파이썬 프로젝트

성적관리프로그램

소속: C 반 1 조

이름: 권구택

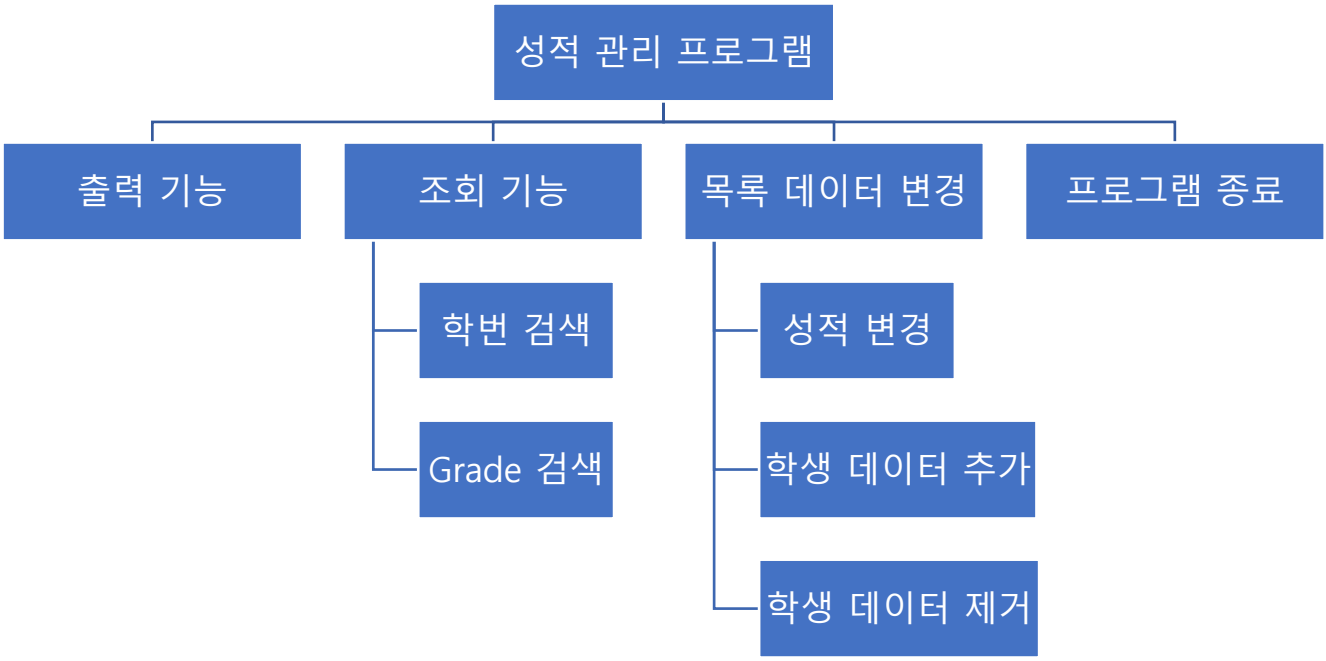
연락처: gt0830@naver.com

1. 개요

본 프로그램을 간략히 설명하면 다음과 같다.

- 사용자가 조회하기 원하는 파일을 입력 받고 특정 형식에 맞춰서 출력해준다.
- 추가적으로 7가지 기능이 있다.
  - Show : 사용자가 원하는 경우 전체 목록을 출력한다.
  - Search : 특정 학생을 검색하고 해당하는 학생 정보를 출력한다.
  - Changescore : 특정 학생의 성적을 변경한다.
  - Searchgrade : 특정 Grade에 해당하는 학생들의 리스트를 출력한다.
  - Add : 기존 목록에 없는 학생 정보를 추가하고 반영한다.
  - Remove : 기존 목록에 있던 학생 정보를 삭제한다.
  - Quit : 파일 저장 여부를 확인하고 프로그램을 종료한다.

이 때 사용되는 기능들을 시각화하면 다음과 같이 표현할 수 있다.



