计算机与信息工程学院实验报告

••••••••••••••••••••••••••••••••• 密 ••••••••••••••••••••••••••••••••• 封 ••••••••••••••••••••••••••••••••• 线 •••••••••••••••••••••••••••••••••

姓名：\_杨佳森\_学号：\_\_2112080106\_\_专业：\_\_数据科学与大数据技术\_\_\_年级：\_ 2021

课程： 数据分析和可视化 主讲教师：\_周黎明 辅导教师：\_无

实验时间： 2023 年\_3\_月\_1\_日 上 午10\_时至\_12\_时，实验地点\_计算机大楼201\_

实验题目： 数据分析基础

实验目的： 掌握 Anaconda 与 Jupyter Notebook 的安装与使用方法 ；掌握 NumPy 创建数组、矩阵的方法以及常用的方法 ；掌握 NumPy 统计分析方法 。

实验环境（硬件和软件） 硬件环境：普通 PC 机 软件环境：Windows 操作系统、Jupyter Notebook

实验内容：

（一）安装 Anaconda 软件并使用 Jupyter Notebook

1、启动 Jupyter Notebook。

2、新建一个 Notebook 。

3、使用 Markdown 功能，创建标题、列表、字体、表格、数学公式 。

（二）NumPy 数值计算

1、创建 NumPy 数组对象 ndarray，并生成随机数，通过索引访问数组，并 变换数组的形态。

2、创建 NumPy 矩阵，使用 ufunc 函数。

3、利用 NumPy 进行统计分析。

实验步骤：

1. 在官网[蟒蛇 |全球最受欢迎的数据科学平台 (anaconda.com)](https://www.anaconda.com/)下载并安装Anaconda。
2. 在终端输入 jupyter notebook 命令，启动Jupyter Notebook。
3. 新建一个 Notebook，选择python3。
4. 练习Markdown功能。
5. 练习NumPy数值计算。

实验数据记录：

**1、Markdown**

#标题代码

# 一级标题

## 二级标题

### 三级标题

#无序列表

\* python

+ python2

- python3

#有序列表

1. python

2. python

3. python

#字体

数据分析和可视化

**\*\*数据分析和可视化\*\***

**\_\_数据分析和可视化\_\_**

**\*\*\*数据分析和可视化\*\***\*

**\_\_\_数据分析和可视化\_\_**\_

#表格

Python | R | MATLAB |

----------- | --------- | ------- |

接口统一，学习曲线平缓 | 接口众多，学习曲线陡峭 | 自由度大，学习曲线平缓 |

开源免费 | 开源免费 | 商业收费 |

#公式表达式:

$ E = mc^2 $

$$ z = \frac{x}{y} $$

**2.1、掌握NumPy数组对象ndarray**

NumPy提供了两种基本的对象：ndarray和ufunc。ndarray是存储单一数据类型的多维数组，而ufunc则是能够对数组进行处理的函数。

**#创建数组并查看数组属性**

import numpy as np

arr1 = np.array([1,2,3,4]) *#创建一维数组*

print('创建的数组为：',arr1)

*#创建的数组为： [1 2 3 4]*

arr2 = np.array([[1,2,3,4],[4,5,6,7],[7,8,9,10]]) *#创建二维数组*

print('创建的数组为：\n',arr2)

*# 创建的数组为：*

*#  [[ 1  2  3  4]*

*#  [ 4  5  6  7]*

*#  [ 7  8  9 10]]*

print（'数组维度为：'，arr2.shape） *#查看数组结构*

*#数组维度为： (3, 4)*

print（'数组类型为：'，arr2.dtype） *#查看数组类型*

*#数组类型为：int32*

print（'数组元素个数为：'，arr2.size） *#查看数组元素个数*

*#数组元素个数为：12*

print（'数组每个元素大小为：'，arr2.itemsize） *# 查看数组每个元素大小*

*#数组每个元素大小为：4*

*#重新设置数组的shape属性*

arr2.shape = 4，3 *#重新设置sahpe*

print（'重新设置形状后的arr2为：'，arr2）

*# 重新设置shape后的arr2为： [[ 1  2  3]*

*#  [ 4  4  5]*

*#  [ 6  7  7]*

*#  [ 8  9 10]]*

print('使用arange函数创建的数组为：\n',np.arange(0,1,0.1))

*# 使用arange函数创建的数组为：*

*#  [0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9]*

print('使用linspace函数创建的数组为：',np.linspace(0,1,12))

