

## 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()

```

## 2. 运行结果

### 实验内容 1-面向对象程序设计

```
C:\Users\yangy\AppData\Local\Programs\Pyt
```

```
5 7 9
0 1 2
2 4 6
0.5 1.0 1.5
3.7416573867739413
[1, 2, 3]
1
[2, 3]
False
False
队列已满, 超时放弃
[2, 3, 4, 5, 6]
The queue is empty
```

```
进程已结束,退出代码0
```

### 实验内容 2-文件操作

```
C:\Users\yangy\AppData\Local\Programs\Python\Python38\python.exe E:/Python/Project/Test/Test5/Test2.py
```

```
请输入文件名: demo6.py
```

```
95c80472b68aab0478f51056b9d494e7
```

```
输入一个大于或等于2的整数n: 4
```

```
演员2: {'电影8', '电影20', '电影10', '电影5', '电影4', '电影2', '电影16', '电影14', '电影13', '电影6', '电影12', '电影15', '电影7'}
```

```
演员6: {'电影17', '电影20', '电影10', '电影19', '电影3', '电影2', '电影16', '电影14', '电影13', '电影12', '电影6', '电影15'}
```

```
演员15: {'电影8', '电影17', '电影20', '电影10', '电影5', '电影3', '电影2', '电影16', '电影14', '电影12', '电影15', '电影7'}
```

```
演员11: {'电影17', '电影10', '电影5', '电影19', '电影4', '电影3', '电影16', '电影13', '电影6', '电影15', '电影9', '电影7'}
```

```
已生成10个Excel文件并写入测试数据。
```

```
Bold text: 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成绩。编写程序, 模拟生成记录若干
```

```
Red text: 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
```

```
Bold text: 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
```

```
Red text: 电影名称和演员清单, 对该电影的参演演员进行
```

```
Bold text: 电影名称和演员清单, 对该电影的参演演员进行
```

```
Red text: 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。
```

```
Bold text: 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。
```

```
Red text: 5、编写程序, 生成
```

```
Bold text: 5、编写程序, 生成
```

```
Red text: 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
```

```
Red text: 加粗; 2、把偶
```

```
Bold text: 加粗; 2、把偶
```

```
Red text: 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
```

```
进程已结束,退出代码0
```

### 题 1:

demo6\_new.py 文件

```
demo6_new.py
文件 编辑 查看

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:

原文件夹:

我的电脑 > HP P500 (E:) > Python > Project > Test > Test5 > test3				
排序 查看 ...				
名称	修改日期	类型	大小	
1.bmp	2023/12/22 19:42	BMP 文件	0 KB	
1.docx	2023/12/22 19:42	DOCX 文档	0 KB	
1.log	2023/12/22 19:41	文本文档	1 KB	
1.obj	2023/12/22 19:41	OBJ 文件	1 KB	
1.tmp	2023/12/22 20:04	TMP 文件	0 KB	
1.txt	2023/12/22 19:39	文本文档	1 KB	
2.log	2023/12/22 20:04	文本文档	0 KB	
2.obj	2023/12/22 20:04	OBJ 文件	0 KB	
2.tmp	2023/12/22 19:40	TMP 文件	1 KB	
2.txt	2023/12/22 20:04	文本文档	0 KB	

