### Python数据分析和展示

<https://www.cnblogs.com/blackclody/p/6970161.html>

[**Python数据分析和展示**](https://www.cnblogs.com/blackclody/p/6970161.html)

1.课程简介

python数据分析和展示

掌握表示、清洗、统计和展示数据的能力

包括 numpy库 ndarray

Matplotlib库 pyplot

Pandas库 Series DataFrame类型

编程工具：

使用anacode IDE集成开发环境

需要理解和掌握如下工具的使用： conda, spyder, IPython

本课程的实例：

实例1：图像的手绘效果

实例2：引力波的绘制

实例3：房价趋势的关联因素分析

实例4：股票数据的趋势分析曲线

坐标系的绘制

饼图的绘制

直方图的绘制

极坐标图的绘制

散点图的绘制

2. Anaconda IDE的基本使用

[https://www.continuum.io](https://www.continuum.io/)

开源免费

支持近800个第三方库

包含多个主流工具

适合数据计算领域开发

win/linux/OS X

包管理和环境管理工具 conda

conda: 一个工具，用于包管理和环境管理 ，其中：

包管理与pip类似，管理python的第三方库

环境管理能够允许用户使用不同版本python，并能灵活切换

anaconda:一个集合，包括conda，某版本的python，一批第三方库

conda将工具，第三方库，Python版本，conda都当作包，同等对待

win平台 cmd 执行 conda --version 获取conda版本

执行 conda update conda升级 conda

编程工具spyder

交互式编程环境 IPython: 是一个功能强大的交互式shell 适合用于进行交互式的数据可视化和GUI的相关应用

变量前或后增加？将显示一些通用信息和包括函数对应的源代码

%run demo.py用于运行一个py程序

IPython的%魔术命令

|  |  |
| --- | --- |
| 常用命令 | 说明 |
| %magic | 显示所有的魔术命令 |
| %hist | IPython命令的输入历史 |
| %pdb | 异常发生后自动进行调试器 |
| %reset | 删除当前命令空间中的全部变量或名称 |
| %who | 显示IPython当前命令空间中已经定义的变量 |
| %time statement | 给出代码的执行时间，statement表示一段代码 |
| %timeit statement | 多次执行代码，计算综合平均执行时间 |

中国大学mooc平台在线开放课程  [www.icourses.cn/imooc](http://www.icourses.cn/imooc)

3. numpy库入门

数据的维度

维度：一组数据的组织形式

一维数据：由对待关系的有序或无序数据构成，采用线性方式组织 数组，对应列表，数组和集合的概念

列表：数据类型可以不同

数组：数据类型相同

二维数据：由多个一维数据组成，是一维数据的组合形式

多维数据：由一维或二维数据在新的维度上的扩展形成的

高维数据：仅利用最基本的二元关系展示数据间的复杂结构

数据维度的python表示

数据维度是数据的组织形式

一维数据： 列表和集合类型  列表有序  集合无序

二维数据：列表类型

多维数据：列表类型

高维数据：字典类型或或JSON,XML和YAML格式

numpy的数据对象ndarray

numpy是一个开源的python科学计算基础库，包含：

* 一个强大的N维数组对象 ndarray
* 广播功能函数
* 整合C/C++/Fortran代码的工具
* 线性代数、傅立叶变换、随机数生成等功能

numpy是Scipy，Pandas等数据处理或科学计算库的基础

numpy库的引用  import numpy as np

ndarray和python自带列表的区别，如下面的例子所示

1. import numpy as np
2. def npSum():
3. a = np.array([0,1,2,3,4,5])
4. b = np.array([9,8,7,6,5,4])
5. c = a\*\*2 + b\*\*3
6. return c
7. def pySum():
8. a = [0,1,2,3,4,5]
9. b = [9,8,7,6,5,4]
10. c = []
11. for i in range(len(a)):
12. c.append(a[i]\*\*2 + b[i]\*\*3)
13. return c
14. print(npSum())
15. print(pySum())

输出

[729 513 347 225 141  89]

[729, 513, 347, 225, 141, 89]

numpy中ndarray的优点：

* ndarray对象可以去掉元素间运算所需的循环，便一维向量更像单个数据
* 设置专门的数组对象，经过优化，可以提升这类应用的运算速度
* 数组对象采用相同的数据类型，有助于节省运算和存储空间

ndarray是一个多维的数组对象，由两部分构成：

* 实际的数据
* 描述这些数据的元数据（数据维度、数据类型等）

ndarray数据一般要求所有元素类型相同（同质），数据下标从0开始

ndarray实例

>>> a = np.array([[0,1,2,3,4],[9,8,7,6,5]]) # np.array()生成一个ndarray数组

>>> a

array([[0, 1, 2, 3, 4],

       [9, 8, 7, 6, 5]])

>>> print(a)#ndarray()输出成[]形式，元素由空格分割

[[0 1 2 3 4]

 [9 8 7 6 5]]

轴：保存数据的维度

秩：轴的数量

ndarray对象的属性

|  |  |
| --- | --- |
| 属性 | 说明 |
| .ndim | 秩，即轴的数量或维度的数量 |
| .shape | ndarray对象尺度，对于 矩阵，n行m列 |
| .size | ndarray对象元素的个数，相当于.shape中的n\*m的值 |
| .dtype | ndarray对象的元素类型 |
| .itemsize | ndarray对象中每个元素的大小，以字节为单位 |

如

>>> a.ndim

2

>>> a.shape

(2, 5)

>>> a.size

10

>>> a.dtype

dtype('int32')

>>> a.itemsize

4

ndarray数组的元素类型

|  |  |
| --- | --- |
| 数据类型 | 说明 |
| bool | 布尔类型，True或False |
| intc | 与C语言中int类型一致，一般是int32或int64 |
| intp | 用于索引的整数，与C语言中ssize\_t一致，int32或int64 |
| int8 | 字节长度的整数，取值：[-128,127] |
| int16 | 16位长度的整数，取值：[-32768,32767] |
| int32 | 32位长度的整数，取值：[-231,231-1] |
| int64 | 64位长度的整数，取值：[-263,263-1] |
| uint8 | 8位无符号整数，取值：[0,255] |
| uint16 | 16位无符号整数，取值：[0,65535] |
| uint32 | 32位无符号整数，取值：[0,232-1] |
| uint64 | 64位无符号整数，取值：[0,264-1] |
| float16 | 16位半精度浮点数 |
| float32 | 32位半精度浮点数 |
| float64 | 64位半精度浮点数 |
| complex64 | 复数类型，实部和虚部都是32位的浮点数 |
| complex128 | 复数类型，实部和虚部都是64位的浮点数 |

ndarray数组的创建方法

1. 从python的列表、元组等类型创建ndarray数组，如 x = np.array(list/tuple)  x = np.array(list/tuple,dytpe = np.float32
2. 使用numpy中函数创建ndarray数组，如arange,ones,zeros等

|  |  |
| --- | --- |
| 函数 | 说明 |
| np.arange(n) | 类似range()函数，返回ndarray类型，元素从0到n-1 |
| np.ones(shape) | 根据shape生成一个全1的数组，shape是元组类型 |
| np.zeros(shape) | 根据shape生成一个全0数组，shape是元组类型 |
| np.full(shape,val) | 根据shape生成一个数组，每个元素都是val |
| np.eye(n) | 创建一个正方的n\*n单位矩阵，对角线为1，其余为0 |
| np.ones\_like(a) | 根据数组a的形状生成一个全1的数组 |
| np.zeros\_like(a) | 根据数据a的形状生成一个全0的数据 |
| np.full\_like(a,val) | 根据数组a的形状生成一个数组，每个元素都是val |
| np.linspace() | 根据起止数据等间距填充数据，形成数组 |
| np.concatenate() | 将两个或多个数组组合成一个新的数组 |

