.用于创建出版质量图表的绘图工具库

2.目的是为Python构建一个Matlab式的绘图接口

3.import matplotlib.pyplot as plt

4.pyploy模块包含了常用的matplotlib API函数

Demo:

plt.plot([1,2,3],[3,2,1])

plt.show()

plt.plot([1,2,3],[-3,-2,-1])

plt.show()

plt.show()可类比与spark中的Action算子，程序只有触发Action算子才会真正的执行计算或输出，而常见的filter、map操作都会暂存起来，当有action操作的时候触发。

## 1.2入门案例

需求：1.打印Matplotlib版本

2.绘制y=x+5和y=2x+5两条曲线

#-\*-coding:utf8-\*-

#1.打印当前matplotlib的版本信息

import matplotlib as mpl

print "matplotlib version",mpl.\_\_version\_\_ #查看当前版本

#2.plot绘制曲线图

import matplotlib.pyplot as plt

import numpy as np

#1.通过Numpy的linspace函数指定横坐标，同时规定起点和终点分别是0和20

x=np.linspace(0,20)

#通过下面画线

plt.plot(x,.5+x)

plt.plot(x,5+2\*x,'--')

#这里既可以用savefig()函数把图形保存到一个文件中，也可以通过show()函数将图像显示在屏幕上

plt.show()

注意：标题加中文u+fontproperties='SimHei'这个

plt.title(u"一次函数",fontproperties='SimHei')

## 2.1 figure对象及多图绘制

Matplotlib的图像均位于figure对象中

创建figure：fig = plt.figure()

#需求1：示例代码：

# 引入matplotlib包

import matplotlib.pyplot as plt

import numpy as np

# 创建figure对象

fig = plt.figure()

运行结果：

<matplotlib.figure.Figure at 0x11a2dd7b8>

#需求2：在一个图中采用面向对象的方式绘制两幅图

#绘制第1张图

fig=plt.figure()

ax1=fig.add\_subplot(1,1,1)

ax1.plot([1,2,3],[3,2,1])

#绘制第2张图

fig2=plt.figure()

ax2=fig2.add\_subplot(1,1,1)

ax2.plot([1,2,3],[-3,-2,-1])

plt.show()

## 2.2通过figure对象创建子图(可以用来做图形的对比)

fig.add\_subplot(a, b, c)

a,b 表示将fig分割成 a\*b 的区域

c 表示当前选中要操作的区域，

注意：从1开始编号（不是从0开始）

plot 绘图的区域是最后一次指定subplot的位置 (jupyter notebook里不能正确显示)

#需求：创建子图，并绘制+-x和+-x的拟合直线图（即：给定相同坐标的x）

示例代码：

#采用的figure对象

fig=plt.figure()

ax1 = fig.add\_subplot(2, 2, 1)

ax2 = fig.add\_subplot(2, 2, 2)

ax3 = fig.add\_subplot(2, 2, 3)

ax4 = fig.add\_subplot(2, 2, 4)

# 在subplot上作图

x = np.arange(1, 100)

ax1.plot(x, x)

ax2.plot(x, -x)

ax3.plot(-x, x)

ax4.plot(-x, -x)

plt.show()

#采用plt的方式

x=np.arange(1,100)

plt.subplot(221)

plt.plot(x,x)

plt.subplot(222)

plt.plot(x,-x)

plt.show()

## 3.1 figure绘制各种图形

#需求：使用figure的方式创建子图并做出直方图、散点图、折线图、饼状图、小提琴图

# 指定切分区域的位置

ax1 = fig.add\_subplot(2,2,1)

ax2 = fig.add\_subplot(2,2,2)

ax3 = fig.add\_subplot(2,2,3)

ax4 = fig.add\_subplot(2,2,4)

plt.show()

(下面的ax1，ax2都是在上面定义的基础上进行的)

12.3直方图：hist

示例代码：

#或者使用plt.hist(np.random.randn(100), bins=10, color='b', alpha=0.3)

ax1.hist(np.random.randn(100), bins=10, color='b', alpha=0.3)

12.4散点图：scatter

散点图是分析数据相关性常用的方法，下面绘制了余弦曲线的散点图

示例代码：

# 绘制散点图

x = np.arange(0,,2\*np.pi,0.1)

y = np.cos(x)

