청년 AI • Big data 아카데미 20 기

AI 개인 프로젝트

<성적 처리 프로그램>

이름 : 오수민

분반 : B 반 1 조

이메일 : osoomin@naver.com

Problem : 성적 처리 프로그램

1. 문제의 개요

본 프로그램을 간략히 설명하면 다음과 같다.

- 사용자는 실행하고자 하는 학생의 성적이 입력된 텍스트 파일을 실행한다.

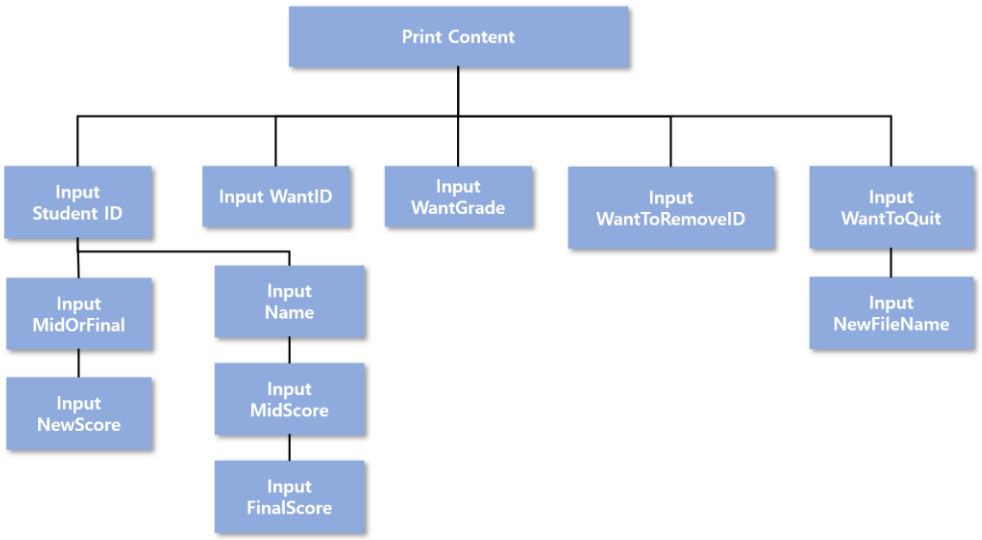
```
선택 명령 프롬프트
C:₩Users₩osoomin₩Desktop>python project.py students.txt
```

혹은 project.py 만 실행한다. (기본값인 students.txt 파일로 실행된다.)

```
명령 프롬프트
C:₩Users₩osoomin₩Desktop>python project.py
```

- 학생들의 ID, 이름, 중간 성적, 기말 성적, 평균, 학점을 출력한 뒤, 사용자로부터 사용자로부터 show, search, changescore, searchgrade, add, remove, quit, 총 7 개의 명령어를 입력 받아 함수마다의 기능을 수행한다.

이 때 입력되는 함수에 따라 구상가능한 구조는 아래와 같이 표현할 수 있다.



- 모든 명령어는 대소문자를 구별하지 않는다. (Show, SHOW, SHow, sHow, shoW 모두 동일하게 동작한다.)

