# Python基础

资源参考：

C语言中文网<http://c.biancheng.net/python/>

菜鸟教程-python基础 <https://www.runoob.com/python/python-tutorial.html>

菜鸟教程-numpy <https://www.runoob.com/numpy/numpy-tutorial.html>

菜鸟教程-pandas <https://www.runoob.com/pandas/pandas-tutorial.html>

菜鸟教程-matplotlib <https://www.runoob.com/matplotlib/matplotlib-tutorial.html>

pyecharts画图 <https://zhuanlan.zhihu.com/p/182532066>

# python环境

Anaconda+spyder/jupyter

Python+pycharm(推荐)

# 使用注意

命名：符合命名规则，命名要有一定的含义，尽量不要使用中文

# Python数字类型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表X python数字类型表 | | | | |
| 数字类型 | 整型 | 浮点型 | 复数类型 | 布尔类型 |
| class | int | float | complex | bool |
| 例子 | 423 | 4.23  4.23E8  4.23e-2 | 3+2j | True 1  False 0 |

通过**type(XXX)**查看数据类型

# 常用数据类型

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表X python常用数据类型表 | | | | | |
| 数据类型 | 字符串 | 列表 | 元组 | 字典 | 集合 |
| class | str | list | tuple | dict | set |
|  | s = “Yes” | l =[1,2,3,”Yes”] | t = (1,2,3,”Yes” ) | d={“a”:1;”b”:2} | set1=(1,) |

# 选择与循环

if语句 if...else.../if...elif...

|  |
| --- |
| **if** 判断条件：  执行语句…… **else**：  执行语句…… |

for循环/ while循环

|  |  |
| --- | --- |
| **while** 判断条件(condition)：  执行语句(statements)…… | **for** iterating\_var **in** sequence:  statements(s) |

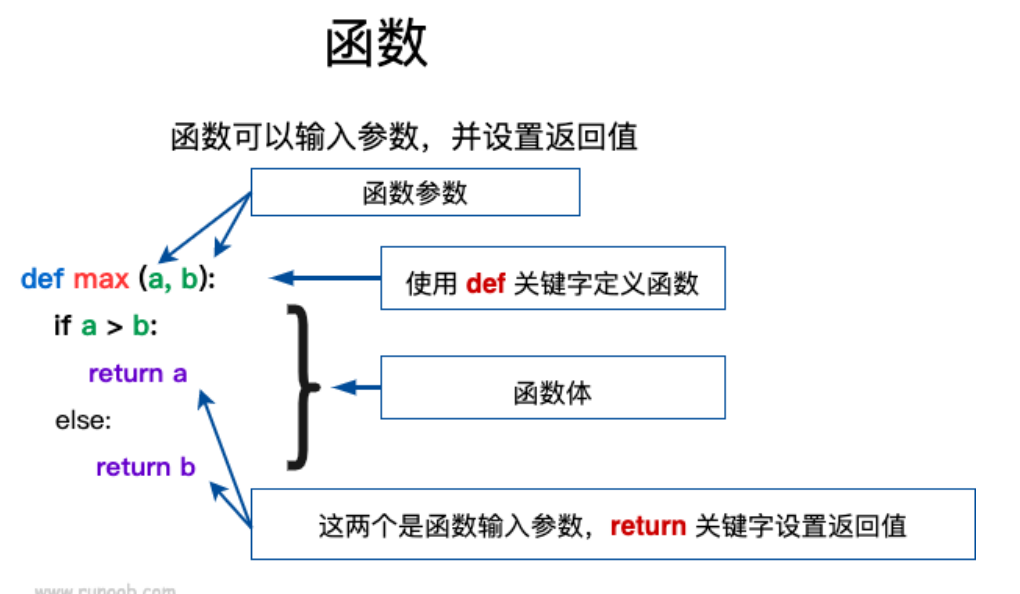
循环控制语句

|  |
| --- |
| **continue break pass** |

range(start,stop,step)

range(起始数,结束数（不包含）,步长)

# 函数



# 导包Numpy-Pandas-Matplotlib

标准库 直接导入

import time

|  |  |
| --- | --- |
| 库 | 作用 |
| math | 提供了数学常数和数学函数 |
| random | 用于生成随机数的库 |
| time | 时间模块 |

https://zhuanlan.zhihu.com/p/515075716

|  |
| --- |
| **import** sys print(sys.path)***#查看所有python路径*** |

第三方包 先安装(pip install)再导入(import)