2. 알고리즘

본 프로그램 작성을 위한 알고리즘을 Pseudo Code로 나타내면 다음과 같다.

```
입력받는 파일명에 해당하는 파일을 연다
파일의 각 행(LINE)을 읽고 리스트 자료형으로 저장한다.
        학생과 학생 사이는 ₩n 을 기준으로 line 을 분류한다.
        나눠진 라인은 ₩t 를 기준으로 학번, 이름, 중간고사 점수, 기말고사 점수로 나누고
        저장한다.
각 리스트에 평균 점수와 Grade 를 계산한다.
        평균 점수 = mean(중간점수, 기말점수)
        평균 점수를 저장한다
        평균 점수를 기준으로 Grade 를 산출하여 리스트에 추가한다.
전체 리스트를 평균 점수 기준 내림차수능로 정렬한다
학생들의 성적 목록을 출력한다
        각 컬럼명을 정리한 줄을 우선 출력한다.
        순서대로 데이터를 출력한다
# 을 출력하고 명령어를 대기한다
        Input 으로 명령어를 입력받는다.
명령어에 적합한 명령을 수행한다.
        명령어를 LOWER()함수를 통해 소문자로 변환한다
        If 명령어 = "show" then
                전체 리스트를 출력한다
                        전체 리스트를 평균 기준 내림차순으로 재정렬하고 출력한다
                        평균 점수는 소수점 이하 첫째 자리까지만 출력한다.
        elif 명령어 = "search" then
                특정 학생을 검색한다
                        학번을 입력받는다
                                If 학번 in 리스트 then
                                        해당 학생 정보를 출력받는다
                                else
                                        "NO SUCH PERSON" 출력한다
        elif 명령어 = "changescore" then
                학번을 입력 받고 검색하고 점수를 변경한다
                학번을 입력반든다.
                        If 학번 in 리스트 then
                                바꾸고 싶은 점수를 입력받고 새 점수도 입력 받는다
                                해당하는 값을 변경한다
                                변경된 값에 맞게 평균을 다시 계산하고 순서를 정렬한다
                        else
                                "NO SUCH PERSON" 출력한다
```

```
elif 명령어 = "add" then
        학번, 이름, 중간 점수, 기말 점수를 차례대로 입력 받는다
        해당 값들로 리스트를 생성하여 추가한다
        "Student added"를 출력한다

elif 명령어 = "searchgrade" then
        Grade 를 입력받는다
        if Grade in [A,B,C,D,F] then
                그 값에 해당하는 학생을 모두 검색한다
                if 검색 결과가 1 개 이상 then
        검색 결과를 모두 출력한다.
                else
                        "NO RESULT'를 출력
        else
                실행하지 않는다

elif 명령어 = "remove" then
        학번을 입력받는다
        if 학번 in 리스트 then
                학번이 속한 리스트를 제거한다
                'Student removed'를 출력한다
        else
                'NO SUCH PERSON'을 출력한다
elif 명령어 = 'quit' then
        저장 여부를 묻는다.
        if Yes 를 입력 then
                파일 명을 입력 받는다
        평균을 기준으로 내림차순으로 정렬한다
입력 받은 파일 명으로 저장한다
else
        다시 입력 받는다
```

3. 프로그램 구조 및 설명

프로그램의 기능 구현 코드는 다음과 같다.

- **cmd에서 실행하기**

cmd에서 해당 프로그램 실행 코드 입력과 동시에 열고자 하는 파일명을 입력하면 해당 파일을 읽고 다음과 같이 출력한다.

코드(Jupyter notebook):

```
import numpy as np
import copy
import sys



### 파일 읽어오기
try:
    readfilename = sys.argv[1]
except:
    readfilename = 'students.txt'

f = open(readfilename,'r') ## Jupyter 환경에서는 파일명을 직접 입력했음

### 사용자 정의 함수 생성 - Grade 산출 함수
def cal_grade(score):
    if score >= 90:
        return "A"
    elif 80 <= score < 90:
        return "B"
    elif 70 <= score < 80:
        return "C"
    elif 60 <= score < 70:
        return "D"
    else:
        return "F"

### 전체 데이터를 관리할 리스트 생성
datalst = []
### 한줄씩 읽으면서 해당 데이터를 추가
for line in f:
    newdata = line.split()
    ### 중간고사 점수와 기말고사 점수를 int형으로 변환
    newdata[-2], newdata[-1] = int(newdata[-2]), int(newdata[-1])
    ### 성과 이름을 합치기
    newdata[1] += " "
    newdata[1] += newdata[2]
    del(newdata[2])
    newdata.append(round(np.mean(newdata[-2:]),1))
    newdata.append(cal_grade(newdata[-1]))
    ### 전체 데이터 리스트 추가
    datalst.append(newdata)
f.close()
```

```
### 전체 데이터 출력 함수 생성
def showall(datalst):
    print("Student number₩tName₩t₩tMidterm₩tFinal₩tAverage₩tGrade₩n")
    print("-----------------------------------------------------------------")
    ### 평균 점수 내림차순 정렬
    datalst.sort(key=lambda x: x[-2], reverse=True)
    for student_data in datalst:
        print('₩t'.join(list(map(str,student_data))))

showall(datalst)
```

이 코드 내에서 뒤에 기능을 구현하는데 필요한 사용자 정의 함수도 같이 작성하였다.