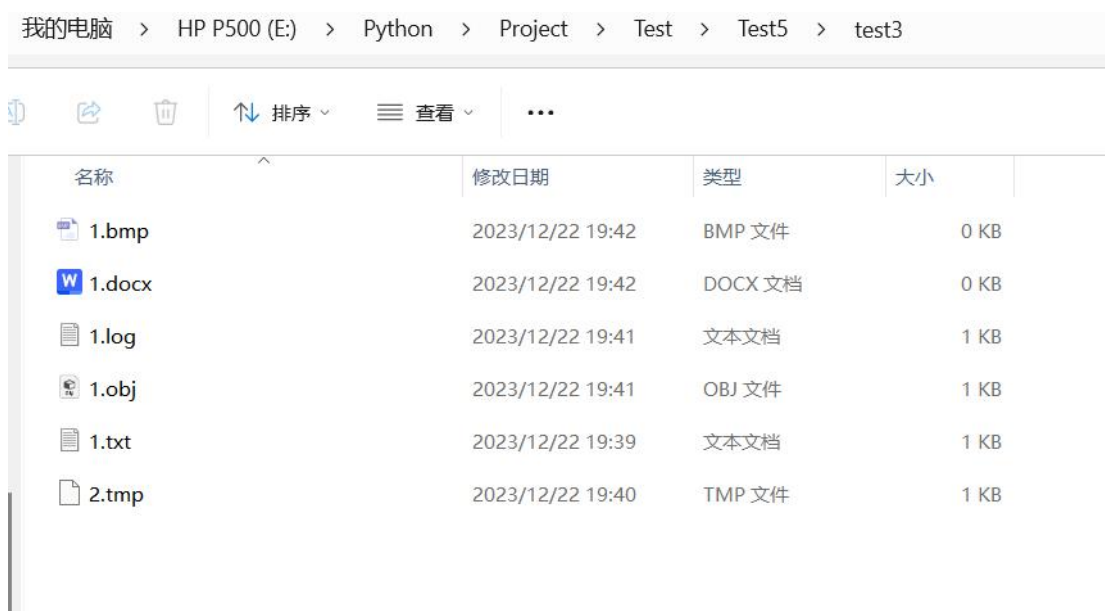
调用 cmd 窗口（终端）:

```
终端: 本地 × + ▾

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Python\Project\Test> cd Test5
PS E:\Python\Project\Test\Test5> python delete_files.py E:\Python\Project
Deleted file: E:\Python\Project\Test\Test5\test3\2.txt
Deleted file: E:\Python\Project\Test\Test5\test3\2.obj
Deleted file: E:\Python\Project\Test\Test5\test3\2.log
Deleted file: E:\Python\Project\Test\Test5\test3\1.tmp
PS E:\Python\Project\Test\Test5> █
```

运行之后：



题 4：

成绩表.xlsx

	A	B	C	D	E	F	G
1	姓名	课程	成绩				
2	KW fjb	英语	7				
3	ZR sik	语文	7				
4	SQ tob	语文	77				
5	DW nyy	英语	70				
6	ZR sik	科学	24				
7	AY dsc	科学	43				
8	ZR sik	英语	53				
9	QY fbm	科学	42				
10	ZD wuk	科学	99				
11	AY dsc	科学	52				
12	ZR sik	数学	31				
13	AY dsc	英语	10				
14	QY fbm	数学	65				
15	SQ tob	英语	56				
16	QR dws	科学	99				
17	QY fbm	科学	88				
18	ZD wuk	数学	19				
19	AY dsc	语文	37				
20	SQ tob	科学	10				
21	ZD wuk	科学	6				
22	DW nyy	数学	27				
23	QY fbm	科学	6				
24	DW nyy	科学	21				
25	AY dsc	英语	82				
26	ZD wuk	英语	36				
27	SQ tob	数学	46				
28	DW nyy	科学	78				
29	ZR sik	科学	16				
30	KW fjb	英语	77				

最高成绩统计表.xlsx

	A	B	C	D	E	F	G	H	I	
1	课程	姓名	最高成绩							
2		AY dsc	53							
3		DS pwa	99							
4		DW nyy	74							
5		KW fjb	97							
6		NM twv	90							
7		QR dws	96							
8		QY fbm	96							
9		SQ tob	86							
10		ZD wuk	98							
11	数学	ZR sik	94							
12		AY dsc	83							
13		DS pwa	85							
14		DW nyy	82							
15		KW fjb	70							
16		NM twv	90							
17		QR dws	99							
18		QY fbm	98							
19		SQ tob	94							
20		ZD wuk	99							
21	科学	ZR sik	90							
22		AY dsc	82							
23		DS pwa	84							
24		DW nyy	95							
25		KW fjb	93							
26		NM twv	39							
27		QR dws	78							
28		QY fbm	81							
29		SQ tob	95							
30		ZD wuk	85							
31	英语	ZR sik	91							
32		AY dsc	100							
33		DS pwa	99							
34		DW nyy	86							
35		KW fjb	91							

题 5:

电影导演演员.xlsx



	A	B	C
1	电影名称	导演	演员
2	电影1	导演1	演员9, 演员5, 演员12, 演员4
3	电影2	导演2	演员4, 演员3, 演员5, 演员6, 演员13, 演员12, 演员7, 演员15, 演员8, 演员1, 演员9, 演员2
4	电影3	导演3	演员11, 演员15, 演员6, 演员10, 演员8
5	电影4	导演4	演员11, 演员7, 演员14, 演员2, 演员4, 演员3
6	电影5	导演5	演员9, 演员11, 演员2, 演员12, 演员15, 演员3, 演员7, 演员4
7	电影6	导演6	演员2, 演员5, 演员12, 演员11, 演员14, 演员6, 演员7, 演员9, 演员8
8	电影7	导演7	演员15, 演员8, 演员14, 演员11, 演员5, 演员4, 演员2, 演员10, 演员13
9	电影8	导演8	演员13, 演员3, 演员14, 演员7, 演员2, 演员15
10	电影9	导演9	演员11, 演员8, 演员6, 演员5, 演员10
11	电影10	导演10	演员12, 演员11, 演员10, 演员5, 演员7, 演员9, 演员3, 演员8, 演员15, 演员4, 演员6, 演员2, 演员13, 演员14, 演员1
12	电影11	导演11	演员3, 演员8, 演员1, 演员12, 演员14
13	电影12	导演12	演员4, 演员6, 演员1, 演员15, 演员12, 演员13, 演员9, 演员2
14	电影13	导演13	演员12, 演员7, 演员8, 演员11, 演员1, 演员14, 演员5, 演员6, 演员2, 演员3, 演员9
15	电影14	导演14	演员10, 演员6, 演员9, 演员8, 演员15, 演员7, 演员2
16	电影15	导演15	演员2, 演员3, 演员1, 演员9, 演员15, 演员5, 演员13, 演员11, 演员7, 演员8, 演员10, 演员12, 演员6
17	电影16	导演16	演员11, 演员10, 演员6, 演员3, 演员12, 演员9, 演员4, 演员1, 演员2, 演员5, 演员13, 演员7, 演员15, 演员14
18	电影17	导演17	演员15, 演员13, 演员11, 演员6, 演员12, 演员14, 演员5
19	电影18	导演18	演员4, 演员5, 演员10, 演员2
20	电影19	导演19	演员4, 演员10, 演员9, 演员3, 演员11, 演员12, 演员14, 演员1, 演员13, 演员6, 演员2, 演员15
21	电影20	导演20	演员10, 演员7, 演员1, 演员2, 演员6, 演员13, 演员3, 演员5, 演员8, 演员15, 演员4

题 6:

文件夹中有十个测试表格， 十个输出表格

我的电脑	>	HP P500 (E:)	>	Python	>	Project	>	Test	>	Test5	>	test6
				排序		查看						
名称	修改日期	类型	大小									
output0.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
output1.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
output2.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
output3.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
output4.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
output5.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
output6.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
output7.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
output8.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
output9.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
test0.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
test1.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
test2.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
test3.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
test4.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
test5.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
test6.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
test7.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
test8.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									
test9.xlsx	2023/12/22 19:30	XLSX 工作表	6 KB									

test0.xlsx



	A	B	C	D
1	A	B	C	
2	83 F		72.6712	
3	78 M		73.55975	
4	92 G		3.432478	
5	24 B		93.53794	
6	47 N		96.16284	
7	45 O		23.51307	
8	67 Z		33.16849	
9	18 H		44.51824	
10	40 E		12.42338	
11	47 E		20.32656	
12				
13				
14				
15				

test7.xlsx

	A	B	C
1	A	B	C
2	26 Q		16.81436
3	99 B		16.8077
4	52 Y		41.31835
5	35 G		1.796949
6	89 Z		4.27381
7	70 Y		35.16405
8	4 B		59.35486
9	46 P		82.27115
10	21 R		89.12671
11	82 D		74.36426
12			
13			
14			
15			

test9.xlsx

	A	B	C
1	A	B	C
2	66 X		23.97937
3	30 O		7.227935
4	78 P		52.2096
5	42 H		71.48835
6	77 L		8.868199
7	29 O		52.99977
8	63 E		79.70315
9	49 A		60.74835
10	61 X		98.84051
11	44 C		29.10981
12			
13			
14			

