中山大學
SUN YAT-SEN UNIVERSITY

《人工智能编程语言》第 2 次作业

学生信息系统

姓　　名：　胡嘉睿
学　　号：　24311019
教学班号：　智慧交通班
专　　业：　智慧交通
院　　系：　智能工程学院

2024~2025 学年第二学期

一. 目标

熟练掌握 Python 中的 os 、random 模块的引用等高级特性。

二. 具体任务

请编写一个 Python 程序， 基于『人工智能编程语言学生名单.txt』开发学生信息系统， 要求使用 Student 类，并仅引用 `os` 、 `random` 模块。要求实现以下功能要求：

1. 信息查找和定位，要求如下：
   - 输入学号，可以查找打印其姓名、性别、班级、学院信息。
2. 随机点名，要求如下：
   - 输入需要回答问题的学生数量，返回对应数量的随机的学生姓名及学号。
3. 打印考场安排表，要求如下：
   - 需要将学生顺序打乱排列，在程序根目录下打印输出“考场安排表.txt ”，包含考场顺序号（1-10）、姓名、学 号。
4. 打印准考证号，要求如下：
   - 根据考场安排信息，在根目录下创建一个名为“准考证号 ”的文件夹，并用考场顺序号命名其准考证文件『01.txt』、『02.txt』…『10.txt』，其中需要包含信息：考场顺序号、姓名、学号。
5. 退出，要求如下：
   - 结束程序。

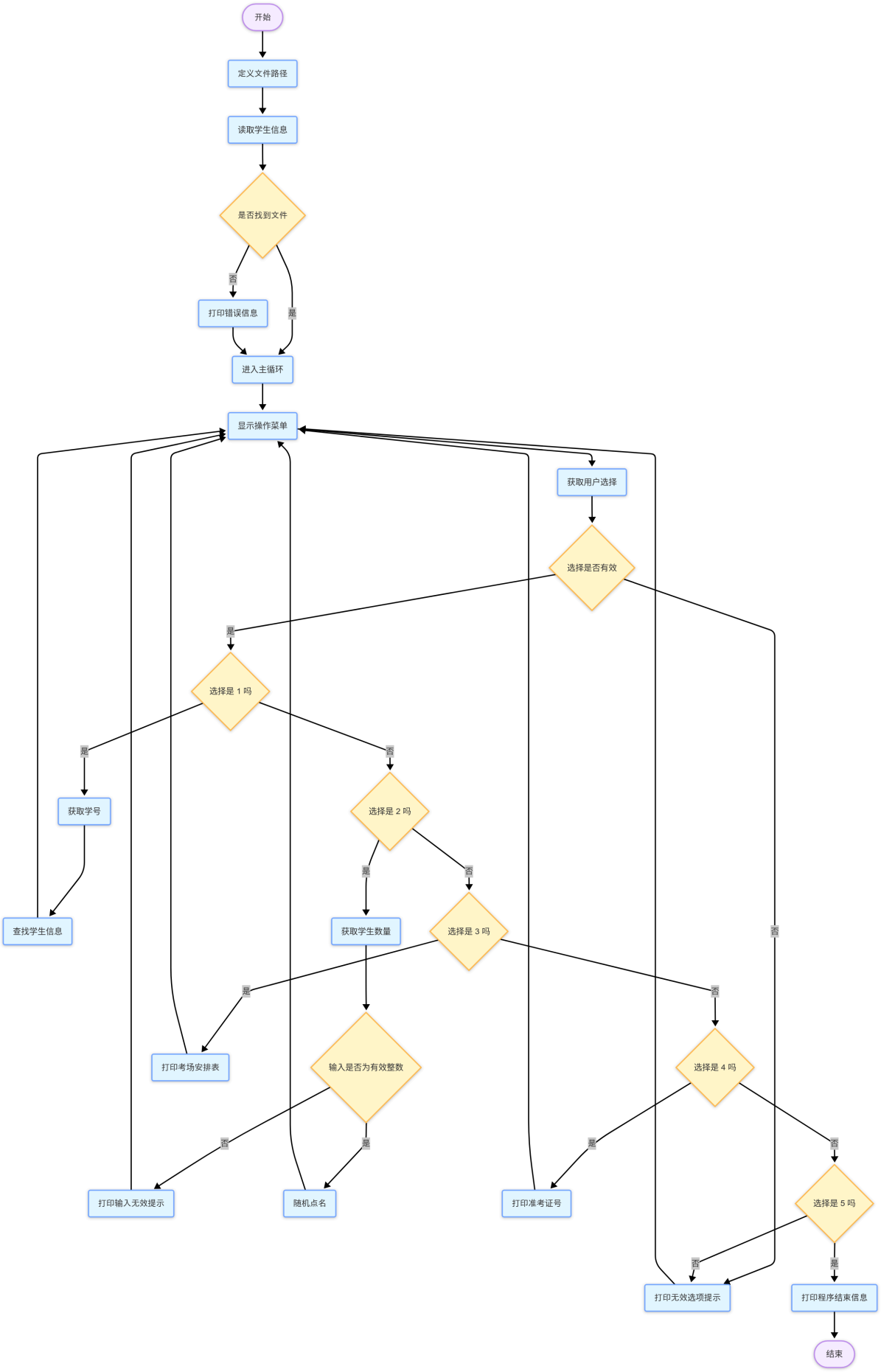
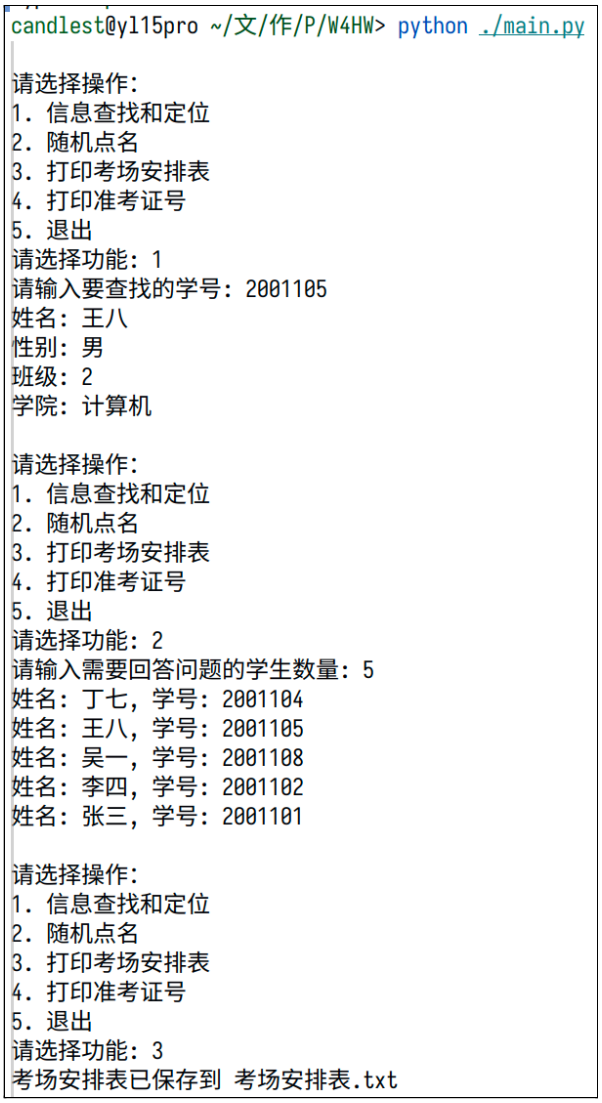