*# 使用linspace函数创建的数组为： [0.         0.09090909 0.18181818 0.27272727 0.36363636 0.45454545*

*#  0.54545455 0.63636364 0.72727273 0.81818182 0.90909091 1.       ]*

print('使用logspace函数创建的数列为：',np.logspace(0,2,20))

*# 使用logspace函数创建的数列为： [  1.           1.27427499   1.62377674   2.06913808   2.6366509*

*#    3.35981829   4.2813324    5.45559478   6.95192796   8.8586679*

*#   11.28837892  14.38449888  18.32980711  23.35721469  29.76351442*

*#   37.92690191  48.32930239  61.58482111  78.47599704 100.        ]*

print('使用zeros函数创建的数组为：',np.zeros((2,3)))

*# 使用zeros函数创建的数组为： [[0. 0. 0.]*

*#  [0. 0. 0.]]*

print('使用eye函数创建的数组为：',np.eye(3))

*# 使用eye函数创建的数组为： [[1. 0. 0.]*

*#  [0. 1. 0.]*

*#  [0. 0. 1.]]*

print('使用diag函数创建的数组为：',np.diag([1,2,3,4]))

*# 使用diag函数创建的数组为： [[1 0 0 0]*

*#  [0 2 0 0]*

*#  [0 0 3 0]*

*#  [0 0 0 4]]*

print('使用ones函数创建的数组为：',np.ones((5,3)))

*# 使用ones函数创建的数组为： [[1. 1. 1.]*

*#  [1. 1. 1.]*

*#  [1. 1. 1.]*

*#  [1. 1. 1.]*

*#  [1. 1. 1.]]*

print('转换结果为：',np.float64(42)) *#整型转换为浮点型*

*# 转换结果为： 42.0*

print('转换结果为：',np.int8(42.0)) *#浮点型转换为整型*

*# 转换结果为： 42*

print('转换结果为：',np.bool(42)) *#整型转换为布尔型*

*# 转换结果为： True*

print('转换结果为：',np.bool(0)) *#整型转换为布尔型*

*# 转换结果为： False*

print('转换结果为：',np.float(True)) *#浮点型转换为布尔型*

*# 转换结果为： 1.0*

print('转换结果为：',np.float(False)) *#浮点型转换为布尔型*

*# 转换结果为： 0.0*

df = np.dtype([("name",np.str\_,40),("numitems",np.int64),("price",np.float64)])

print('数据类型为：',df)

*# 数据类型为： [('name', '<U40'), ('numitems', '<i8'), ('price', '<f8')]*

print('数据类型为：',df["name"])

*# 数据类型为： <U40*

print('数据类型为：',np.dtype(df["name"]))

*# 数据类型为： <U40*

itemz = np.array([("tomatoes",42,4.14),("cabbages",13,1.72)],dtype=df)

print('自定义数据为：',itemz)