1. 从字节流(raw bytes)中创建ndarray数组
2. 从文件中读取特定格式，创建ndarray数组

ndarray数组的变换

|  |  |
| --- | --- |
| 方法 | 说明 |
| .reshape(shape) | 不改变数组元素，返回一个shape形状的数组，原数组不变 |
| .resize(shape) | 与.shape()功能一致，但修改原数组 |
| .swapaxes(ax1,ax2) | 将数组n个维度中两个维度进行调换 |
| .flatten() | 对数组进行降维，返回折叠后的一维数组，原数组不变 |

ndarray数组的类型变换  new\_a = a.astype(new\_type)

>>> a = np.array([1,2,3],dtype = np.int)

>>> a.dtype

dtype('int32')

>>> b = a.astype(np.float)

>>> b.dtype

dtype('float64')

ndarray数组向列表的转换 ls = a.tolist()

>>> c = a.tolist()

>>> c

[1, 2, 3]

ndarray数组的操作

索引和切片：与python的列表类似

ndarray数组的运算

数组与标量之间的运算作用于数组的每一个元素

对ndarray中的数据执行元素级运算的函数

|  |  |
| --- | --- |
| 函数 | 说明 |
| np.abs() np.fabs() | 计算数组各元素的绝对值 |
| np.sqrt() | 计算数组各元素的平方根 |
| np.square() | 计算数组各元素的平方 |
| np.log(x),np.log10(x),np.log2(x) | 计算数组各元素的自然对数、10底对数和2底对数 |
| np.ceil(x),np.floor(x) | 计算数组各元素的ceiling值或floor值 |
| np.rint(x) | 计算数组各元素的四舍五入值 |
| np.modf(x) | 将数据各元素的整数和小数部分以两个独立的数组形式返回 |
| np.cos/cosh/sin/sinh/tan/tanh | 计算数据各元素的普通型和双典型的三角函数 |
| np.exp(x) | 计算数组各元素的指数值 |
| np.sign(x) | 计算数组各元素的符号值，1(+),0,-1(-) |

numpy的二元函数

|  |  |
| --- | --- |
| 函数 | 说明 |
| + - \* / \*\* | 两个数组对应的元素进行相应的运算 |
| np.maximum/minimum/fmax/fmin | 元素级的最大值/最小值 |
| np.mod(x,y) | 元素级的模运算 |
| np.copysign(x,y) | 将元素y中各元素值的符号赋值给数据x对应的元素 |
| > < >= <= == != | 算术比较，产生布尔型的数组 |

numpy数据存取函数

数据的csv文件存取

CSV（Comma-Separated Value,逗号分隔值）是一种常见的文件格式，用来存储批量数据

np.savetxt(frame,array,fmt='%.18e',delimiter=None)

frame: 文件、字符串或产生器，可以是.gz或.bz2的压缩文件

array: 存入文件的数组

fmt:写入文件的格式 ，例如%d %.2f %.18e

delimiter: 分割字符串，默认是空格

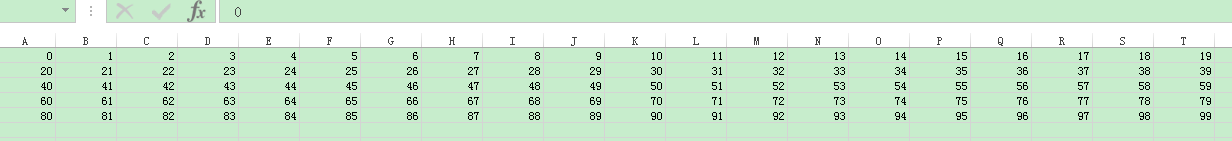
如

>>> import numpy as np

>>> a = np.arange(100).reshape(5,20)

>>> np.savetxt('a.csv',a,fmt='%d',delimiter=',')

a.csv文件



np.loadtxt(frame,dtype=np.float,delimiter=None,unpack=False)

frame: 文件、字符串或产生器，可以是.gz或.bz2的压缩文件

dtype:数据类型，可选

delimiter: 分割字符串，默认是空格

unpack:如果True，读入属性将分别写入不同的变量

csv只能有效存储一维和二维数组

多维数据的存取

a.tofile(frame,seq=' ',format='%s')

frame:文件，字符串

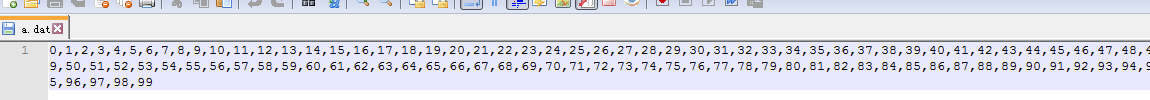
seq:数据分割字符串，如果是空串，写入文件为二进制

format:写入数据的格式

如

>>> a = np.arange(100).reshape(5,20)

>>> a.tofile('a.dat',sep=',',format='%d')



np.fromfile(frame,dtype=float,count=-1,sep=' ')

frame:文件，字符串

dtype:读取的数据类型

count:读入元素的个数，-1表示读入整个文件

seq:数据分割字符串，如果是空串，写入文件为二进制

 numpy的便捷文件读取

np.save(fname,array)

np.savez(fname,array)

fname:文件名，以.npy为扩展名，压缩扩展名为.npz

array:数据变量

np.load(fname)

fname:文件名，以.npy为扩展名，压缩扩展名为.npz

numpy的随机数函数

np.random.\*

|  |  |
| --- | --- |
| 函数 | 说明 |
| rand(d0,d1,...,dn) | 根据d0-dn创建随机数数组，浮点数，[0,1)，均匀分布 |
| randn(d0,d1,...,dn) | 根据d0-dn创建随机数数组，标准正态分布 |
| randint(low[,high,shape]) | 根据shape创建随机整数或整数数组，范围是[low,high) |
| seed(s) | 随机数种子，s是给定的种子值 |
| shuffle(a) | 根据数组a的第1轴进行随排列，改变数组a |
| permutation(a) | 根据数组a的第1轴产生一个新的乱序数组，不改变数组a |
| choice(a[,size,replace,p]) | 从一维数组a中以概率p抽取元素，形成size形状新数组replace表示是否可能重用元素，默认为False |
| uniform(low,high,size) | 产生具有均匀分布的数组，low起始值，high结束值，size为形状 |
| normal(loc,scale,size) | 产生具有正态分布的数组，loc为均值，scale标准差，size为形状 |
| poisson(lam,size) | 产生具有泊松分布的数组，lam为随机事件发生率，size为形状 |

numpy的统计函数

注意： axis = None是统计函数的标配参数

|  |  |
| --- | --- |
| 函数 | 说明 |
| sum(a,axis=None) | 根据给定axis计算数组a相关元素之和，axis整数或元组 |
| mean(a,axis=None) | 根据给定axis计算数组a相关元素的期望，axis整数或元组 |
| average(a,axis=None,weights=None) | 根据给定axis计算数组a相关元素的加权平均值 |
| std(a,axis=None) | 根据给定轴axis计算数组a相关元素的标准差 |
| var(a,axis = None) | 根据给定轴axis计算数组a相关元素的方差 |
| min(a)  max(a) | 计算数组a中元素的最小值，最大值 |
| argmin(a) argmax(a) | 计算数组a中元素的最小值，最大值的降一维后下标 |
| unravel\_index(index,shape) | 根据shape将一维下标index转换成多维下标 |
| ptp(a) | 计算数组a中元素最大值和最小值的差 |
| median(a) | 计算数组a中元素的中位数(中值) |

numpy的梯度函数

|  |  |
| --- | --- |
| 函数 | 说明 |
| np.gradient(f) | 计算数组f中元素的梯度，当f为多维时，返回每个维度梯度 |