- 7 개 이외의 명령어가 입력될 시, 에러 메시지 없이 다시 명령어를 입력 받도록 한다.

```
# look
#
```

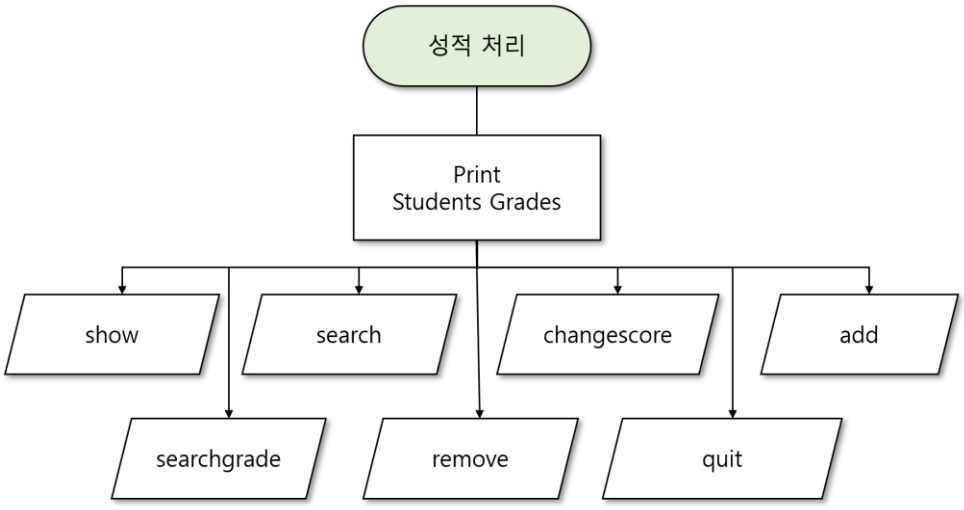
2. 알고리즘

본 프로그램 반복 실행을 위한 알고리즘을 코드 형태로 나타내면 다음과 같다.

Repeat algorithm for Run

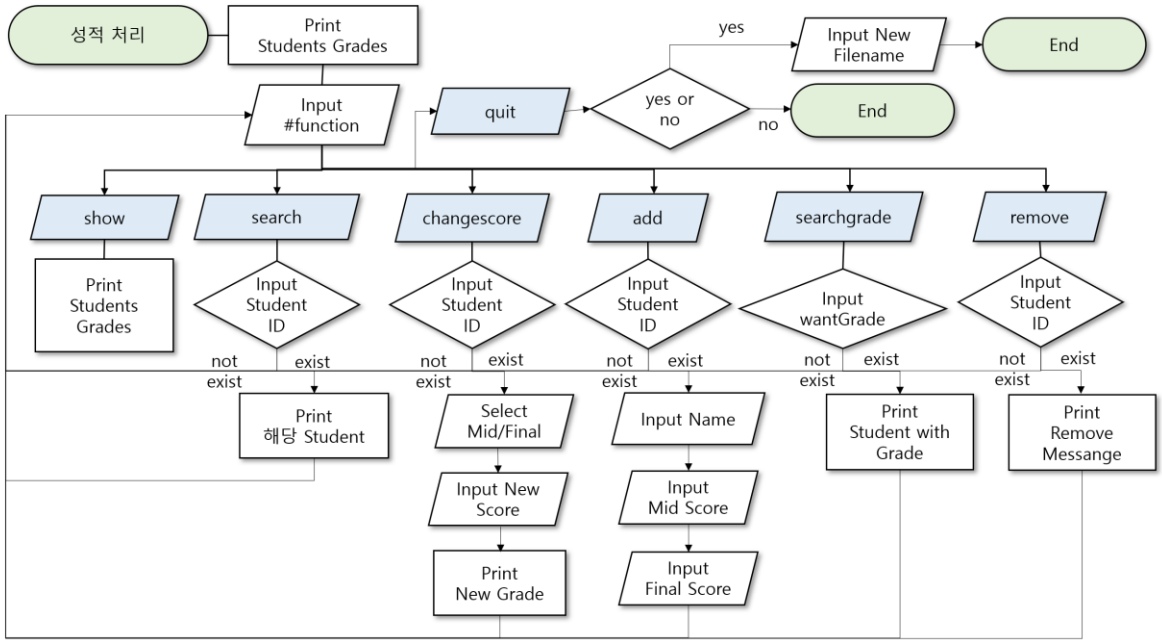
```
## 프로그램에 필요한 함수와 변수는 미리 정의해둔 것으로 가정한다.
while True:
    command = input("# ").lower()
    if command == "show":
        show_function(stu_list)
    elif command == "search":
        search_function()
    elif command == "changescore":
        changescore_function()
    elif command == "add":
        add_function()
    elif command == "searchgrade":
        searchgrade_function()
    elif command == "remove":
        remove_function()
    elif command == "quit":
        quit_function()
        break
    else:
        continue
```

위의 의사 알고리즘을 간단한 Flowchart 를 통해 표현하면 아래와 같다.



3. 프로그램 구조 및 설명

프로그램의 전체적인 구조는 아래의 플로우 차트(Flow Chart)와 같다.



a) 원하는 함수 입력

- 프로그램을 실행하면 학생들의 성적 목록이 출력 된 후에 명령어 입력을 대기하는 #표시가 뜨며, 입력을 요청한다.
- 7 가지 명령어 외의 명령어가 입력될 경우 무시하고 다시 명령어 입력을 대기한다.

b) show 함수

- show 입력 시, 저장되어 있는 전체 목록을 아래와 같이 평균 점수를 기준으로 내림차순으로 출력한다. 평균 점수는 소수점 이하 첫째 자리까지만 표시한다.