*# 自定义数据为： [('tomatoes', 42, 4.14) ('cabbages', 13, 1.72)]*

print('生成的随机数组为：',np.random.random(100)) #随机数

*# 生成的随机数组为： [0.53930514 0.84450389 0.01317895 0.68622834 0.23670795 0.58745146*

*#  0.44637637 0.87780438 0.58615098 0.04094569 0.58374124 0.03225865*

*#  0.80116206 0.63558673 0.06958795 0.19291817 0.65705074 0.72046962*

*#  0.55017025 0.26480873 0.9359779  0.16445065 0.00849382 0.36996273*

*#  0.56575854 0.45305267 0.97610962 0.44268653 0.97055055 0.55053466*

*#  0.62143885 0.84938341 0.73157828 0.96871069 0.29287076 0.75781211*

*#  0.67592803 0.27016642 0.12059112 0.0304136  0.4870118  0.46050311*

*#  0.81862425 0.60461839 0.19819039 0.57886891 0.67130008 0.0209593*

*#  0.05798908 0.35118276 0.27123874 0.76480324 0.2819527  0.75499786*

*#  0.49013706 0.02722568 0.95441286 0.37924686 0.45812459 0.16810892*

*#  0.61441896 0.92944406 0.85663386 0.56415564 0.63139827 0.25423349*

*#  0.60560596 0.48642021 0.66352376 0.32539252 0.30096644 0.74704438*

*#  0.62120803 0.42360001 0.75802353 0.69290324 0.14413101 0.81574804*

*#  0.61366728 0.24679437 0.9129756  0.20335574 0.03566932 0.70048207*

*#  0.05579011 0.9224947  0.18170251 0.23445088 0.24588901 0.13253485*

*#  0.51188186 0.86509472 0.97236073 0.38382471 0.64598023 0.04011793*

*#  0.86056525 0.7426921  0.18989518 0.45289637]*

print('生成的随机数组为：\n',np.random.rand(5,5)) *#生成服从均匀分布的随机数*

*# 生成的随机数组为：*

*#  [[0.92401414 0.29075855 0.59211202 0.33541952 0.06472536]*

*#  [0.32494565 0.58316906 0.18659544 0.58227486 0.66979874]*

*#  [0.96342811 0.95186437 0.88161405 0.21508702 0.07084674]*

*#  [0.96667877 0.90278393 0.98292452 0.84950626 0.76455417]*

*#  [0.7164529  0.20253333 0.25245437 0.72863703 0.19477701]]*

print('生成的随机数组为：\n',np.random.randn(5,5)) *#生成服从正态分布的随机数*

*# 生成的随机数组为：*

*#  [[-0.32354535 -0.38402012 -0.72083559  0.81422679 -1.48168735]*

*#  [-0.76959161 -1.26159562 -1.58773179 -0.39938354  0.46393018]*

*#  [-1.01445832  1.94099431  0.41028955 -0.3663865   0.72267596]*

*#  [-0.84199764 -1.13925743 -0.9337132  -0.51566632 -0.83067743]*

*#  [-1.4155873   1.60607239 -0.34619076  0.67296898  0.43626799]]*

print('生成的随机数组为：\n',np.random.randint(2,10,size = [2,5])) #生成给定上下限范围的随机数

# 生成的随机数组为：

#  [[3 3 6 5 4]

#  [9 4 3 4 3]]

*# 使用索引访问一维数组*

arr = np.arange(10)

print('索引结果为：',arr[5])

*# 索引结果为： 5*

*#用范围作为下标获取数组的一个切片，包括arr[3]，不包括arr[5]*

print('索引结果为：',arr[3:5])

*# 索引结果为： [3 4]*

print('索引结果为：',arr[:5])

*# 索引结果为： [0 1 2 3 4]*

print('索引结果为：',arr[-1])

*# 索引结果为： 9*

arr[2:4] = 100,101

print('索引结果为：',arr)

*# 索引结果为： [0 1 100 101 4 5 6 7 8 9]*

print('索引结果为：',arr[1:-1:2])

*# 索引结果为： [1 101 5 7]*

print('索引结果为：',arr[5:1:-2])

*# 索引结果为： [  5 101]*

# 使用索引访问多维数组

arr = np.array([[1,2,3,4,5],[4,5,6,7,8],[7,8,9,10,11]])

print('创建的二维数组为：',arr)

# 创建的二维数组为： [[ 1  2  3  4  5]

#  [ 4  5  6  7  8]

#  [ 7  8  9 10 11]]

print('索引结果为：',arr[0,3:5])

# 索引结果为： [4 5]

print('索引结果为：',arr[1:,2:])

# 索引结果为： [[ 6  7  8]

#  [ 9 10 11]]

print('索引结果为：',arr[:,2])

# 索引结果为： [3 6 9]

#使用整数函数和布尔值索引访问多维数组

*# 创建一个二维数组*

arr = np.array([

 [1, 2, 3, 4, 5],

 [4, 5, 6, 7, 8],

 [7, 8, 9, 10, 11]

])

*# 如若想一下获取，arr[0, 1], arr[1, 2], arr[2, 3]这三个元素组成的数组，便可通过下面这种方式*

print(arr[

 (0, 1, 2),

 (1, 2, 3)

])

*# 获取非连续列的元素也可通过序列完成*

*# 比如索引第1、2行中第0、2、3列的元素*

print(arr[1:, (0, 2, 3)])

*# 下面是通过布尔值索引访问多维数组的方式*

mask = np.array([1, 0, 1], dtype = np.bool)

print(arr[mask, 2]) *# 索引第0行，第2行中第2列的元素*

arr = np.arange(12) *# 创建一个一维数组*

arr1 = arr.reshape(3, 4) *# 将一维数组变成3行4列的二维数组*

arr = np.arange(12).reshape(3, 4) *# 创建一个二维数组*

print(arr)

print(arr.ravel()) *# 将数组横向展平*

arr = np.arange(12).reshape(3, 4) *# 创建一个二维数组*

print(arr.flatten()) *# 默认为横向展平*

print(arr.flattem('F')) *# 加上参数F之后，改为纵向展平*

arr1 = np.arange(12).reshape(3, 4)

arr2 = arr1 \* 3

*# 两个数组进行横向组合*

print(np.hstack((arr1, arr2)))

arr1 = np.arange(12).reshape(3, 4)

arr2 = arr1 \* 3

*# 实现两个数组的纵向组合*

print(np.vstack((arr1, arr2)))