梯度是指连续值之间的变化率，即斜率

XY坐标轴连续三个x坐标对应的y轴值：a,b,c，其中b的梯度是(c-a)/2

二、matplotlib库入门

matplotlib是Python优秀的数据可视化第三方库

matplotlib库的效果可参考

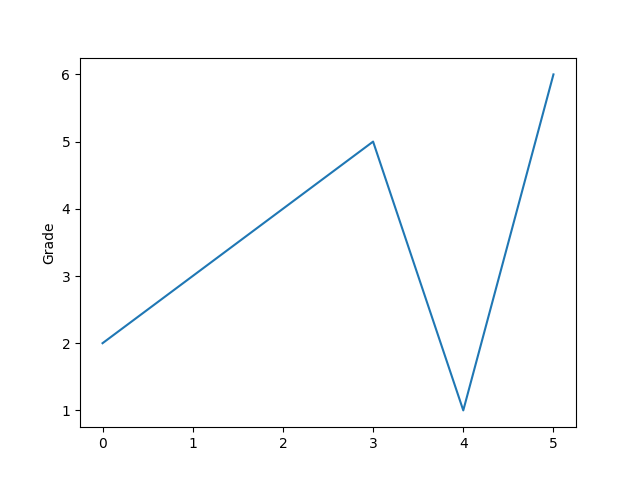
<http://matplotlib.org/gallery.html>

matplotlib的使用 由各种可视化类构成，内部结构复杂，受matlab库启发，matplotlab.pyplot是绘制种类可视化图形的命令子库，相当于快捷方式

import matplotlib.pyplot as plt

1. import matplotlib.pyplot as plt
2. plt.plot([2,3,4,5,1,6])
3. plt.ylabel("Grade")
4. plt.savefig('test',dpi=600)#plt.savefig()将输出图形存储为文件，默认为png格式，可以通过dpi修改输出质量
5. plt.show()

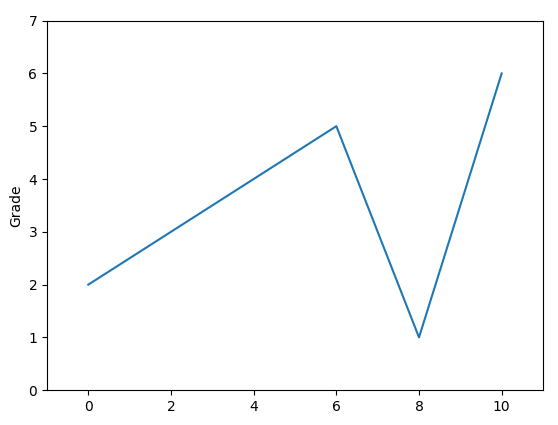
输出



plt.plot(x,y)当有两个以上参数时，按照x轴和y轴顺序绘制数据点

1. import matplotlib.pyplot as plt
2. plt.plot([0,2,4,6,8,10],[2,3,4,5,1,6])
3. plt.ylabel("Grade")
4. plt.axis([-1,11,0,7])
5. plt.show()

输出效果



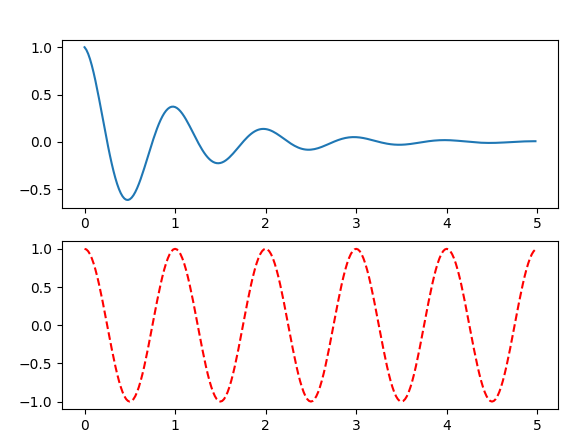
plot.subplot(nrows,ncols,plot\_number)

在全局绘制区域中创建一个分区体系，并定位到一个子绘图区域

pyplot绘图区域示例

1. import numpy as np
2. import matplotlib.pyplot as plt
3. def f(t):
4. return np.exp(-t) \* np.cos(2\*np.pi\*t)
5. a = np.arange(0.0,5.0,0.02)
6. plt.subplot(211)
7. plt.plot(a,f(a))
8. plt.subplot(2,1,2)
9. plt.plot(a,np.cos(2\*np.pi\*a),'r--')
10. plt.show()

输出



plt.plot(x,y,format\_string,\*\*kwargs)

x:x轴数据，列表或数组，可选

y:y轴数据，列表或数组

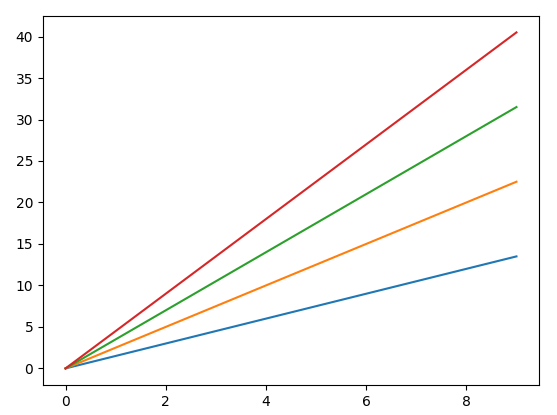
format\_string: 控制曲线的格式字符串，可迁

\*\*kwargs: 第二组或更多的(x,y,format\_string)

注意：当绘制多条曲线时，各条曲线的x不能省略

如

1. import numpy as np
2. import matplotlib.pyplot as plt
3. a = np.arange(10)
4. plt.plot(a,a\*1.5,a,a\*2.5,a,a\*3.5,a,a\*4.5)
5. plt.show()



 format\_string：由颜色字符、风格字符和标记字符组成

|  |  |  |  |
| --- | --- | --- | --- |
| 颜色字符 | 说明 | 颜色字符 | 说明 |
| 'b' | blue | 'm' | magenta洋红色 |
| 'g' | green | 'y' | 黄色 |
| 'r' | red | 'k' | 黑色 |
| 'c' | cyan青绿色 | 'w' | 白色 |
| '#008000' | RGB某颜色 | '0.8' | 灰度值字符串 |

|  |  |
| --- | --- |
| 风格字符 | 说明 |
| '-' | 实线 |
| '--' | 破折线 |
| '-.' | 点划线 |
| ':' | 虚线 |
| ' ' | 无线条 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 标记字符 | 说明 | 标记字符 | 说明 | 标记字符 | 说明 |
| '.' | 点标记 | '1' | 下花三角标记 | 'h' | 竖六边形标记 |
| ',' | 像素标记(极小点) | '2' | 上花三角标记 | 'H' | 横六边形标记 |
| 'o' | 实心圏标记 | '3' | 左花三角标记 | '+' | 十字形标记 |
| 'v' | 倒三角标记 | '4' | 右花三角标记 | 'x' | x标记 |
| '^' | 上三角标记 | 's' | 实心方形标记 | 'D' | 菱形标记 |
| '>' | 右三角标记 | 'p' | 实心五角标记 | 'd' | 瘦菱形标记 |
| '<' | 左三角标记 | '\*' | 星形标记 | '|' | 垂直线标记 |

