

성적 관리 프로그램

파이썬 1주차 과제

포스코 청년 AI·Big Data 아카데미

담당 교수 윤은영

이름 이찬

이메일 lc9542@naver.com

ID s_1605

명예서약(Honor code)

“나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.”

1. 성적 관리 프로그램 개요

본 프로그램을 간략히 설명하면 다음과 같습니다.

- 기존의 txt 파일을 열어 7개의 명령어 중 하나를 입력합니다.
- 7개의 명령어는 다음과 같습니다.
- show, search, changescore, searchgrade, add, remove, quit
- 각각의 명령어 알맞은 코드를 수행합니다.

2. 알고리즘

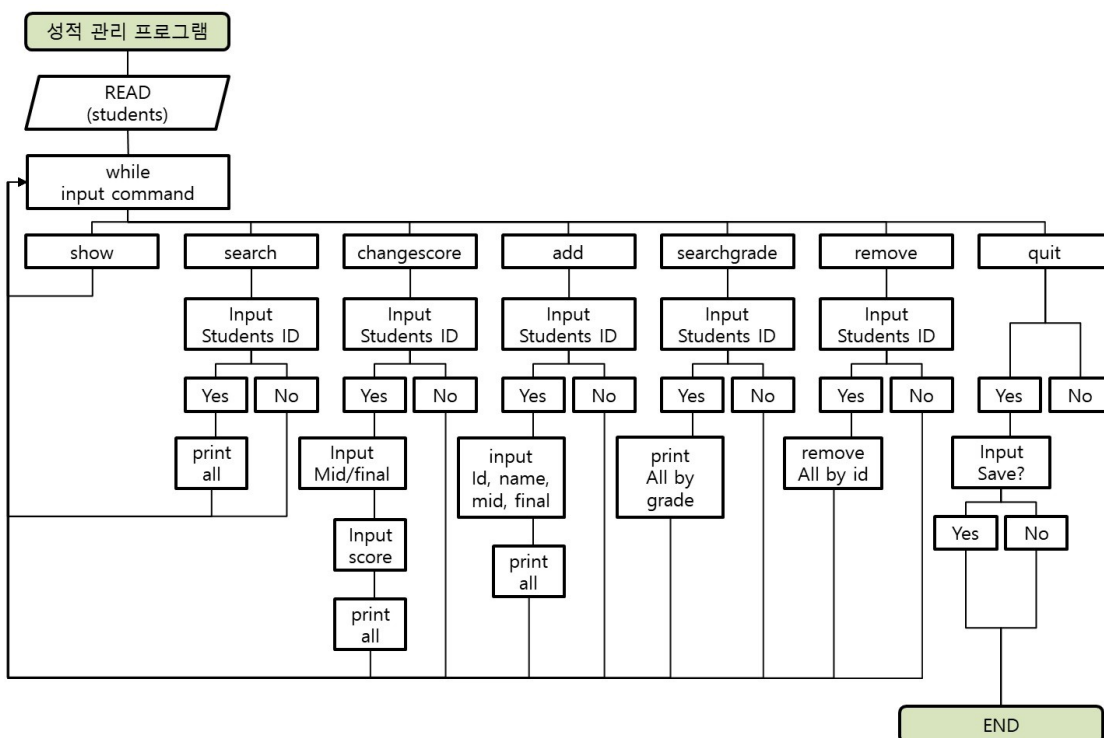
본 프로그램 작성을 위한 알고리즘은 다음과 같습니다.

Pseudo-algorithm for 성적 관리 프로그램

// 프로그램에 필요한 변수는 미리 선언해놓은 것으로 가정한다.

```
1 input command
2 while True
3     if command :
4         def command_function :
5             if command == quit
6                 break
```

위의 알고리즘을 Flowchart를 통해 표현하면 아래와 같습니다.



3. 프로그램 구조 및 설명

a) 원하는 명령어 입력

- 프로그램을 실행하면 실행할 명령어에 대한 입력을 요청한다. 사용자가 입력하면 command 변수에 명령어를 저장한다.

b) 명령어 실행

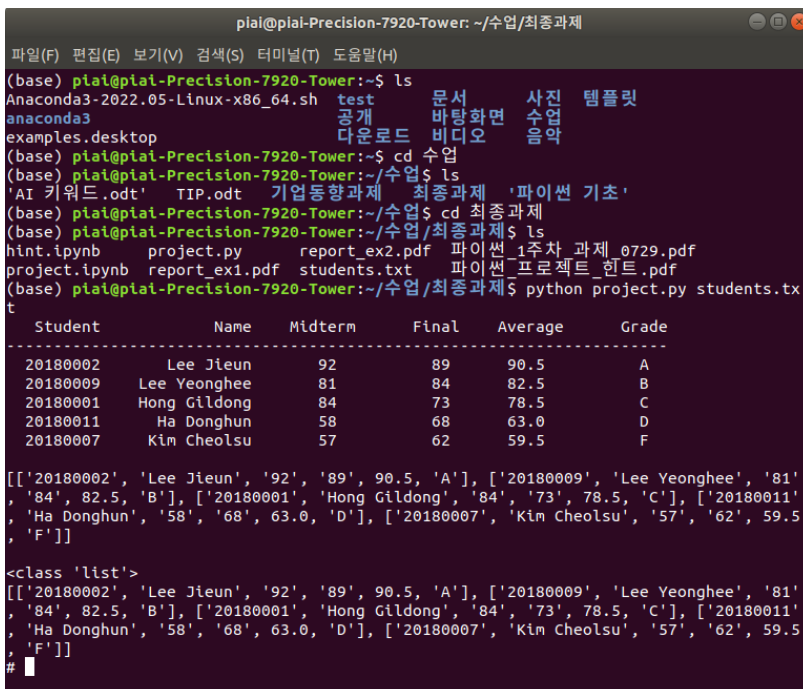
- 입력을 마친 후 command 변수에 저장된 값을 통해 명령어에 맞는 함수를 실행한다.
- 결과를 출력할 때 평균을 기준으로 내림차순으로 출력 혹은 저장된다.