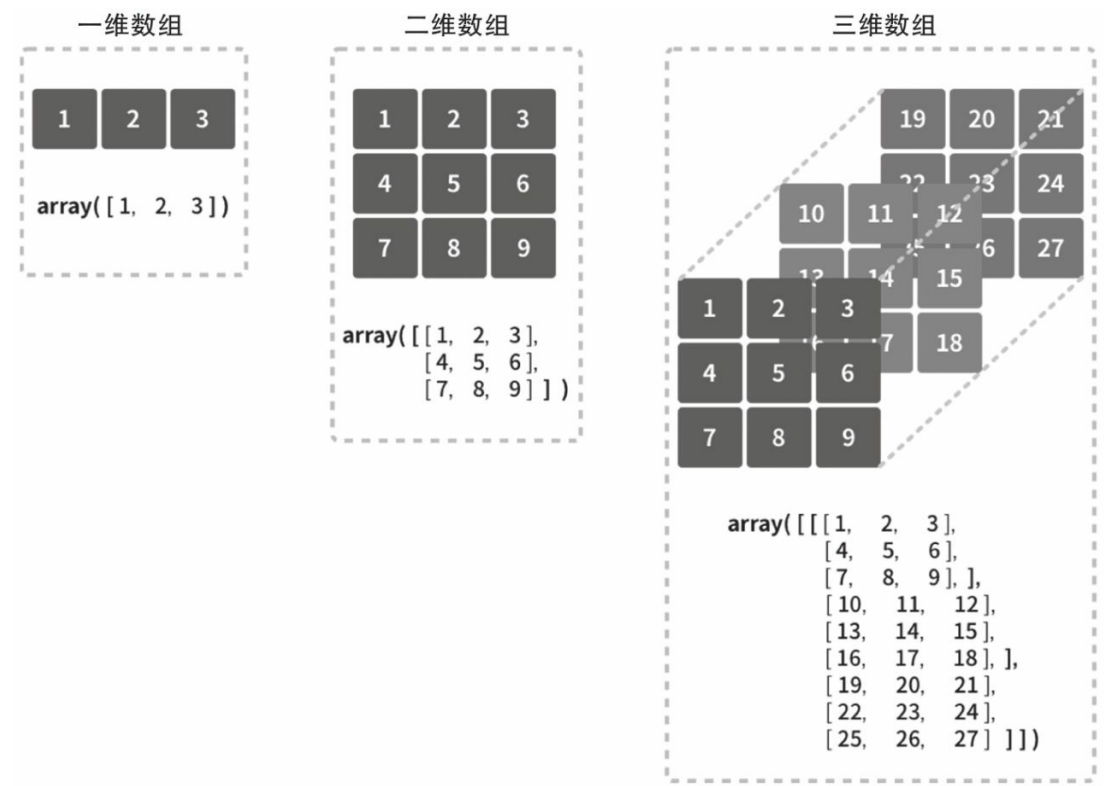
|  |  |
| --- | --- |
| 库 | 作用 |
| Numpy | 支持多维数组与矩阵运算 |
| scipy | 众多库函数,如线性代数、常微分方程数值求解、信号处理、图像处理、稀疏矩阵等，可进行插值处理、信号滤波，以及使用C语言加速计算。 |
| pandas | 读入数据 |
| matplotlib | 可视化 |

## 7.1 Numpy

### 7.1.1 NumPy 数组对象 ndarray

|  |
| --- |
| numpy.array(**object**, dtype=None, copy=True, order='K',subok=False, ndmin=0) |

|  |  |
| --- | --- |
| **属性** | **说明** |
| **ndim** | 返回 int。表示数组的维数 |
| **shape** | 返回 tuple。表示数组的尺寸，对于 n 行 m 列的矩阵，形状为(n,m) |
| **size** | 返回 int。表示数组的元素总数，等于数组形状的乘积 |
| **dtype** | 返回 data-type。描述数组中元素的类型 |
| **itemsize** | 返回 int。表示数组的每个元素的大小（以字节为单位）。 |



|  |
| --- |
| #创建数组 |
| **import** numpy **as** np  a = np.array([[1, 2], [3, 4]])  print(a) print(a.shape) |
| #arange 函数创建数组  #arange(开始值,结束值,步长) |
| arr2 = np.array(range(1,7))  arr3 = np.arange(1,7)  arr2.shape = 3,2  print(arr2) |
| #linspace 函数创建数组  #linspace(开始值,结束值,步数) |
| print(np.linspace(0, 1, 12)) |
| #zeros函数创建的数组 |
| print('使用zeros函数创建的数组为：',np.zeros((2,3))) |
| #eye函数创建数组 |
| print('使用eye函数创建的数组为：',np.eye(3)) |
| #np.random.random(n)  randn randint rand |
| print(np.random.random(10)) |
|  |