output0.xlsx

	A	B	C
1	A	B	C
2	52 F		72.673
3	78 M		73.55975
4	64 G		3.43247
5	24 B		93.53794
6	45 N		96.1626
7	45 O		23.51307
8	64 Z		33.1694
9	18 H		44.51824
10	45 E		12.4233
11	47 E		20.32656
12			
13			
14			

output7.xlsx












	A	B	C	D
1	A	B	C	
2	54 Q		16.8143	
3	99 B		16.8077	
4	54 Y		41.3183	
5	35 G		1.796949	
6	84 Z		4.2738	
7	70 Y		35.16405	
8	54 B		59.3548	
9	46 P		82.27115	
10	54 R		89.1967	
11	82 D		74.36426	
12				
13				
14				

output9.xlsx

	A	B	C	
1	A	B	C	
2	64 X		23.9793	
3	30 O		7.227935	
4	74 P		52.2007	
5	42 H		71.48835	
6	74 L		8.86819	
7	29 O		52.99977	
8	64 E		79.7033	
9	49 A		60.74835	
10	64 X		93.8407	
11	44 C		29.10981	
12				
13				
14				

题 7:

文件夹中十个原始文件和一个输出文件

我的电脑 > HP P500 (E:) > Python > Project > Test > Test5 > test7				
<div><div></div><div></div><div></div><div>↑↓ 排序</div><div>≡ 查看</div><div>...</div></div>				
名称	修改日期	类型	大小	
 file_1.xlsx	2023/12/22 18:17	XLSX 工作表	6 KB	
 file_2.xlsx	2023/12/22 18:17	XLSX 工作表	6 KB	
 file_3.xlsx	2023/12/22 18:17	XLSX 工作表	6 KB	
 file_4.xlsx	2023/12/22 18:17	XLSX 工作表	6 KB	
 file_5.xlsx	2023/12/22 18:17	XLSX 工作表	6 KB	
 file_6.xlsx	2023/12/22 18:17	XLSX 工作表	6 KB	
 file_7.xlsx	2023/12/22 18:17	XLSX 工作表	6 KB	
 file_8.xlsx	2023/12/22 18:17	XLSX 工作表	6 KB	
 file_9.xlsx	2023/12/22 18:17	XLSX 工作表	6 KB	
 file_10.xlsx	2023/12/22 18:17	XLSX 工作表	6 KB	
 merged_file.xlsx	2023/12/22 18:17	XLSX 工作表	16 KB	



file\_1.xlsx

	A	B	C	D	E	F
4	ID7	4	64 Male	Address 3		
5		8	45 Male	Address 4		
6		4	60 Female	Address 5		
7		1	71 Male	Address 6		
8		5	88 Male	Address 7		
9		3	8 Female	Address 8		
10		7	27 Male	Address 9		
11	ID6	10	60 Male	Address 1		
12		10	94 Male	Address 2		
13		3	42 Female	Address 3		
14		7	71 Female	Address 4		
15		4	11 Female	Address 5		
16		8	62 Female	Address 6		
17		4	8 Male	Address 7		
18	ID4	5	15 Male	Address 8		
19		6	26 Male	Address 9		
20		6	20 Female	Address 1		
21		10	72 Female	Address 2		
22		2	63 Male	Address 3		
23		7	11 Female	Address 4		
24		5	34 Female	Address 5		
25	ID2	4	8 Male	Address 6		
26		1	76 Female	Address 7		
27		9	18 Female	Address 8		
28		6	2 Female	Address 9		
29		5	13 Female	Address 1		
30		7	18 Female	Address 2		
31		7	90 Female	Address 3		
32	ID1	10	78 Male	Address 4		
33		4	67 Male	Address 5		
34		10	68 Female	Address 6		
35		2	84 Female	Address 7		
36		9	99 Female	Address 8		
37		1	55 Female	Address 9		
38		4	62 Male	Address 1		
39		5	73 Female	Address 2		
40		6	35 Female	Address 3		
41		4	11 Male	Address 4		
42		2	64 Female	Address 5		
43		1	54 Male	Address 6		
44		5	98 Male	Address 7		
45		3	7 Male	Address 8		
46		4	55 Female	Address 9		
47						
48						
49						
50						
51						
52						
53						