*# 实现横向组合*

np.concatenate((arr1, arr2), axis = 1)

*# 实现纵向组合*

np.concatenate((arr1, arr2), axis = 0)

arr = np.arange(16).reshape(4, 4)

print(np.hsplit(arr, 2)) *# 指定分割为2个大小相同的子数组*

arr = np.arange(16).reshape(4, 4)

*# 横向分割为2个大小相同的子数组*

print(np.split(arr, 2, axis = 1))

*# 纵向分割为2个大小相同的子数组*

print(np.split(arr, 2, axis = 0))

**2.2、掌握NumPy矩阵与通用函数**

#使用mat函数与matrix函数创建矩阵

matr1 = np.mat("1,2,3;4,5,6;7,8,9")

matr2 = np.matrix([[1,2,3],[4,5,6],[7,8,7]])

print(matr1)

*# [[1 2 3]*

*#  [4 5 6]*

*#  [7 8 9]]*

print(matr2)

*# [[1 2 3]*

*#  [4 5 6]*

*#  [7 8 7]]*

*#使用bat函数创建矩阵*

np.bmat("matr1 matr2;matr2 matr1")

*# matrix([[1, 2, 3, 1, 2, 3],*

*#         [4, 5, 6, 4, 5, 6],*

*#         [7, 8, 9, 7, 8, 7],*

*#         [1, 2, 3, 1, 2, 3],*

*#         [4, 5, 6, 4, 5, 6],*

*#         [7, 8, 7, 7, 8, 9]])*

#矩阵与数相乘

arr1 = np.ones([3,3])

np.asmatrix(arr1)

print(arr1)

arr2 = arr1\*3

print(arr2)

#矩阵相加减

arr2[0,1] = 2

arr1[0,1] = 2

print(arr2)

arr3 = arr1\*arr2

print(arr3)

#矩阵与对应元素相乘

np.multiply(arr1,arr2)

matr1 = np.mat("1 2 3;4 5 6;7 8 9")

print('矩阵转置结果为：',matr1.T)

*# 矩阵转置结果为： [[1 4 7]*

*#  [2 5 8]*

*#  [3 6 9]]*

print('矩阵共轭转置结果为：',matr1.H)

*# 矩阵共轭转置结果为： [[1 4 7]*

*#  [2 5 8]*

*#  [3 6 9]]*

print('矩阵逆矩阵结果为：',matr1.I)

*# 矩阵逆矩阵结果为： [[ 3.15251974e+15 -6.30503948e+15  3.15251974e+15]*

*#  [-6.30503948e+15  1.26100790e+16 -6.30503948e+15]*

*#  [ 3.15251974e+15 -6.30503948e+15  3.15251974e+15]]*

print('矩阵的二维数组结果为：',matr1.A)

*# 矩阵的二维数组结果为： [[1 2 3]*

*#  [4 5 6]*

*#  [7 8 9]]*

x = np.array([1,2,3])

y = np.array([4,5,6])

print('数组相加结果为：',x + y)  *#数组相加*

*#数组相加结果为： [5 7 9]*

print('数组相减结果为：',x - y)  *#数组相减*

*#数组相减结果为： [-3 -3 -3]*

print('数组相乘结果为：',x \* y)  *#数组相乘*

*#数组相乘结果为： [ 4 10 18]*

print('数组相除结果为：',x / y)  *#数组相除*

*#数组相除结果为： [ 0.25  0.4   0.5 ]*

print('数组幂运算结果为：',x \*\* y)  *#数组幂运算*

*#数组幂运算结果为： [  1  32 729]*

x = np.array([1,3,5])

y = np.array([2,3,4])

print('数组比较结果为：',x < y)

print('数组比较结果为：',x > y)

print('数组比较结果为：',x == y)

print('数组比较结果为：',x >= y)

print('数组比较结果为：',x <= y)

print('数组比较结果为：',x != y)