#使用plt.scatter(x, y)

ax2.scatter(x, y)

12.4 折线图

#画出-10到10区间的二次函数图

#100改为10就能看出折线图

x=np.linspace(-10,10,100) #平均分为100份

y=x\*\*2

ax3.plot(x,y)

12.5饼状图

饼状图显示一个数据中各项的大小与各项和的比例

饼状图中显示为在整个饼状图中的比例

#饼状图代码

labelx=['A','B','C','D']

fracs=[15,30,45,10]

#1.必须设置比例为1:1才能显示为圆形

plt.axes(aspect=1)

#2.加每一块所占有具体比例的值autopct

#3.突出显示explode

explode=[0,0.2,0,0] #块B远离了饼状图中心

#3.加阴影shadow

ax4.pie(x=fracs,labels=labelx,autopct='%.1f%%',explode=explode,shadow=True)

plt.show()

12.6箱线图

#箱线图

data=np.random.normal(size=1000,loc=0,scale=1)

#调整异常点的形状sym='o'

#whis参数表示:虚线的长度，默认1.5(比例)，盒子距离上下四分位数的距离

#距离越大虚线越长，设置成0.5和100分别观察

ax4.boxplot(data,sym='o',whis=1.5)

参考代码：<https://matplotlib.org/gallery/statistics/barchart_demo.html>

## 4.1网格Grid

#绘制1-5的随机数的二次函数

#以网格的形式作为背景，可以知道坐标轴的位置

y=np.arange(1,5)

plt.plot(y,y\*\*2)

plt.grid(True)#True表示显示网格

plt.grid(color='r',linewidth=2,linestyle="-")

plt.show()

#面向对象的创建方式：

#plt在交互中使用很多，面向对象不适合用交互式的方式

y=np.arange(1,5)

fig=plt.figure()

ax=fig.add\_subplot(111)

ax.plot(y,y\*\*2)

ax.grid(color='r'，linestyle='-')

plt.show()

## 5.1图例的用法

x=np.arange(1,11)

plt.plot(x,x\*\*2,label='Normal')

plt.plot(x,x\*\*3,label="Fast")

plt.plot(x,x\*\*4,label="Faster")

#方式1

#ncol=3扁平的效果，排为一列

plt.legend(loc=2,ncol=3)#1是右上角2是左上角3左下角4右下角0是best

# 方式2：

# plt.legend(['Normal',"Fast","Faster"])

plt.show()

## 6.1颜色、标记、线型

ax.plot(x, y, ‘r--’)

等价于ax.plot(x, y, linestyle=‘--’, color=‘r’)

示例代码：

import matplotlib.pyplot as plt

import numpy as np

fig, axes = plt.subplots(2)

axes[0].plot(np.random.randint(0, 100, 50), 'ro--')

# 等价

axes[1].plot(np.random.randint(0, 100, 50), color='r', linestyle='dashed', marker='o')

常用的颜色、标记、线型：<>

marker

. point

, pixel

o circle

v 下三角形

^ 上三角形

< 左三角形

color

b：blue

g:green

r:red

c:cyan

m:magenta

y:yellow

k:black

w:white

linestyle

- or solid 粗线

-- or dashed dashed line

-. or dashdot dash-dotted

: or dotted dotted line

'None' draw nothing

' ' or '' 什么也不绘画

## 7.1刻度、标签、图例（补充）

面向对象和面向过程两种方式在设置方面的区别：

设置刻度范围

plt.xlim(), plt.ylim()

ax.set\_xlim(), ax.set\_ylim()

设置显示的刻度

plt.xticks(), plt.yticks()

ax.set\_xticks(), ax.set\_yticks()

设置刻度标签

ax.set\_xticklabels(), ax.set\_yticklabels()

设置坐标轴标签

plt.xlabel(),plt.ylabel()

ax.set\_xlabel(), ax.set\_ylabel()

设置标题

ax.set\_title()

图例

ax.plot(label=‘legend’)

ax.legend(), plt.legend()

loc=‘best’和loc=0一致：自动选择放置图例最佳位置

示例代码：

import matplotlib.pyplot as plt

import numpy as np

fig, ax = plt.subplots(1)

ax.plot(np.random.randn(1000).cumsum(), label='line0')

# 设置刻度

#plt.xlim([0,500])

ax.set\_xlim([0, 800])