\*\*kwargs: 第二组或更多(x,y,format\_string)

color: 控制颜色 如color='green'

linestyle:线条控制 如linestyle='dashed'

marker:标记风格，marker='o'

markerfacecolor:标记颜色，markerfacecolor='blue'

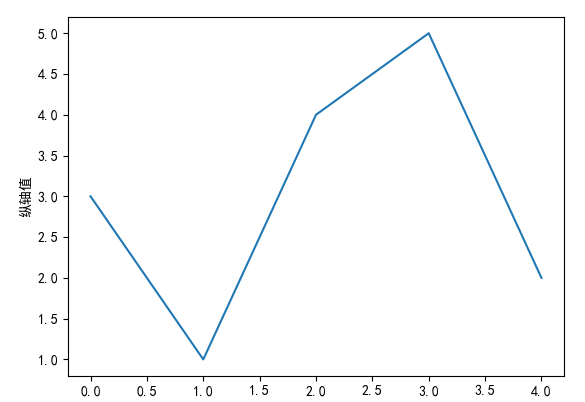
markersize:标记尺寸，markersize=20

...

pyplot的中文显示

pyplot并不默认支持中文显示，需要rcParams修改字体实现

1. import matplotlib
2. import matplotlib.pyplot as plt
3. matplotlib.rcParams['font.family']='SimHei'
4. plt.plot([3,1,4,5,2])
5. plt.ylabel('纵轴值')
6. plt.savefig('test',dpi=600)
7. plt.show()

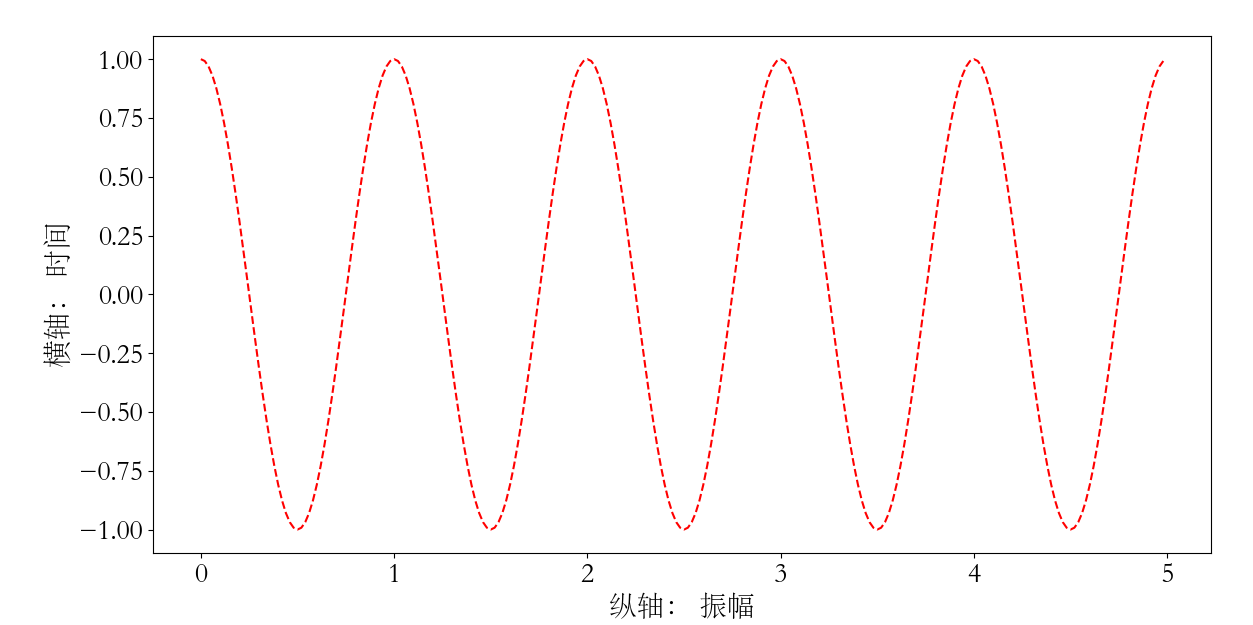


|  |  |
| --- | --- |
| 属性 | 说明 |
| 'font.family' | 用于显示字体的名字 |
| 'font.style' | 字体风格，正常'normal'或斜体'italic' |
| 'font.size' | 字体大小，整数字号或者'large','x-small' |

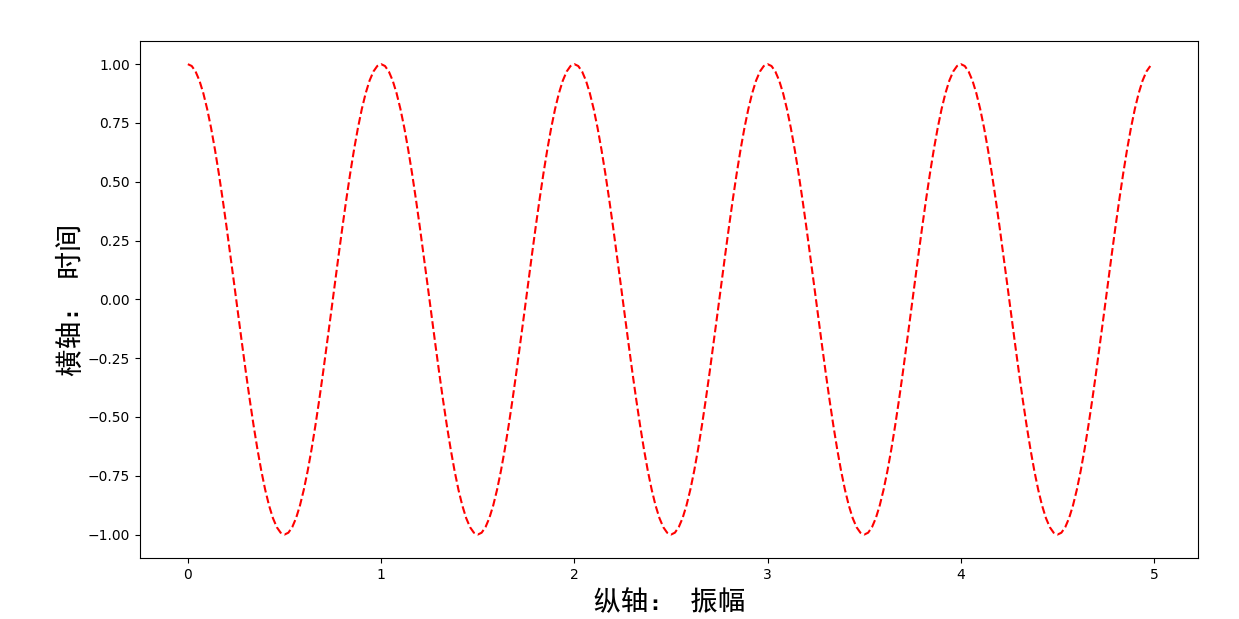
rcParams['font.family']

|  |  |
| --- | --- |
| 中文字体 | 说明 |
| 'SimHei' | 中文黑体 |
| 'Kaiti' | 中文楷体 |
| 'LiSu' | 中文隶书 |
| 'FangSong' | 中文仿宋 |
| 'YouYuan' | 中文幼圆 |
| STSong | 华文宋体 |

