**实验报告5**

WA2214014 杨跃浙 人工智能2班

**实验内容：**

**实验内容1-****面向对象程序设计**

1、定义一个三维向量类，并定义相应的特殊方法实现两个该类对象之间的加、减运算（要求支持运算+、-），实现该类对象与标量的乘、除（要求支持运算\* 、/ ），以及向量长度的计算（要求使用属性实现）。

2、编写程序，实现自定义类，模拟队列结构。要求实现入队、出队以及修改队列 大小和判断队列是否为空，是否为满的功能，同时要求在入队时队列已满则等待 指定时间、出队时如果队列已空则等待指定时间等辅助功能。

**实验内容2-文件操作**

1、编写程序，保存为demo6.py，运行后生成文件demo6\_new.py，其中的内容

与demo6.py一致，但是在每行的行尾加上了行号。

2、编写程序，要求输入一个文件名，然后输出该文件的MD5值，如果文件不存

在就进行相应的提示。

3、编写程序，实现磁盘垃圾文件清理功能。要求程序运行时，通过命令行参数指 定要清理的文件夹，然后删除该文件夹及其子文件夹中所有扩展名为tmp、log、obj、txt以及大小为0的文件。

4、假设一个学期内所有课程允许多次考试，学生可以随时参加考试，系统自动将每次成绩添加到Excel文件

中（包含姓名、课程、成绩三列）。现期末开始统计所有学生每门课程的最高绩。编写程序，模拟生成记 录若干同学各课程多次成绩的Excel文件，统计所有学生每门课程的最高成绩，并将结果写入新的Excel文件。

5、假设当前文件夹中有Excel文件“电影导演演员.xlsx”，其中内容按照：电影名称、导演、演员三列分别 存放。要求统计所有演员中关系最好的n个演员及共同参演电影数量，其中n可以指定为大于或等于2的整数。这里关系好的定义为共同参演电影数量最多。编写程序，使用python扩展库openpyxl读取Excel文件中的数据，返回一个字典。在字典中，使用演员名字作为键，使用包含该演员参演电影名称的集合作为“值”。读取数据时，跳过表头，对于每一行有效数据，获取每一行的电影名称和演员清单，对该电影的参演演员进行分割得到演员列表，列中的每个研究都参演过该行对应的电影。

6、编写程序，生成一些Excel文件并写入一些测试数据，然后批量修改这些文件的格式。要求：1、每列的表头变为黑体加粗；2、把偶数行所有列的文本设置为宋体、红色，并且使用从红色到蓝色的渐变色对背景进行填充；3、奇书行所有单元格的文本设置为浅蓝色、宋体。

7、准备多个具有相同表头结构的Excel文件，每个文件中第一列具有不同的单元格合并方式。编写程序，合并这些Excel文件，并进行适当的合并。

8、创建测试用的Word文档test.docx，写入测试内容，并根据需要设置红色文本和加粗文本。编写程序查找并输出Word文档test.docx中红色文本和加粗文本。

**实验原理：**

1. **主要代码**

**实验内容1-面向对象程序设计**

def test1():  
 import math  
  
 class Vector:  
 def \_\_init\_\_(self, x=0, y=0, z=0):  
 self.x = x  
 self.y = y  
 self.z = z  
  
 def \_\_add\_\_(self, other):  
 return Vector(self.x + other.x, self.y + other.y, self.z + other.z)  
  
 def \_\_sub\_\_(self, other):  
 return Vector(self.x - other.x, self.y - other.y, self.z - other.z)  
  
 def \_\_mul\_\_(self, scalar):  
 return Vector(self.x \* scalar, self.y \* scalar, self.z \* scalar)  
  
 def \_\_truediv\_\_(self, scalar):  
 return Vector(self.x / scalar, self.y / scalar, self.z / scalar)  
  
 def \_\_abs\_\_(self):  
 return math.sqrt(self.x\*\*2 + self.y\*\*2 + self.z\*\*2)  
   
 v1 = Vector(1, 2, 3)  
 v2 = Vector(4, 5, 6)  
 v3 = v1 + v2  
 print(v3.x, v3.y, v3.z) *# 5 7 9* v4 = Vector(1, 1, 1)  
 v5 = v1 - v4  
 print(v5.x, v5.y, v5.z) *# 0 1 2* v6 = v1 \* 2  
 print(v6.x, v6.y, v6.z) *# 2 4 6* v7 = v1 / 2  
 print(v7.x, v7.y, v7.z) *# 0.5 1.0 1.5* length = abs(v1)  
 print(length) *# 3.7416573867739413*def test2():  
  
 import time  
  
 class Queue:  
 def \_\_init\_\_(self, size=20):  
 self.\_content = []  
 self.\_size = size  
 self.\_current = 0  
  
 def setSize(self, size):  
 if size < self.\_current:  
 for i in range(size, self.\_current)[::-1]:  
 del self.\_content[i]  
 self.\_current = size  
 self.\_size = size  
  
 def put(self, v, timeout=9):  
  
 if self.\_current < self.\_size:  
 self.\_content.append(v)  
 self.\_current = self.\_current + 1  
 else:  
 for i in range(timeout):  
 time.sleep(1)  
 if self.\_current < self.\_size:  
 self.\_content.append(v)  
 self.\_current = self.\_current + 1  
 break  
 else:  
 return '队列已满，超时放弃'  
  
 def get(self, timeout=9):  
  
 if self.\_content:  
 self.\_current = self.\_current - 1  
 return self.\_content.pop(0)  
 else:  
 for i in range(timeout):  
 time.sleep(1)  
 if self.\_content:  
 self.\_current = self.\_current - 1  
 return self.\_content.pop(0)  
 else:  
 return '队列为空，超时放弃'  
 def show(self):  
  
 if self.\_content:  
 print(self.\_content)  
 else:  
 print('The queue is empty')  
 def empty(self):  
 self.\_content = []  
 self.\_current = 0  
  
 def isEmpty(self):  
 return not self.\_content  
  
  
 def isFull(self):  
 return self.\_current == self.\_size  
  
 q = Queue(5)  
 q.put(1)  
 q.put(2)  
 q.put(3)  
 q.show() *# [1, 2, 3]* print(q.get())  
 q.show() *# [2, 3]* print(q.isEmpty()) *# False* print(q.isFull()) *# False* q.put(4)  
 q.put(5)  
 q.put(6)  
 print(q.put(7)) *# 队列已满，超时放弃* q.show() *# [2,3,4,5,6]* q.empty()  
 q.show() *# The queue is empty*if \_\_name\_\_=='\_\_main\_\_':  
 test1()  
 test2()

