HuStar 아카데미

Python 프로그래밍

1주차 프로젝트

담당교수 : 윤은영

이름 : 이재윤

Problem 1 : 파일로부터 데이터를 읽어서 성적 목록을 만들어 관리하는 성적 관리 프로그램을 작성한다.

1. 문제의 개요

A. 주의사항

- 사용자로부터 7개의 명령어(show, search, changescore, searchgrade, add, remove, quit)를 입력 받아 각 기능을 수항하게 된다. 최소한 각 명령어 별로 함수를 정의하여 사용한다. 명령어 외에 필요한 함수는 추가로 정의하여 사용할 수 있다.

- 소스코드 제출 파일 이름은 "project1.py"로 한다.

- 보고서 파일은 projoect1.doc(x) 로 한다.

- 기본적으로 문제에서 요구한 사항대로 구현을 하되, 실제로 프로그램을 작성한다 보면 결정해야 할 세부 사항이 많아진다. 명시되어 있지 않은 세부사항에 대해서 처리한 방법, 이유 등을 보고서에 기록하도록 한다.

B. 설명

- 프로그램 실행은 다음과 같이 한다. (Python project1.py students.txt)

- 파일명을 입력하지 않을 경우, default로 "students.txt"로부터 데이터를 읽는다

- 파일명이 입력될 경우, 입력된 파일로부터 데이터를 읽는다.

- 파일명에는 공백이 없다고 가정한다. (즉, 공백이 있는 파일명의 입력에 대해서는 고려하지 않는다.

- 프로그램 실행 시 텍스트 파일(로부터 학생들의 성적 목록 작성을 위한 데이터를 읽으며, 텍스트 파일의 내용 및 구성은 아래와 같다.

20180001	Hong Gildong	84	73
20180002	Lee Jieun	92	89
20180007	Kim Cheolsu	57	62
20180009	Lee Yeonghee	81	84
20180011	Ha Donghun	58	68

- 각 줄은 각 학생의 학번, 이름, 중간고사 점수, 기말고사 점수로 구성되어 있으며 각 항목 사이는 tab(₩t)으로 구분된다.

- 학생과 학생 사이는 줄 바꿈 문자(₩n)으로 구분된다.

Ex. [Student number][₩t][Name][₩t][Midterm][₩t][Final][₩n]

- 프로그램을 실행시키면 텍스트 파일로부터 데이터를 읽어 목록을 리스트(list) 자료형 또는 딕셔너리(dict) 자료형을 사용하여 저장하고, 전체 목록을 평균 점수를 기준으로 내림차순으로 정렬하여 아래의 예제처럼 출력한다. 동일한 평균 점수를 가진 학생들이 있는 경우 순서는 상관없다.

```
   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

#
```

- 평균(Average) 항목은 중간고사 점수와 기말고사 점수의 평균을 계산하여 저장한다.
- 학점(Grade)항목의 기준은 아래와 같다.

```
A: 평균이 90 점 이상
B: 평균이 80 점 이상, 90 점 미만
C: 평균이 70 점 이상, 80 점 미만
D: 평균이 60 점 이상, 70 점 미만
F: 평균이 60 점 미만
```