Run 'Show' Function

```
## 프로그램에 사용한 변수들은 미리 정의 및 선언해놓은 것으로 가정한다.
1 def show_function(stu_list):
2       stu_list.sort(key = lambda x:x[4], reverse=True)
3       print('  Student              Name    Midterm    Final    Average   Grade')
4       print('- ' * 30)
5       for i in stu_list:
6           print('   {:<8} {:>13}{:8}{:8}{:10.1f}     {:^8}'.format(i[0],i[1],i[2],i[3],i[4], i[5]))
```

c) search 함수

- search 입력 시, 검색하고자 하는 학생의 학번(ID)을 입력 받아 학번, 이름, 중간고사 점수, 기말고사 점수, 평균, 학점을 출력한다.
- 찾고자 하는 학생이 목록에 없는 경우에는 "NO SUCH PERSON."를 출력한다.

Run 'search' Function

```
## 프로그램에 사용한 변수들은 미리 정의 및 선언해놓은 것으로 가정한다.
def search_function():
    sl = []
    for i in range(len(stu_list)):
        sl.append(stu_list[i][0])
    # 학번 입력 받기
    stu_id = int(input("Student ID: "))
    if stu_id in sl :
        print('  Student              Name    Midterm    Final    Average   Grade')
        print('- ' * 30)
        i = stu_list[sl.index(stu_id)]
        print('   {:<8} {:>13}{:8}{:8}{:10.1f}     {:^8}'.format(i[0],i[1],i[2],i[3],i[4], i[5]))
    else:
        print("NO SUCH PERSON")
```

---

d) changescore 함수

- changescore 입력 시, 수정하고자 하는 학생의 학번(ID), 수정하고자 하는 점수가 중간고사인지 기말고사인지와 수정하고자 하는 점수(0-100 사이)를 순서대로 입력 받아 해당 학생의 점수를 수정한다.
- 목록에 저장된 학생의 중간고사(mid) 혹은 기말고사(final)의 점수를 수정한 뒤, 바뀐 점수에 따라 Grade 도 다시 계산하여 저장한다.

Run 'changescore' Function

```
## 프로그램에 사용한 변수들은 미리 정의 및 선언해놓은 것으로 가정한다.
def changescore_function():
    sl = []
    for i in range(len(stu_list)):
        sl.append(stu_list[i][0])
    # 학번 입력 받기
    stu_id = int(input("Student ID: "))
```

```
7       if stu_id in sl :
8           # 변경할 점수 고르기(중간/기말 중)
9           what_score = input("Mid/Final? ")
10           # 중간일 때
11           if what_score == "mid" :
12               how_much = int(input('Input new score: '))
13               if (how_much<0) | (how_much>100) :
14                   pass
15               else:
16                   print('   Student               Name     Midterm     Final     Average   Grade')
17                   print('- ' * 30)
18                   i = stu_list[sl.index(stu_id)]
19                   print('    {:<8} {:>13}{:8}{:8}{:10.1f}     {:^8}'.format(i[0],i[1],i[2],i[3],i[4], i[5]))
20                   print("Score changed.")
21                   # 업데이트
22                   stu_list[sl.index(stu_id)][2] = how_much
23                   stu_list[sl.index(stu_id)][4] = (how_much + stu_list[sl.index(stu_id)][3])/2
24                   stu_list[sl.index(stu_id)][5] = score(stu_list[sl.index(stu_id)][4])
25                   i = stu_list[sl.index(stu_id)]
26                   print('    {:<8} {:>13}{:8}{:8}{:10.1f}     {:^8}'.format(i[0],i[1],i[2],i[3],i[4], i[5]))
27           # 기말일 때
28           elif what_score == "final":
29               how_much = int(input('Input new score: '))
30               if (how_much<0) | (how_much>100) :
31                   pass
32               else:
33                   print('   Student               Name     Midterm     Final     Average   Grade')
34                   print('- ' * 30)
35                   i = stu_list[sl.index(stu_id)]
36                   print('    {:<8} {:>13}{:8}{:8}{:10.1f}     {:^8}'.format(i[0],i[1],i[2],i[3],i[4], i[5]))
37                   print("Score changed.")
38                   # 업데이트
39                   stu_list[sl.index(stu_id)][3] = how_much
40                   stu_list[sl.index(stu_id)][4] = (how_much + stu_list[sl.index(stu_id)][2])/2
41                   stu_list[sl.index(stu_id)][5] = score(stu_list[sl.index(stu_id)][4])
42                   i = stu_list[sl.index(stu_id)]
43                   print('    {:<8} {:>13}{:8}{:8}{:10.1f}     {:^8}'.format(i[0],i[1],i[2],i[3],i[4], i[5]))
44           else:
45               pass
46       else:
47           print("NO SUCH PERSON")
```