**实验内容2-文件操作**

def test2():  
 import hashlib  
 import os  
  
 def get\_md5(file\_path):  
 if not os.path.exists(file\_path):  
 return "文件不存在"  
 md5 = hashlib.md5()  
 with open(file\_path, 'rb') as f:  
 for chunk in iter(lambda: f.read(4096), b""):  
 md5.update(chunk)  
 return md5.hexdigest()  
  
 file\_path = input("请输入文件名：")  
 print(get\_md5(file\_path))  
  
def test4():  
 import random  
 import string  
 import pandas as pd  
 from openpyxl import Workbook  
 from openpyxl.utils.dataframe import dataframe\_to\_rows  
  
 def generate\_name():  
 first\_name = ''.join(random.choices(string.ascii\_uppercase, k=2))  
 last\_name = ''.join(random.choices(string.ascii\_lowercase, k=3))  
 return f"{first\_name} {last\_name}"  
  
 def generate\_score():  
 return random.randint(0, 100)  
  
 names=[]  
 for i in range(10):  
 names.append ( generate\_name())  
 data = []  
 courses = ['语文', '数学', '英语','科学']  
 for i in range(200):  
 name=names[random.randint(0,9)]  
 course=courses[random.randint(0,3)]  
 score = generate\_score()  
 l=[]  
 l.append(name)  
 l.append(course)  
 l.append(score)  
 data.append(l)  
  
 df = pd.DataFrame(data, columns=['姓名', '课程', '成绩'])  
 wb = Workbook()  
 ws = wb.active  
 for r in dataframe\_to\_rows(df, index=False, header=True):  
 ws.append(r)  
 wb.save('成绩表.xlsx')  
  
 result = df.groupby(['课程', '姓名'])['成绩'].max().reset\_index()  
 result.columns = ['课程', '姓名', '最高成绩']  
  
 wb\_result = Workbook()  
 ws\_result = wb\_result.active  
 for r in dataframe\_to\_rows(result, index=False, header=True):  
 ws\_result.append(r)  
 for i in range(4):  
 a=str(i\*10+2)  
 b=str(i\*10+11)  
 ws\_result.merge\_cells('A'+a+':A'+b)  
 wb\_result.save('最高成绩统计表.xlsx')  
  
def test5():  
 import openpyxl  
 from collections import defaultdict  
  
 def read\_excel(file\_name):  
 workbook = openpyxl.load\_workbook(file\_name)  
 sheet = workbook.active  
 data = []  
 for row in sheet.iter\_rows(min\_row=2, values\_only=True):  
 movie\_name, director, actors = row  
 actors\_list = actors.split(',')  
 data.append((movie\_name, actors\_list))  
 return data  
  
 def find\_best\_actors(data, n):  
 actor\_movies = defaultdict(set)  
 for movie\_name, actors\_list in data:  
 for actor in actors\_list:  
 actor\_movies[actor].add(movie\_name)  
  
 best\_actors = sorted(actor\_movies.items(), key=lambda x: len(x[1]), reverse=True)[:n]  
 result = {actor: movies for actor, movies in best\_actors}  
 return result  
  
 file\_name = "电影导演演员.xlsx"  
 n=int(input("输入一个大于或等于2的整数n:"))  
 data = read\_excel(file\_name)  
 best\_actors = find\_best\_actors(data, n)  
 for key,value in best\_actors.items():  
 print(f"{key}:{value}")  
  
def test6():  
  
 def generate():  
 import pandas as pd  
 import random  
 import string  
  
 file\_names = ['E:/Python/Project/Test/Test5/test6/test'+str(i) + '.xlsx' for i in range(10)]  
  
 for file\_name in file\_names:  
 data = {'A': [random.randint(1, 100) for \_ in range(10)],  
 'B': [random.choice(string.ascii\_uppercase) for \_ in range(10)],  
 'C': [random.uniform(1, 100) for \_ in range(10)]}  
 df = pd.DataFrame(data)  
 df.to\_excel(file\_name, index=False,)  
  
 print("已生成10个Excel文件并写入测试数据。")  
  
 def manage(i):  
 import openpyxl  
 from openpyxl.styles import Font, PatternFill, GradientFill, Color, colors  
 workbook = openpyxl.load\_workbook('E:/Python/Project/Test/Test5/test6/test'+str(i)+'.xlsx')  
 for sheet in workbook.worksheets:  
 for cell in sheet[1]:  
 cell.font = Font(bold=True, color=colors.BLACK)  
 for row in sheet.iter\_rows(min\_row=2):  
 if row[0].row % 2 == 0:  
 for cell in row:  
 cell.font = Font(name='宋体', color="FF0000")  
 fill = GradientFill(stop=("0000FF", "FF0000"))  
 for cell in row:  
 cell.fill = fill  
 else:  
 for cell in row:  
 cell.font = Font(name='宋体', color="ADD8E6")  
 *#fill = PatternFill(patternType='solid', fgColor=Color('CCE5FF'))  
 #for cell in row:  
 # cell.fill = fill* workbook.save('E:/Python/Project/Test/Test5/test6/output'+str(i)+'.xlsx')  
  
 generate()  
 for i in range(10):  
 manage(i)  
  
def test8():  
 import docx  
  
 doc = docx.Document('test.docx')  
 for para in doc.paragraphs:  
 for run in para.runs:  
 if run.font.color.rgb == docx.shared.RGBColor(255, 0, 0):  
 print(f'Red text: {run.text}')  
  
 if run.bold:  
 print(f'Bold text: {run.text}')  
  
  
if \_\_name\_\_=='\_\_main\_\_':  
 test2()  
 test4()  
 test5()  
 test6()  
 test8()

**题1：demo6.py**

with open("demo6.py", "r") as f:  
 lines = f.readlines()  
  
with open("demo6\_new.py", "w") as f:  
 for index, line in enumerate(lines):  
 f.write(line.strip('\n').ljust(100) + "#"+str(index + 1) + "\n")

题3：delete\_files.py

import os  
import sys  
  
def delete\_files(directory):  
 for foldername, subfolders, filenames in os.walk(directory):  
 for filename in filenames:  
 if filename.endswith('tmp') or filename.endswith('log') or filename.endswith('obj') or filename.endswith('txt'):  
 file\_path = os.path.join(foldername, filename)  
 try:  
 if os.path.getsize(file\_path) == 0:  
 os.remove(file\_path)  
 print(f'Deleted file: {file\_path}')  
 except OSError as e:  
 print(f'Error: {file\_path} : {e.strerror}')  
  
directory = sys.argv[1]  
delete\_files(directory)

