성적 관리 시스템

1. 성적 관리 시스템 개요

- 학생들의 정보를 입력받고 성적을 산출하거나 정보를 변경, 추가하는 시스템을 생성하도록 한다.
- 학생들을 구분하는 id로는 학생마다 부여된 고유한 번호인 학번을 이용하도록 한다.
- 학생들의 성적 목록은 언제나 평균 기준 내림차순으로 정렬한다.

입력 및 기본 출력 사항

- 학생 별로 한 행에 학번, 이름, 중간고사 점수, 기말고사 점수가 포함된 텍스트 파일을 입력받는다.
  - 모든 점수는 0이상 100이하이다.
- 학생 별로 한 줄 씩 출력하고 중간, 기말고사 점수에 대한 평균 ,학점을 계산하여 추가한다.
  - 평균은 중간고사와 기말고사 점수의 평균을 의미하며 소수점 첫째 자리까지 표시한다.
  - 학점의 경우 평균 기준으로 90점 이상이면 ‘A’, 80점 이상 90점 미만이면 ‘B’, 70점 이상 80점 미만이면 ‘C’, 60점 이상 70점 미만이면 ‘D’, 60점 미만이면 ‘F’이다.
- 프로그램 실행 시 현재 입력된 모든 학생들의 성적 목록을 한 번 출력한다.
  - 성적 목록에 포함된 정보는 좌측부터 학번, 이름, 중간고사 점수, 기말고사 점수, 평균, 학점이다.