e) add 함수

- add 입력 시, 새 학생의 학번(ID), 이름, 중간고사 점수, 기말고사 점수를 차례로 입력 받는다. Average 와 Grade 는 중간고사와 기말고사 점수로 계산하며 새 학생이 추가되면, "Student added."를 출력한다.
- 이미 존재하는 학생의 학번을 입력하면 'ALREADY EXISTS.'를 출력한다.

Run 'add' Function

```
## 프로그램에 필요한 함수와 변수는 미리 정의해둔 것으로 가정한다.
def add_function():
    sl = []
    for i in range(len(stu_list)):
        sl.append(stu_list[i][0])
    # 학번 입력 받기
    stu_id = int(input("Student ID: "))
    if stu_id in sl :
        print("ALREADY EXISTS.")
    else:
        who_name = input("Name: ")
        mid_score = int(input("Midterm Score: "))
        if (mid_score<0) | (mid_score>100) :
            pass
        else :
            fin_score = int(input("Final Score: "))
            if (fin_score<0) | (fin_score>100) :
                pass
            else:
                total = (mid_score+fin_score)/2
                grade = score(total)
                name = [stu_id, who_name, mid_score, fin_score, total, grade]
                stu_list.append(name)
                print("Student added")
```

f) searchgrade 함수

- searchgrade 입력 시, 특정 grade 를 입력 받아 그 grade 에 해당하는 학생을 모두 출력한다.

- A, B, C, D, F 외의 값이 입력된 경우 실행되지 않고, 해당 grade 의 학생이 없는 경우 "NO RESULTS."를 출력한다.

Run 'searchgrade' Function

```
## 프로그램에 필요한 함수와 변수는 미리 정의해둔 것으로 가정한다.
def searchgrade_function():
    # 학점 입력받기
    stu_grade = input("Grade to search: ")
    # 가능한 학점
    list_grade = ['A','B','C','D','F']
    # 학생들의 학점
    gl = []
    for i in range(len(stu_list)):
        gl.append(stu_list[i][-1])
    if stu_grade not in list_grade :
        pass
    else:
        if stu_grade not in gl:
            print("NO RESULTS.")
        else:
            print('   Student              Name     Midterm     Final     Average    Grade')
            print('- ' * 30)
            for a in range(len(stu_list)):
                if stu_list[a][-1] == stu_grade:
                    i = stu_list[a]
                    print('    {:<8} {:>13}{:8}{:8}{:10.1f}      {:^8}'.format(i[0],i[1],i[2],i[3],i[4],i[5]))
```

g) remove 함수

- remove 입력 시, 삭제하고자 하는 학생의 학번을 입력 받은 후, 학생이 목록에 있는 경우 삭제한다. 삭제가 끝나면 "Student removed."를 출력한다.
- 학생이 목록에 없는 경우에는 "NO SUCH PERSON."를 출력한다.

Run 'remove' Function

```
## 프로그램에 필요한 함수와 변수는 미리 정의해둔 것으로 가정한다.
def remove_function():
    if len(stu_list) == 0:
        print("List is empty")
```

```
4       else:
5           sl = []
6           for i in range(len(stu_list)):
7               sl.append(stu_list[i][0])
8           # 학번 입력 받기
9           stu_id = int(input("Student ID: "))
10           if stu_id in sl :
11               del stu_list[sl.index(stu_id)]
12               print('Student removed.')
13           else:
14               print("NO SUCH PERSON")
```