|  |  |
| --- | --- |
| **函数** | **说明** |
| **seed** | 确定随机数生成器的种子。 |
| **permutation** | 返回一个序列的随机排列或返回一个随机排列的范围。 |
| **shuffle** | 对一个序列进行随机排序。 |
| **binomial** | 产生二项分布的随机数。 |
| **normal** | 产生正态（高斯）分布的随机数。 |
| **beta** | 产生beta分布的随机数。 |
| **chisquare** | 产生卡方分布的随机数。 |
| **gamma** | 产生gamma分布的随机数。 |
| **uniform** | 产生在[0,1)中均匀分布的随机数。 |

|  |
| --- |
| #一维数组索引 |
| **import** numpy **as** np  a = np.array([[1, 2,3, 4]])  print(a[0:3]) |
| #二维数组索引 |
| **import** numpy **as** np  a = np.array([[1, 2, 3, 4],  [5, 6, 7, 8],  [9, 10, 11, 12],  [13, 14, 15, 16]]) print(a[0,0]) print(a) print(a[::2,::2]) print(a[(1,3),(2,3)]) |
| #更改形状 |
| a.shape = 4,1***#更改原数组的形状***  print(a) print(a.shape)  b = a.reshape(1,4)#***更改形状生成新数组***  a.***ravel()*** #***函数展平数组***  a.flatten()#横向展平  a.flatten(‘F’)#纵向展平 |
| 组合数组 |
| 使用hstack函数实现数组横向组合：np.hstack((arr1,arr2))  使用vstack函数实现数组纵向组合：np.vstack((arr1,arr2))  使用concatenate函数实现数组横向组合：np.concatenate((arr1,arr2),axis = 1)  使用concatenate函数实现数组纵向组合：np.concatenate((arr1,arr2),axis = 0) |
| 切割数组 |
| 使用hsplit函数实现数组横向分割： np.hsplit(arr1, 2)  使用vsplit函数实现数组纵向分割： np.vsplit(arr, 2)  使用split函数实现数组横向分割： np.split(arr, 2, axis=1)  使用split函数实现数组纵向分割： np.split(arr, 2, axis=0) |