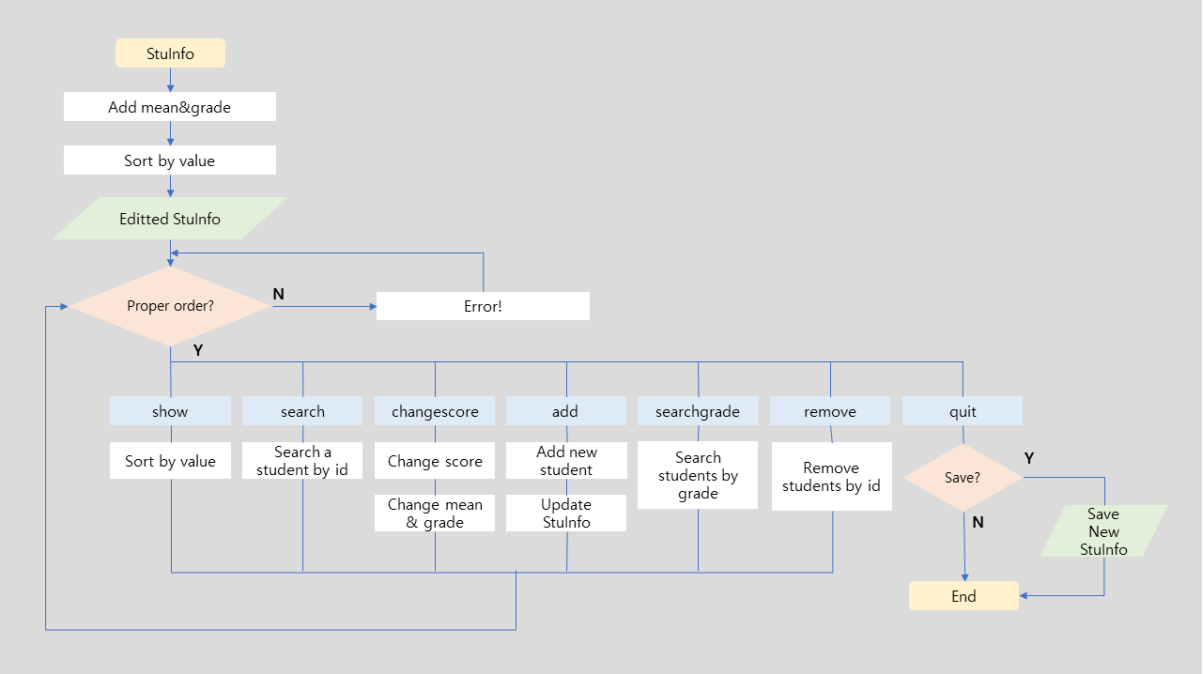
명령어 부분

- 기본 출력이하 ‘#’를 표시하면서 명령어 입력을 기다린다.
- 이용자에게 명령어를 직접 입력받고 이에 대한 output을 시스템 상에 출력한다.
  - 이때, 명령어는 대소문자를 구분하지 않고 모두 같은 기능을 수행하도록 한다.
- 각 명령어에 대한 동작은 함수로 정의하도록 하고 이외 필요한 경우 함수를 추가하여 사용한다.
- 명령어에 따라 학생의 정보가 추가되거나 변경될 수 있다.
- 명령어는 show, search, changescore, add, searchgrade, remove, quit 7개이다.

종료 부분

- quit 명령어에 의해 프로그램은 종료된다.
- 시스템 상 변경된 내용은 프로그램 종료시 선택에 따라 처음 입력된 텍스트 파일과 같은 형태의 파일에 포함되어 저장할 수 있다.

2. 알고리즘 설계



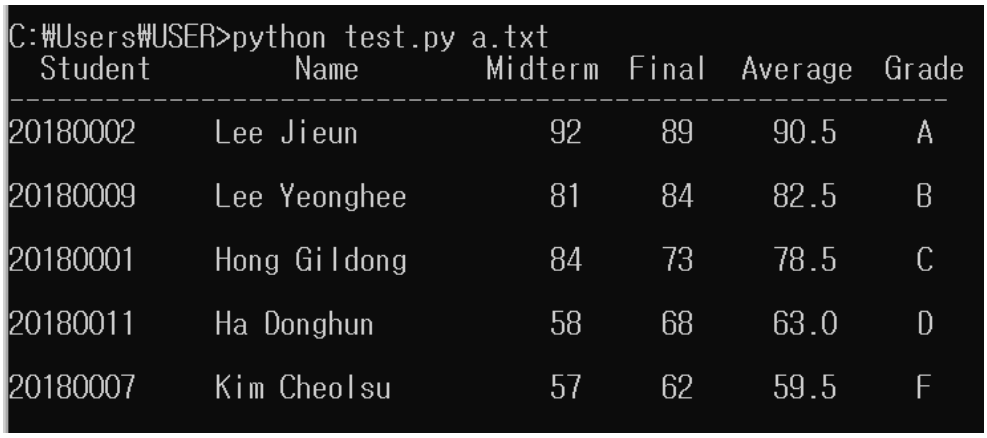

```
(1) stu_info = open('studentinfo.txt', 'r')
    add mean&grade to stu_info
    def show(stu_info):
      stu_info sorted by mean
      show(stu_info)

(2) order = input('#')

- show
  show(stu_info)

- search
    def search(id, stu_info):
      if id in stu_info:
         find a student in stu_info by id
      else:
        back to (2)inputting order

- changescore
    def changscore(id, test_type, stu_info):
      if id not in stu_info:
        back to (2)inputting order
      else:
        searching a student by id
        changing the scor
        caculating mean&grade again

- add
    def add(id, name, mid_score, final_score, stu_info):
    adding a student to stu_info

- searchgrade
    def searchgrade(grade, stu_info):
      searching every student having the exact grade in stu_info

- remove
    def remove(id, stu_info):
      searching a student by id
      removing the student in stu_info
```

```
- quit
    def quit(stu_info):
      if save == yes:
        save stu_info in new file
        quit the program
      else:
        quit the program
```