**题3：delete\_files.py**

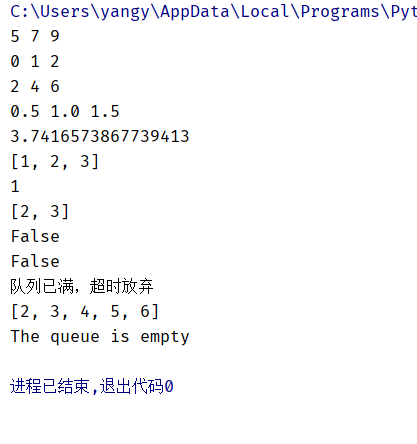
import os  
import sys  
  
def delete\_files(directory):  
 for foldername, subfolders, filenames in os.walk(directory):  
 for filename in filenames:  
 if filename.endswith('tmp') or filename.endswith('log') or filename.endswith('obj') or filename.endswith('txt'):  
 file\_path = os.path.join(foldername, filename)  
 try:  
 if os.path.getsize(file\_path) == 0:  
 os.remove(file\_path)  
 print(f'Deleted file: {file\_path}')  
 except OSError as e:  
 print(f'Error: {file\_path} : {e.strerror}')  
  
directory = sys.argv[1]  
delete\_files(directory)

**题7：work\_on\_excel.py**

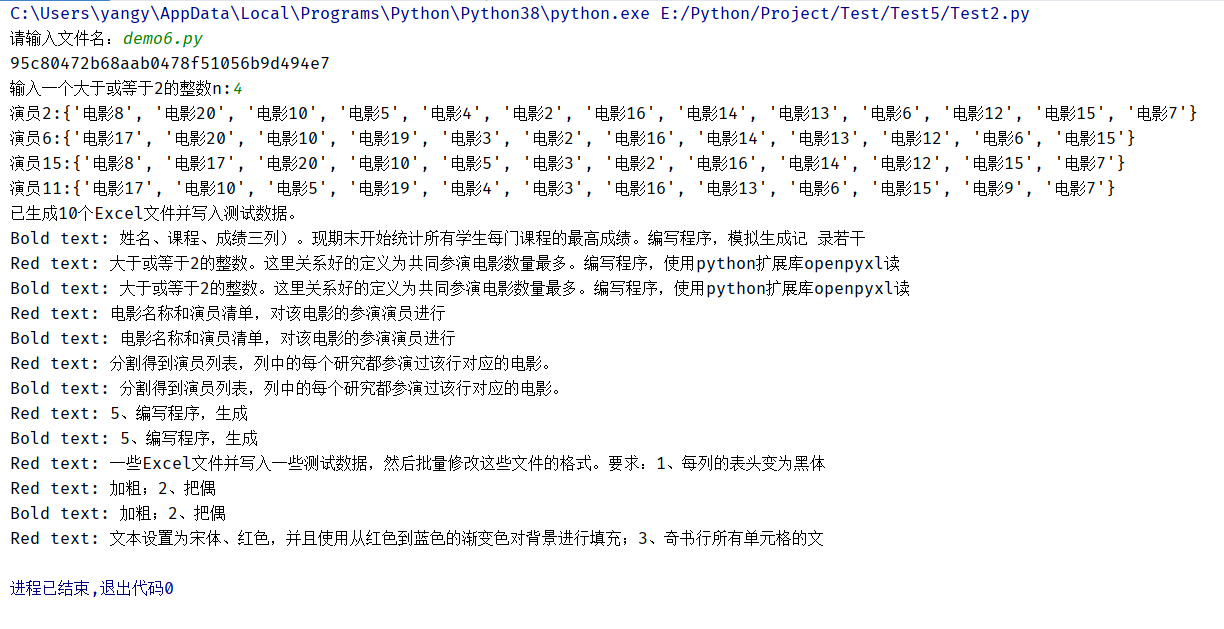
import openpyxl  
import pandas as pd  
import random  
import os  
  
header = ['ID', 'Group', 'Age', 'Gender', 'Address']  
  
id\_values\_all = ['ID1', 'ID2', 'ID3', 'ID4', 'ID5','ID6','ID7']  
  
def generate():  
 save\_path = r"E:\Python\Project\Test\Test5\test7"  
 for i in range(1, 11):  
 workbook = openpyxl.Workbook()  
 sheet = workbook.active  
 for j, col in enumerate(header, start=1):  
 sheet.cell(row=1, column=j).value = col  
 num\_rows\_per\_id=random.randint(1,10)  
 data = []  
  
 id\_values=random.sample(id\_values\_all,5)  
  
 for id\_value in id\_values:  
 for k in range(1, num\_rows\_per\_id + 1):  
 group\_value = random.randint(1, 10)  
 age\_value = random.randint(0, 100)  
 gender\_value = random.choice(['Male', 'Female'])  
 address\_value = f'Address {k}'  
 data.append([id\_value, group\_value, age\_value, gender\_value, address\_value])  
  
 df = pd.DataFrame(data, columns=header)  
 df = df.sort\_values(by=['ID'])  
  
 for l, row in df.iterrows():  
 sheet.cell(row=l + 2, column=1).value = row['ID']  
 sheet.cell(row=l + 2, column=2).value = row['Group']  
 sheet.cell(row=l + 2, column=3).value = row['Age']  
 sheet.cell(row=l + 2, column=4).value = row['Gender']  
 sheet.cell(row=l + 2, column=5).value = row['Address']  
  
 for lis in range(0,5):  
 start\_row = 2+lis\*num\_rows\_per\_id  
 end\_row = 1+(lis+1)\*num\_rows\_per\_id  
 sheet.merge\_cells(start\_row=start\_row, start\_column=1, end\_row=end\_row, end\_column=1)  
  
 file\_name = f'file\_{i}.xlsx'  
 file\_path = f"{save\_path}\\{file\_name}"  
 workbook.save(file\_path)  
  
  
def work():  
 folder\_path = r"E:\Python\Project\Test\Test5\test7"  
 save\_path = r"E:\Python\Project\Test\Test5\test7"  
 file\_list = [file for file in os.listdir(folder\_path) if file.endswith('.xlsx')]  
 merged\_data = pd.DataFrame()  
  