file\_8.xlsx

	A	B	C	D	E	F	G
1	ID	Group	Age	Gender	Address		
2		2	68	Male	Address 1		
3	ID1	6	95	Male	Address 2		
4		5	47	Female	Address 1		
5	ID7	2	70	Male	Address 2		
6		7	79	Female	Address 1		
7	ID6	10	58	Male	Address 2		
8		6	22	Male	Address 1		
9	ID3	10	17	Male	Address 2		
10		8	44	Female	Address 1		
11	ID4	6	40	Male	Address 2		
12							
13							
14							
15							

merged\_file.xlsx

	A	B	C	D	E
1	ID	Group	Age	Gender	Address
2		4	55	Female	Address 9
3		8	11	Male	Address 5
4		10	93	Female	Address 4
5		5	35	Male	Address 3
6		5	67	Male	Address 2
7		5	60	Male	Address 1
8		4	62	Male	Address 1
9		3	96	Male	Address 1
10		5	73	Female	Address 2
11		4	11	Male	Address 4
12		2	64	Female	Address 5
13		1	54	Male	Address 6
14		5	98	Male	Address 7
15		3	7	Male	Address 8
16		6	95	Male	Address 2
17		6	35	Female	Address 3
18		7	74	Female	Address 2
19		2	68	Male	Address 1
20		4	29	Male	Address 4
21		10	65	Female	Address 8
22		2	7	Female	Address 6
23		8	72	Female	Address 3
24		2	34	Male	Address 7
25		7	98	Male	Address 8
26		5	6	Male	Address 5
27		8	55	Male	Address 4
28		7	47	Male	Address 3
29		7	73	Male	Address 2
30		9	27	Male	Address 1
31		7	9	Female	Address 9
32		9	67	Male	Address 10
33		4	36	Male	Address 5
34		6	69	Male	Address 6
35	ID1	5	65	Male	Address 7
36		3	32	Female	Address 6
37		5	32	Male	Address 1
38		10	36	Male	Address 7
39		4	66	Male	Address 8
40		8	73	Female	Address 3
41		4	10	Female	Address 4
42		6	97	Female	Address 2
43		9	85	Female	Address 1
44		4	19	Male	Address 7
45		10	55	Female	Address 9
46		5	32	Female	Address 6
47		9	96	Male	Address 5
48		7	80	Male	Address 4
49		6	12	Female	Address 3
50		9	92	Female	Address 5
51		8	96	Female	Address 10
52		5	96	Female	Address 6
53		7	29	Male	Address 2
54		2	77	Male	Address 1
55		1	50	Male	Address 2
56		7	35	Female	Address 3
57		5	19	Male	Address 2
58		5	91	Female	Address 1
59		3	1	Female	Address 8
60		7	18	Female	Address 2
61		7	90	Female	Address 3
62		5	13	Female	Address 1
63		4	67	Male	Address 5
64		10	68	Female	Address 6
65		2	84	Female	Address 7
66		9	99	Female	Address 8
67		1	55	Female	Address 9
68		5	4	Female	Address 5
69		4	77	Male	Address 4
70		10	78	Male	Address 4
71	ID2	4	27	Male	Address 7
72		7	40	Female	Address 6
73		4	82	Female	Address 5
74		10	94	Male	Address 4
75		10	81	Female	Address 3
76		2	65	Female	Address 2
77		7	34	Female	Address 1
78		7	41	Male	Address 7
79		9	88	Female	Address 1
80		3	90	Male	Address 2
81		10	71	Female	Address 3
82		6	29	Male	Address 4
83		10	17	Male	Address 2
84		7	4	Male	Address 6
85		1	58	Male	Address 7
86		6	27	Female	Address 8
87		7	75	Female	Address 7
88		6	22	Male	Address 1
89		2	97	Male	Address 2
90		5	10	Male	Address 1
91		9	77	Female	Address 5
92		6	62	Male	Address 8
93		9	17	Female	Address 1

题 8:

test.docx



- 3、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成绩。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。
- 4、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。
- 5、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、奇数行所有单元格的文本设置为浅蓝色、宋体

## 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。