h) quit 함수

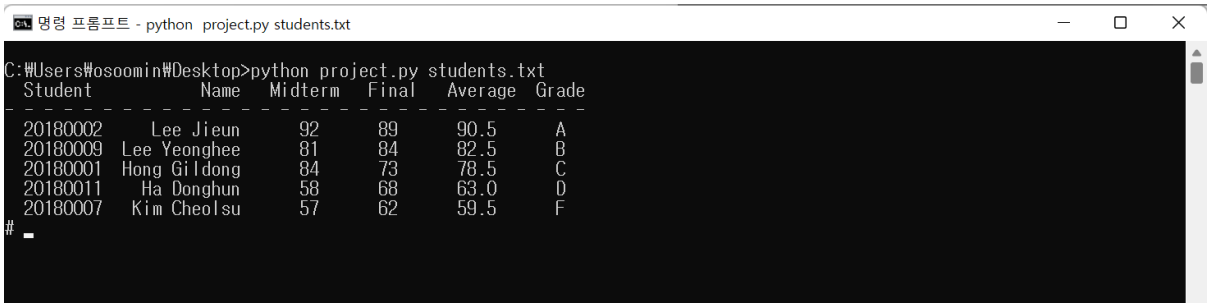
- quit 입력 시, 현재까지 편집한 내용의 저장 여부를 묻고 저장을 선택(yes 입력)할 경우 파일명을 입력 받아서 저장한 뒤, 프로그램을 종료한다.
- 저장할 때 목록의 순서는 평균을 기준으로 내림차순으로 한다.

Run 'quit' Function

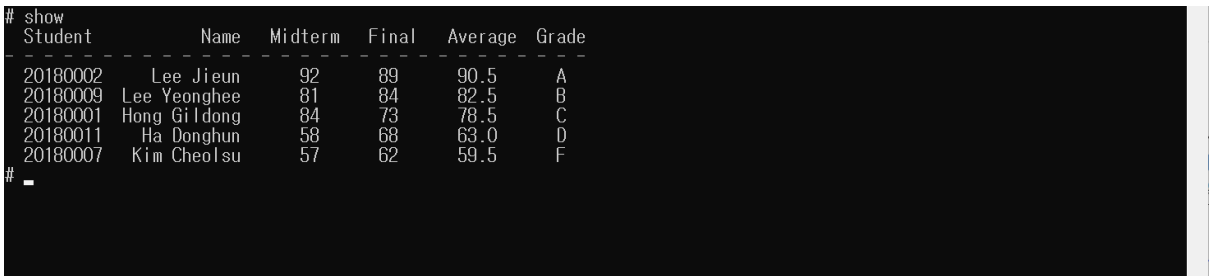
```
## 프로그램에 필요한 함수와 변수는 미리 정의해둔 것으로 가정한다.
1 def quit_function():
2       stu_list.sort(key = lambda x:x[4], reverse=True)
3       finish = input("Save data?[yes/no] ")
4       if finish == 'yes':
5           file_name = input("File name: ")
6           fw = open(file_name,'w')
7           for i in stu_list:
8               data = '{}₩t{}₩t{}₩t{}₩n'.format(i[0],i[1],i[2],i[3])
9               fw.write(data)
10          fw.close()
11      elif finish =="no":
12          pass
```

4. 프로그램 실행방법 및 예제

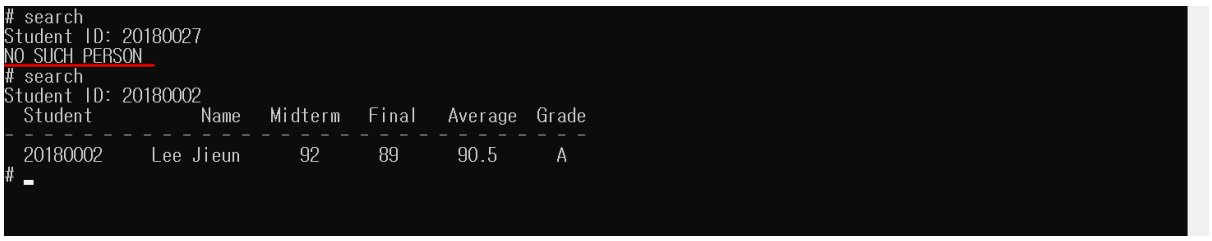
- 소스 코드가 있는 파일과 텍스트 파일이 있는 폴더로 이동한 뒤, 프로그램을 실행한다. 그럼 학생 목록이 출력됨을 확인할 수 있고 함수를 입력받는 # 표시가 뜬다.