 for file in file\_list:  
 file\_path = os.path.join(folder\_path, file)  
 df = pd.read\_excel(file\_path)  
 value=''  
 for i in range(0,len(df['ID'])):  
 if df['ID'][i] not in id\_values\_all:  
 df['ID'][i]=value  
 else:  
 value=df['ID'][i]  
 *#print(df)* merged\_data = pd.concat([merged\_data, df],ignore\_index=True)  
 *#print(merged\_data)* merged\_data = merged\_data.sort\_values(by=['ID'],ignore\_index=True)  
 start\_row\_1 = 2  
 end\_row\_1=0  
 *#print(len(merged\_data['ID']))* workbook = openpyxl.Workbook()  
 sheet = workbook.active  
 for j, col in enumerate(header, start=1):  
 sheet.cell(row=1, column=j).value = col  
 for l, row in merged\_data.iterrows():  
 sheet.cell(row=l + 2, column=1).value = row['ID']  
 sheet.cell(row=l + 2, column=2).value = row['Group']  
 sheet.cell(row=l + 2, column=3).value = row['Age']  
 sheet.cell(row=l + 2, column=4).value = row['Gender']  
 sheet.cell(row=l + 2, column=5).value = row['Address']  
 for i in range(1, len(merged\_data['ID'])):  
 if not(merged\_data['ID'][i]==merged\_data['ID'][i-1]):  
 end\_row\_1=i+1  
 sheet.merge\_cells(start\_row=start\_row\_1, start\_column=1, end\_row=end\_row\_1, end\_column=1)  
 start\_row\_1=i+2  
 *#print(start\_row\_1)  
 #print(i)* sheet.merge\_cells(start\_row=start\_row\_1, start\_column=1, end\_row=i+2, end\_column=1)  
 file\_name = 'merged\_file.xlsx'  
 file\_path\_1 = f"{save\_path}\\{file\_name}"  
 workbook.save(file\_path\_1)  
  
if \_\_name\_\_=='\_\_main\_\_':  
 generate()  
 work()