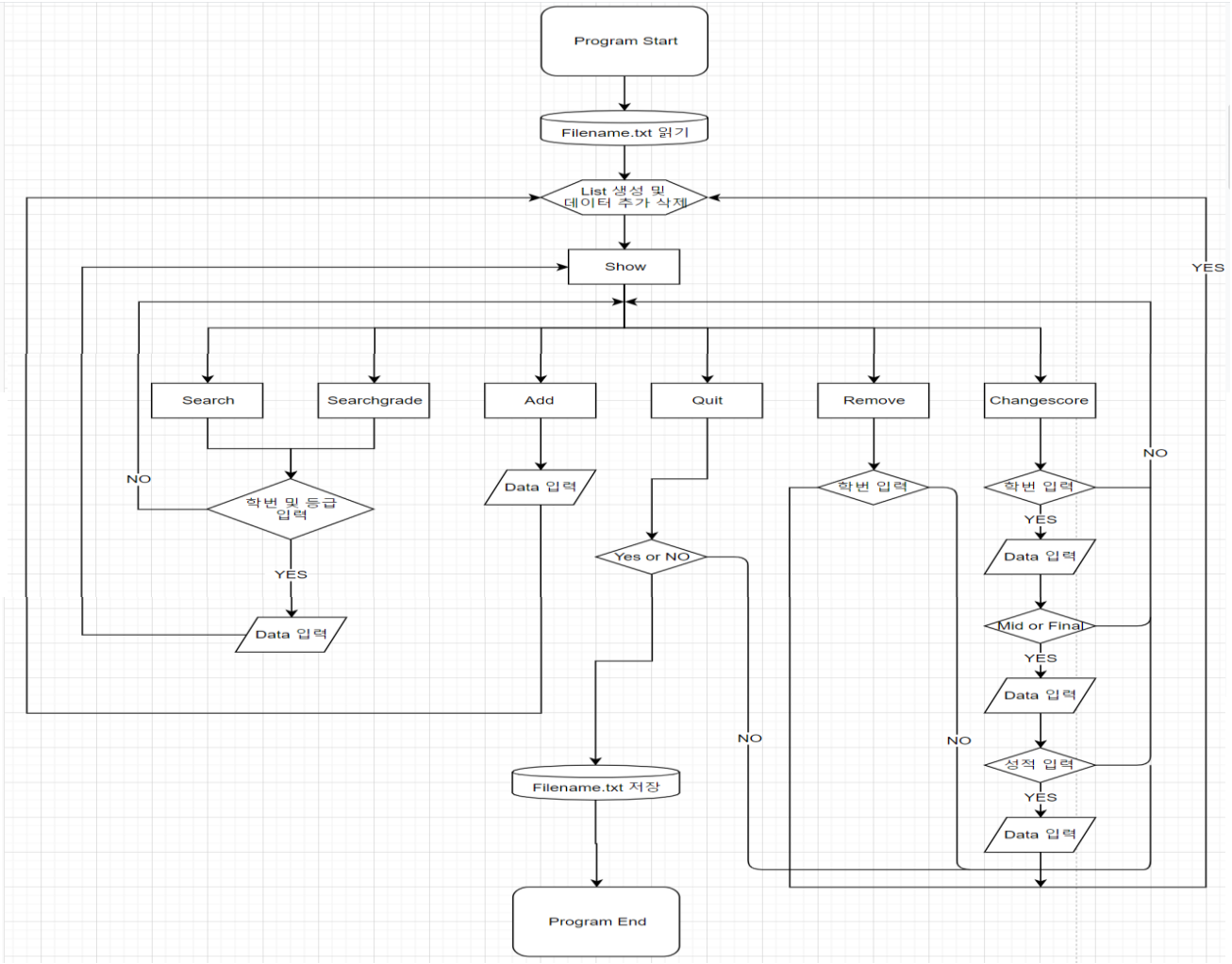
- 위와 같이 학생들의 성적 목록이 출력 된 후에는 명령어 입력을 대기하는 #표시가 뜨며, 이 상태에서 사용자는 명령어를 입력할 수 있다.

- 사용자는 7개의 명령어(show, search, changesocre, searchgrade, add, remove, quit)를 사용할 수 있으며, 명령어를 입력하였을 때만 기능이 실행된다. 이 명령어는 사용자가 명령어 입력 시, 대소문자를 구분하지 않고 동일한 명령어의 기능을 수행하도록 작성한다. 예를 들면, show, SHOW, Show, shoW 는 동일한 동작을 수행한다. 잘못된 명령어 입력 시, 에러 메시지 없이 아래와 같이 다시 명령어를 입력 받을 준비를 한다.

```
# find
#
```

2. 알고리즘

A. Flowchart



B. 설명

- 처음에 list를 하나 만들어 놓고, 그 list 안에 학생 성적 데이터를 학번당 dictionary구조로 집어 넣어 key 값을 학번, 이름, 점수 등으로 할 예정이다.

```
[{'ID': '20180001', 'Name': 'Hong Gildong', 'Midterm': '84', 'Final': '73', 'Average': 78.5, 'Grade': 'C'}
, {'ID': '20180002', 'Name': 'Lee Jieun', 'Midterm': '92', 'Final': '89', 'Average': 90.5, 'Grade': 'A'},
{'ID': '20180007', 'Name': 'Kim Cheolsu', 'Midterm': '57', 'Final': '62', 'Average': 59.5, 'Grade': 'F'},
{'ID': '20180009', 'Name': 'Lee Yeonghee', 'Midterm': '81', 'Final': '84', 'Average': 82.5, 'Grade': 'B'},
 {'ID': '20180011', 'Name': 'Ha Donghun', 'Midterm': '58', 'Final': '68', 'Average': 63.0, 'Grade': 'D'}]
```

- search, searchgrade, remove, changescore 같은 기능들은 데이터에 학번으로 접근을 해야 한다. 딕셔너리의 key값 'ID'가 학번이므로 해당하는 key 값을 불러와서 접근하는 방식으로 구현할 예정이다.
- add기능은 처음에 만들어 놓은 list에 접근하여 새로운 dictionary구조의 학생 성적 데이터를 추가시켜주는 방식을 구현할 예정이다.
- show 함수는 학생 성적 프로그램에서 잘 많이 쓰일 것 같아서 show함수를 다른 함수에서도 사용할 수 있게 구현할 예정이다.