- **show 기능**

저장되어 있는 전체 목록을 특정 형식에 맞게 출력한다. 출력 순서는 평균 점수를 기준으로 내림차순으로 출력한다.

코드(Jupyter notebook):

```
orderlst = ['show', 'search','changescore','searchgrade','add','remove','quit']
while True:
    order = input("# ")
    order = order.lower()
    if order in orderlst:
        if order == 'show':  # show 기능 구현
            showall(datalst)
```

위에서 showall 함수를 구현해씩 때문에 해당 기능은 쉽게 구현가능했다.

- **search 기능**

검색하고자 하는 학번을 입력하면 해당 학번이 목록 안에 있는지 확인하고 없으면 에러 메시지를 출력하고 존재하면 기존 출력 형식에 맞춰서 출력한다.

코드(Jupyter notebook):

```
        elif order == 'search': # search 기능 구현
            student_num = input("Student ID: ")
            student_numlst = [x[0] for x in datalst]
            if student_num in student_numlst:
                order_result = datalst[student_numlst.index(student_num)]
                showall([order_result])
            else:
                print("NO SUCH PERSON")
```

전체 목록에서 학번들만 모아놓은 리스트를 생성하고 해당 리스트에 존재하지 않으면 에러 메시지를 출력하고 아닐 경우 해당 값이 있는 Index 값을 받아 해당 학번에 해당하는 학생의 정보를 가져와 showall 함수를 이용해 출력한다.

- **changescore기능**

원하는 학생의 학번을 입력 후 수정하고자 하는 시험과 원하는 점수를 입력하면 해당 값으로 수정하고 평균과 Grade를 다시 계산한다.

코드(Jupyter notebook):

```
        elif order == 'changescore': # changescore 기능 구현
            student_num = input("Student ID: ")
            student_numlst = [x[0] for x in datalst]
            if student_num not in student_numlst:
                print("NO SUCH PERSON")
            else:
                score_type = input("Mid/Final? ")
                score_type = score_type.lower()
                if score_type == 'mid':
                    newscore = int(input("Input new score: "))
                    if newscore not in range(0, 101):
                        pass
                    else:
                        changeindex = student_numlst.index(student_num)
                        beforedata = copy.deepcopy(datalst[changeindex])
                        datalst[changeindex][2] = newscore
                        datalst[changeindex][4] = round(np.mean(datalst[changeindex]
                        datalst[changeindex][-1] = cal_grade(datalst[changeindex][4]
                        showall([beforedata])
                        print('Score changed')
                        print('₩t'.join(list(map(str, datalst[changeindex]))))
                elif score_type == 'final':
                    newscore = int(input("Input new score: "))
                    if newscore not in range(0, 101):
                        pass
                    else:
                        changeindex = student_numlst.index(student_num)
                        beforedata = copy.deepcopy(datalst[changeindex])
                        datalst[changeindex][3] = newscore
                        datalst[changeindex][4] = round(np.mean(datalst[changeindex]
                        datalst[changeindex][-1] = cal_grade(datalst[changeindex][4]
                        showall([beforedata])
                        print('Score changed')
                        print('₩t'.join(list(map(str, datalst[changeindex]))))
                else:
                    pass
```

우선 학번을 입력 받고 해당 학번이 목록 내에 있는지 확인한다. 이후 성적을 바꾸고 싶은 시험을 입력 받고 소문자로 변환한다. 기존 요구 사항은 mid나 final이 아니면 무시하라는 내용이었지만 명령어처럼 대소문자 구별 없이 입력 받으면 좀 더 좋다고 생각하여 해당 코드를 작성하였다. 이후 원하는 점수를 입력을 받는다. 단, 0에서 100 사이의 수가 아닌 경우 무시한다. 이후 입력 받은 값으로 성적을 바꾸고 평균과 Grade를 다시 계산한다. 마지막으로 showall 함수를 이용해 출력한다.

아쉬운 점은 여러 기능 중에서 코드가 제일 길었던 코드인데 지금 다시 생각해보니 이 부분에서도 사용자 정의 함수를 만들 수 있을 것 같다. Mid 나 final 값을 입력 받고 진행되는 코드는 해

당 시험 성적의 index 번호를 제외하고 다 일치하기 때문에 함수로 정의했다면 보다 더 간결한 코드를 작성할 수 있을 것 같다.