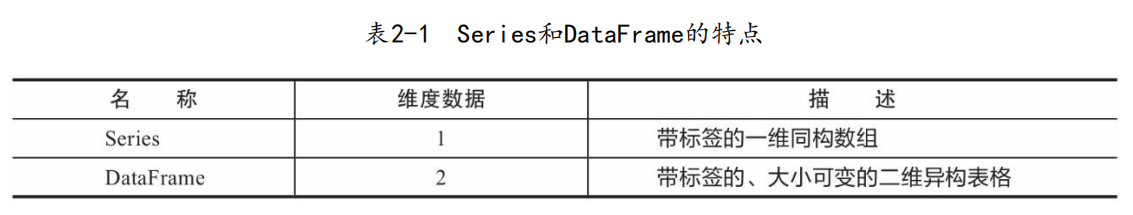
### 7.1.2 NumPy 矩阵与通用函数

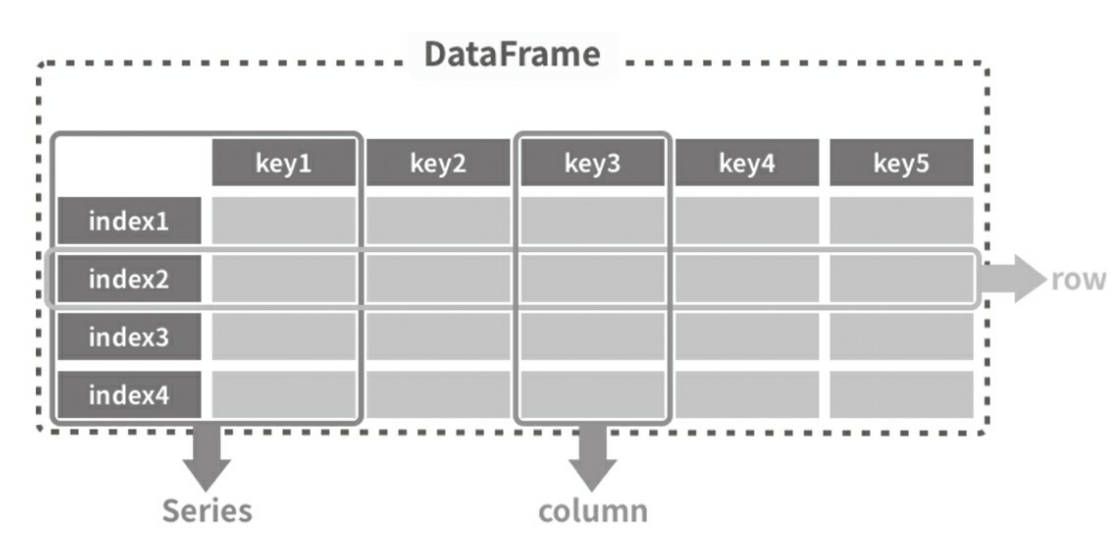
|  |
| --- |
| 创建与组合矩阵 |
| 使用mat函数创建矩阵： matr1 = np.mat("1 2 3;4 5 6;7 8 9")  使用matrix函数创建矩阵：matr2 = np.matrix([[123],[456],[789]])  使用bmat函数合成矩阵：np.bmat("arr1 arr2; arr1 arr2") |
| 矩阵的运算 |
| 矩阵与数相乘：matr1\*3  矩阵相加减：matr1±matr2  矩阵相乘：matr1\*matr2  矩阵对应元素相乘：np.multiply(matr1,matr2)   |  |  | | --- | --- | | **属性** | **说明** | | **T** | 返回自身的转置 | | **H** | 返回自身的共轭转置 | | **I** | 返回自身的逆矩阵 | | **A** | 返回自身数据的2维数组的一个视图 | |