*#数组比较结果为： [ True False False]*

*#数组比较结果为： [False False  True]*

*#数组比较结果为： [False  True False]*

*#数组比较结果为： [False  True  True]*

*#数组比较结果为： [ True  True False]*

*#数组比较结果为： [ True False  True]*

print('数组逻辑运算结果为：',np.all(x == y))

print('数组逻辑运算结果为：',np.any(x == y))

*#数组逻辑运算结果为： False*

*#数组逻辑运算结果为： True*

*#一维数组的广播机制*

arr1 = np.array([[0,0,0],[1,1,1],[2,2,2],[3,3,3]])

print('创建的数组1为：\n',arr1)

print('数组1的shape为：',arr1.shape)

arr2 = np.array([1,2,3])

print('创建的数组2为：\n',arr2)

print('数组2的shape为：',arr2.shape)

print('数组相加结果为：\n',arr1 + arr2)

#创建的数组1为：

  [[0 0 0]

  [1 1 1]

  [2 2 2]

  [3 3 3]]

#数组1的shape为： (4, 3)

#创建的数组2为：

  [1 2 3]

#数组2的shape为： (3,)

#数组相加结果为：

  [[1 2 3]

  [2 3 4]

  [3 4 5]

  [4 5 6]]

*#二维数组的广播机制*

arr1 = np.array([[0,0,0],[1,1,1],[2,2,2],[3,3,3]])

print('创建的数组1为：\n',arr1)

print('数组1的shape为：',arr1.shape)

arr2 = np.array([1,2,3,4]).reshape((4,1))

print('创建的数组2为：\n',arr2)

print('数组2的shape为：',arr2.shape)

print('数组相加结果为：\n',arr1 + arr2)

*#创建的数组1为：*

*#[[0 0 0]*

*#[1 1 1]*

*#[2 2 2]*

*#[3 3 3]]*

*#数组1的shape为： (4, 3)*

*#创建的数组2为：*

*#[[1]*

*#[2]*

*#[3]*

*#[4]]*

*#数组2的shape为： (4, 1)*

*#数组相加结果为：*

*#[[1 1 1]*

*#[3 3 3]*

*#[5 5 5]*

*#[7 7 7]]*

**2.3、利用NumPy进行统计分析**

**读/写文件**

save函数是以二进制的格式保存数据。

np.save("…/tmp/save\_arr",arr)

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)

保存好数据后可以用load函数从二进制的文件中读取数据。

np.load("…/tmp/save\_arr.npy")

存储时可以省略扩展名，但读取时不能省略扩展名。

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函数执行的是把文本格式的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**

sort函数是最常用的排序方法。

arr.sort()

sort函数也可以指定一个axis参数，使得它可以沿着指定轴对数据集进行排序。axis=1为沿横轴排序； axis=0为沿纵轴排序。

argsort函数返回值为重新排序值的下标。

arr.argsort()

lexsort函数返回值是按照最后一个传入数据排序的。

np.lexsort((a,b,c))

**去重与重复数据**

通过unique函数可以找出数组中的唯一值并返回已排序的结果。

tile函数主要有2个参数，“A”指定重复的数组，“reps”指定重复的次数。

np.tile(A，reps)

repeat函数主要有三个参数，参数“a”是需要重复的数组元素，参数“repeats”是重复次数，参数“axis”指定沿着哪个轴进行重复，axis= 0表示按行进行元素重复；axis = 1表示按列进行元素重复。

numpy.repeat(a, repeats, axis=None)

这两个函数的主要区别在于，tile函数是对数组进行重复操作，repeat函数是对数组中的每个元素进行重复操作。



*#花萼长度数据统计分析*

import numpy as np

iris\_sepal\_length =np.loadtxt("E:/Desktop/第2章 数据集/iris\_sepal\_length.csv",delimiter=",")*#读取文件数据的方法*

iris\_sepal\_length.sort()*#对文件排序*

print("排序后的花萼长度表为:",iris\_sepal\_length)*#展示排序后的矩阵*

print("去重后的花萼长度表为：",np.unique(iris\_sepal\_length))*#去重*

print("花萼长度表的总和为：",np.sum(iris\_sepal\_length))*#求和*

print("花萼长度表为：",iris\_sepal\_length)*#展示矩阵*

print("花萼长度表的均值为：",np.mean(iris\_sepal\_length))*#平均值*

print("花萼长度表的标准差为：",np.std(iris\_sepal\_length))*#标准差*

print("花萼长度表的方差为：",np.var(iris\_sepal\_length))*#方差*

print("花萼长度表的最小值为：",np.min(iris\_sepal\_length))*#最小值*

print("花萼长度表的最大值为：",np.max(iris\_sepal\_length))*#最大值*

*# 排序后的花萼长度表为: [4.3 4.4 4.4 4.4 4.5 4.6 4.6 4.6 4.6 4.7 4.7 4.8 4.8 4.8 4.8 4.8 4.9 4.9*