- **add 기능**

학번, 이름, 중간고사 점수, 기말고사 점수를 입력하여 새로운 학생의 정보를 목록에 저장한다.

코드(Jupyter notebook):

```
        elif order == 'add':  # add 구현
            student_num = input("Student ID: ")
            student_numlst = [x[0] for x in datalst]
            if student_num in student_numlst:
                print("ALREADY EXISTS")
            else:
                newname = input("Name: ")
                newmid = int(input("Midterm Score: "))
                newfinal = int(input("Final Score: "))
                newavg = round(np.mean([newmid, newfinal]),1)
                newgrade = cal_grade(newavg)
                datalst.append([student_num, newname, newmid, newfinal, newavg, newgr
                print('Student added')
```

*** 잘린 부분의 코드는 다음과 같다.**

datalst.append([student_num, newname, newmid, newfinal, newavg, newgrade])

학번을 입력 받고 이미 전체 목록에 존재할 경우 에러 메시지를 출력한다. 만약 새로운 학번일 경우, 전체 목록에 입력 받은 데이터들을 입력 받고 리스트 형태로 전체 목록에 추가하고 메시지를 출력한다.

- **searchgrade 기능**

원하는 Grade에 해당하는 성적을 가지고 있는 학생이 있는지 확인하고 해당 학생들의 리스트를 출력한다.

코드(Jupyter notebook):

```
        elif order == 'searchgrade': # searchgrade 구현
            gradeinput = input("Grade to search: ")
            if gradeinput not in ['A','B','C','D','F']:
                pass
            else:
                gradelst = [x[-1] for x in datalst]
                if gradeinput not in gradelst:
                    print('NO RESULTS')
                else:
                    searchresult = [studata for studata in datalst if studata[-1] ==
                    showall(searchresult)
```

*** 잘린 부분의 코드는 다음과 같다.**

searchresult = [studata for studata in datalst if studata[-1] == gradeinput]

정해진 Grade 외의 입력값은 무시하고 만약 전체 목록에 해당 값이 없으면 에러 메시지를 출력한다. 만약 존재한다면 해당 값에 해당하는 학생 정보들을 리스트화하여 showall 함수를 통해 출력한다.

- **remove 기능**

삭제하고자 하는 학생의 학번을 입력 받고 목록에 존재할 경우, 삭제한다.

코드(Jupyter notebook):

```
        elif order == 'remove': # remove 구현
            if len(datalst) == 0:
                print("List is empty")
            else:
                student_num = input("Student ID: ")
                student_numlst = [x[0] for x in datalst]
                if student_num not in student_numlst:
                    print("NO SUCH PERSON")
                else:
                    changeindex = student_numlst.index(student_num)
                    datalst.remove(datalst[changeindex])
#                     del(datalst[changeindex])
                    print("Student removed")
```

만약 전체 목록 내 데이터가 없을 경우 에러 메시지를 발생한다. 학번을 입력 받고 해당 학번에 해당하는 학생이 존재하면 삭제하고 없는 경우 에러 메시지를 출력한다.

- **quit 기능**

저장을 원하는 경우 새로운 이름을 입력 받아 해당 파일명으로 저장하고 아닐 경우 그대로 프로그램을 종료한다.

코드(Jupyter notebook):

```
        elif order == 'quit': # quit 기능 구현
            savedata = input("Save data?[yes/no] ")
            if savedata == 'yes':
                filename = input("File name: ")
                newf = open(filename, 'w')
                datalst.sort(key=lambda x: x[-2], reverse=True)
                for studata in datalst:
                    dataline = '₩t'.join(list(map(str,studata[:-2]))) +'₩n'
                    newf.write(dataline)
                newf.close()
            break
```

저장 여부를 묻고 yes라고 답하면 파일명을 입력 받아 평균 기준 내림차순으로 정렬한 전체 목록을 저장 후 종료한다. no라고 할 경우 바로 프로그램을 종료한다.