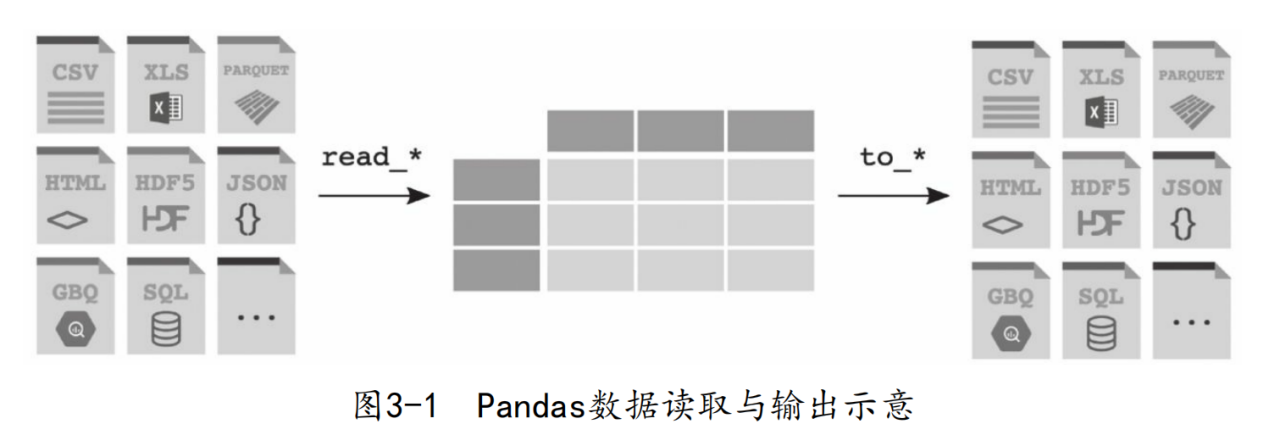
### 7.1.3 **NumPy 进行统计分析**

|  |
| --- |
| 数据读写  多个数组：savez函数可以将多个数组保存到一个文件中。 |
| #savez函数可以将多个数组保存到一个文件中。  np.savez(‘../tmp/savez\_arr’,arr1,arr2)，文件扩展名为.npz  #load函数是从二进制的文件中读取数据。 np.load("../tmp/savez\_arr.npz") |
| 保存和读取文本格式的数据 |
| savetxt函数是将数组写到某种分隔符隔开的文本文件中。  np.savetxt("../tmp/arr.txt", arr, fmt="%d", delimiter=",")  loadtxt函数执行的是把文件加载到一个二维数组中。  np.loadtxt("../tmp/arr.txt",delimiter=",")  genfromtxt函数面向的是结构化数组和缺失数据。  np.genfromtxt("../tmp/arr.txt", delimiter = ",") |
| 排序 |
| arr.sort() |
| 统计  当axis=0时，表示沿着纵轴计算。当axis=1时，表示沿着横轴计算。默认时计算一个总值。  print(a.sum(axis=1)) |
| |  |  | | --- | --- | | **函数** | **说明** | | **sum** | 计算数组的和 | | **mean** | 计算数组均值 | | **std** | 计算数组标准差 | | **var** | 计算数组方差 | | **min** | 计算数组最小值 | | **max** | 计算数组最大值 | | **argmin** | 返回数组最小元素的索引 | | **argmax** | 返回数组最小元素的索引 | | **cumsum** | 计算所有元素的累计和 | | **cumprod** | 计算所有元素的累计积 | | **median()** | 计算数组中元素的中位数（中值） | |
| #去重unique  numpy.where() 函数返回输入数组中满足给定条件的元素的索引。 |
|  |