*#  4.9 4.9 4.9 4.9 5.  5.  5.  5.  5.  5.  5.  5.  5.  5.  5.1 5.1 5.1 5.1*

*#  5.1 5.1 5.1 5.1 5.1 5.2 5.2 5.2 5.2 5.3 5.4 5.4 5.4 5.4 5.4 5.4 5.5 5.5*

*#  5.5 5.5 5.5 5.5 5.5 5.6 5.6 5.6 5.6 5.6 5.6 5.7 5.7 5.7 5.7 5.7 5.7 5.7*

*#  5.7 5.8 5.8 5.8 5.8 5.8 5.8 5.8 5.9 5.9 5.9 6.  6.  6.  6.  6.  6.  6.1*

*#  6.1 6.1 6.1 6.1 6.1 6.2 6.2 6.2 6.2 6.3 6.3 6.3 6.3 6.3 6.3 6.3 6.3 6.3*

*#  6.4 6.4 6.4 6.4 6.4 6.4 6.4 6.5 6.5 6.5 6.5 6.5 6.6 6.6 6.7 6.7 6.7 6.7*

*#  6.7 6.7 6.7 6.7 6.8 6.8 6.8 6.9 6.9 6.9 6.9 7.  7.1 7.2 7.2 7.2 7.3 7.4*

*#  7.6 7.7 7.7 7.7 7.7 7.9]*

*# 去重后的花萼长度表为： [4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.  5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 6.*

*#  6.1 6.2 6.3 6.4 6.5 6.6 6.7 6.8 6.9 7.  7.1 7.2 7.3 7.4 7.6 7.7 7.9]*

*# 花萼长度表的总和为： 876.5*

*# 花萼长度表为： [4.3 4.4 4.4 4.4 4.5 4.6 4.6 4.6 4.6 4.7 4.7 4.8 4.8 4.8 4.8 4.8 4.9 4.9*

*#  4.9 4.9 4.9 4.9 5.  5.  5.  5.  5.  5.  5.  5.  5.  5.  5.1 5.1 5.1 5.1*

*#  5.1 5.1 5.1 5.1 5.1 5.2 5.2 5.2 5.2 5.3 5.4 5.4 5.4 5.4 5.4 5.4 5.5 5.5*

*#  5.5 5.5 5.5 5.5 5.5 5.6 5.6 5.6 5.6 5.6 5.6 5.7 5.7 5.7 5.7 5.7 5.7 5.7*

*#  5.7 5.8 5.8 5.8 5.8 5.8 5.8 5.8 5.9 5.9 5.9 6.  6.  6.  6.  6.  6.  6.1*

*#  6.1 6.1 6.1 6.1 6.1 6.2 6.2 6.2 6.2 6.3 6.3 6.3 6.3 6.3 6.3 6.3 6.3 6.3*

*#  6.4 6.4 6.4 6.4 6.4 6.4 6.4 6.5 6.5 6.5 6.5 6.5 6.6 6.6 6.7 6.7 6.7 6.7*

*#  6.7 6.7 6.7 6.7 6.8 6.8 6.8 6.9 6.9 6.9 6.9 7.  7.1 7.2 7.2 7.2 7.3 7.4*

*#  7.6 7.7 7.7 7.7 7.7 7.9]*

*# 花萼长度表的均值为： 5.843333333333334*

*# 花萼长度表的标准差为： 0.8253012917851409*

*# 花萼长度表的方差为： 0.6811222222222223*

*# 花萼长度表的最小值为： 4.3*

*# 花萼长度表的最大值为： 7.9*

问题讨论：

1、在使用Markdown标题时，应注意#后面应该加空格再加文字。

2、读取文件路径时应该使用 / 斜杠，如果复制粘贴的话就是反斜杠。

/: 支持linux系统和Windows系统下的路径间隔；

\: 仅支持Windows系统下的路径间隔，在linux系统下该符号作为命令行的强制换行符号。