1. import matplotlib
2. import matplotlib.pyplot as plt
3. import numpy as np
4. matplotlib.rcParams['font.family']='STSong'
5. matplotlib.rcParams['font.size']=20
6. a = np.arange(0.0,5.0,0.02)
7. plt.xlabel('纵轴： 振幅')
8. plt.ylabel('横轴： 时间')
9. plt.plot(a,np.cos(2\*np.pi\*a),'r--')
10. plt.show()



在有中文输出的地方，增加一个属性：fontproperties

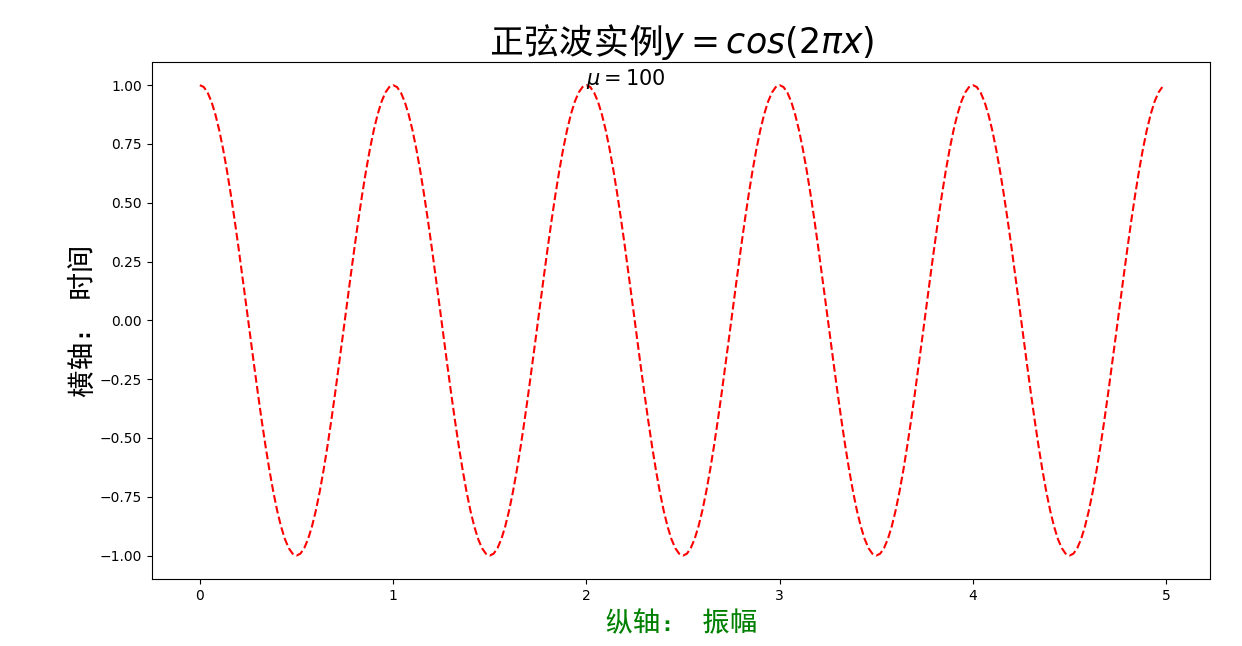


pyplot的文本显示

|  |  |
| --- | --- |
| 函数 | 说明 |
| plt.xlabel() | 对x轴增加文本标签 |
| plt.ylabel() | 对y轴增加文本标签 |
| plt.title() | 对图形本整体增加文本标签 |
| plt.text() | 在任意位置增加文本 |
| plt.annotate() | 在图形中增加带箭头的注释 |

1. import matplotlib.pyplot as plt
2. import numpy as np
3. a = np.arange(0.0,5.0,0.02)
4. plt.xlabel('纵轴： 振幅', fontproperties='SimHei', fontsize=20, color = 'green')
5. plt.ylabel('横轴： 时间', fontproperties='SimHei', fontsize=20)
6. plt.title(r'正弦波实例$y=cos(2\pi x)$',fontproperties='SimHei',fontsize=25)
7. plt.text(2,1,r'$\mu=100$',fontsize=15)
8. plt.plot(a,np.cos(2\*np.pi\*a),'r--')
9. plt.show()

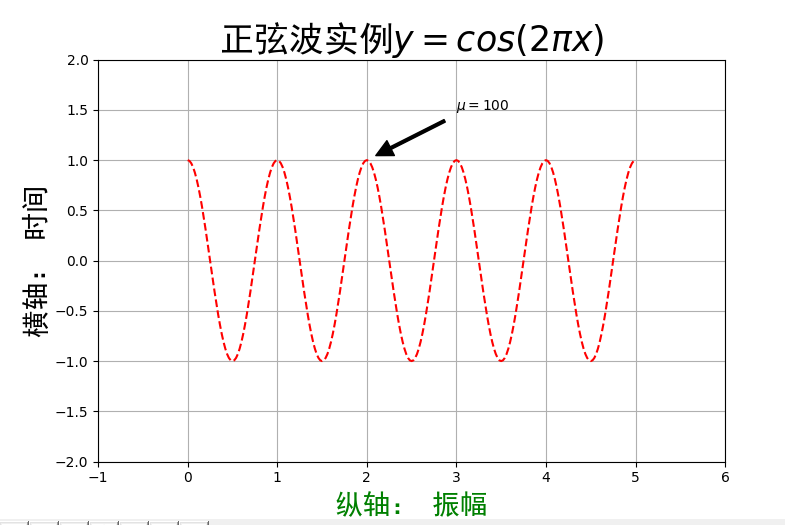
输出效果



plt.annotate(s,xy=arrow\_crd,xytext=text\_crd,arrowprops=dict)

1. import matplotlib.pyplot as plt
2. import numpy as np
3. a = np.arange(0.0,5.0,0.02)
4. plt.plot(a,np.cos(2\*np.pi\*a),'r--')
5. plt.xlabel('纵轴： 振幅', fontproperties='SimHei', fontsize=20, color = 'green')
6. plt.ylabel('横轴： 时间', fontproperties='SimHei', fontsize=20)
7. plt.title(r'正弦波实例$y=cos(2\pi x)$',fontproperties='SimHei',fontsize=25)
8. plt.annotate(r'$\mu=100$',xy=(2,1),xytext=(3,1.5),
9. arrowprops=dict(facecolor='black',shrink=0.1,width=2))
10. plt.axis([-1,6,-2,2])
11. plt.grid()
12. plt.show()

输出



 pyplot的子绘图区域

plt.subplot2grid(GridSpec,CurSpec,colspan=1,rowspan=1)

如绘图区域2可表示为

plt.subplot2grid((3,3),(1,0),colspan=2,rowspan=1)

ax1: plt.subplot2grid((3,3),(0,0),colspan=3)

ax2: plt.subplot2grid((3,3),(1,0),colspan=1)

ax3: plt.subplot2grid((3,3),(1,2),rowspan=2)

ax4: plt.subplot2grid((3,3),(2,0))

ax1: plt.subplot2grid((3,3),(2,1))

|  |  |  |
| --- | --- | --- |
| ax1 | | |
| ax2 | | ax3 |
| ax4 | ax5 |

GridSpec类

1. import matplotlib.gridspec as gridspec
2. gs = gridspec.GridSpec(3,3)
3. ax1 = plt.subplot(gs[0,:])
4. ax2 = plt.subplot(gs[1,:-1])
5. ax3 = plt.subplot(gs[1:,-1])
6. ax4 = plt.subplot(gs[2,0])
7. ax5 = plt.subplot(gs[2,1])