开始
定义文件路径
读取学生信息
是否找到文件
否
打印错误信息
是
进入主循环
显示操作菜单
获取用户选择
选择是否有效
是
选择是 1 吗
是
获取学号
查找学生信息
否
选择是 2 吗
是
获取学生数量
否
选择是 3 吗
是
打印考场安排表
输入是否为有效整数
否
打印输入无效提示
是
随机点名
否
选择是 4 吗
是
打印准考证号
否
选择是 5 吗
否
打印无效选项提示
是
打印程序结束信息
结束

图 1　示意图

三. 运行结果

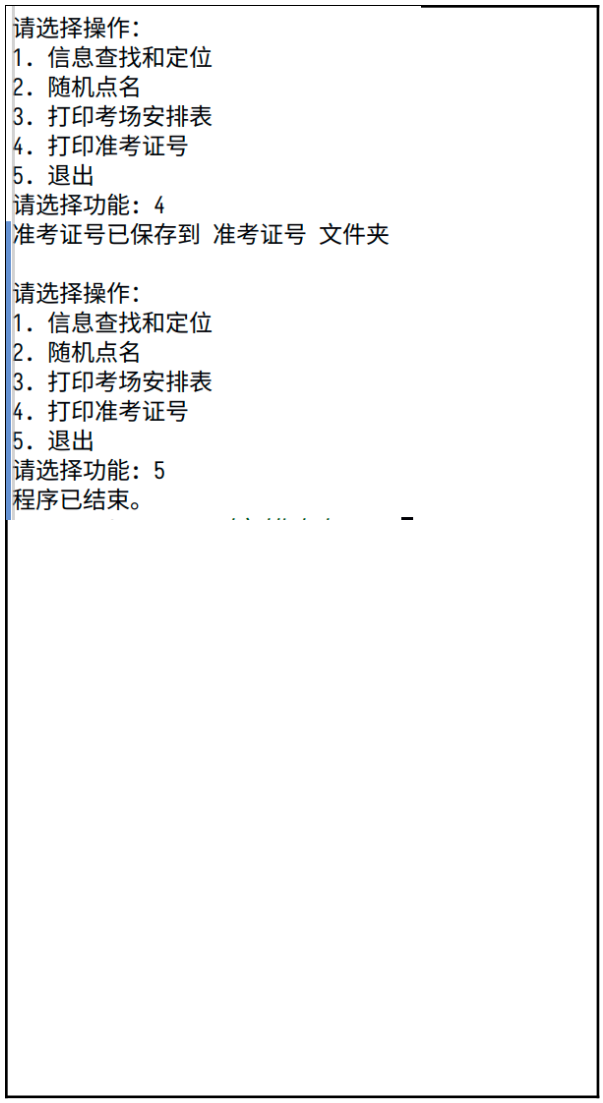


```
candlest@yl15pro ~/文/作/P/W4HW> python ./main.py

请选择操作:
1. 信息查找和定位
2. 随机点名
3. 打印考场安排表
4. 打印准考证号
5. 退出
请选择功能: 1
请输入要查找的学号: 2001105
姓名: 王八
性别: 男
班级: 2
学院: 计算机

请选择操作:
1. 信息查找和定位
2. 随机点名
3. 打印考场安排表
4. 打印准考证号
5. 退出
请选择功能: 2
请输入需要回答问题的学生数量: 5
姓名: 丁七, 学号: 2001104
姓名: 王八, 学号: 2001105
姓名: 吴一, 学号: 2001108
姓名: 李四, 学号: 2001102
姓名: 张三, 学号: 2001101

请选择操作:
1. 信息查找和定位
2. 随机点名
3. 打印考场安排表
4. 打印准考证号
5. 退出
请选择功能: 3
考场安排表已保存到 考场安排表.txt
```



```
请选择操作:
1. 信息查找和定位
2. 随机点名
3. 打印考场安排表
4. 打印准考证号
5. 退出
请选择功能: 4
准考证号已保存到 准考证号 文件夹

请选择操作:
1. 信息查找和定位
2. 随机点名
3. 打印考场安排表
4. 打印准考证号
5. 退出
请选择功能: 5
程序已结束。
```