## 7.2 Pandas



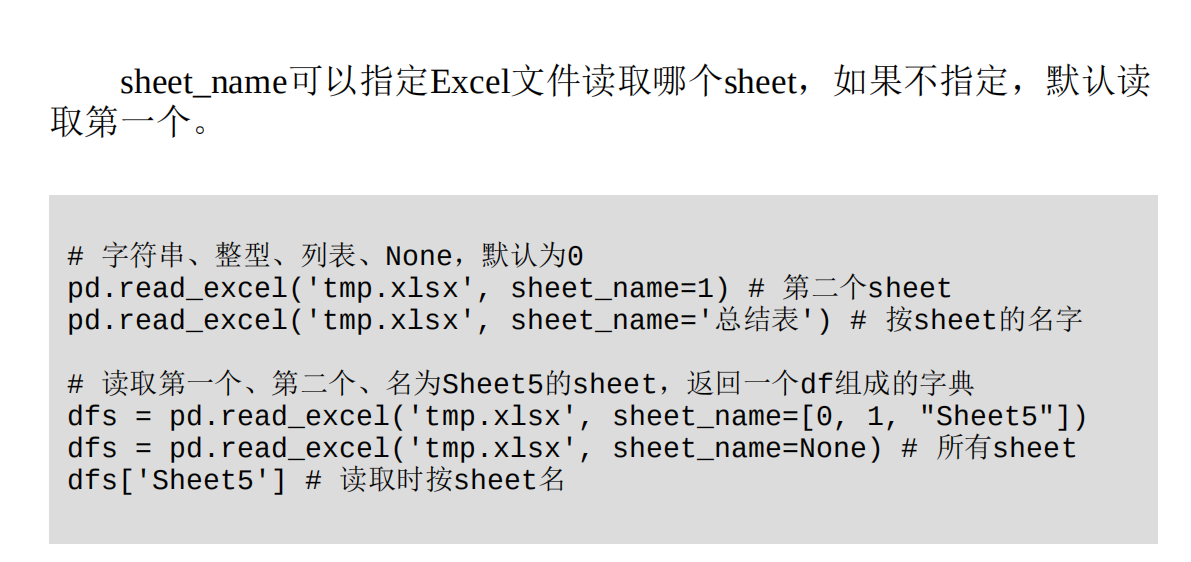




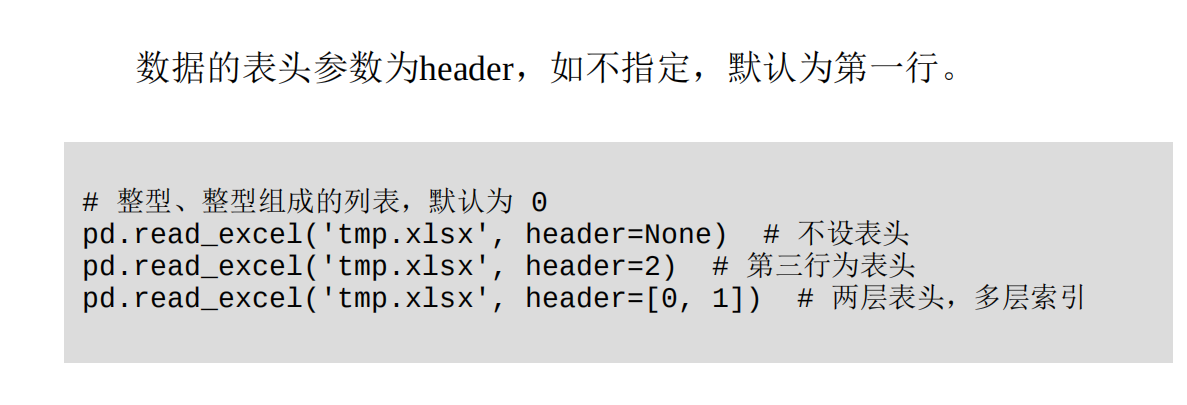
### 7.2.1读入excel

|  |
| --- |
| 读入**read\_excel(xlsxpath1)** |
| **import** pandas **as** pd **import** numpy **as** np ***#设置读入数据的路径*** xlsxpath0 = **'../data01/附件1：物流网络历史货量数据.xlsx'** xlsxpath1 =**'../data01/练习数据30行.xlsx' *#读入excel数据*** data = **pd.read\_excel(xlsxpath1)** |

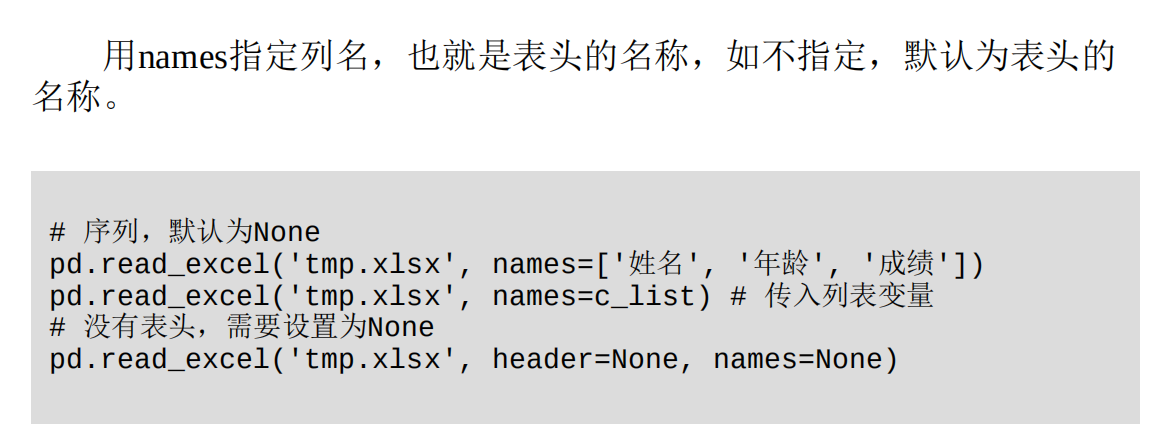
### 7.2.2选择读取excel的表格——sheet

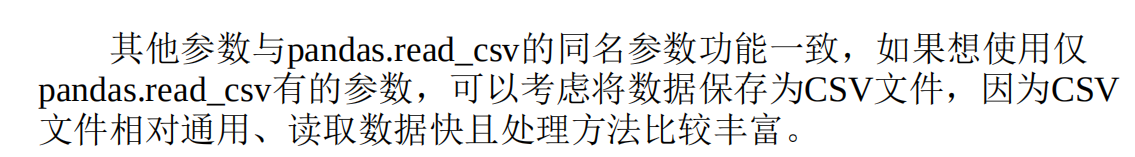


### 7.2.3设置excel的表头



### 7.2.4设置excel的列名





## 7.3 Matplotlib

### 7.3.1 Pyplot

|  |  |
| --- | --- |
| plot() | 用于绘制线图和散点图 |
| scatter() | 用于绘制散点图 |
| bar() | 用于绘制垂直条形图和水平条形图 |
| hist() | 用于绘制直方图 |
| pie() | 用于绘制饼图 |
| imshow() | 用于绘制图像 |
| subplots() | 用于创建子图 |