3. 프로그램 구조 및 설명

A. openfile 함수 구현

Source-code

```
def openfile():
    file_name = 'students.txt'

    if len(sys.argv) > 1:
        file_name = sys.argv[1]
    print(file_name)
    f = open(file_name, 'r')
    lines = f.readlines()
    f.close()
    return lines
```

- default값으로 'students.txt'파일을 불러오게끔 했다. 조건문을 써서 input에 파일명이 입력되지 않으면, default값을 불러오고, 만약에 다른 파일명이 입력되면 그 파일명을 가진 파일을 읽어올 수 있게 하였다.

- 불러온 txt파일을 한 줄씩 불러와서 lines에 담고 return 하였다.

B. start 함수 구현

Source-code

```
def start(lines):
    for index, line in enumerate(lines):
        lines[index] = line.strip().split('\t')

    students = list()
    for i in lines:
        temp = dict()
        temp['ID'], temp['Name'], temp['Midterm'], temp['Final'] = i
        temp['Average'], temp['Grade'] = '', ''
        students.append(temp)
    for index, student in enumerate(students):
        student['Average'] = (int(student['Midterm']) + int(student['Final'])) / 2
        get_grade(students[index])
    return students
```

- return 값으로 받음 lines를 불러와 빈 list에 for 문으로 dictionary 구조를 만들고, keys값을 "ID", "Name", "Midterm", "Final", "Average", "Grade"로 만들어주고 기존에 있는 values값은 "ID", "Name", "Midterm", "Final" 밖에 없으므로 "Midterm"과 "Final"을 이용해서 "Average"와 "Grade" values 값을 구한다.

- "Average"와 "Grade" values 값은 "Average"는 ("Midterm" + "Final") / 2로 구하고, "Grade"는 get_grade 함수로 구현하였다.

C. get_grade 함수 구현

Source-code

```
def get_grade(para):
    if para['Average'] >= 90:
        para["Grade"] = "A"
    elif para['Average'] >= 80:
        para["Grade"] = "B"
    elif para['Average'] >= 70:
        para["Grade"] = "C"
    elif para['Average'] >= 60:
        para["Grade"] = "D"
    else:
        para["Grade"] = "F"
```

- key['Average']의 value 값이 90점 이상이면 "A", 80점 이상이면 "B", 70점 이상이면 "C", 60점 이상이면 "D", 60점 미만이면 "F"로 key['Grade"]의 value값에 넣어주었다.

D. show 함수 구현.

Source-code

```
def show(students):
    sorted_students = sorted(students, key=lambda x: x['Average'], reverse=True)
    student_list = list()
    print("Student\t\t\tName\t\tMidterm\tFinal\tAverage\tGrade")
    print('-----------------------------------------------------------------------')
    for student in sorted_students:
        temp = list(student.values())
        student_list.append(temp)
        data = '%s\t%15s\t\t%5s\t%4s\t%6.1f\t%3s\n' % (temp[0], temp[1], temp[2], temp[3], temp[4], temp[5])
        print(data)
```

- lambda 함수 중 reverse=True를 이용해서 dictionary의 keys 값 중 하나 'Average'를 기준으로 내림차순 정렬한다. for 문을 이용해서 dictionary의 values값을 불러온다.

- print 함수를 이용하여 dictionary의 keys값을 양식에 맞게끔 출력시키고, 평균 점수는 소수점 이하 첫째 자리 까지만 표시한다.

E. search 함수 구현.

Source-code

```
def search(students):
    stud_found = input('Student ID: ')
    found_or_not = 0
    return_stud = list()
    for i in students:
        if i['ID'] == stud_found:
            found_or_not = i
            break
    if not found_or_not:
        print('NO SUCH PERSON.')
        return
    return_stud.append(found_or_not)
    show(return_stud)
```

- students(데이터 수정 삭제가 바로바로 되는 list) list에 들어있는 dictionary에서 ["ID"] value값이 stud_found(input 값)과 같으면 해당되는 dictionary를 return_stud list에 담아 그 list를 show함수에 담아준다.

<예외처리> 찾고자 하는 학생이 목록에 없는 경우에는 "NO SUCH PERSON." 이라는 에러 메시지를 출력 : 만약에 처리하고자 하는 학생이 목록에 없으면 if not found_or_not 조건문에 들어가서 "NO SUCH PERSON." 출력이 된다.