四. 附录：代码

```python
# 导入 os 模块，用于处理文件和目录路径、创建目录等操作
import os
# 导入 random 模块，用于生成随机数、随机抽样和打乱列表顺序等操作
import random


# Student 类，用于表示学生对象
class Student:
    # 类的构造函数，用于初始化学生对象的属性
```

```
    def __init__(self, id, name, gender, class_, school):
        # 学生的学号
        self.id = id
        # 学生的姓名
        self.name = name
        # 学生的性别
        self.gender = gender
        # 学生所在的班级
        self.class_ = class_
        # 学生所在的学院
        self.school = school


# 从文件中读取学生信息
def read_students(file_path):
    # 初始化一个空列表，用于存储读取到的学生对象
    students = []
    try:
        # 以只读模式打开指定路径的文件，并使用 UTF-8 编码
        with open(file_path, 'r', encoding='utf-8') as file:
            # 跳过文件的第一行，通常第一行为表头
            next(file)
            # 逐行读取文件中的剩余内容
            for line in file:
                # 去除每行末尾的换行符，并按制表符分割字符串
                _, name, gender, class_, id, school = line.strip().split('\t')
                # 创建一个新的 Student 对象
                student = Student(id, name, gender, class_, school)
                # 将新创建的学生对象添加到 students 列表中
                students.append(student)
        # 返回存储所有学生对象的列表
        return students
    except FileNotFoundError:
        # 若文件未找到，打印错误信息
        print("错误：未找到学生名单文件。")
        # 返回空列表
        return []


# 根据学号查找学生信息并打印
def find_student(students, id):
    # 遍历存储学生对象的列表
    for student in students:
        # 检查当前学生的学号是否与输入的学号匹配
        if student.id == id:
            # 若匹配，打印该学生的详细信息
```

```
56              print(f"姓名: {student.name}\n性别: {student.gender}\n班级: {student.class_}\n学院: {student.school}")
57              return
58      # 若遍历完列表未找到匹配的学生，打印提示信息
59      print("未找到该学号对应的学生信息。")
60
61
62  # 随机点名指定数量的学生
63  def random_roll_call(students, num):
64      # 检查所需学生数量是否超过总学生数量
65      if num > len(students):
66          # 若超过，打印提示信息
67          print("所需学生数量超出总学生数量。")
68          return
69      # 从学生列表中随机抽取指定数量的学生
70      selected_students = random.sample(students, num)
71      # 遍历抽取到的学生列表
72      for student in selected_students:
73          # 打印每个被选中学生的姓名和学号
74          print(f"姓名: {student.name}, 学号: {student.id}")
75
76
77  # 打印考场安排表并保存到文件
78  def print_exam_arrangement(students):
79      # 随机打乱学生列表的顺序
80      random.shuffle(students)
81      # 以写入模式打开名为 "考场安排表.txt" 的文件，并使用 UTF-8 编码
82      with open("考场安排表.txt", 'w', encoding='utf-8') as file:
83          # 遍历前 10 个学生，并为每个学生分配一个考场顺序号
84          for i, student in enumerate(students[:10], start=1):
85              # 将每个学生的考场顺序号、姓名和学号写入文件
86              file.write(f"考场顺序号: {i}, 姓名: {student.name}, 学号: {student.id}\n")
87      # 打印提示信息，告知考场安排表已保存
88      print("考场安排表已保存到 考场安排表.txt")
89
90
91  # 打印准考证号并保存到文件夹
92  def print_admission_tickets(students):
93      # 随机打乱学生列表的顺序
94      random.shuffle(students)
95      # 检查 "准考证号" 文件夹是否存在
96      if not os.path.exists("准考证号"):
97          # 若不存在，创建该文件夹
98          os.makedirs("准考证号")
99      # 遍历前 10 个学生，并为每个学生分配一个考场顺序号
100     for i, student in enumerate(students[:10], start=1):
101         # 生成准考证号文件的完整路径
```

```
        file_name = os.path.join("准考证号", f"{i:02d}.txt")
        # 以写入模式打开准考证号文件，并使用 UTF-8 编码
        with open(file_name, 'w', encoding='utf-8') as file:
            # 将每个学生的考场顺序号、姓名和学号写入文件
            file.write(f"考场顺序号: {i}\n姓名: {student.name}\n学号: {student.id}\n")
    # 打印提示信息，告知准考证号已保存
    print("准考证号已保存到 准考证号 文件夹")


# 定义主函数，作为程序的入口点
def main():
    # 定义存储学生名单的文件路径
    file_path = "人工智能编程语言学生名单.txt"
    # 调用 read_students 函数，从文件中读取学生信息
    students = read_students(file_path)
    # 进入一个无限循环，直到用户选择退出
    while True:
        # 打印操作菜单
        print("\n请选择操作：")
        print("1. 信息查找和定位")
        print("2. 随机点名")
        print("3. 打印考场安排表")
        print("4. 打印准考证号")
        print("5. 退出")
        # 获取用户输入的选择
        choice = input("请选择功能: ")
        if choice == '1':
            # 若用户选择 1，获取用户输入的学号
            id = input("请输入要查找的学号: ")
            # 调用 find_student 函数，查找并打印该学号对应的学生信息
            find_student(students, id)
        elif choice == '2':
            try:
                # 若用户选择 2，获取用户输入的需要回答问题的学生数量
                num = int(input("请输入需要回答问题的学生数量: "))
                # 调用 random_roll_call 函数，进行随机点名
                random_roll_call(students, num)
            except ValueError:
                # 若用户输入的不是有效的整数，打印错误提示信息
                print("输入无效，请输入一个整数。")
        elif choice == '3':
            # 若用户选择 3，调用 print_exam_arrangement 函数，打印并保存考场安排表
            print_exam_arrangement(students)
        elif choice == '4':
            # 若用户选择 4，调用 print_admission_tickets 函数，打印并保存准考证号
            print_admission_tickets(students)
```

```
        elif choice == '5':
            # 若用户选择 5，打印程序结束信息
            print("程序已结束。")
            # 跳出循环，结束程序
            break
        else:
            # 若用户输入的选项无效，打印提示信息
            print("无效的选项，请重新输入。")


# 只在作为脚本运行时生效
if __name__ == "__main__":
    main()
```