- numpy
- pandas
- matplotlib

numpy

```
import numpy as np
#数组的创建：
# 一维数组
data_1 = np.array([1,2,3,4])
print(data_1)
data_1 = np.arange(1,10,2)
print(data_1)

# 二维数组
data_2 = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
print(data_2)

# 全0数组
#shape代表形状，比如我这里创建的就是2行三列的2维数组
data_3 = np.zeros(shape=(2,3))
print(data_3)

# 全1数组
data_4 = np.ones(shape=(2,3))
print(data_4)

# 全空数组
data_5 = np.empty(shape=(2,3))
print(data_5)

# 有连续序列的数组 arange
data_6 = np.arange(10,16,2) # 10-16的数据，步长为2
print(data_6)

# 有连续间隔的数组 linspace
# 创建线段型数据
data_7= np.linspace(1,10,10) # 开始端1，结束端10，且分割成10个数据，生成线段
print(data_7)

# 随机数组
data_8 = np.random.rand(3,4) # 3行4列的随机数组
print(data_8)
data_8=np.random.randint(2,5,size=(3,4)) # 3行4列,每个元素值在2~5之间的随机数组
print(data_8)

# 改变数组形状
data1=[1,2,3,4,5]
data2=[1,2,3,4,5]
data=np.array([data1,data2])
```

```
print("改之前的数组形状为:")
print(data.shape)
data=data.reshape((5,2))
print("改之后的数组形状为:")
print(data.shape)

# 数组转置
data = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
data_array = np.array(data)
print("没有转置数组之前数组为：")
print(data)
print("转置数组之后数组为：")
print(data_array.T)

# 数组的查看:
# 数组维度 ndim
print(data_1.ndim)
# 数组形状shape
print(data_1.shape)
# 数组中元素个数
print(data_1.size)
# 数组的数据类型 dtype
print(data_1.dtype)

#数组的运算:
# 数组相加
array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])
result = array1 + array2
print(result)
# 数组相乘
result=array1*array2
print(result)

# 数据统计:
```