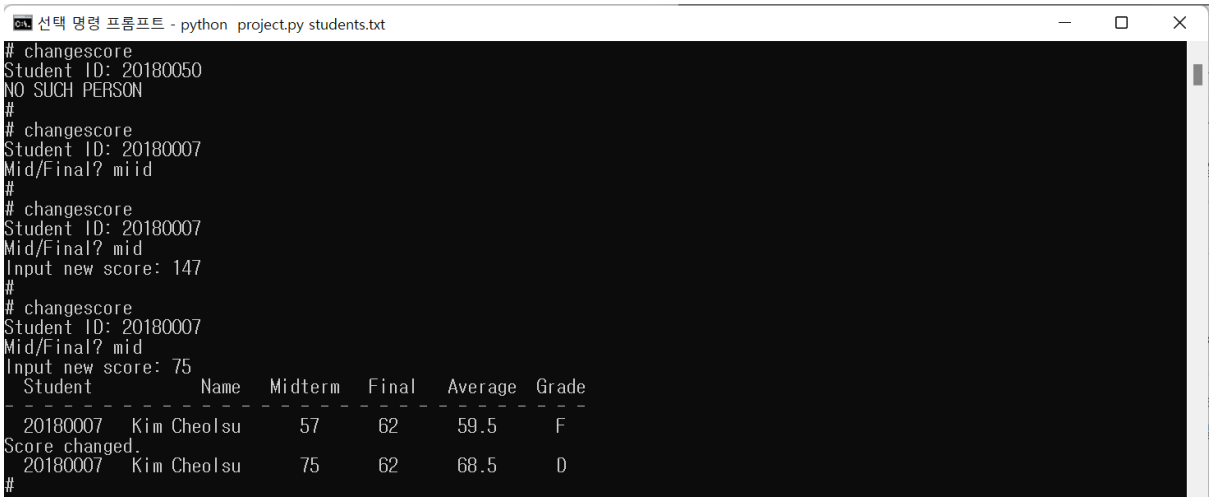
- Show 함수를 입력해 전체 학생 정보를 평균을 기준으로 내림차순으로 출력한다.

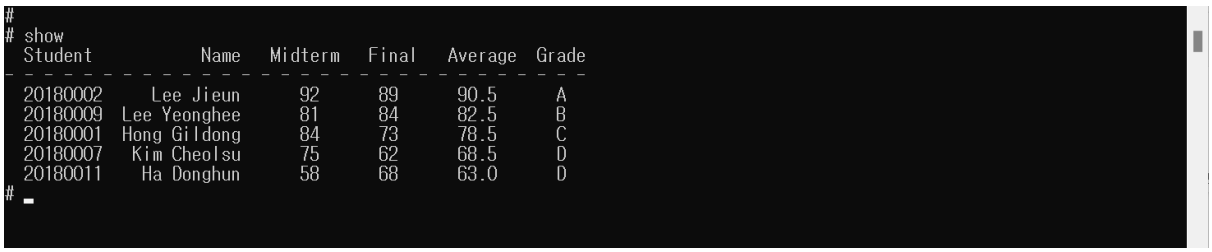


- Search 함수로 학생 학번을 입력하여 해당 학생의 이름, 점수, 성적을 출력한다.
- 이 때, 파일에 없는 학생을 입력하게 되면 예외 처리로 'NO SUCH PERSON'이 출력된다.