c) 프로그램 종료

- 저장 여부에 따라 txt 파일에 대한 처리 후 프로그램을 종료한다.

4. 프로그램 실행 방법 및 예제

- 리눅스에서 ls로 폴더와 파일을 확인하고, cd를 통해 폴더를 열어 파일을 읽습니다.



```
pi@pi@piat-Precision-7920-Tower: ~/수업/최종과제
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
(base) pi@piat-Precision-7920-Tower:~$ ls
Anaconda3-2022.05-Linux-x86_64.sh  test      문서      사진      템플릿
anaconda3                          공개      바탕화면  수업
examples.desktop                  다운로드  비디오    음악
(base) pi@piat-Precision-7920-Tower:~$ cd 수업
(base) pi@piat-Precision-7920-Tower:~/수업$ ls
'AI 키워드.odt'  TIP.odt  기업동향과제  최종과제  '파이썬 기초'
(base) pi@piat-Precision-7920-Tower:~/수업$ cd 최종과제
(base) pi@piat-Precision-7920-Tower:~/수업/최종과제$ ls
hint.ipynb  project.py  report_ex2.pdf  파이썬_1주차_과제_0729.pdf
project.ipynb  report_ex1.pdf  students.txt  파이썬_프로젝트_힌트.pdf
(base) pi@piat-Precision-7920-Tower:~/수업/최종과제$ python project.py students.tx
t
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

[['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']]

<class 'list'>
[['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']]
#
```

- 이후 실행 화면은 jupyter notebook으로 진행하였습니다.

- 가장 먼저 프로그램의 실행을 위한 txt 파일을 읽습니다.
- txt 파일의 데이터를 통해 평균과 학점을 계산하여 출력합니다.

```
In [4]: f = open("students.txt", "r")

data = f.readlines()

for i in range(0, len(data)) :
    data[i] = data[i].split("\t")

for i in range(0, len(data)) :
    data[i][3] = data[i][3][:2]

for i in range(0, len(data)) :
    data[i].append((int(data[i][2]) + int(data[i][3]))/2)

for i in range(0, len(data)) :
    if data[i][4] >= 90 :
        data[i].append("A")
    elif data[i][4] >= 80 :
        data[i].append("B")
    elif data[i][4] >= 70 :
        data[i].append("C")
    elif data[i][4] >= 60 :
        data[i].append("D")
    else :
        data[i].append("F")

data.sort(key = lambda e : e[5])

print("%+10s %+15s %10s %10s %10s %10s" %("Student", "Name", "Midterm", "Final", "Average", "Grade"))
print("-"*70)
for i in range(len(data)) :
    print("%+10s %+15s %8s %11s %9s %10s" % (data[i][0], data[i][1], data[i][2], data[i][3], data[i][4], data[i][5]))
```

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

- while 문을 통해 실행할 command를 입력하고 해당 command에 맞는 함수를 실행합니다.

```
while True:
    command = input("# ")
    if command == "show" :
        show_function()

    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:
        print("wrong input!")
```

- 또한 7개의 명령어를 제외한 단어가 입력될 시 잘못 입력되었음을 알리고, 다시 프로그램을 진행합니다.

```
# find
wrong input!
```

1). show(전체 학생 정보 출력)

- show 입력 시, 저장되어 있는 전체 목록을 평균 점수를 기준으로 내림차순 출력합니다.

```
def show_function() :
    data.sort(key = lambda e : e[4], reverse = True)

    print("%+10s %+15s %10s %10s %10s %10s" %("Student", "Name", "Midterm", "Final", "Average", "Grade"))
    print("-"*70)
    for i in range(len(data)) :
        print("%+10s %+15s %8s %11s %9s %10s" %(data[i][0], data[i][1], data[i][2], data[i][3], data[i][4], data[i][5]))
```

```
# 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
```

2). search(특정 학생 검색)