F. changescore 함수 구현.

Source-code

```
def changescore(students):
    stud_change = input('Students ID: ')
    for index, i in enumerate(students):
        if i['ID'] not in stud_change:
            pass
        else:
            tmp_student = dict()
            for key in i:
                tmp_student[key] = i[key]
            stud_score = str(input('Mid or Final?'))
            if stud_score == 'mid' or stud_score == 'final':
                stud_new = input('Input new score :')
                if 0 < int(stud_new) <= 100:
                    if stud_score == 'mid':
                        students[index]['Midterm'] = int(stud_new)
                    else:
                        students[index]['Final'] = int(stud_new)
                    show([tmp_student])
                    print('\n<Score Changed>\n')
                    students[index]['Average'] = (int(students[index]['Final']) + int(students[index]['Midterm'])) / 2
                    get_grade(students[index])
                    data = '%s\t%15s\t\t%5s\t%4s\t%6.1f\t%3s\n' % (
                    students[index]['ID'], students[index]['Name'], students[index]['Midterm'],
                    students[index]['Final'], students[index]['Average'], students[index]['Grade'])
                    print(data)
```

```
                    return
                else:
                    return
            else:
                return
    print('NO SUCH PERSON')
```

- students(데이터 수정 삭제가 바로바로 되는 list) list에 들어있는 dictionary에서 ["ID"] value값이 stud_change(input 값)와 다르면 pass, 들어있으면 dictionary를 temp_stud list에 담아 그 list를 show함수에 담아준다.(점수가 바뀌기 전 데이터를 보여줘야 함) 그리고 해당되는 "ID"의 dictionary에 접근하여 mid or final이라는 string 구조의 input 값이 들어가게 되면 다음 조건문에 들어가서 int형태의 stud_new라는 변수에 바꾸는 mid 혹은 final 값을 저장하고 해당되는 "Midterm" 이나 "Final"의 key값에 접근하여 해당되는 value값을 stud_new로 바꾼다. 그리고 바꾼 데이터를 print해준다.

<예외처리> 학번이 목록에 없는 경우에는 "NO SUCH PERSON."이라는 에러 메시지를 출력 : 가장 위에 있는 조건문 i['ID'] not in stud_change 에 들어가면 pass가 되어 조건문을 빠져나오므로 조건문 가장 밖에 있는 "NO SUCH PERSON." 출력이 된다.

"mid" 또는 "final" 외의 값이 입력된 경우에는 실행되지 않음 : if stud_score == 'mid' or stud_score == 'final' 조건문의 else: 를 return 해줌으로써 stud_score에 'mid'나 'final'이 들어가지 않으면 else에 들어가 return이 된다.

점수에 0~100 외의 값이 입력된 경우에는 실행되지 않음 : if 0 < int(stud_new) <= 100: 조건문에 else: 를 return 해줌으로써 stud_new 의 값이 0 이상 100 이하면 else 에 들어가 return 이 된다.

G. add 함수 구현

Source-code

```
def add(students):
    stu_add = input('Sudent ID : ')
    for i in students:
        if stu_add == i['ID']:
            print('ALREADY EXISTS')
            return
    stu_temp = dict()
    stu_temp['ID'] = stu_add
    stu_temp['Name'] = input('Name :')
    mid_score = input('Miderm Score :')
    final_score = input('Final Score :')
    if 0 <= int(mid_score) and int(final_score) <= 100 :
        stu_temp['Midterm'] = mid_score
        stu_temp['Final'] = final_score
```

```
    else:
        print("점수는 0 ~ 100 외의 값은 입력될 수 없습니다.")
        return
    stu_temp['Average'] = (int(stu_temp['Final']) + int(stu_temp['Midterm'])) / 2
    get_grade(stu_temp)
    students.append(stu_temp)
    print('\n<Student Added>\n')
    print("Student\t\t\tName\t\tMidterm\tFinal\tAverage\tGrade")
    print('-------------------------------------------------------------------------')
    data = '%s\t%15s\t\t%5s\t%4s\t%6.1f\t%3s\n' % (
        stu_temp['ID'], stu_temp['Name'], stu_temp['Midterm'],
        stu_temp['Final'], stu_temp['Average'], stu_temp['Grade'])
    print(data)
```

- students(데이터 수정 삭제가 바로바로 되는 list) list에 들어있는 dictionary에서 ["ID"] value값이 stud_add(input 값)와 같으면 return 들어 있지 않으면 stu_temp 라는 dictionary에 keys값은 "ID", "Name", "Midterm", "Final", "Average", "Grade"로 만들어주고 새로운 values 들을 input 해주고 "Average"의 value는 새로 구해주고 "Grade"의 value는 get_grade함수에 넣어 구해준다. 그리고 형식에 맞게 print 해준다.

<예외처리> 목록에 있는 학생의 학번을 입력 시, 'ALREADY EXISTS.' 이라는 에러 메시지 출력 : if stu_add == i['ID']: 조건문에 만족하면 "ALREADY EXISTS"가 출력이 된다.

점수에 0~100외의 값이 입력될 수 없음 : "점수 0~100 외의 값은 입력될 수 없습니다."