|  |  |
| --- | --- |
| **#plot()用于绘制线图和散点图** |  |
| **import** matplotlib.pyplot **as** plt **import** numpy **as** np xpoints = np.array([0, 6]) ypoints = np.array([0, 100]) plt.plot(xpoints, ypoints) plt.show() |  |
| **import** matplotlib.pyplot **as** plt **import** numpy **as** np x = np.arange(0,4\*np.pi,0.1) ***# start,stop,step*** y = np.sin(x) z = np.cos(x) plt.plot(x,y,x,z) plt.show() |  |
| **#maker绘图标记** |  |
| **import** matplotlib.pyplot **as** plt **import** numpy **as** np ypoints = np.array([1,3,4,5,8,9,6,1,3,4,5,2,4]) plt.plot(ypoints, marker = **'o'**) plt.show() |  |
| **#linestyle绘图线** |  |
| **import** matplotlib.pyplot **as** plt **import** numpy **as** np ypoints = np.array([6, 2, 13, 10]) plt.plot(ypoints, linestyle = **'dotted'**) plt.show()  **#color颜色** |  |
| **import** matplotlib.pyplot **as** plt **import** numpy **as** np ypoints = np.array([6, 2, 13, 10]) plt.plot(ypoints, color = **'r'**) plt.show()  **#https://www.runoob.com/html/html-colorvalues.html** |  |
| **#linewidth线的宽度** |  |
| **import** matplotlib.pyplot **as** plt **import** numpy **as** np ypoints = np.array([6, 2, 13, 10]) plt.plot(ypoints, linewidth = **'12.5'**) plt.show() |  |
| **#绘制多条线** |  |
| **import** matplotlib.pyplot **as** plt **import** numpy **as** np ypoints = np.array([6, 2, 13, 10]) ypoints2 = np.array([6.1, 2.9, 13, 10.22]) plt.plot(ypoints, linewidth = **'12.5'**) plt.plot(ypoints2, linewidth = **'3'**) plt.show() |  |
| **#轴标签和标题** |  |
| **import** matplotlib.pyplot **as** plt **import** numpy **as** np x = np.arange(0,4\*np.pi,0.1) ***# start,stop,step*** y = np.sin(x) z = np.cos(x) plt.plot(x,y,x,z)  plt.xlabel(**"x - label"**) plt.ylabel(**"y - label"**) plt.title(**"sin(x) and cos(x)"**) plt.show() |  |
| **#轴标签和标题的字体和颜色** |  |
| **import** matplotlib.pyplot **as** plt **import** numpy **as** np x = np.array([0, 1, 2, 3, 4]) y = np.array([4, 3, 2, 1, 4]) plt.bar(x, y) plt.title(**r'Example of $\frac{1}{2}$'**, font={**'family'**:**'Arial'**, **'size'**:18}) plt.ylabel(**r'Example of $\beta$'**, font={**'family'**:**'Arial'**, **'size'**:16}) plt.xlabel(**r'Example of $\alpha$'**, font={**'family'**:**'Arial'**, **'size'**:16}) plt.show()  **#https://www.latexlive.com/** |  |
| **#刻度** |  |
| set\_xlim()  set\_ylim() |  |
| **#图例** |  |
| **import** matplotlib.pyplot **as** plt **import** numpy **as** np x = np.arange(0,4\*np.pi,0.1) y = np.sin(x) z = np.cos(x) plt.plot(x,y,label=**"sin(x)"**) plt.plot(x,z,label=**"cos(x)"**) plt.legend() plt.xlabel(**"x - label"**) plt.ylabel(**"y - label"**) plt.title(**"sin(x) and cos(x)"**) plt.show() |  |
| **#子图** |  |
| **import** matplotlib .pyplot **as** plt x=[1,2,3,4,5] y=[3,6,7,9,2] ***# 实例化两个子图(1,2)表示1行2列*** fig,ax=plt.subplots(1,2) ax[0].plot(x,y,label=**'trend'**) ax[1].plot(x,y,color=**'cyan'**) ax[0].set\_title(**'title 1'**) ax[1].set\_title(**'title 2'**) plt.show() |  |
| **#网格线** |  |
| **import** numpy **as** np **import** matplotlib.pyplot **as** plt x = np.array([1, 2, 3, 4]) y = np.array([1, 4, 9, 16]) plt.xlabel(**"x - label"**) plt.ylabel(**"y - label"**) plt.plot(x, y) plt.grid(color = **'r'**, linestyle = **'--'**, linewidth = 0.5) plt.show() |  |
| **#保存图片** |  |
| **import** numpy **as** np **import** matplotlib.pyplot **as** plt x = np.array([1, 2, 3, 4]) y = np.array([1, 4, 9, 16]) plt.xlabel(**"x - label"**) plt.ylabel(**"y - label"**) plt.plot(x, y) plt.grid(color = **'r'**, linestyle = **'--'**, linewidth = 0.5) plt.savefig(**'MyFigure.png'**, dpi=200) plt.show() |  |
| **#双坐标轴** |  |
| **import** numpy **as** np **import** matplotlib.pyplot **as** plt  x = np.arange(0., np.e, 0.01) y1 = np.exp(-x) y2 = np.log(x) fig = plt.figure() ax1 = fig.add\_subplot(111) ax1.plot(x, y1) ax1.set\_ylabel(**'Y values for exp(-x)'**) ax1.set\_title(**"Double Y axis"**) ax2 = ax1.twinx() ***# this is the important function*** ax2.plot(x, y2, **'r'**) ax2.set\_xlim([0, np.e]) ax2.set\_ylabel(**'Y values for ln(x)'**) ax2.set\_xlabel(**'Same X for both exp(-x) and ln(x)'**) plt.show() |  |