# 设置显示的刻度

#plt.xticks([0,500])

ax.set\_xticks(range(0,500,100))

# 设置刻度标签

ax.set\_yticklabels(['Jan', 'Feb', 'Mar'])

# 设置坐标轴标签

#plt.xlabel('Number')

#plt.ylabel('Month')

ax.set\_xlabel('Number')

ax.set\_ylabel('Month')

# 设置标题

#plt.title('Example')

ax.set\_title('Example')

**#案例部分：**

import matplotlib.pyplot as plt  
def fig\_way():  
 fig = plt.figure()  
 ax = fig.add\_subplot(111)  
 ax.plot([1, 2, 3], [3, 2, 1])  
 ax.set\_xlim(-10, 10)  
 ax.set\_ylim(-10, 10)  
 ax.set\_title("This is Demo!")  
 ax.set\_xlabel("X")  
 ax.set\_ylabel("Y")  
 ax.legend(["best"], loc=0)  
 plt.show()  
  
def plt\_way():  
 plt.figure()  
 ax1 = plt.subplot(111)  
 plt.sca(ax1)  
 plt.plot([1, 2, 3], [3, 2, 1])  
 plt.xlabel("X")  
 plt.ylabel("Y")  
 plt.title("This is Demo!")  
 plt.xlim(-10, 10)  
 plt.ylim(-10, 10)  
 plt.legend(["best"], loc=0)  
 plt.show()  
  
if \_\_name\_\_=="\_\_main\_\_":  
 # fig\_way()  
 plt\_way()

## 8.1Matplotlib综合案例分析

import numpy as np

import matplotlib.pyplot as plt

x=np.linspace(0,2\*np.pi,500)

y1=np.sin(x)

y2=np.cos(x)

y3=np.sin(x\*x)

plt.figure(1) #创建图形

#创建三个轴

ax1=plt.subplot(2,2,1) #第一行第一列

ax2=plt.subplot(2,2,2) #第一行第二列

ax3=plt.subplot(2,1,2) #第二行

plt.sca(ax1) #选择ax1

plt.plot(x,y1,color='red')

plt.ylim(-1.2,1.2)

plt.sca(ax2) #选择ax2

plt.plot(x,y2,'b--')

plt.ylim(-1.2,1.2)

plt.sca(ax3) #选择ax3

plt.plot(x,y3,'g--',label="sin x\*x")

plt.ylim(-1.2,1.2)

plt.legend(loc=0)

plt.title("Programe")

plt.show()

#补充

plt.subplots()

1.同时返回新创建的figure和subplot对象数组

2.生成2行2列subplot:fig, subplot\_arr = plt.subplots(2,2)

示例代码：

import matplotlib.pyplot as plt

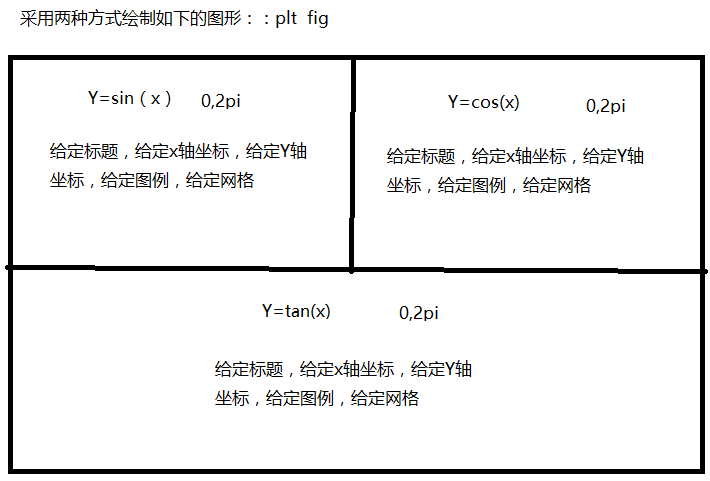
import numpy as np

fig, subplot\_arr = plt.subplots(2,2)

# bins 为显示个数，一般小于等于数值个数

subplot\_arr[1,0].hist(np.random.randn(100), bins=10, color='b', alpha=0.3)

plt.show()