H. searchgrade 함수 구현

Source-code

```
def searchgrade(students):
    stud_sg = input('searchgrade: ')
    found_or_not = 0
    Grades = ('A','B','C','D','F')
    return_stud = list()
    for i in students:
        if stud_sg not in Grades:
            return
        if i['Grade'] in stud_sg:
            print('Grade to search: %s'%i['Grade'])
            if not isinstance(found_or_not, list):
                found_or_not = list()
            found_or_not.append(i)
    if not found_or_not:
        print('NO RESULTS.')
        return
    return_stud.append(found_or_not)
    show(found_or_not)
```

- Grades 변수에 string 타입의 'A', 'B', 'C', 'D', 'F'를 만들어주고 stud_sg(input 값)에 들어있으면 return_stud list에 stud_sg에 해당되는 ['Grade']의 value를 가진 dictionary를 불러와서 넣어주고 show해준다.

<예외처리> A, B, C, D, F 외의 값이 입력된 경우 실행되지 않음 : stud_sg가 Grades변수에 들어있지 않으면 return이 되어 조건문을 빠져나감

해당 grade의 학생이 없는 경우 아래와 같이 메시지 "NO RESULTS." 출력 : stud_sg가 Grades 변수에는 들어있지만, 해당되는 학생이 없는 경우는 if no found_or_not: 조건문에 들어가서 'NO RESULTS."가 출력이 됨

I. remove 함수 구현

Source-code

```
def remove(students):
    found_or_not = 0
    stud_remove = input('Students ID: ')
    if len(students) == 0:
        print('List is empty')
        return
    for index, i in enumerate(students):
        if i['ID'] not in stud_remove:
            pass
        elif i['ID'] in stud_remove:
            del students[index]
            print('%s Student removed' %i['ID'])
            found_or_not = i
    if not found_or_not:
        print('NO SUCH PERSON')
```

- students(데이터 수정 삭제가 바로바로 되는 list) list에 들어있는 dictionary에서 ["ID"] value값이 stud_remove(input 값)와 같으면 del 함수를 이용해서 해당되는 ['ID']의 dictionary를 삭제한다.

<예외처리> 목록에 아무도 없을 경우 아래의 예제와 같이 "List is empty." 메시지 출력 : students(데이터 수정 삭제가 바로바로 되는 list) list의 길이가 0이면 "List is empty" 출력

학생이 목록에 없는 경우에는 "NO SUCH PERSON."이라는 에러 메시지를 출력 : students(데이터 수정 삭제가 바로바로 되는 list) list에 해당되는 ["ID"]의 key값이 없는 경우는 if i['ID'] not in stud_remove: 조건문에 들어가 pass하게 되어 if not found_or_not: 조건문에 들어가 "NO SUCH PERSON"이 출력된다.

J. quit 함수 구현

Source-code

```
def quit(students):
    while True:
        stud_quit = input('Save data?[yes/no]')
        if stud_quit == 'no':
            print("")
            return
        elif stud_quit == 'yes':
            break
    stud_file = input('File name :')
    students.sort(key=lambda x:x['Average'], reverse=True)
    data = list()
    for x in students:
        data.append(list(x.values())[:-2])
    f = open(stud_file, 'w')
    for i in data:
        a='\t'.join(i)
        a += '\n'
        f.write(a)
    f.close
    exit()
```

- stud_quit(input 값)이 "no"를 입력하게 되면 return으로 다시 프로그램으로 돌아가고, "yes"를 입력하면 break로 조건문을 빠져나와 stud_file(File name)입력을 하게 된다.

- 저장할 때 목록의 순서는 평균을 기준으로 내림차순 해야하고, 불러온 파일과 동일하게 "ID", "Name", "Midterm", "Final", 의 values값만 가져야 하므로, data라는 이름을 가진 빈 list를 만들어 주어 dictionary의 items 중 "ID", "Name", "Midterm", "Final"의 keys값의 values를 담아준다.

- data가 담겨진 list를 for문을 이용해서 한 줄씩 뽑아내어 stud_file(File name)의 이름을 가진 파일에 넣어주고, 프로그램을 종료 시켜준다.

K. how 함수 구현

Source-code

```
def how():
    print('==================')
    print('    <Command>\n')
    command = ['search', 'show', 'changescore', 'changename', 'searchgrade', 'add', 'remove', 'quit']
    for i,j in enumerate(command):
        print('%d. %s'%(i+1,j))
```

- Terminal 창에서 입력되지 않은 명령어를 칠 경우 혹은 내가 무슨 명령어가 있는지 알고 싶은 경우 '?'를 input하면 내가 사용할 수 있는 명령어를 볼 수 있다.