pandas

```
import pandas as pd
import numpy as np

# Series数据结构
# Series：一维的带索引数据结构(单列)
# 第一列为索引，第二列为Series数据
sdata = pd.Series(np.arange(1,4), index=list('abc'))
print(sdata)
# index查看下标(索引/标签)，values查看下标的值()
print(sdata.index)
print(sdata.values)
# iloc and loc的使用
# iloc是原下标，也就是默认值，计算机的记忆
```

```
# loc是修改过的下标，我们把他叫作标签，标签是由我们自主给的，计算机并不会自己产生
print(sdata.iloc[0])
# 使用标签[a,b,c]
print(sdata['b'])
# 使用loc方式，只能使用标签
print(sdata.loc['c'])

# 创建对象
s1 = pd.Series({"小李":90,"小王":85}) #前键后值
print(s1)
# 使用loc修改数据
s1.loc["小李"]=100
print("小李的成绩：",s1.loc["小李"])

# 数据相加、相减、相乘、相除
# 标签会自动对应
s2 = pd.Series([1,2,3,4,5],index=[1,2,3,4,5])
s3 = pd.Series([2,3,1,6,4],index=[2,1,3,5,6])
print(s2+s3) # 使用这种相加，没有对应标签的情况下会错误，显示NAN
print(s2.add(s3,fill_value=0)) # s2.add（s3，fill_value =“如果出现没有值的情况，以0代
替”）
# 同理
print(s2.sub(s3,fill_value=0))
print(s2.mul(s3,fill_value=1)) # 需注意的是，乘除不要设为0
print(s2.div(s3,fill_value=1))

# 求最大值、最小值、求和值、平均值
print(s2.max())
print(s2.min())
print(s2.sum())
print(s2.mean())

# DataFrame创建二维数组
# 一维数据
data = pd.DataFrame(data=np.arange(1,5))
print(data)
# 多维数据 data为4X4
data = np.arange(16).reshape(4,4)
data = pd.DataFrame(data=data)
print(data)
# 设置index与columns
data = np.arange(16).reshape(4,4)
df = pd.DataFrame(data=data, index=list('abcd'), columns=['c1','c2','c3','c4'])
print(df)
#
s_id = pd.Series(["01","02","03","04"],index=["小明","小李","小红","小王"])
s_class = pd.Series(["一班","二班","三班","一班"],index=["小明","小李","小红","小王"])
s_grade = pd.Series([92,89,95,87],index=["小明","小李","小红","小王"])
df = pd.DataFrame({"学号":s_id,"班级":s_class,"成绩":s_grade})
print(df)


# 查看行
print(df.index)
# 查看列
print(df.columns)
# 查看数据
```

```python
print(df.values)
# 数据倒置(行列转换)
print(df.T)

# 切片
# 标签切片
print(df.loc["小明":"小红"])
# 下标切片
print(df.iloc[0:3])
# 行列切片
print(df.loc["小明":"小红","学号":"班级"])
# 行列切片(下标)
print(df.iloc[0:2,0:2])

import pandas as pd
import os
# 数据导入
# 使用绝对路径
# fpath_excel = r"D:\知行合一\数据分析\python\data\test.xlsx"
# fpath_csv = r"D:\知行合一\数据分析\python\data\Fitness_Tracker_Data.csv"

# 使用相对路径
# 获取脚本所在的目录
script_dir = os.path.dirname(os.path.abspath(__file__))
# 设置当前工作目录为脚本所在的目录
os.chdir(script_dir)

fpath_excel = r"data\test.xlsx"
fpath_csv = r"data\Fitness_Tracker_Data.csv"

# 读取 Excel 文件
pdata_excel = pd.read_excel(fpath_excel)
print("Excel 文件内容：")
print(pdata_excel)

# 读取 CSV 文件
pdata_csv = pd.read_csv(fpath_csv, encoding='gbk')
print("\nCSV 文件内容：")
print(pdata_csv)

# 数据保存
# 保存为 CSV 文件
df = pd.DataFrame({'Name': ['Al', 'B', 'C'], 'Age': [25, 30, 35]})
df.to_csv('output.csv', index=False, encoding='utf-8') # index=False表示不带行索引
# 保存为 Excel 文件
df.to_excel('output.xlsx', sheet_name='Sheet1', index=False)

# 保存为 JSON 文件
df.to_json('output.json', orient='records', indent=4)

# 保存为 HTML 文件
df.to_html('output.html', index=False)

#

import pandas as pd
import numpy as np
```

```python
# 缺失数据处理
s1 = [1,2,3,4,5]
s2 = [2,3,None,1,None]
s3 = [7, 6.9,7.5,None,1]
s4 = [4, 6.9,3.3,7.2,4]

data = pd.DataFrame({'s1':s1,'s2':s2,'s3':s3,'s4':s4})
print(data)

# 判断方法
sdata = pd.isnull(data) # 缺省值对应的值为True，返回值为Boolean的Series或者DataFrame对象
print(sdata)
sdata = pd.notnull(data) # 缺省值对应的值为False，返回值为Boolean的Series或者DataFrame对象
print(sdata)

# 判断是否有缺失值
# pd.notnull,若包含缺省值，缺省值对应值为False
# np.all：若对象中包含假，返回False， 否则返回真
# 返回False， 说明包含缺省值
sdata = np.all(pd.notnull(data))
print(sdata)

# isnull：缺省值对应值为True
# any:对象中包含真，返回True
sdata = np.any(pd.isnull(data))
print(sdata)
# 返回False,说明不含缺省值

# 过滤数据
# 布尔索引方法
# 获取boolean索引
sdata = np.all(data.notnull(), axis=1) # axis 在布尔索引和 dropna 方法中有不同的含义

# 获取无空值的数据
print(data[sdata])

sdata_columns = np.all(data.notnull(), axis=0)  # axis=0 表示按列操作
print("按行布尔索引筛选后的数据：\n",sdata_columns)

# 使用布尔索引按列筛选数据
filtered_data = data.loc[:, sdata_columns]  # 使用 loc 选择列
print("按列布尔索引筛选后的数据：")
print(filtered_data)




# 删除包含缺省值的行
print(data.dropna())
# 删除包含缺省值列
print(data.dropna(axis=1)) # axis参数：0或'index'：按行操作，1或'columns'：按列操作

# 填充数据
# 固定值0
```

```
print(data.fillna(0))
# 使用前一行数据填充
print(data.fillna(method='ffill'))
# 使用后一行数据填充
print(data.fillna(method='bfill'))
# 插入均值
print(data.fillna(data.mean()))
```

matplotlib

```
import numpy as np
import matplotlib.pyplot as plt

# Figure对象（包含一个或多个子图（Axes）的顶层容器。）
fig = plt.figure(facecolor='c')
# Axes（图形中的一个绘图区域，可以包含多个绘图元素（如线条、文本、标签等）。）
ax = fig.add_subplot(111) # 在画板的第1行第1列的第一个位置生成一个Axes对象来准备作画

ax.set(xlim=[0.5, 4.5], ylim=[-2, 8], title='An Example Axes',
       ylabel='Y-Axis', xlabel='X-Axis')
plt.show() # 每次调用plt.show()方法后，会将所有绘图元素释放出来，所以对元素的修改应该在该方法之前

# 折线图
x=[1,2,3,4,5]
y=[2,1,3,5,4]
plt.plot(x,y,color='#A7D676',linestyle='--
',marker='*',markerfacecolor='b',markersize=15)
plt.show()

#点密集后，折线图会变得光滑
x=np.linspace(0,2*np.pi,200) # 使用linspace 函数生成从 0 到 2π 的 200 个等间隔点。
y=np.sin(x)
fig=plt.figure()
plt.plot(x,y,linestyle='-',color='c')
plt.title('Line Plot',fontsize=18) # 添加标题，字体大小设置为 18。
plt.xlabel('x',loc='right')
plt.ylabel('y',loc='top',rotation=0) # 设置 y 轴标签为 'y'，loc='top' 表示将标签放置在 y 轴的上方，rotation=0 表示不旋转标签，使其水平显示。
plt.grid(axis='y',linestyle='--',color='k',alpha=0.5) # 只显示y轴网格线

plt.show()
```