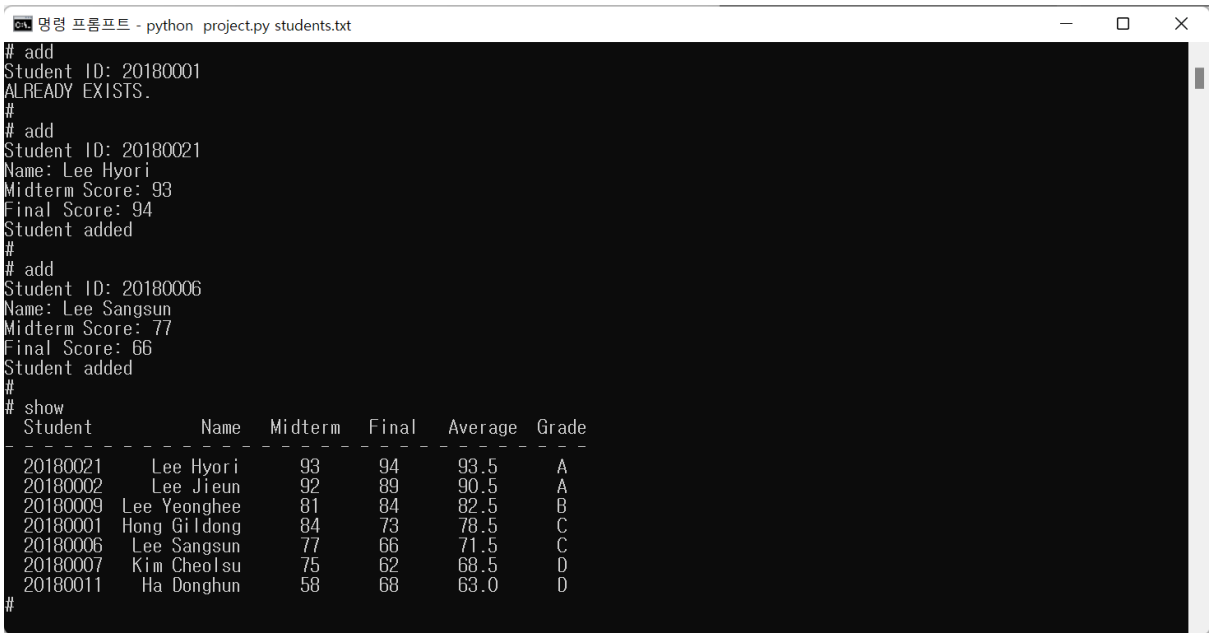


- Changescore 함수로 기존 존재하는 학생의 중간 혹은 기말 점수를 수정하여 평균과 성적까지 업데이트한 뒤 출력한다.
- 이 때, 없는 학생의 학번을 입력하게 되면 'NO SUCH PERSON'을 출력하며, 올바른 학번을 입력하더라도 이후의 단계에서 mid/final 이 아닌 문자를 입력하거나 0~100 사이가 아닌 수를 입력하면 초기화면으로 돌아가는 것을 볼 수 있다.

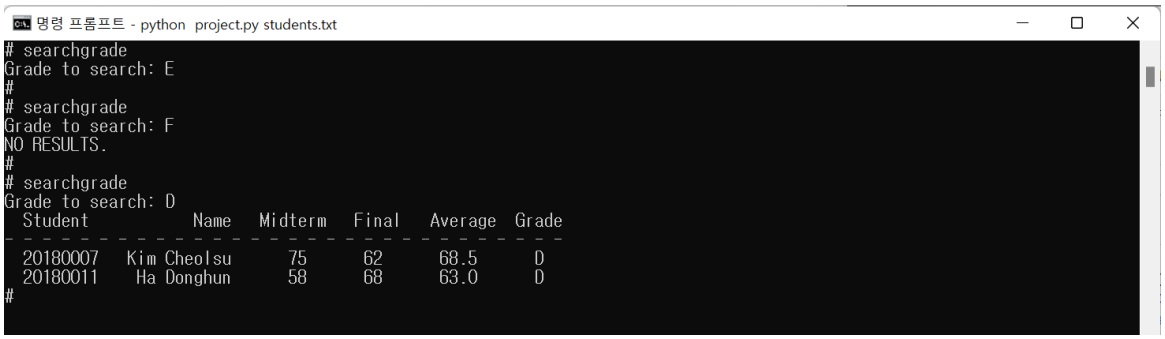




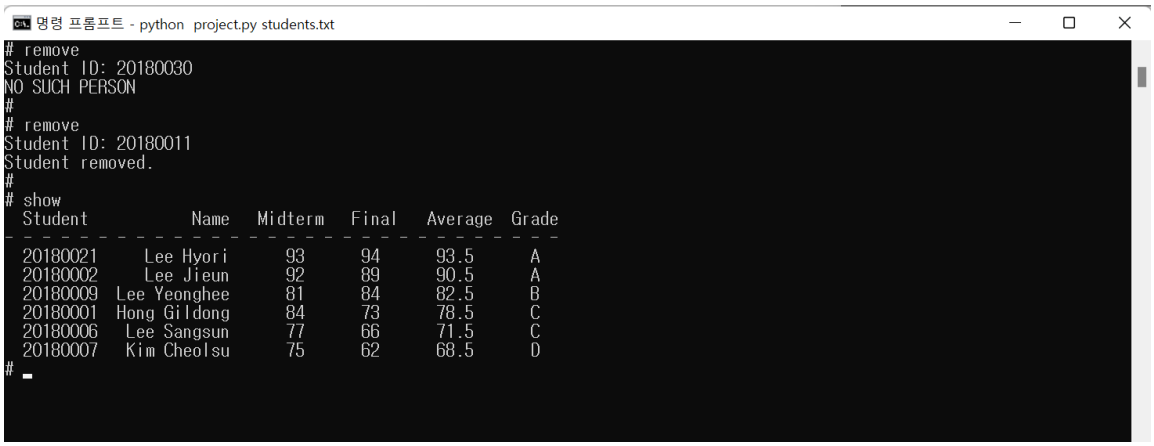
- Add 함수로 학번, 이름, 중간, 기말 점수를 입력 받아 학생을 추가하고 저장한다.
- 이 때, 이미 존재하는 학생의 학번을 입력하게 되면 'ALREADY EXISTS'을 출력하며 초기화면으로 돌아가는 것을 볼 수 있다.