L. changename 함수 구현

Source-code

```
def changename(students):
    stud_changename = input('Students ID: ')
    for index, i in enumerate(students):
        if i['ID'] not in stud_changename:
            pass
        else:
            tmp_student = dict()
            for key in i:
                tmp_student[key] = i[key]
            stud_name = str(input("How would you change the student's name?"))
            students[index]['Name'] = stud_name
            print('\n<Name Changed>\n')
            show([tmp_student])
            data = '%s\t%15s\t\t%5s\t%4s\t%6.1f\t%3s\n' % (
                students[index]['ID'], students[index]['Name'], students[index]['Midterm'],
                students[index]['Final'], students[index]['Average'], students[index]['Grade'])
            print(data)
            return
    print('NO SUCH PERSON')
```

- students(데이터 수정 삭제가 바로바로 되는 list) list에 들어있는 dictionary에서 ["ID"] value값이 stud_change(input 값)와 다르면 pass해서 'NO SUCH PERSON'을 출력해주고, 들어있으면 dictionary를 temp_stud list₩에 담아 그 list를 show함수에 담아준다.(점수가 바뀌기 전 데이터를 보여줘야 함) 그리고 해당되는 "ID"의 dictionary에 접근하여 stud_name이라는 string 구조의 input 값이 들어가게 되면 다음 조건문에 들어가서 int형태의 stud_name라는 변수에 값을 저장하고 해당되는 "Name"의 key값에 접근하여 해당되는 value값을 stud_name로 바꾼다. 그리고 바꾼 데이터를 print해준다.

M. loop 함수 구현

Source-code

```
def loop(students):
    show(students)
    while True:
        user_input = input('#').lower()
        if user_input == 'search':
            search(students)
        elif user_input == 'show':
            show(students)
        elif user_input == 'changescore':
            changescore(students)
        elif user_input == 'searchgrade':
            searchgrade(students)
        elif user_input == 'changename':
            changename(students)
        elif user_input == 'add':
            add(students)
        elif user_input == 'remove':
            remove(students)
        elif user_input == 'quit':
            quit(students)
        elif user_input == 'how':
            how()
        else:
            print('명령어가 잘못 되었습니다')
            how()
            pass
```

- 프로그램 실행 후 바로 내가 불러온 파일의 데이터를 양식에 맞춰서 보여줘야 하므로 show함수에 students(초기 학생 데이터 상태)list를 넣어 불러오게 끔 구현하였다.

- 프로그램을 실행한 후 Terminal 명령어 입력 창에서 '#'을 보여준 뒤 명령어를 입력하게 끔 해준다. 그리고 명령어는 lower()함수를 이용하여 대소문자 구분 없이 단어만 똑같으면 같은 명령어로 인식하게끔 해준다. 만약 내가 설정한 명령어 외에 다른 명령어가 입력될 시 위에 만들어 놓은 how함수로 들어가게끔 되어 내가 사용할 수 있는 명령어를 볼 수 있게끔 해주었다.

N. Main 함수 구현

Source-code

```
if __name__ == '__main__':
    opens = openfile()
    students = start(opens)
    loop(students)
```

- project1.py 프로그램이 실행되면 main함수가 실행되는데 첨에는 openfile함수를 불러와 파일을 불러오게끔 하였다.

- openfile함수의 return 값인 lines(데이터 파일의 학생 성적 데이터를 한 줄씩 불러와 담겨져 있는 것)을 start 함수에 넣어 사용자가 원하는 양식으로 바꾸어서 list에 담아준다.

- loop함수에 start함수의 students list를 넣어 loop함수에 들어있는 내가 원하는 명령어 함수 search, searchgrade, remove, changescore, add, show, quit 등을 실행 할 수 있게끔 해준다.

4. 프로그램 실행방법 및 예제

A. 프로그램실행

Python project1.py students.txt [파일명(기본값:students.txt)을 입력할 경우]

```
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트 master python project1.py students.txt
students.txt
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

#
```

Python project1.py [파일명을 입력하지 않을 경우]

```
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트 master python project1.py
students.txt
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

#
```

Python project1.py mystudents.txt [다른 파일명을 입력할 경우]

```
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트 master python project1.py mystudents.txt
mystudents.txt
Student                 Name            Midterm Final   Average Grade
-------------------------------------------------------------------
20190003         Jang Yunjeong            100    100      100.0    A

20190002              Kim Minsu             40     80       60.0    D

20190001            Lee Jaeyoon             55     55       55.0    F

20190005             Ha Dongjun             22     22       22.0    F

20190004             Lee Saeyun              0      0        0.0    F

#
```

B. show 입력

show [평균 점수를 기준으로 내림차순 출력, 평균점수는 소수점 첫째 자리]

```
#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

#
```

C. search 입력

search [학번이 목록에 있는 경우]

```
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트  master python project1.py students.txt
students.txt
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
Student ID: 20180002
Student                 Name           Midterm Final   Average Grade
------------------------------------------------------------------
20180002              Lee Jieun            92     89      90.5    A

#
```

search [학번이 목록에 없는 경우 "NO SUCH PERSON"]

```
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트  master python project1.py students.txt
students.txt
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
Student ID: 1
NO SUCH PERSON.
#
```