1. **运行结果**

**实验内容1-面向对象程序设计**

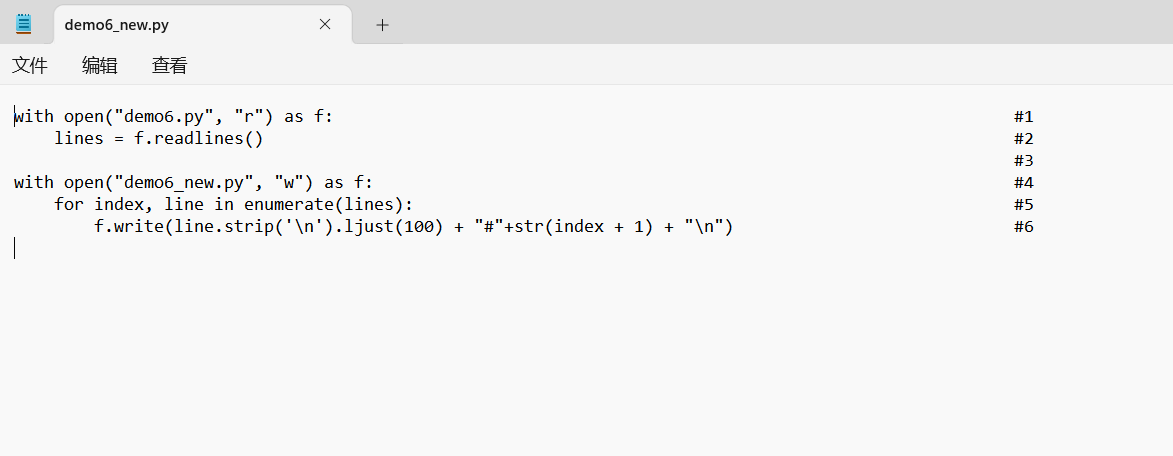


**实验内容2-文件操作**



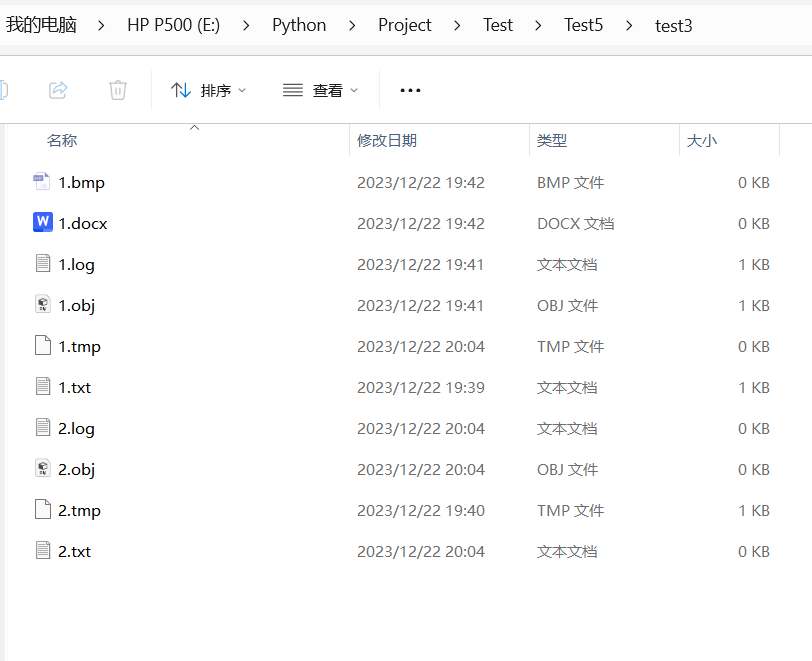
**题1：**

**demo6\_new.py 文件**



**题3：**

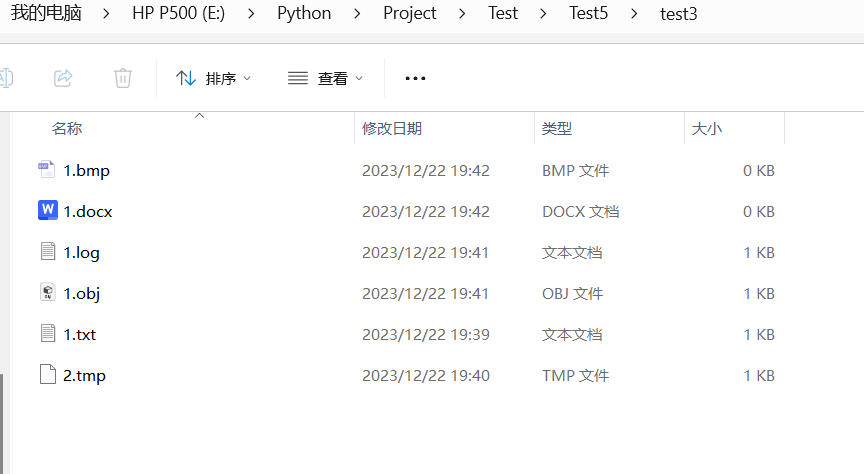
**原文件夹：**



**调用cmd窗口（终端）：**

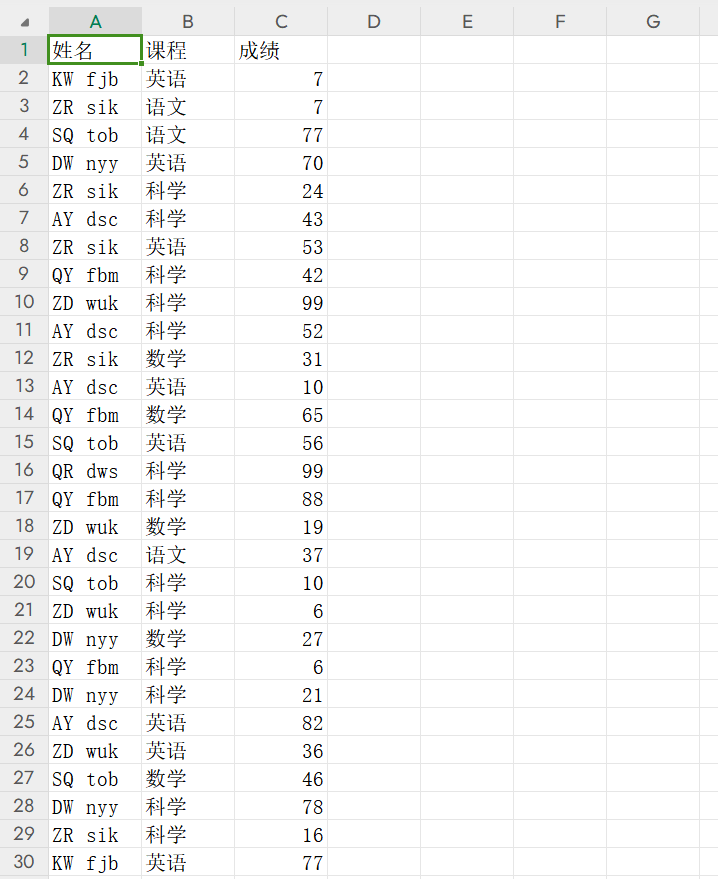


**运行之后：**

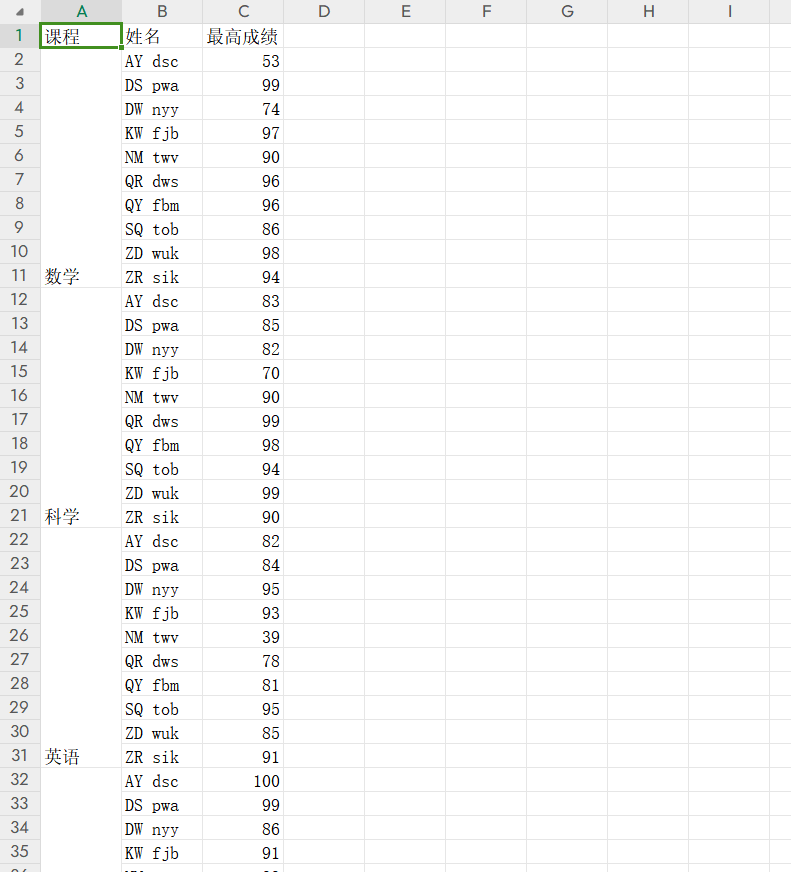


**题4：**

**成绩表.xlsx**

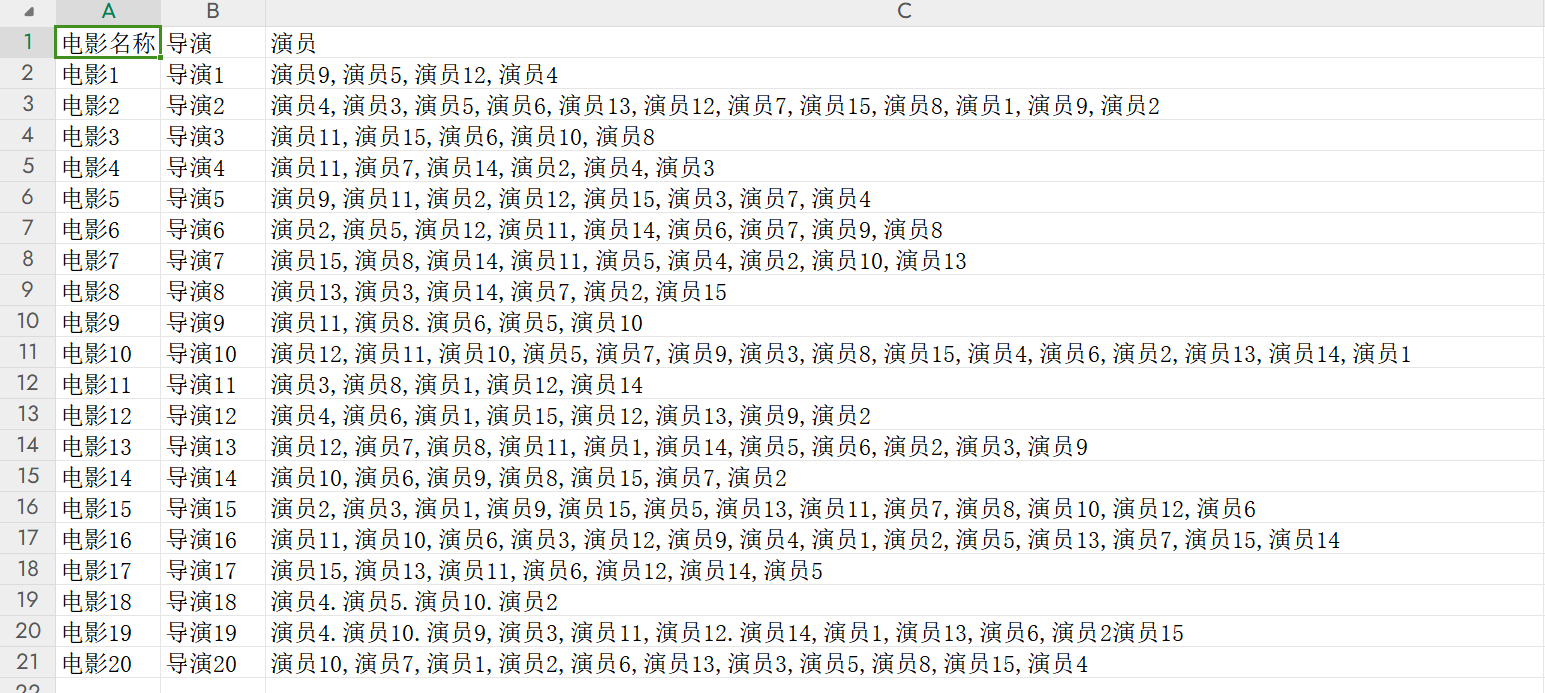


**最高成绩统计表.xlsx**



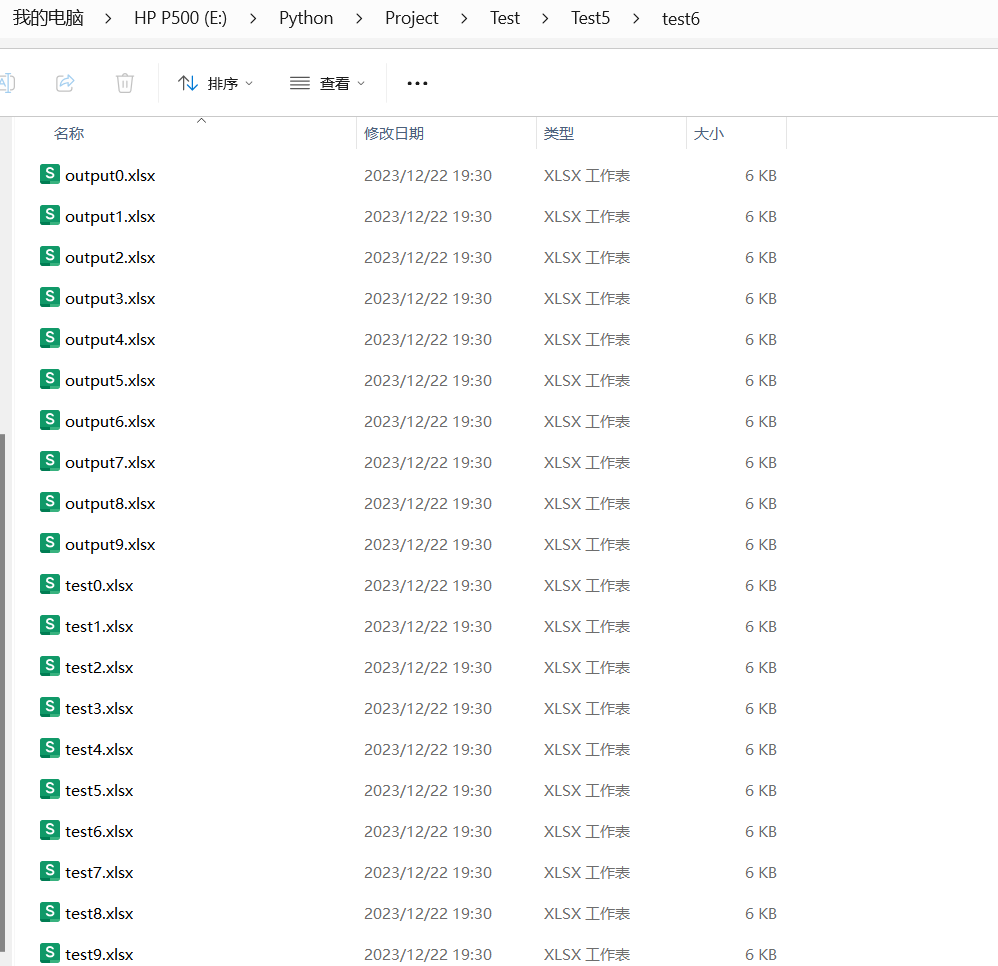
**题5：**

**电影导演演员.xlsx**

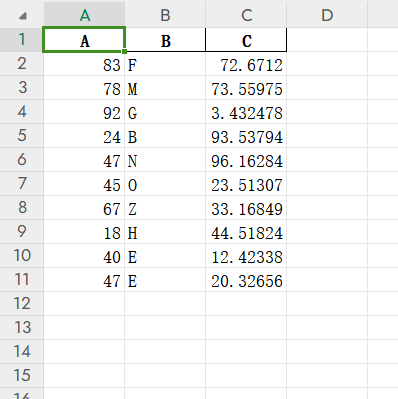


**题6：**

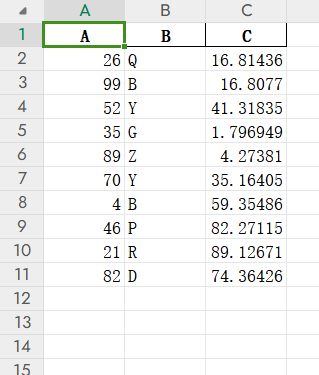
**文件夹中有十个测试表格，十个输出表格**



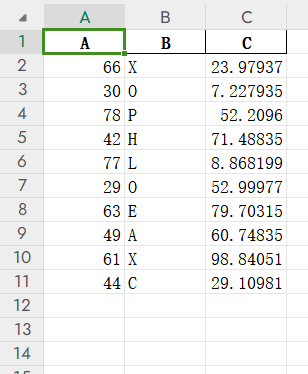
**test0.xlsx**



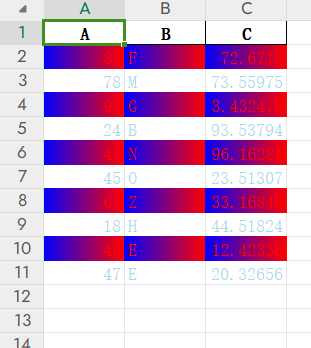
**test7.xlsx**



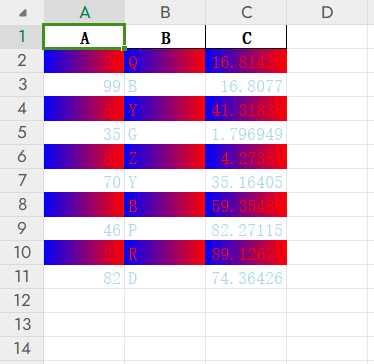
**test9.xlsx**



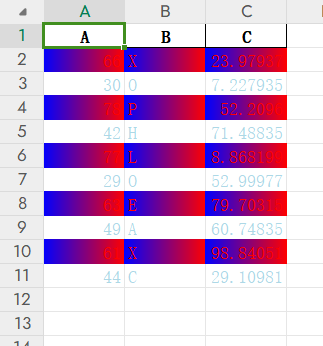