- Searchgrade 함수로 찾고자 하는 성적을 입력하여 그 성적에 해당하는 학생을 모두 출력한다. 이 때, 입력되는 성적이 A, B, C, D, F 에 없다면 초기화면으로 돌아가는 것을 볼 수 있고, 입력되는 성적인 학생이 없다면 'NO RESULTS.'이 출력되는 것을 볼 수 있다.

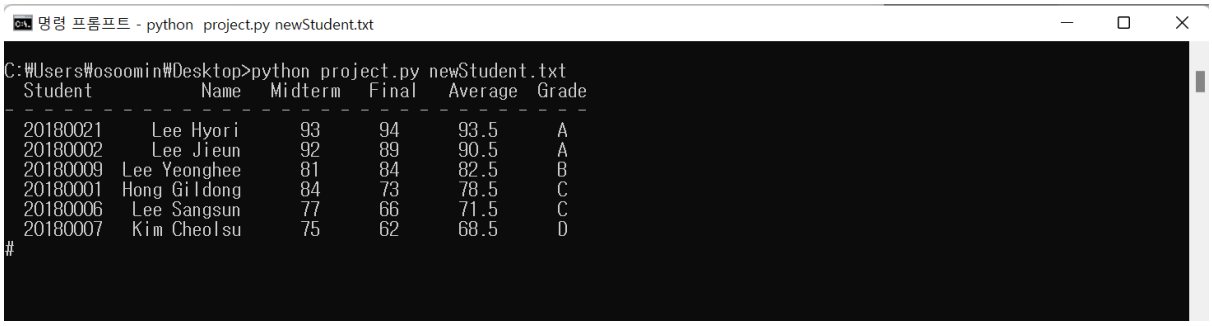


- Remove 함수로 지우고자 하는 학생의 학번을 입력하여 삭제한다. 이 때, 목록에 없는 학생을 입력하면 'NO SUCH PERSON'을 출력되는 것을 볼 수 있다.



- Quit 함수를 입력하면 수정된 데이터에 대해 저장여부를 물은 뒤 yes 를 입력하면 새 파일 이름 입력 후 프로그램을 종료한다. No 를 입력하면 파일을 저장하지 않고 프로그램을 종료한다.

- 생성된 새 텍스트 파일을 확인하면 다음과 같이 출력된다.



5. 토론

- Searchgrade 함수를 구현하면서 특정 성적에 해당하는 학생이 2 명 이상일 때 1 명만 출력되는 문제가 발생하였다.
- If 문과 list, index 를 사용하여 데이터를 추출하고 출력했는데, index 는 맨 앞의 데이터부터 읽기 때문에 오류가 발생했음을 알게 되었다.
- 예를 들어, A 를 입력했고 학생의 성적은 [A,A,B,C]라고 가정했을 때, 첫번째 A 는 index=0 으로 올바르게 출력되지만, 두번째 A 는 index=1 이 아닌 0 으로 출력된 것이다.
- 이는 for 문으로 stu_list 의 길이만큼 반복하는데, stu_list 에 있는 학생의 성적이 입력받은 성적과 같은 학생만을 고른 다음, 이를 한 줄씩 출력하도록 설정하여 해결하였다.

6. 결론

- 본 과제에서는 성적 처리 프로그램을 제작하여 파이썬 프로그램으로 텍스트 파일에 데이터를 업데이트하는 방법, 출력 형태를 다루는 방법을 익히는데 도움이 되었으며, 논리구조문을 직접 제작하는 과정을 통해 함수 정의 역량을 함양하게 되었다.

7. 개선 방향

- 본 과제는 요구사항만 적절히 만족시켜주면 오류가 발생하지 않기 때문에 프로그램 기능 자체의 개선보다는 사용자 입력 시의 주의사항 고지 추가를 통해 안정적인 사용이 가능할 것으로 예상한다.