

Problem 2 – Grades Processing

17101992 HongSumin

Index

1. About Problem
2. Analysis & Function
3. Code
4. Q&A

About Problem

성적 처리 프로그램을 작성하자. “컴퓨터 프로그래밍” 과목에서는 중간 시험, 기말 시험, 과제를 각각 100점 만점씩, 300점 만점으로 하여 성적을 산출한다.

입력 정보: 학번, 이름, 중간시험 점수, 기말시험 점수, 과제 점수

입출력 예:

```
01 손흥민 99 99 99
02 최철순 67 89 77
03 권창훈 55 73 85
04 권순태 78 90 56
05 전가을 89 95 78
06 이동국 78 98 75
07 이청용 35 76 68
08 이재성 89 67 88
09 지소연 66 89 85
10 김신욱 78 82 60
```

학번	이름	중간	기말	과제	총점	평균	학점
1	손흥민	99	99	99	297	99.00	A
2	최철순	67	89	77	233	77.67	C
3	권창훈	55	73	85	213	71.00	C
4	권순태	78	90	56	224	74.67	C
5	전가을	89	95	78	262	87.33	A
6	이동국	78	98	75	251	83.67	B
7	이청용	35	76	68	179	59.67	D
8	이재성	89	67	88	244	81.33	B
9	지소연	66	89	85	240	80.00	B
10	김신욱	78	82	60	220	73.33	C

About Problem

제한 요소 및 요구 사항

- 프로그램은 다음과 같은 순서로 실행되며, 명시된 제한 사항을 지켜야 한다.

- 각 학생의 학번, 이름, 성적을 파일에서 입력 받는다. 여러 반에 대해 처리할 필요가 있으므로 학생의 수는 입력 시마다 변할 수 있도록 처리한다.
- 학생별 평균을 계산하여 저장한다.
- 평균을 정렬하여 상위부터 20% A, 30% B, 40% C, 10% D의 학점을 산출하여 기록한다.
- 전체 성적표를 학번, 이름, 중간, 기말, 과제, 총점, 평균, 학점의 순서로 출력하도록 한다. 마지막 줄에는 평균을 출력한다.
- 종료하기 전 결과를 파일에 기록할지 여부를 물어 보아 그에 따라 기록한다.
- 정렬할 때 전체 배열이나 구조체를 복사하는 일이 없도록 효율성을 고려해 프로그램을 작성한다.
- 위에 나열된 각 작업은 별도의 함수(메소드)로 작성해야 한다.
- 단계 별로 작업이 진행될 때마다 적절한 출력 메시지를 출력한다.

- 파일을 이용해 입출력한다.

- 여러 개의 소스 파일과 헤더 파일을 사용한다. 또는 여러 개의 클래스를 사용한다.

1. 가변적인 학생 수
2. 클래스 활용
3. 파일 입출력

Analysis & Function

- **Student Class**

- Student ID
- Name
- Midterm score
- Final score
- Assignment score
- Total score
- Average
- Grade

- Set Average
- Set Grade

- Get student scores' file and save at Student Class
- Ordering / Grading
- Print out result
- Save result to file
- Exception Handling

Code : Class

Student Class

It will receive value by input method, so int() to string -> integer

Student Class's method

Can set average & grade

idx+1: index starts at 0

```
class Student(object):  
    def __init__(self, num, name, midterm, final, assignment):  
        self.num = int(num)  
        self.name = name  
        self.midterm = int(midterm)  
        self.final = int(final)  
        self.assignment = int(assignment)  
        self.totalscore = 0  
        self.average = 0  
        self.grade = ""  
  
    def setAverage(self):  
        self.totalscore = self.midterm+self.final+self.assignment  
        self.average = float(format((self.totalscore)/3, ".2f"))  
  
    def setGrade(self, idx, total):  
        percentage = (idx+1)/total  
        if percentage <= 0.2:  
            self.grade = 'A'  
        elif 0.2 <= percentage <= 0.5:  
            self.grade = 'B'  
        elif 0.5 <= percentage <= 0.9:  
            self.grade = 'C'  
        else:  
            self.grade = 'D'
```