**output0.xlsx**



**output7.xlsx**

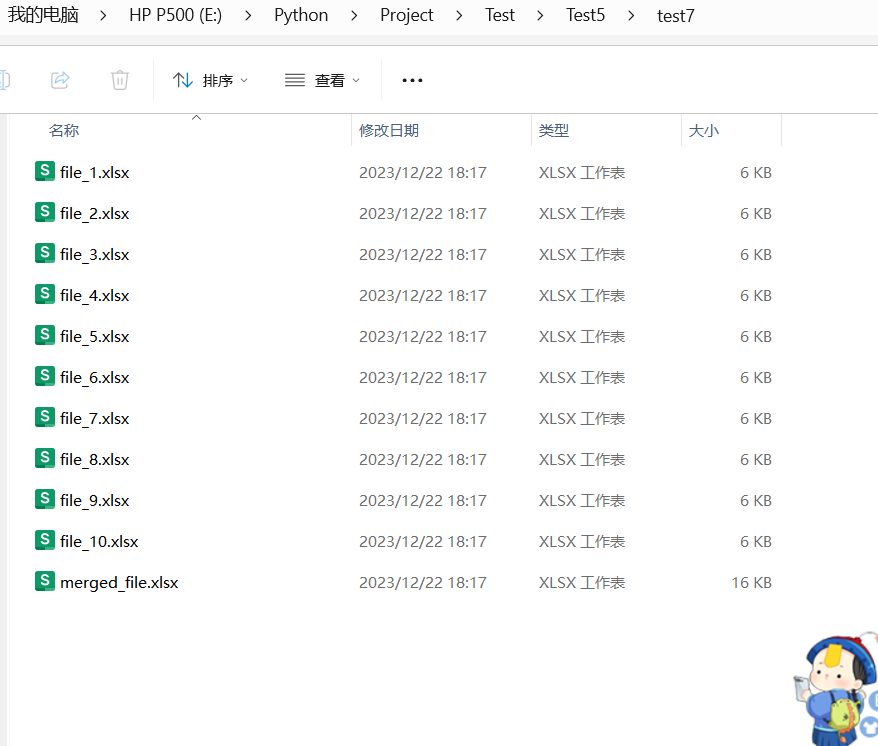


**output9.xlsx**

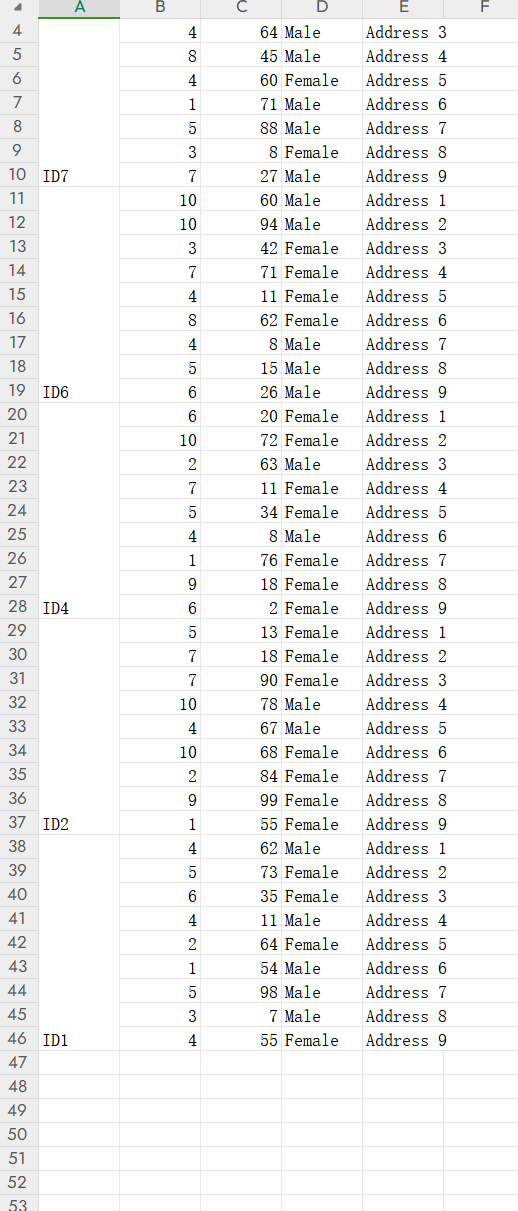


**题7：**

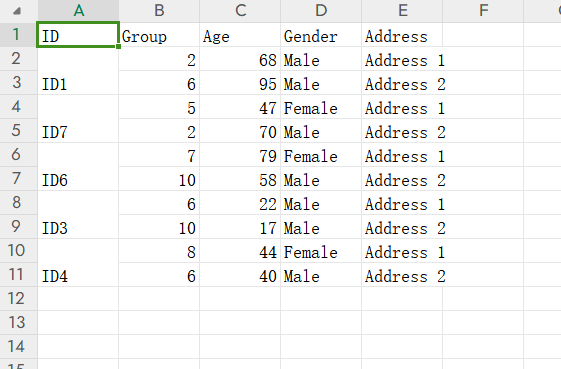
**文件夹中十个原始文件和一个输出文件**



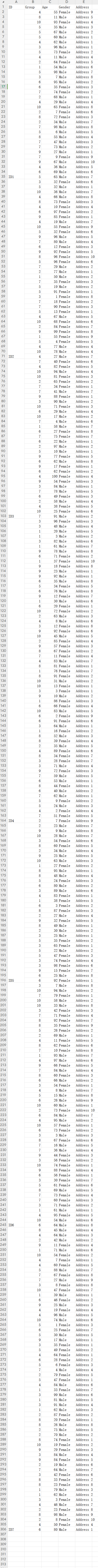
**file\_1.xlsx**



**file\_8.xlsx**

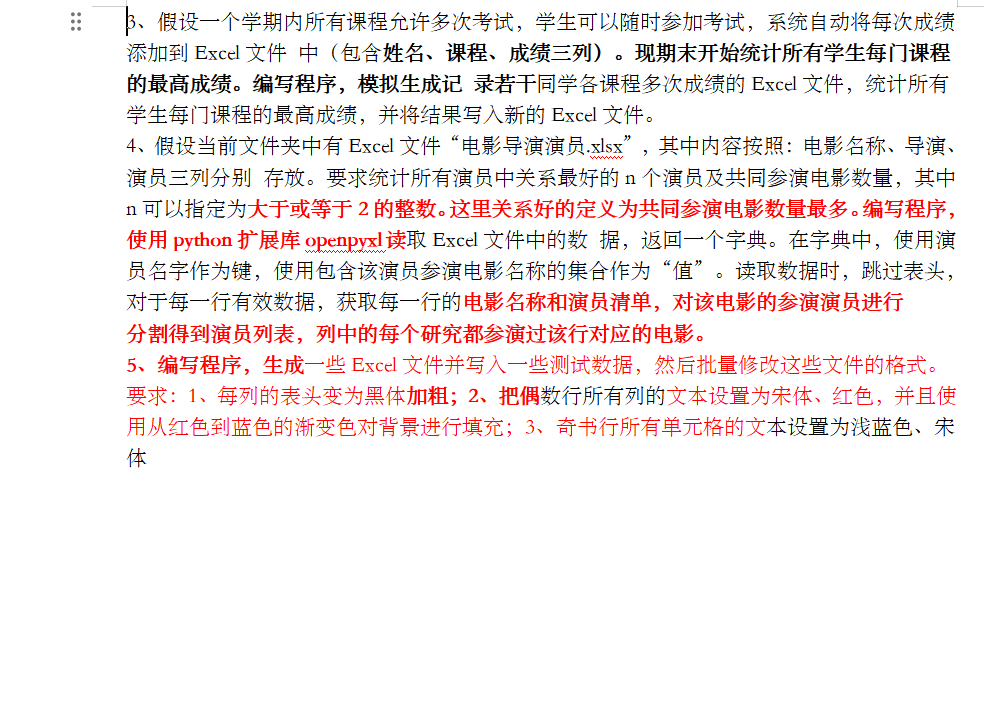


**merged\_file.xlsx**



**题8：**

**test.docx**



**小结与讨论：**

**实验内容1-面向对象程序设计**

题1：用一个三维向量列类型，add表示+，sub表示-，mul表示\*，truediv表示/，abs表示向量的长度，通过重构这五种计算实现三维向量的功能。

定义了一个名为Vector的类，表示三维向量。该类具有以下功能：

构造函数\_\_init\_\_：初始化向量的x、y、z分量，默认为0。

运算符重载：

\_\_add\_\_：实现向量的加法，将两个向量的对应分量相加。

\_\_sub\_\_：实现向量的减法，将两个向量的对应分量相减。

\_\_mul\_\_：实现向量的标量乘法，将向量的每个分量与标量相乘。

\_\_truediv\_\_：实现向量的标量除法，将向量的每个分量与标量相除。

\_\_abs\_\_：计算向量的模（长度），使用欧几里得范数计算。

示例代码：

创建两个向量v1和v2，分别为(1, 2, 3)和(4, 5, 6)。

使用+运算符将v1和v2相加，得到向量v3，并打印其分量。