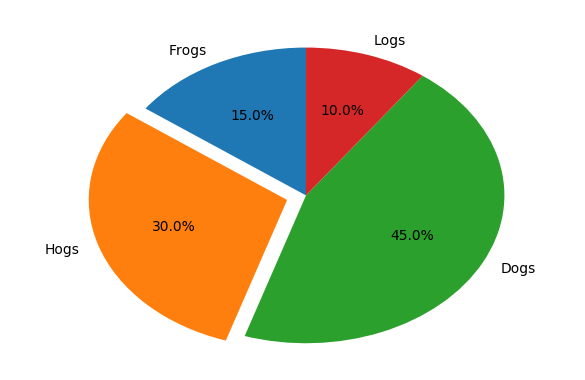
matplotlib库基础图函数示例

|  |  |
| --- | --- |
| 函数 | 说明 |
| plt.plot(x,y,fmt,...) | 绘制一个坐标图 |
| plt.boxplot(data,notch,position) | 绘制一个箱形图 |
| plt.bar(left,height,width,bottom) | 绘制一个条形图 |
| plt.barh(width,bottom,left,heitht) | 绘制一个横向条形图 |
| plt.polar(theta,r) | 绘制极坐标图 |
| plt.pie(data,explode) | 绘制饼图 |
| plt.psd(x,NFFT=256,pad\_to,Fs) | 绘制功率谱密度图 |
| plt.specgram(x,NFFT=256,pad\_to,F) | 绘制谱图 |
| plt.cohere(x,y,NTTF=256,Fs) | 绘制X-Y相关性函数 |
| plt.scatter(x,y) | 绘制散点图，其中x,y长度相同 |
| plt.step(x,y,where) | 绘制步阶图 |
| plt.hist(x,bins,normed) | 绘制直方图 |
| plt.contour(X,Y,Z,N) | 绘制等值图 |
| plt.vlines() | 绘制垂直图 |
| plt.stem(x,y,linefmt,markerfmt) | 绘制柴火图 |
| plt.plot\_date() | 绘制数据日期 |
|  |  |

pyplot饼图的绘制

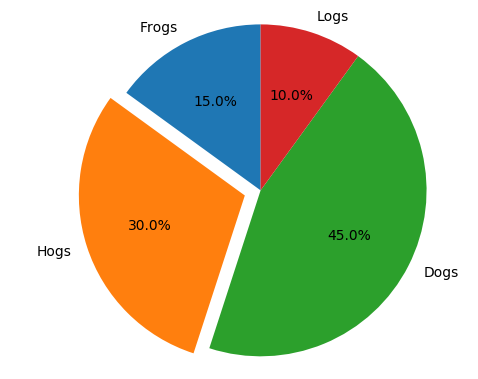
1. import matplotlib.pyplot as plt
2. labels = 'Frogs','Hogs','Dogs','Logs'
3. sizes = [15,30,45,10]
4. explode = [0,0.1,0,0]
5. plt.pie(sizes, explode = explode,labels = labels, autopct = '%1.1f%%',
6. shadow = False,startangle = 90)
7. plt.show()

输出



1. import matplotlib.pyplot as plt
2. labels = 'Frogs','Hogs','Dogs','Logs'
3. sizes = [15,30,45,10]
4. explode = [0,0.1,0,0]
5. plt.pie(sizes, explode = explode,labels = labels, autopct = '%1.1f%%',
6. shadow = False,startangle = 90)
7. plt.axis('equal')
8. plt.show()

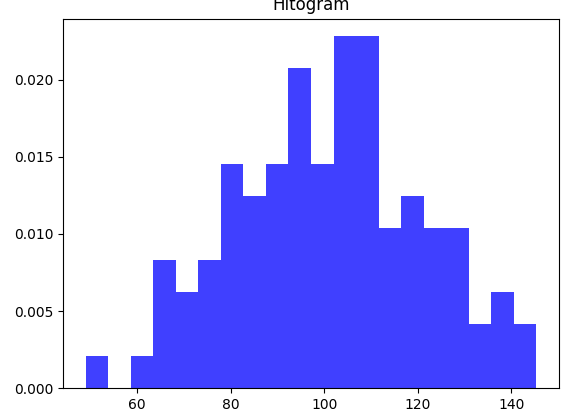
输出



绘制直方图

1. import matplotlib.pyplot as plt
2. import numpy as np
3. np.random.seed(0)
4. mu,sigma = 100,20
5. a = np.random.normal(mu,sigma,size=100)
6. plt.hist(a,20,normed=1,histtype='stepfilled',facecolor='b',alpha=0.75)
7. plt.title('Hitogram')
8. plt.show()

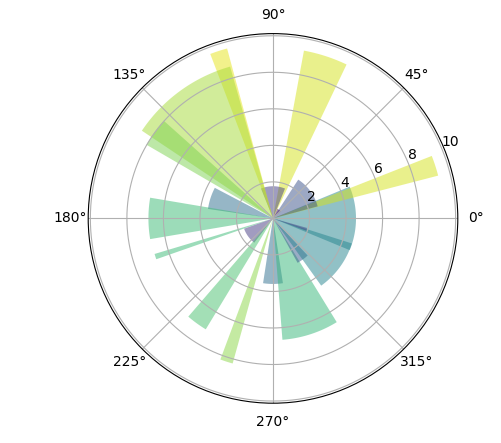
输出



pyplot极坐标图的绘制

1. import matplotlib.pyplot as plt
2. import numpy as np
3. N = 20
4. theta = np.linspace(0.0, 2\*np.pi, N, endpoint = False)
5. radii = 10\*np.random.rand(N)
6. width = np.pi/4\*np.random.rand(N)
7. ax = plt.subplot(111,projection = 'polar')
8. bars = ax.bar(theta, radii, width=width, bottom = 0.0)
9. for r, bar in zip(radii, bars):
10. bar.set\_facecolor(plt.cm.viridis(r/10.))
11. bar.set\_alpha(0.5)
12. plt.show()

输出



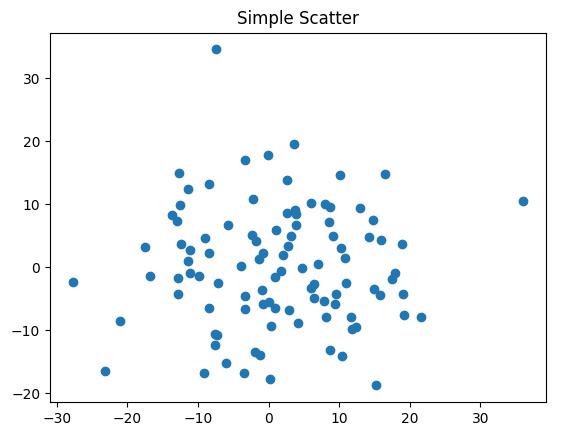
绘制散点图

1. #-\*- coding=utf8 -\*-
2. # 面向对象绘制散点图
3. import numpy as np
4. import matplotlib.pyplot as plt

fig, ax = plt.subplots()

1. ax.plot(10\*np.random.randn(100),10\*np.random.randn(100),'o')
2. ax.set\_title('Simple Scatter')
3. plt.show()

输出



pandasy库的介绍

[http://pandas.pydata.org](http://pandas.pydata.org/)

pandas库的使用

import pandas as pd

pandas基于numpy实现，常与numpy和matplotlib库一同使用

两个数据类型 Series DataFrame

基于上述两个数据类型的各类操作

基本操作，运算操作，特征类操作 关联类操作

numpy 基础数据类型 关注数据的结构表达 维度: 数据间关系

pandas:扩展数据类型 关注数据的应用表达 数据与索引间的关系

Series类型:由一组数据及与之相关的数据索引组成

如

>>> import pandas as pd

>>> a = pd.Series(range(10))