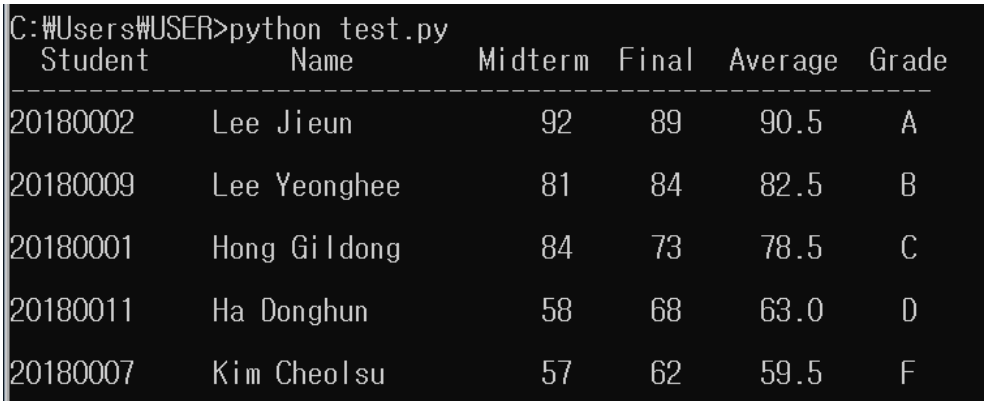
3. 기능 실행 결과

처음 실행

- 파일을 받아 바로 mean, grade를 계산한 후 mean 기준으로 정렬된 성적 목록을 출력하는 단계
- 파일명이 있을 때
  - a.txt 읽어왔을 때



- 파일 명이 없을 때
  - default인 ‘students.txt’를 읽은 상태



명령어 실행

- 명령어가 없을 때

```
# seachgrade
error!
```

show

- 전체 성적 목록을 평균 기준 내림차순 출력
- 평균은 소수점 이하 첫째 자리까지만

```
# show
  Student        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

- id가 성적 목록에 없을 때
- id가 성적 목록에 있을 때

```
# search
Student ID:2018
NO SUCH PERSON.
# search
Student ID:20180002
  Student        Name      Midterm  Final  Average  Grade
-------------------------------------------------------------
20180002         Lee Jieun     92     89     90.5     A
# S
```

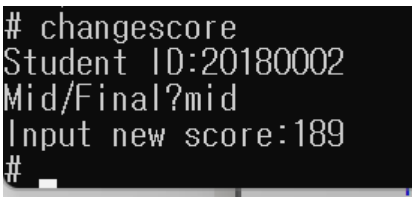
changescore

- id가 성적 목록에 없을 때

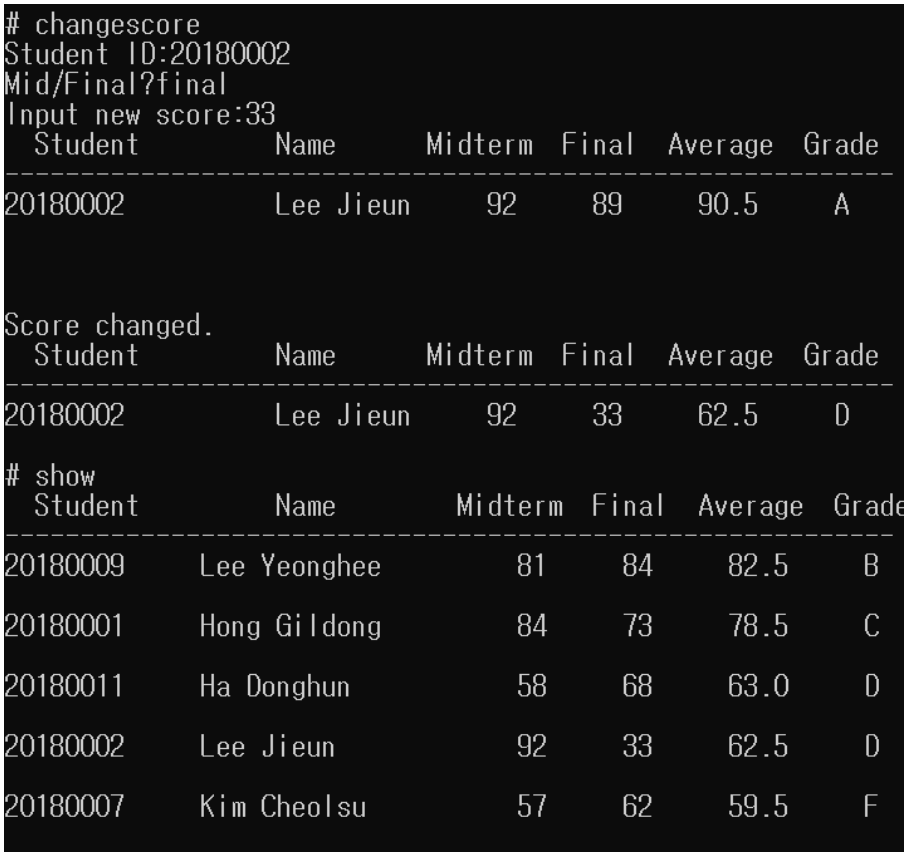
```
# changescore
Student ID:13
NO SUCH PERSON.
#
```

- mid/final 입력이 잘못 되었을 때

- 입력된 점수의 값이 0~100 이외일 때



- 제대로 입력되어 mean, grade가 바뀐 모습
  - 전체 성적 목록이 변경된 값에 따라 값, 정렬이 바뀐 모습



add

- id가 이미 성적 목록에 존재할 때
- id가 성적 목록에 존재하지 않을 때
- 전체 성적 목록에 새로운 학생이 추가된 모습