使用-运算符将v1和另一个向量(1, 1, 1)相减，得到向量v5，并打印其分量。

使用\*运算符将v1乘以2，得到向量v6，并打印其分量。

使用/运算符将v1除以2，得到向量v7，并打印其分量。

使用abs函数计算向量v1的长度，并打印结果。

题2：代码定义了一个名为Queue的队列类，用于实现队列的基本操作。该类具有以下功能：

构造函数\_\_init\_\_：初始化队列的内容、大小和当前元素个数。默认大小为20。

setSize方法：设置队列的大小，如果新的大小小于当前元素个数，则删除多余的元素。

put方法：向队列中添加元素。如果队列未满，则直接添加；否则，会等待一定时间，直到队列有空间为止。如果超时仍未有空间，则返回提示信息。

get方法：从队列中获取元素。如果队列非空，则返回队列的第一个元素并将其从队列中删除；否则，会等待一定时间，直到队列非空为止。如果超时仍未有元素，则返回提示信息。

show方法：打印当前队列的内容。

empty方法：清空队列。

isEmpty方法：判断队列是否为空。

isFull方法：判断队列是否已满。

在运行过程中，无法实现在等待过程中继续读入并处理指令，如果要实现在等待过程中继续读入指令，可能需要多线程实现，比较复杂。

**实验内容2-文件操作**

题1中用with open方法打开文件，在加行号时用了ljust方法实现右对齐，在行号前加上注释符‘#’，确保了程序仍可运行。

题2中定义了一个名为get\_md5的函数，用于计算给定文件的MD5哈希值。

导入hashlib和os模块，分别用于计算哈希值和操作文件。定义get\_md5函数，接受一个文件路径作为参数。首先判断给定的文件路径是否存在，如果不存在，则返回字符串"文件不存在"。创建一个md5对象，用于计算MD5哈希值。使用open函数打开文件，以二进制模式读取文件内容。使用iter函数和lambda表达式来迭代读取文件内容，每次读取4096字节（4KB）。