>>> a

0    0

1    1

2    2

3    3

4    4

5    5

6    6

7    7

8    8

9    9

dtype: int32

还可以指定数据索引

>>> a = pd.Series(range(4),index=['a','b','c','d'])

>>> a

a    0

b    1

c    2

d    3

dtype: int32

Series类型可以由如下类型创建：

* Python列表
* 标量值
* python字典
* ndarray
* 其他函数

由标量值创建时，index不能省略

如

>>> pd.Series(25,index=['a','b','c'])

a    25

b    25

c    25

dtype: int64

从字典类型创建

>>> pd.Series({'a':1,'b':2,'c':3})

a    1

b    2

c    3

dtype: int64

>>> pd.Series({'a':1,'b':2,'c':3},index = ['b','c','d'])#index从字典中进行选择操作

b    2.0

c    3.0

d    NaN

dtype: float64

从ndarray类型创建

>>> pd.Series(np.arange(5),index=np.arange(9,4,-1))

9    0

8    1

7    2

6    3

5    4

dtype: int32

Series类型的基本操作

* 包括index和value两部分

>>> a.index#获得索引

Index(['a', 'b', 'c', 'd'], dtype='object')

>>> a.values#获得数据

array([0, 1, 2, 3])

 >>> a[1]#自动索引

1

>>> a['b']#自定义索引

1

>>> a[['b','c']]

b    1

c    2

dtype: int32

* 操作类似于ndarray类型
  + 索引方法相同，采用[]
  + numpy中运算和操作可用于Series类型
  + 可以通过自定义的列表进行切片
  + 可以通过自动索引进行切片，如果存在自定义索引，则一同被切片

如

>>> b=pd.Series([9,8,7,6],['a','b','c','d'])

>>> b

a    9

b    8

c    7

d    6

dtype: int64

>>> b[3]

6

>>> b[:3]

a    9

b    8

c    7

dtype: int64

>>> b[b>b.median()]

a    9

b    8

dtype: int64

>>> np.exp(b)

a    8103.083928

b    2980.957987

c    1096.633158

d     403.428793

dtype: float64

* 操作类似于python字典类型
  + 通过自定义索引访问
  + 保留字in操作
  + 使用.get方法

如

>>> b=pd.Series([9,8,7,6],['a','b','c','d'])

>>> b['b']

8

>>> 'c' in b

True

>>> 0 in b

False

>>> b.get('f',100)

100

Series类型对齐操作

Series类型的运算中会自动对齐不同索引的数据

>>> b=pd.Series([9,8,7,6],['a','b','c','d'])

>>> a=pd.Series([1,2,3],['c','d','e'])

>>> a+b

a    NaN

b    NaN

c    8.0

d    8.0

e    NaN

dtype: float64

Series类型的name属性

Series对象和索引都可以有名字，存储在在属性.name中

>>> b=pd.Series([9,8,7,6],['a','b','c','d'])

>>> b.name

>>> b.name='Series对象'

>>> b.index.name='索引列'

>>> b

索引列

a    9

b    8

c    7

d    6

Name: Series对象, dtype: int64

Series对象可以随时修改并立即生效

>>> b=pd.Series([9,8,7,6],['a','b','c','d'])

>>> b['a']=15

>>> b.name='Series'

>>> b

a    15

b     8

c     7

d     6

Name: Series, dtype: int64

>>> b.name='New Series'

>>> b['b','c']=20

>>> b

a    15

b    20

c    20

d     6

Name: New Series, dtype: int64

Series是一维带‘标签’数组

DataFrame类型

DataFrame类型由共用相同索引的一组列组成

index\_0   data\_a data\_1 ... data\_w

index\_1   data\_b data\_2 ... data\_x

index\_2   data\_c data\_3 ... data\_y

index\_3   data\_d data\_4 ... data\_z

索引                多列数据

DataFrame是一个表格型的数据类型，每列值类型可以不同，即有行索引，也有列索引

常用于表达二维数据，但可以表达多维数据

DataFrame类型可以由如下类型创建

二维ndarray对象

>>> d=pd.DataFrame(np.arange(10).reshape(2,5))

>>> d

   0  1  2  3  4

0  0  1  2  3  4

1  5  6  7  8  9

由一维ndarray、列表、字典、元组或Series类型构成的字典

>>> dt=pd.DataFrame({'one':pd.Series([1,2,3],['a','b','c']),

... 'two':pd.Series([9,8,7,6],['a','b','c','d'])})

>>> dt

   one  two

a  1.0    9

b  2.0    8

c  3.0    7

d  NaN    6

>>> pd.DataFrame(dt,index=['b','c','d'],columns=['two','three'])

   two  three

b    8    NaN

c    7    NaN

d    6    NaN

>>> d1={'one':[1,2,3,4],'two':[9,8,7,6]}

>>> d = pd.DataFrame(d1,index=['a','b','c','d'])

>>> d

   one  two

a    1    9

b    2    8

c    3    7

d    4    6

Series类型

其他的DataFrame类型

数据类型操作

如何改变Series和DataFrame类型

增加或重排：重新索引

删除：drop

重新索引

.reindex(index=None,columns=None,...)

|  |  |
| --- | --- |
| 参数 | 说明 |
| index, columns | 新的行列自定义索引 |
| fill\_value | 重新索引中，用于填充缺失位置的值 |
| method | 填充方法，ffill当前值向前填充，bfill向后填充 |
| limit | 最大填充量 |
| copy | 默认True，生成新的对象，False时，新旧相等不复制 |

索引类型

Series和DataFrame的索引是Index类型，

Index对象是不可修改对象类型

|  |  |
| --- | --- |
| 方法 | 说明 |
| .append(idx) | 连接另一个Index对象，产生新的Index对象 |
| .diff(idx) | 计算差集，产生新的Index对象 |
| .intersection(idx) | 计算交集 |
| union(idx) | 计算并集 |
| delete(loc) | 删除loc位置处的元素 |
| insert(loc,e) | 在loc位置上增加一个元素e |

.drop方法能够删除Series和DataFrame指定行或列索引

算术运算法则

算术运算根据行列索引，补齐后运算，运算默认产生浮点数

补齐时缺项填充NaN

二维和一维，一维和零维间为广播运算

采用+ - \* / 符号进行的二元运算产生新的的对象

>>> a = pd.DataFrame(np.arange(12).reshape(3,4))

>>> b = pd.DataFrame(np.arange(20).reshape(4,5))

>>> a+b

      0     1     2     3   4

0   0.0   2.0   4.0   6.0 NaN

1   9.0  11.0  13.0  15.0 NaN

2  18.0  20.0  22.0  24.0 NaN

3   NaN   NaN   NaN   NaN NaN

>>> a\*b

      0     1      2      3   4

0   0.0   1.0    4.0    9.0 NaN

1  20.0  30.0   42.0   56.0 NaN

2  80.0  99.0  120.0  143.0 NaN

3   NaN   NaN    NaN    NaN NaN

|  |  |
| --- | --- |
| 方法 | 说明 |
| .add(d,\*\*argws) | 类型间加法运算，可选参数 |
| .sub(d,\*\*argws) | 类型间减法运算，可选参数 |
| .mul(d,\*\*argws) | 类型间乘法运算，可选参数 |
| .div(d,\*\*argws) | 类型间除法运算，可选参数 |