```
# add
Student ID:20180002
ALREADY EXISTS.
# add
Student ID:20180101
Name:Kim Haeun
Midterm Score:99
Final Score:88
Score added.
# show
  Student         Name        Midterm  Final  Average  Grade
------------------------------------------------------------
20180101     Kim Haeun            99     88     93.5     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

20180011     Ha Donghun           58     68     63.0     D

20180007     Kim Cheolsu          57     62     59.5     F
```

remove

- 정보 삭제
- id가 전체 학생 목록에 없을 때

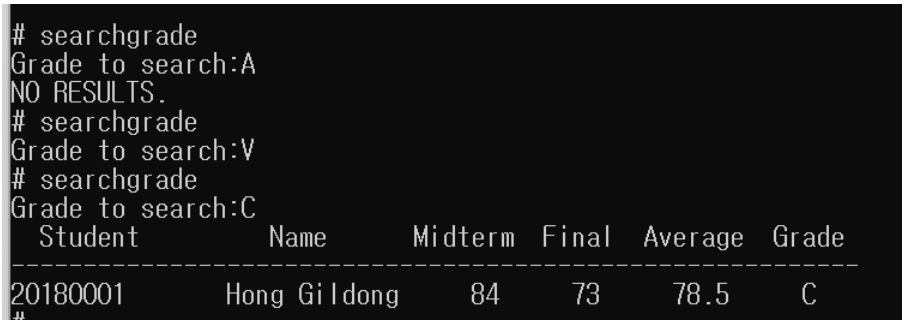
```
# remove
Student ID:2019
NO SUCH PERSON.
# remove
Student ID:20180101
Student removed.
# remove
Student ID:20180002
Student removed.
# show
  Student         Name        Midterm  Final  Average  Grade
------------------------------------------------------------
20180009     Lee Yeonghee         81     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
```

- 전체 성적 목록에 아무도 없을 때

```
# show
  Student         Name        Midterm  Final  Average  Grade
------------------------------------------------------------
# remove
List is empty
```

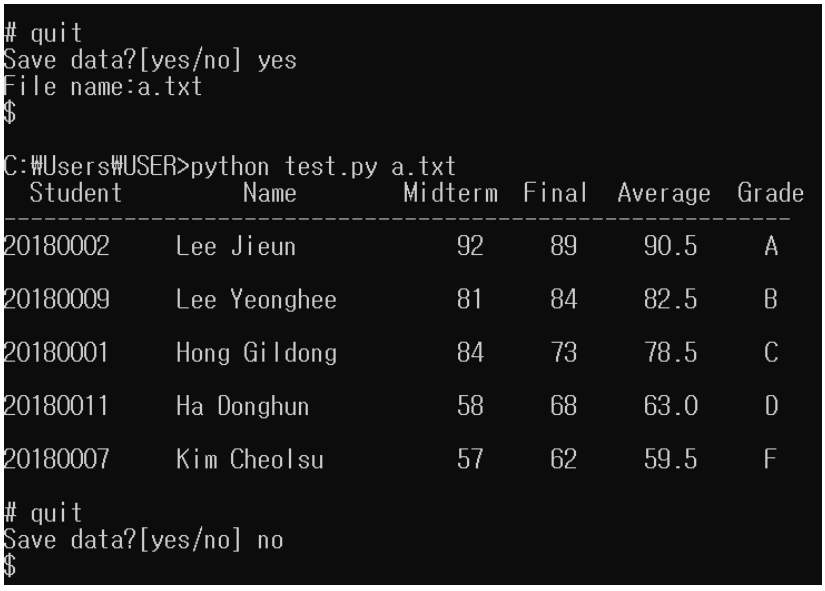
searchgrade

- 입력된 학점이 A, B, C, D, F 이외 값일 때
- 해당 학점에 속하는 학생이 없을 때
- 학점에 속하는 학생이 있을 때



quit

- 저장할 때
- 저장하지 않을 때



4. 토론

개발 중 문제

- **전체 성적 목록 출력 시 원본이 같이 변경되는 문제**
  - 시스템에 출력을 바르게 하기 위해 이름의 글자수를 일정하게 유지하기 위해 이름+(20-이름의 글자수)*'공백'으로 출력했다.
  - 이때 출력리스트=원본으로 하면 shallow copy되어 원본 자체의 이름이 바뀐다.
  - 해결 : deepcopy 이용