**案例部分1：**

#1.面向对象的方式  
import matplotlib.pyplot as plt  
import numpy as np  
#2.绘制画布  
fig=plt.figure()  
ax1 = fig.add\_subplot(221)  
ax2 = fig.add\_subplot(222)  
ax3 = fig.add\_subplot(212)  
#3.准备数据  
X=np.linspace(0,2\*np.pi,100)  
Y1=np.sin(X)  
Y2=np.cos(X)  
Y3=np.tan(X)  
#4.绘制图像  
ax1.plot(X,Y1,color="b",linestyle="-.",linewidth=2)  
ax1.set\_title("Y=Sin(x)")  
ax1.set\_xlabel("X")  
ax1.set\_ylabel("Y")  
ax1.legend(["Y=sin(x)"],loc=0)  
#第二幅图像  
ax2.plot(X,Y2,"r--",linewidth=3)  
ax2.set\_title("Y=Cos(x)")  
ax2.set\_xlabel("X")  
ax2.set\_ylabel("Y")  
ax2.legend(["Y=Cos(x)"],loc=0)  
#第三幅图像  
ax3.plot(X,Y3,color="g",linewidth=4,linestyle="-")  
ax3.set\_title("Y=tan(x)")  
ax3.set\_xlabel("X")  
ax3.set\_ylabel("Y")  
ax3.legend(["Y=tan(x)"],loc=0)  
plt.show()

**案例部分2：**

#面向过程的方式  
import matplotlib.pyplot as plt  
import numpy as np  
plt.figure()  
#指定第1个图示  
ax1 = plt.subplot(221)  
#指定第2个图示  
ax2 = plt.subplot(222)  
#指定第3个图示  
ax3 = plt.subplot(212)  
#准备数据  
X=np.linspace(-np.pi,np.pi,100)  
Y1=np.sin(X)  
Y2=np.cos(X)  
Y3=np.tan(X)  
#占位置--绘制第一幅图像  
plt.sca(ax1)  
plt.plot(X,Y1,color='r',linestyle="-.",linewidth=4)  
plt.title("Y=Sin(x)")  
plt.xlabel("X")  
plt.ylabel("Y")  
plt.legend(["Y=sin(x)"],loc=0)  
#占位置--绘制第二幅图像  
plt.sca(ax2)  
plt.plot(X,Y2,'g--',linewidth=4)  
plt.title("Y=Cos(X)")  
plt.xlabel("X")  
plt.ylabel("Y")  
plt.legend(["Y=Cos(X)"],loc=0)  
#占位置--绘制第三度图像  
plt.sca(ax3)  
plt.plot(X,Y3,color="gray",linestyle="--",marker="^")  
plt.title("Y=tan(X)")  
plt.xlabel("X")  
plt.ylabel("Y")  
plt.legend(["Y=tan(X)"],loc=2)  
plt.show()

## 9.1案例补充

### 9.1.1散点图：

https://baike.baidu.com/item/%E6%95%A3%E7%82%B9%E5%9B%BE/10065276?fr=aladdin

定义：用两组数据构成多个坐标点，考察坐标点的分布，判断两变量之间是否存在某种关联或总结坐标点的分布模式。如身高和体重的分布就可以用散点图描绘。

简单的例子：身高和体重

#身高

height=[160,170,182,175,173]

#体重

weight=[50,58,80,70,55]

plt.scatter(height,weight)

plt.show()

正相关、负相关和不相关

N=1000 #样本

x=np.random.randn(N)

#不相关

y1=np.random.randn(N)

plt.scatter(x,y1)

#正相关

y2=x+np.random.randn(N)\*0.5

plt.scatter(x,y2)

#负相关

y2=-x+np.random.randn(N)\*0.5

plt.scatter(x,y2)

plt.show()

### 9.1.2折线图

折线图是用直线段将各数据连接起来形成的图形

常用来观察数据随时间变化的趋势

比如温度等的的变化

#画出-10到10区间的二次函数图

#100改为10就能看出折线图

x=np.linspace(-10,10,100) #平均分为100份

y=x\*\*2

plt.plot(x,y)

plt.show()

#画出0-10的正弦函数图

x=np.linspace(0,10,100) #平均分为100份

y=np.sin(x)

plt.plot(x,y)

plt.show()