D. changescore 입력

changescore [학생의 학번이 목록에 있으면 해당 점수를 바꿈]

```
#changescore
Students ID: 20180001
Mid or Final?final
Input new score :100
Student                 Name            Midterm Final   Average Grade
------------------------------------------------------------------------
20180001            Hong Gildong            84     73       78.5    C


<Score Changed>

20180001            Hong Gildong            84    100       92.0    A

#
#show
Student                 Name            Midterm Final   Average Grade
------------------------------------------------------------------------
20180001            Hong Gildong            84    100       92.0    A

20180002               Lee Jieun            92     89       90.5    A

20180009            Lee Yeonghee            81     84       82.5    B

20180011              Ha Donghun            58     68       63.0    D

20180007             Kim Cheolsu            57     62       59.5    F

#
```

changescore [학생의 학번이 목록에 없는 경우 "NO SUCH PERSON"]

```
 pirl@pirl-Precision-7920-Tower  ~/huster_python/프로젝트  master  python project1.py students.txt
students.txt
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

#changescore
Students ID: 1
NO SUCH PERSON
#
```

changescore ["mid" or "final 외의 값 입력된 경우 실행되지 않음]

```
 pirl@pirl-Precision-7920-Tower  ~/huster_python/프로젝트  master  python project1.py students.txt
students.txt
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

#changescore
Students ID: 20180001
Mid or Final?a
#
```

changescore [점수가 0~100외의 값 입력된 경우 실행되지 않음]

```
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트 master python project1.py students.txt
students.txt
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

# ChaNGEScore
Students ID: 20180001
Mid or Final?mid
Input new score :200
#
```

E. add 입력

add [학생의 학번, 중간고사, 기말고사 점수를 입력]

```
#ADD
Sudent ID : 20190001
Name :Lee Jaeyoon
Miderm Score :80
Final Score :65

<Student Added>

Student                 Name            Midterm Final   Average Grade
-------------------------------------------------------------------
20190001           Lee Jaeyoon           80      65      72.5    C

#
#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

20190001            Lee Jaeyoon           80      65      72.5    C

20180011             Ha Donghun           58      68      63.0    D

20180007            Kim Cheolsu           57      62      59.5    F

#
```

add [학생의 학번이 목록에 있는 경우 "ALREADY EXISTS"]

```
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트 master python project1.py students.txt
students.txt
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

#add
Sudent ID : 20180001
ALREADY EXISTS
#
```

F. searchgrade 입력

searchgrade ['A' 인 학생을 하나 추가하고 기존에 있던 'A'와 같이 2명 출력]

```
#add
Sudent ID : 201900001
Name :Lee Jaeyoon
Miderm Score :100
Final Score :100

<Student Added>

Student                 Name            Midterm Final   Average Grade
------------------------------------------------------------------------
201900001           Lee Jaeyoon           100    100     100.0    A

#searchgrade
searchgrade: A
Grade to search: A
Grade to search: A
Student                 Name            Midterm Final   Average Grade
------------------------------------------------------------------------
201900001           Lee Jaeyoon           100    100     100.0    A

20180002              Lee Jieun            92     89      90.5    A

#
```

searchgrade ['A'인 학생을 하나 추가하고 기존에 있던 'A'와 같이 2명 출력]

```
#add
Sudent ID : 201900001
Name :Lee Jaeyoon
Miderm Score :100
Final Score :100

<Student Added>

Student                 Name            Midterm Final   Average Grade
------------------------------------------------------------------------
201900001           Lee Jaeyoon           100    100     100.0    A

#searchgrade
searchgrade: A
Grade to search: A
Grade to search: A
Student                 Name            Midterm Final   Average Grade
------------------------------------------------------------------------
201900001           Lee Jaeyoon           100    100     100.0    A

20180002              Lee Jieun            92     89      90.5    A

#
```

searchgrade [기존에 'F'인 학생을 제거하고 'F' 등급 출력 "NO RESULTS"]

```
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트 master python project1.py students.txt
students.txt
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

#remove
Students ID: 20180007
20180007 Student removed
#searchgrade
searchgrade: F
NO RESULTS.
#
```

searchgrade [A,B,C,D,F 외의 등급을 입력할 경우 실행되지 않음]

```
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트 master python project1.py students.txt
students.txt
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

#searchGrade
searchgrade: E
#
```