>>> a

   0  1   2   3

0  0  1   2   3

1  4  5   6   7

2  8  9  10  11

>>> b

    0   1   2   3   4

0   0   1   2   3   4

1   5   6   7   8   9

2  10  11  12  13  14

3  15  16  17  18  19

>>> b.add(a,fill\_value=100)#fill\_value替代NaN，替代后参与运算

       0      1      2      3      4

0    0.0    2.0    4.0    6.0  104.0

1    9.0   11.0   13.0   15.0  109.0

2   18.0   20.0   22.0   24.0  114.0

3  115.0  116.0  117.0  118.0  119.0

>>> a.mul(b,fill\_value=0)

      0     1      2      3    4

0   0.0   1.0    4.0    9.0  0.0

1  20.0  30.0   42.0   56.0  0.0

2  80.0  99.0  120.0  143.0  0.0

3   0.0   0.0    0.0    0.0  0.0

数据的排序

.sort\_index(axis=0,ascending=True)

https://www.icourse163.org/topics/python-sp

### 利用爬虫技术能做到哪些很酷很有趣很有用的事情？

**利用爬虫技术能做到哪些很酷很有趣很有用的事情？**

准备学习python爬虫。各位大神都会用爬虫做哪些有趣的事情？ 今天突然想玩玩爬虫，就提了这个问题。跟着YouTube上的一个tutor写了个简单的程序…显示全部

关注者

**74,488**

被浏览

**6,957,984**

关注问题写回答

​62 条评论

​分享

​邀请回答

​

**655 个回答**

默认排序​

[Pacino](https://www.zhihu.com/people/ljx95)

[**Pacino**](https://www.zhihu.com/people/ljx95)

忙于追求更好的人生。

3,644 人赞同了该回答

我当初是看到这个帖子才知道Python这门语言的功能，才开始去学的，现在也学了一小段时间。不得不说，Python爬虫对于我来说真是个神器。之前在分析一些经济数据的时候，需要从网上抓取一些数据下来，想了很多方法，一开始是通过Excel，但是Excel只能爬下表格，局限性太大了。之后问了学编程的朋友，他说JavaScript也能实现，于是懵懵懂懂地就去学Java（我那朋友是学Java的，我当时问他用Java能不能实现，他说JavaScript好像可以，当时我什么都不懂，就把JavaScript理解成是Java下的一个分支，以为JavaScript只是Java其中一个包什么的，于是我便去学了一小会Java，无知惹的祸啊。。。）。但整个Java体系也太庞大了，学起来力不从心，毕竟我只是要运用其中一部分功能而已，于是学没多久我就放弃了。就在我迷茫的时候，我发现了Python大法……

废话说多了，说说自己的学习经历吧。也给想学Python，想写爬虫的人一个参考。  
一开始我是在网易云课堂上自己找了个基础的视频来学，Python是真是门简单的语言，之前懂一点Visual Basic，感觉Python也很适合给无编程基础的人学习。  
入门视频到最后，就做出了我的第一个爬虫——百度贴吧图片爬虫（相信很多的教程都是以百度贴吧爬虫为经典例子来说的。）

一开始代码很简单，只能爬取第一页的数据，于是我加了一个循环，就能够爬取制定页数的图片了。并且图片是有按顺序排列的，非常方便。在筛选网址的时候用正则表达式就好了。正则表达式使用：[[精华] 正则表达式30分钟入门教程](https://link.zhihu.com/?target=http%3A//www.oschina.net/question/12_9507)

### 【Python数据分析与展示】（八）数据合并、规整化

<https://blog.csdn.net/polarislove36/article/details/78847869>

【Python数据分析与展示】（八）数据合并、规整化

### Python数据分析与展示

**课程大纲**

【第〇周】数据分析之前奏

“数据分析”课程内容导学

Python语言开发工具选择

Anaconda IDE的基本使用方法

【第一周】数据分析之表示

第1周课程导学

单元1：NumPy库入门

单元2：NumPy数据存取与函数

单元3：实例1：图像的手绘效果

第一周测验：数据分析之表示

【第二周】数据分析之展示

第2周课程导学

单元4：Matplotlib库入门

单元5：Matplotlib基础绘图函数示例（5个实例）

单元6：实例2：引力波的绘制

第二周测验：数据分析之展示

【第三周】数据分析之概要

第3周课程导学

单元7：Pandas库入门

单元8：Pandas数据特征分析

单元9：实例3：房价趋势的关联因素分析

第三周测验：数据分析之概要

【第四周】数据分析之处理

第4周课程导学

单元10：数据的清洗和规约

单元11：时间序列的处理和展示

单元12：商品数据时序分析图

第四周测验：数据分析之处理

**预备知识**

本课程需要学习者具备Python语言编程的基本知识和初步技能，建议Python零基础学习者先修嵩老师的“[Python语言程序设计](https://www.icourse163.org/course/BIT-268001)”课程。具体地，学习者需要预先掌握Python的数字类型、字符串类型、分支、循环、函数、列表类型、字典类型、文件和第三方库使用等概念和编程方法。

**证书要求**

本课程采取百分制，60分-79分可获得合格证书，80分以上可获得优秀证书。

**参考资料**

**Python集成开发环境(IDE)**

[1] Anaconda: <https://www.continuum.io/> （推荐）

[2] IDLE: Python解释器默认工具

[3] PyCharm: <https://www.jetbrains.com/pycharm/>

**参考教程**

[1] Python零基础入门教程:《Python语言程序设计基础(第2版)》，嵩天、礼欣、黄天羽著，高等教育出版社，2017.2

教程链接：[京东自营旗舰店](https://item.jd.com/12128326.html?dist=jd)  [高教社天猫旗舰店](https://detail.tmall.com/item.htm?spm=a1z10.1-b-s.w19095786-17077471552.4.761f613c29xVTL&id=562950803430)  [亚马逊地址](https://www.amazon.cn/Python%E8%AF%AD%E8%A8%80%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1%E5%9F%BA%E7%A1%80-%E5%B5%A9%E5%A4%A9-%E7%A4%BC%E6%AC%A3-%E9%BB%84%E5%A4%A9%E7%BE%BD/dp/B06W9KM5P5/ref=sr_1_1?s=books&ie=UTF8&qid=1488437402&sr=1-1)

[2] 专题参考资料：《利用Python进行数据分析》，Wes McKinney著，O’Reilly & 机械工业出版社，2014.1（该书使用Python 2.x系列，内容略微陈旧，**仅做参考**，**不建议跟踪学习**）

**参考网站**

[1] Python NumPy: <https://www.numpy.org/>

[2] Python Matplotlib: <https://matplotlib.org/>

[3] Python Pandas: <https://pandas.pydata.org/>

**常见问题**

**Q1：**除了Python，这个课程需要其他编程语言基础吗？

**A1：**

不需要，只要具备基本的Python编程能力，就可以学习本课程。

**Q2：**Python 2.x和Python 3.x，这个课程采用哪个版本？

**A2：**

Python 3.x已经足够成熟，这是Python语言的现在和未来，嵩老师所有Python课程都采用Python 3.x系列版本。

**Q3：**在线开放课程看不到老师，有问题谁来解答？

**A3：**

为了更好服务同学们，本课程教师和助教会经常在线答疑，尽快解决与课程相关的各类问题。

**Q4：**有没有好的参考书推荐？

**A4：**

本课程提供了视频、教学资源，但暂时没有参考书推荐。如果大家发现该领域优秀的参考书，可以反馈给老师

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*