### 9.1.3条形图

以长方形的长度为变量的统计图表

用来比较多个分类的数据大小

通常用于较小的数据集

如不同季度的销量

介绍：https://baike.baidu.com/item/%E6%9D%A1%E5%BD%A2%E5%9B%BE

#主要介绍层叠的和并列的散点图

#一组数据的条形图

index=np.arange(5)

y=[20,10,30,34,15]

#left条形图最左边的横坐标，height条形图的高度

# plt.bar(left=index,height=y)

#2.外观调整color='b',width=0.5

# plt.bar(left=index,height=y,color='b',width=0.5)

#3.水平条形图-加参数 bottom和height、width变化

# plt.bar(left=0,bottom=index,width=y,height=0.5,orientation='horizontal')

plt.barh(left=0,bottom=index,width=y)

plt.show()

##层叠图

index=np.arange(4)

sale1=[52,55,63,53]

sale2=[44,66,55,41]

bar\_width=0.3

#蓝色对应的是sale1

plt.bar(index,sale1,bar\_width,color='b')

#红色对应的比蓝色多0.3宽度

plt.bar(index+bar\_width,sale2,bar\_width,color='r')

plt.show()

### 9.1.4直方图画法

由一些列高度不等的纵向条形组成，表示数据分布的情况

例如学校身高的分布情况

和条形图的区别：直方图是连续的值，使用连续的方式分组

条形图展示不同类别数据，不连续的且不能自定义的

#自定义生成随机数

mu=100

sigma=20

x=mu+sigma\*np.random.randn(200)

#bins在直方图中有几个直方，color颜色，normed标准化(个数变为频率)

plt.hist(x,bins=10,color='r',normed=True)

#调整数据量，可以接近总体密度曲线

plt.show()

#双变量的直方分布图

x=np.random.randn(1000)+2 #中心在2

y=np.random.randn(1000)+3 #中心在3

plt.hist2d(x,y,bins=40)#颜色深浅表示频率大小

plt.show()

### 9.1.5饼状图

饼状图显示一个数据中各项的大小与各项和的比例

饼状图中显示为在整个饼状图中的比例

#饼状图代码

labelx=['A','B','C','D']

fracs=[15,30,45,10]

#1.必须设置比例为1:1才能显示为圆形

plt.axes(aspect=1)

#2.加每一块所占有具体比例的值autopct

#3.突出显示explode

explode=[0,0.2,0,0] #块B远离了饼状图中心

#3.加阴影shadow

plt.pie(x=fracs,labels=labelx,autopct='%.1f%%',explode=explode,shadow=True)

plt.show()

### 9.1.6箱线图boxplot

显示数据的分散情况

上四分位数，中位数，下四分位数，下边缘，异常值（最外面的点）

#箱线图

np.random.seed(10)

data=np.random.normal(size=1000,loc=0,scale=1)

#调整异常点的形状sym='o'

#whis参数表示:虚线的长度，默认1.5(比例)，盒子距离上下四分位数的距离

#距离越大虚线越长，设置成0.5和100分别观察

plt.boxplot(data,sym='o',whis=1.5)

#同时画几组数据在一个图形上

np.random.seed(10)

#4列数据

data=np.random.normal(size=(1000,4),loc=0,scale=1)

labels=['A','B','C','D']

plt.boxplot(data,labels=labels)

plt.show()

# Seaborn绘图实践

基于matplotlib的高层绘图API：

**案例1：基础图形绘制**

**import** matplotlib.pyplot **as** plt  
**import** seaborn **as** sns  
sns.set(style=**"darkgrid"**)  
tips = sns.load\_dataset(**"tips"**)  
*#1.处理基础数据  
#total\_bill和tip关系  
# sns.relplot(x="total\_bill", y="tip", data=tips);  
# sns.relplot(x="total\_bill", y="tip", hue="smoker", data=tips);  
# sns.relplot(x="total\_bill", y="tip", hue="smoker",  
 # col="time", data=tips);  
#2.处理时间数据  
# df = pd.DataFrame(dict(time=pd.date\_range("2017-1-1", periods=500),  
# value=np.random.randn(500).cumsum()))  
# print(df.head())  
# g = sns.relplot(x="time", y="value", kind="line", data=df)  
# g.fig.autofmt\_xdate()  
# 3.分类变量  
# sns.catplot(x="day", y="total\_bill", data=tips);  
# sns.catplot(x="day", y="total\_bill", hue="sex", kind="swarm", data=tips);  
# sns.catplot(x="smoker", y="tip", order=["No", "Yes"], data=tips);  
# sns.catplot(x="day", y="total\_bill", kind="box", data=tips);*sns.boxplot(x=**"day"**,y=**"total\_bill"**,hue=**"smoker"**,data=tips)  
plt.show()