Code : Get information

```
def getStudentInfo():  
    while True:  
        try:  
            filename = input("처리할 성적 파일 루트를 입력해주세요.")  
            f = open_file(filename).split('\n')  
            break  
  
        except (FileNotFoundError, IOError, OSError):  
            print("정확한 파일 경로를 입력해주세요. ex)C:\LabAssignment\\testfile.txt")  
            pass  
  
    studentList = []  
    for idx, value in enumerate(f):  
        num, name, midterm, final, assignment = value.split(' ')  
        studentList.append(Student(num, name, midterm, final, assignment))  
        studentList[idx].setAverage()  
  
    return studentList
```

getStudentInfo()

get filename that contains
students' information

Exception Handling

studentList

Put object in each list elements &
set averages

Code : Processing

```
def orderStudent(studentList):  
    # studentList.sort(key=studentList.average, reverse=True)  
    studentList = sorted(studentList, key=lambda student: student.average, reverse=True)  
  
    return studentList  
  
def gradeStudent(studentList):  
    for idx, val in enumerate(studentList):  
        val.setGrade(idx, len(studentList))  
  
    return studentList
```

orderStudent(studentList)
Use 'sorted' method to sort student by setting key to average

gradeStudent(studentList)
Use setGrade and give this index & length of studentList

Code : Output

```
def output_result(studentList):  
    studentList = sorted(studentList, key=lambda student: student.num)  
    result = "학번 이름 중간 기말 과제 총점 평균 학점\n"  
    for obj in studentList:  
        obj_list=[obj.num, obj.name, obj.midterm, obj.final, obj.assignment, obj.totalscore, obj.average, obj.grade]  
        obj_list = list(map(str, obj_list))  
  
        result += " ".join(obj_list) + "\n"  
  
    # result += str(obj.num) + " " + obj.name + " " + str(obj.midterm) + " " + str(obj.final) + " " + str(obj.assig  
  
print(result)  
  
while True:  
    choice = input("결과를 파일로 저장하시겠습니까? [Y/N]").upper()  
    if choice == 'Y':  
        write_file(result)  
        break  
    elif choice == "N":  
        print("결과를 저장하지 않습니다.")  
        break  
    else:  
        print("올바른 값을 입력해주세요!")  
        pass
```

output_result(studentList)
Sorting student's list by student ID again
Make each object's attributes list & convert them to string
Make result by using for statement

Exception Handling

Code : File Handling

```
def write_file(result):  
    newFile = open("ResultScore.txt", 'w')  
    newFile.write(result)  
    newFile.close()  
    print('결과가 파일로 출력되었습니다!')  
  
def open_file(filename):  
    lines = ""  
    with open(filename, 'r', encoding='utf8') as fh:  
        for line in fh:  
            lines += line  
  
    return lines
```

File Handling

Write file & open file

Code : Main

```
# Press the green button in the gutter to run the script.
```

```
if __name__ == '__main__':
```

```
    score_data = getStudentInfo()
```

```
    order_student = orderStudent(score_data)
```

```
    grade_student = gradeStudent(order_student)
```

```
    output_result(grade_student)
```

```
# See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

Main function

Run the preceding function

Code : Result

```
print(result)

while True:
    choice = input("결과를 파일로 저장하시겠습니까? [Y/N]").upper()
    if choice == 'Y':
        write_file(result)
        break
    elif choice == "N":
        print("결과를 저장하지 않습니다.")
        break
    else:
        print("올바른 값을 입력해주세요!")
        pass

def write_file(result):
    newFile = open("ResultScore.txt", 'w')
    newFile.write(result)
    newFile.close()
    print('결과가 파일로 출력되었습니다!')

write_file()

main <
main <
학번 이름 중간 기말 과제 총점 평균 학점
1 손흥민 99 99 99 297 99.0 A
2 최철순 67 89 77 233 77.67 C
3 권창훈 55 73 85 213 71.0 C
4 권순태 78 90 56 224 74.67 C
5 전가을 89 95 78 262 87.33 A
6 이동국 78 98 75 251 83.67 B
7 이청용 35 76 68 179 59.67 D
8 이재성 89 67 88 244 81.33 B
9 지소연 66 89 85 240 80.0 B
10 김신욱 78 82 60 220 73.33 C

결과를 파일로 저장하시겠습니까? [Y/N]Y
결과가 파일로 출력되었습니다!

Process finished with exit code 0
```

```
ResultScore - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
학번 이름 중간 기말 과제 총점 평균 학점
1 손흥민 99 99 99 297 99.0 A
2 최철순 67 89 77 233 77.67 C
3 권창훈 55 73 85 213 71.0 C
4 권순태 78 90 56 224 74.67 C
5 전가을 89 95 78 262 87.33 A
6 이동국 78 98 75 251 83.67 B
7 이청용 35 76 68 179 59.67 D
8 이재성 89 67 88 244 81.33 B
9 지소연 66 89 85 240 80.0 B
10 김신욱 78 82 60 220 73.33 C
```

Q&A