4. 프로그램 실행 결과

- 터미널에서 해당 파일 실행하기

```
(base) piai@piai-Precision-7920-Tower:~/python project$ python project.py students.txt
Student number  Name            Midterm Final   Average Grade

------------------------------------------------------------
20180002        Lee Jieun       92      89      90.5    A
20180009        Lee Yeonghee    81      84      82.5    B
20180001        Hong Gildong    84      73      78.5    C
20180011        Ha Donghun      58      68      63.0    D
20180007        Kim Cheolsu     57      62      59.5    F
```

- show 기능

```
# show
Student number  Name            Midterm Final   Average Grade

------------------------------------------------------------
20180002        Lee Jieun       92      89      90.5    A
20180009        Lee Yeonghee    81      84      82.5    B
20180001        Hong Gildong    84      73      78.5    C
20180011        Ha Donghun      58      68      63.0    D
20180007        Kim Cheolsu     57      62      59.5    F
```

- search 기능

```
# search
Student ID: 20180050
NO SUCH PERSON
# search
Student ID: 20180002
Student number  Name            Midterm Final   Average Grade

------------------------------------------------------------
20180002        Lee Jieun       92      89      90.5    A
```

- changescore 기능

```
# changescore
Student ID: 20180050
NO SUCH PERSON
# changescore
Student ID: 20180007
Mid/Final? miid
# changescore
Student ID: 20180007
Mid/Final? MID
Input new score: 147
# changescore
Student ID: 20180007
Mid/Final? mid
Input new score: 75
Student number  Name            Midterm Final   Average Grade

------------------------------------------------------------
20180007        Kim Cheolsu     57      62      59.5    F
Score changed
20180007        Kim Cheolsu     75      62      68.5    D
# show
Student number  Name            Midterm Final   Average Grade

------------------------------------------------------------
20180002        Lee Jieun       92      89      90.5    A
20180009        Lee Yeonghee    81      84      82.5    B
20180001        Hong Gildong    84      73      78.5    C
20180007        Kim Cheolsu     75      62      68.5    D
20180011        Ha Donghun      58      68      63.0    D
```

- searchgrade 기능

```
# searchgrade
Grade to search: E
# searchgrade
Grade to search: F
NO RESULTS
# serachgrade
# searchgrade
Grade to search: D
Student number  Name            Midterm Final   Average Grade

-------------------------------------------------------------
20180007        Kim Cheolsu     75      62      68.5    D
20180011        Ha Donghun      58      68      63.0    D
```

- add 기능

```
# add
Student ID: 20180001
ALREADY EXISTS
# add
Student ID: 20180021
Name: Lee Hyori
Midterm Score: 93
Final Score: 95
Student added
# add
Student ID: 20180006
Name: Lee Sangsun
Midterm Score: 77
Final Score: 66
Student added
# show
Student number  Name            Midterm Final   Average Grade

-------------------------------------------------------------
20180021        Lee Hyori       93      95      94.0    A
20180002        Lee Jieun       92      89      90.5    A
20180009        Lee Yeonghee    81      84      82.5    B
20180001        Hong Gildong    84      73      78.5    C
20180006        Lee Sangsun     77      66      71.5    C
20180007        Kim Cheolsu     75      62      68.5    D
20180011        Ha Donghun      58      68      63.0    D
```

- remove 기능

```
# remove
Student ID: 20180030
NO SUCH PERSON
# remove
Student ID: 20180011
Student removed
# show
Student number  Name            Midterm Final   Average Grade

-------------------------------------------------------------
20180021        Lee Hyori       93      95      94.0    A
20180002        Lee Jieun       92      89      90.5    A
20180009        Lee Yeonghee    81      84      82.5    B
20180001        Hong Gildong    84      73      78.5    C
20180006        Lee Sangsun     77      66      71.5    C
20180007        Kim Cheolsu     75      62      68.5    D
```

- quit 기능

```
# quit
Save data?[yes/no] yes
File name: newStudents.txt
(base) piai@piai-Precision-7920-Tower:~/python project$
```

5. 토론

- 의사코드 작성법에 대해서 찾아보고 Flow Chart 관련해서 어려운 점이 많았다. 실제로 작성한 코드도 올바르게 작성한 것이 맞는지 잘 모르겠다.
- 추가 기능으로 changescore 부분에서 중간기말은 선택하는 과정에서 명령어 입력 때 사용했던 것처럼 대소문자를 구별하지 않고 받아내는 방식으로 수정하였다.

6. 결론

이번 과제를 통해서 코드 작성 시에 바로 코드를 작성하기보다 의사 코드나 Flow Chart 작성을 통해서 좀 더 구체적인 방향을 잡아놓으면 더욱 더 빠르게 코드를 작성할 수 있다는 것을 알았다. 왜냐하면 평소 코드 작성 시에는 생각을 하면서 코드를 작성하지만 의사코드를 미리 작성해놓으니깐 거기에 적힌 내용을 바로 작성하면 되기 때문에 시간이 단축될 것 같다. 물론 아직 의사 코드 작성에 시간이 많이 걸려 전체적인 시간은 비슷하지만 익숙해진다면 효율성을 높일 수 있다고 판단하였다.