G. remove 입력

remove [목록에 있는 학생 학번을 입력 받은 후 목록에 있으면 삭제]

```
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트 master python project1.py students.txt
students.txt
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

# Remove
Students ID: 20180002
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

#
```

remove [학생의 학번이 목록에 없는 경우 "NO SUCH PERSON"]

```
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트 master python project1.py students.txt
students.txt
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

# remove
Students ID: 1
NO SUCH PERSON.
#
```

remove [학생의 학번이 목록에 없는 경우 "NO SUCH PERSON"]

```
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트 master python project1.py students.txt
students.txt
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

# remove
Students ID: 1
NO SUCH PERSON.
#
```

remove [목록에 아무도 없을 경우 "List is empty"]

```
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트 master python project1.py students.txt
students.txt
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

#remove
Students ID: 20180001
20180001 Student removed
#remove
Students ID: 20180002
20180002 Student removed
#remove
Students ID: 20180009
20180009 Student removed
#remove
Students ID: 20180007
20180007 Student removed
#remove
Students ID: 20180011
20180011 Student removed
#remove
List is empty.
#show
Student                 Name            Midterm Final   Average Grade
-------------------------------------------------------------------
#
```

H. changename 입력

changename [학생의 학번이 목록에 있으면 이름을 바꿈]

```
#changename
Students ID: 20180002
How would you change the student's name?Lee Jaeyoon

<Name Changed>

Student                 Name            Midterm Final   Average Grade
-------------------------------------------------------------------
20180002              Lee Jieun            92     89      90.5    A

20180002            Lee Jaeyoon            92     89      90.5    A

#
```

changename [학생의 학번이 목록에 없는 경우 "NO SUCH PERSON"]

```
#changename
Students ID: 20180002
How would you change the student's name?Lee Jaeyoon

<Name Changed>

Student                 Name            Midterm Final   Average Grade
-------------------------------------------------------------------
20180002              Lee Jieun            92     89      90.5    A

20180002            Lee Jaeyoon            92     89      90.5    A

#
```

I. '?' 입력 or 등록되지 않은 명령어 입력

'?', 등록되지 않은 명령어 입력

```
#?
==================
    <Command>

1. search
2. show
3. changescore
4. changename
5. searchgrade
6. add
7. remove
8. quit
#searchh
명령어가 잘못 되었습니다
==================
    <Command>

1. search
2. show
3. changescore
4. changename
5. searchgrade
6. add
7. remove
8. quit
#
```

H. quit 입력

quit ['yes' 입력 시 저장파일명을 입력 받은 후 파일명으로 저장]

```
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트 master python project1.py students.txt
students.txt
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

#quit
Save data?[yes/no]yes
File name :newStudents.txt
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트 master ls
파이썬_1주차_과제_1227.pdf  newStudents.txt  report_ex1.docx  report_ex2.pdf
mystudents.txt              project1.py      report_ex1.pdf   students.txt
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트 master
```

Open ▾ | newStudents.txt (~/huster_python/프로젝트) | Save

```
20190001        Lee Jaeyoon     100     100
20180002        Lee Jieun       92      89
20180009        Lee Yeonghee    81      84
20190002        Kang Mingung    80      80
20180001        Hong Gildong    84      73
20190003        Kim hyojin      70      70
20180011        Ha Donghun      58      68
20190004        Lee Jaekook     60      60
20180007        Kim Cheolsu     57      62
20190005        Lee Shangho     0       0
```

quit ['no'입력 시 프로그램으로 다시 돌아옴]

```
pirl@pirl-Precision-7920-Tower ~/huster_python/프로젝트 master python project1.py students.txt
students.txt
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

#quit
Save data?[yes/no]no

#
```

5. 결론

- 본 과제는 for문과 if문 while문 등 반복문과 조건문을 사용하는 방법을 익히는데 유용했다. 또한 미숙하지만 같은 logic을 쓰는 것들을 묶어서 함수로 정의 해둔 다음 필요할 때 마다 그 함수를 불러와서 파이썬스럽게(oop, 객체지향) 쓰는 방법도 알 수 있었다.

- 내가 원하는 파일을 불러오고 파일에 들어있는 데이터를 사용자가 보기 편하게끔 자료구조에 담고 또 그 자료구조에 접근하여 내가 원하는 데이터를 읽어오고, 바꾸고, 삭제해 보면서 파이썬의 list, dictionary 자료구조에 대해 공부할 수 있었다.

6. 개선방향

- changescore 및 add 함수에서 학생의 점수를 바꾸거나 새로운 학생을 추가 했을 경우 해당하는 학생이 한번 출력되어야 하는데 이때 만들어 놓은 show함수를 사용하고 싶었다. 하지만 show함수 자체에 기본 틀까지 보여지게 끔 만들어 놓았는데, 이 부분을 따로 만들어 놓았으면 changescore 와 add 함수의 코드가 조금 더 간략하게 만들 수 있을 것 같다.

- show함수를 미리 구현 해놓고 코드를 짜다 보니까 만들어 놓은 함수를 고치는 것 보다 그 함수에 맞게끔 코드를 짜려고 하니까 코드를 짜는데 제약이 있었다. 차라리 미리 구현하지 말고, 전부다 코드를 구현한 다음 비슷한 코드를 묶어 함수로 만들어서 사용하면 조금 더 깔끔한 코드가 나올 것 같다.