**案例2：泰坦尼克获救问题绘图**

import matplotlib.pyplot as plt  
import seaborn as sns  
sns.set(style="darkgrid")  
tips = sns.load\_dataset("tips")  
  
titanic = sns.load\_dataset("titanic")  
# sns.catplot(x="sex", y="survived", hue="class", kind="bar", data=titanic);  
# sns.catplot(x="deck", kind="count", palette="ch:.25", data=titanic);  
sns.catplot(y="deck", hue="class", kind="count",  
 palette="pastel", edgecolor=".6",  
 data=titanic);  
plt.show()

# 4.了解Scipy

在Python科学计算中，Scipy为数学、物理、工程方面的科学计算提供了无可代替的支持

它是一个Numpy的高级模块

主要凸显在符号计算、信号处理、数值优化

最重要的是向量化的思想（包括符号计算和函数向量化）

Scipy子模块的汇总表：

scipy.cluster 主流的聚类算法

scipy.constants 数学和物理常数

scipy.fftpack 快速傅里叶变换

scipy.integrate 求解积分和常微分方程

scipy.linalg 线性代数

scipy.ndimage n维图像处理

scipy.signal 信号处理

scipy.spatial 空间数据结构和算法

scipy.stats 统计分布及其相关函数

#-\*-coding:utf8-\*-

#0.常数

from scipy import constants as C

print (C.pi) #圆周率

print (C.golden) #黄金比例

print (C.c) #光速

#1.特殊函数模块

from scipy import special as S

print (S.cbrt(8)) #8的立方根

print (S.exp10(3)) #10\*\*3

print (S.comb(5,3)) #从5个中任选3个

print (S.perm(5,3))#排列数

#2.符号计算案例

from scipy import poly1d

p=poly1d([3,4,5]) #3 x^2 + 4 x + 5

print p

print p\*p # 9 x^4 + 24 x ^3+ 46 x ^2+ 40 x + 25

# 4 3 2

# 9 x + 24 x + 46 x + 40 x + 25

print p.integ(k=6) #求解p(x)的不定积分，指定常数项为6

print p.deriv() #求解P(x)的一阶导数

print p([1,5]) #计算每个值代入p(x)的结果

#3.数值积分

from scipy import integrate #导入积分函数

def g(x):# 定义被积函数

return (1-x\*\*2)\*\*0.5

pi\_2,err=integrate.quad(g,-1,1) #积分结果和误差

print (pi\_2\*2) #根据微积分知识知道积分结果为圆周率一半，3.14159265359

#4.函数向量化

import numpy as np

def addsubtract(a,b):

if a>b:

return a-b

else:

return a+b

vec\_addsubtract=np.vectorize(addsubtract)

print vec\_addsubtract([0,3,6,9],[1,3,5,7]) #[1 6 1 2]

# 5.一维卷积运算

from scipy import signal

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

h=np.array([4,5,6])

print "一维卷积运算",signal.convolve(x,h)

#5.信号处理模块-卷积处理图像

from scipy import signal,misc

import matplotlib.pyplot as plt

image=misc.ascent() #二维图像，公寓图像

w=np.zeros((50,50))

w[0][0]=1.0 #修改参数调整滤波器

w[49][25]=1.0 #可以根据需要调整

image\_new=signal.fftconvolve(image,w) #使用FFT算法进行卷积

plt.figure()

plt.imshow(image\_new) #显示滤波后的图像

plt.gray()

plt.title("Filteres image!")

plt.show()

# 了解Scikit-learn

**from** sklearn **import** linear\_model  
reg = linear\_model.LinearRegression()  
x = [[0, 0], [1, 1], [2, 2]]  
y = [0, 1, 2]  
reg.fit (x, y) *#Fit linear model.  
# LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None,  
# normalize=False)*print(reg.coef\_)  
print(reg.intercept\_)  
**import** numpy **as** np  
print(np.allclose(0,reg.intercept\_))  
*# array([0.5, 0.5])*