```
def show(stu_dict):
    # 기본으로 출력하기
    print('  Student         Name        Midterm  Final  Average  Grade    ')
    print('-------------------------------------------------------------')

    stu_dict = sorted(stu_dict.items(), key = lambda a:a[1][3], reverse=True)
    print_list = copy.deepcopy(stu_dict)
    for stu in print_list:
        print(stu[0], end='      ')
        if len(stu[1][0]) < 16:
            stu[1][0] = str(stu[1][0]+' '*(16-len(stu[1][0])))
        for num in stu[1]:
            print(num, end='      ')
        print('₩n')
```

- **불필요한 코드의 반복**
  - changescore
    - 다른 점수가 바뀌면 평균, 학점이 다시 바뀌게 된다.
      - 해결 : 평균, 학점 계산의 함수화

```
# mean, grade 구하기
def mean_grade(stu_dict):
    for stu in stu_dict:
        mean = round((int(stu_dict[stu][1])+int(stu_dict[stu][2]))/2, 1)
        if mean >= 90:
            grade = 'A'
        elif mean >= 80:
            grade = 'B'
        elif mean >= 70:
            grade = 'C'
        elif mean >= 60:
            grade = 'D'
        else:
            grade = 'F'
        stu_dict[stu].append(mean)
        stu_dict[stu].append(grade)
```

    - 바뀐 평균에 따라 전체 성적 목록의 정렬이 바뀌어 한다.
      - 해결 : show 함수 활용
  - add

        - 새로운 학생이 추가될 때 평균, 학점 계산은 앞서 만든 mean_grade함수를 이용해야한다. 하지만, 해당 함수는 딕셔너리 value에 값을 덧붙이는 형태여서 변경이 되지 않고 추가된다.
        - 해결 : 전체 성적 목록(딕셔너리)에서 모든 학생들의 평균, 학점을 지운 상태에서 mean_grade 함수를 수행한다.

```
for stu in stu_dict:
    stu_dict[stu] = stu_dict[stu][:-2]
```

    - 프로그램 시작시 입력 부분 코드와 명령어 함수의 불필요한 중복
        - 입력 부분 이전에 함수를 정의한 후 함수 이용

5. 결론

- 이 과제 이전엔 python을 터미널을 이용하여 구동한 경험이 없었다. 이번 과제를 통해 프로그램을 실제 이용 가능한 형태로 만들 수 있는 능력을 갖추게 되었다.
- pseudo code와 순서도를 통해 프로그램 하나의 구동 흐름을 가시적으로 표현할 수 있었고 어느 부분까지 하나의 함수, 즉 하나의 기능에 포함할 것인지 고민하는 시간을 가질 수 있었다. 뿐만 아니라 어떤 함수에서 전체 성적 목록(원본 데이터)를 변경하고 어떤 함수에서 변경하지 않을 지 판단할 수 있게 되었다.

개선점

- 각각의 기능을 함수 단위로 나누었지만 함수 간에 아직도 겹치는 코드들이 존재한다. 이는 기능 단위가 다소 컸던 것이라고 생각한다. 함수를 다시 분할하여 정비할 필요가 있다. 정비 후 코드가 더 효율적이고 가독성 높아질 것이라고 예상한다.