### 7.3.2 bar

|  |  |
| --- | --- |
| **#柱形图** |  |
| **import** matplotlib.pyplot **as** plt **import** numpy **as** np x = np.array([**"Runoob-1"**, **"Runoob-2"**, **"Runoob-3"**, **"C-RUNOOB"**]) y = np.array([12, 22, 6, 18]) plt.bar(x,y) plt.show() |  |

### 7.3.3 pie

|  |  |
| --- | --- |
| **#扇形图** |  |
| **import** matplotlib.pyplot **as** plt **import** numpy **as** np y = np.array([35, 25, 25, 15]) plt.pie(y) plt.show() |  |

### 7.3.4 scatter

|  |  |
| --- | --- |
| **#散点图** |  |
| **import** matplotlib.pyplot **as** plt **import** numpy **as** np x = np.array([1, 2, 3, 4, 5, 6, 7, 8]) y = np.array([1, 4, 9, 16, 7, 11, 23, 18]) plt.scatter(x, y) plt.show() |  |

# 练习题

1.2022年赛题

全国大学生数学建模竞赛赛题

<http://www.mcm.edu.cn/html_cn/block/8579f5fce999cdc896f78bca5d4f8237.html>

**针对E题数据进行处理**

1.统计所有物料的编号以及出现的次数。

2.选取6种物料，分别统计6种物料，最大需求值，最高单价，需求的平均值和方差，单价的平均值和方差，以及其他认为应当统计的数据，并填入表格。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 物料1 | 物料2 | 物料3 | 物料4 | 物料5 | 物料6 |
| 物料编号 |  |  |  |  |  |  |
| 需求最大值 |  |  |  |  |  |  |
| 需求平均数 |  |  |  |  |  |  |
| 需求中位数 |  |  |  |  |  |  |
| 需求方差 |  |  |  |  |  |  |
| 单价最大值 |  |  |  |  |  |  |
| 单价平均值 |  |  |  |  |  |  |
| 单价方差 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

3.分别绘制6种物料需求与价格，需求与时间之间的散点图或折线图。

|  |
| --- |
| ***仅供参考*** |
| ***#导包* import** pandas **as** pd **import** numpy **as** np **import** matplotlib.dates **as** mdates **import** matplotlib.pyplot **as** plt **from** datetime **import** datetime ***#设置读入数据的路径*** xlsxpath0 = **'../data01/附件.xlsx' *#读入excel数据*** data = pd.read\_excel(xlsxpath0) ***#将数据转换为numpy数组*** data1 = np.array(data) ***#筛选物料编号为6004020664的物料索引*** index1=np.where(data1 == 6004020664) ***#取出物料编号为6004020664的物料信息*** dt3 = data1[index1[0],:] demand = dt3[:,2] ***#6004020664的物料需求*** price = dt3[:,3] ***#6004020664的物料价格*** time1 = dt3[:,0] ***#6004020664的物料时间*** fig,ax=plt.subplots(2,1) ax[0].plot(time1, price) ***#6004020664物料价格随时间的波动*** ax[1].plot(time1, demand) ***#6004020664物料需求随时间的波动*** ax[1].set\_xlabel(**"time"**) ax[1].set\_ylabel(**"demand"**) ax[0].set\_ylabel(**"price"**) ***# ax[2].scatter(price, demand)*** ax[1].xaxis.set\_major\_formatter(mdates.DateFormatter(**'%Y%m'**)) ax[0].xaxis.set\_major\_formatter(mdates.DateFormatter(**'%Y%m'**)) plt.gcf().autofmt\_xdate() ***# ax[1].xaxis.set\_major\_locator(mdates.YearLocator()) # https://zhuanlan.zhihu.com/p/399872495*** plt.show() |
|  |