对每次读取的内容调用md5.update方法，更新MD5哈希值。循环结束后，使用md5.hexdigest方法获取最终的MD5哈希值，并将其作为函数的返回值。在主程序中，通过input函数获取用户输入的文件名，并将其赋值给file\_path变量。调用get\_md5函数，传入file\_path作为参数，并打印返回的MD5哈希值。

题3中利用sys实现在终端输入的响应，注意是sys.argv[1]

题4中先按要求随机生成一个有“姓名”，“课程”，“成绩”，三个字段的成绩表，然后按照要求统计每个人每门课的最高成绩，并把相同的课程单元格合并起来。

题5中先通过read\_excel函数读取表格数据，之后通过调用find\_best\_actors函数用集合的方式找到n个符合要求的演员。

题6中先用程序实现批量产生写有随机数据的Excel表格，即generate函数，然后通过manage函数批量按要求处理这些表格，generate产生的文件为test0.xlsx-test9.xlsx,manage文件产生的是output0.xlsx-output9.xlsx

题7中通过调用generate函数批量化产生符合题目要求的表格，其中他们具有相同的表头，“ID”, “Group”, “Age”, “Gender”, “Address”这五个字段，其中ID只有“ID1”,“ID2”,“ID3”,“ID4”,“ID5”,这五个字段值，每张表格的ID列都有不同的合并。之后调用work函数，对生成的表格进行合并，在实验过程中发现pandas对于合并单元格的处理不满足预期，读取ID合并单元格的数据出现了Nan（Not a number）的情况，所以就考虑先拆分单元格，再对表格进行合并处理，同时去除原本列索引，再进行排序和单元格的合并来实现想要达到的效果。

题8中调用docx外部库，按要求匹配test.docx文件中红色字和加粗的字，并分别显示出来。