- search 입력 시, 검색하고자 하는 학생의 학번을 받아 해당 학생만 출력합니다.

```
def search_function() :
    stu_id = input("Student ID : ")

    for i in range(0, len(data)) :
        if data[i][0] == stu_id :
            print("%+10s %+15s %10s %10s %10s %10s" %("Student", "Name", "Midterm", "Final", "Average", "Grade"))
            print("-"*70)
            print("%+10s %+15s %8s %11s %9s %10s" %(data[i][0], data[i][1], data[i][2], data[i][3], data[i][4], data[i][5]))
            return

    print("NO SUCH PERSON.")
```

```
# search
Student ID : 20180050
NO SUCH PERSON.
# search
Student ID : 20180002
Student          Name      Midterm      Final      Average      Grade
-----
20180002      Lee Jieun      92          89        90.5          A
```

3). changescore(점수 수정)

- 목록에 저장된 학생 중 1명의 중간, 기말고사 점수를 수정하고 점수가 바뀐 학생의 평균, 학점 변경 후 기존의 파일을 수정합니다.

```
def changescore_function() :
    stu_id = input("Student ID : ")

    for i in range(0, len(data)) :
        if data[i][0] == stu_id :
            question = input("Mid/Final? ")
            if question == "mid" :
                newscore = int(input("Input new score : "))
                if (newscore < 101) & (newscore >= 0) :
                    print("%+10s %+15s %10s %10s %10s %10s" % ("Student", "Name", "Midterm", "Final", "Average", "Grade"))
                    print("-"*70)
                    print("%+10s %+15s %8s %11s %9s %10s" % (data[i][0], data[i][1], data[i][2], data[i][3], data[i][4], data[i][5]))

                    data[i][2] = newscore
                    data[i][4] = (int(data[i][2]) + int(data[i][3]))/2

                    if data[i][4] >= 90 :
                        data[i][5] = "A"
                    elif data[i][4] >= 80 :
                        data[i][5] = "B"
                    elif data[i][4] >= 70 :
                        data[i][5] = "C"
                    elif data[i][4] >= 60 :
                        data[i][5] = "D"
                    else :
                        data[i][5] = "F"

                    print("Score changed.")
                    print("%+10s %+15s %8s %11s %9s %10s" % (data[i][0], data[i][1], data[i][2], data[i][3], data[i][4], data[i][5]))
            elif question == "final" :
                newscore = int(input("Input new score : "))
                if (newscore < 101) & (newscore >= 0) :
                    print("%+10s %+15s %10s %10s %10s %10s" % ("Student", "Name", "Midterm", "Final", "Average", "Grade"))
                    print("-"*70)
                    print("%+10s %+15s %8s %11s %9s %10s" % (data[i][0], data[i][1], data[i][2], data[i][3], data[i][4], data[i][5]))

                    data[i][3] = newscore
                    data[i][4] = (int(data[i][2]) + int(data[i][3]))/2

                    if data[i][4] >= 90 :
                        data[i][5] = "A"
                    elif data[i][4] >= 80 :
                        data[i][5] = "B"
                    elif data[i][4] >= 70 :
                        data[i][5] = "C"
                    elif data[i][4] >= 60 :
                        data[i][5] = "D"
                    else :
                        data[i][5] = "F"

                    print("Score changed.")
                    print("%+10s %+15s %8s %11s %9s %10s" % (data[i][0], data[i][1], data[i][2], data[i][3], data[i][4], data[i][5]))

    return

print("NO SUCH PERSON.")
```

```
# 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	Name	Midterm	Final	Average	Grade
20180007	Kim Cheolsu	57	62	59.5	F

Score changed.

Student	Name	Midterm	Final	Average	Grade
20180007	Kim Cheolsu	75	62	68.5	D

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

4). add(학생 추가)

- add 입력 시, 새로운 학생의 정보를 입력 받아 기존의 파일에 추가한 후 평균을 기준으로 내림차순 출력합니다.

```
def add_function() :
    stu_id = input("Student ID : ")

    idlist = []
    for i in range(0, len(data)) :
        idlist.append(data[i][0])

    if stu_id not in idlist :
        new_name = input("Name : ")
        new_mid = input("Midterm Score : ")
        new_final = input("Final Score : ")
        new_avg = (int(new_mid) + int(new_final))/2
        if new_avg >= 90 :
            new_grade = "A"
        elif new_avg >= 80 :
            new_grade = "B"
        elif new_avg >= 70 :
            new_grade = "C"
        elif new_avg >= 60 :
            new_grade = "D"
        else :
            new_grade = "F"

        data.append([stu_id, new_name, new_mid, new_final, new_avg, new_grade])

        print("Student added.")
        return

    print("ALREADY EXISTS.")
```

add

Student ID : 20180001

ALREADY EXISTS.

add

Student ID : 20180021

Name : Lee H

Midterm Score : 93

Final Score : 95

Student added.

add

Student ID : 20180006

Name : Lee S

Midterm Score : 77

Final Score : 66

Student added.

show

Student	Name	Midterm	Final	Average	Grade
20180021	Lee H	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 S	77	66	71.5	C
20180007	Kim Cheolsu	75	62	68.5	D
20180011	Ha Donghun	58	68	63.0	D

5). searchgrade(학점 검색)

- searchgrade 입력 시, 특정 학점을 입력 받아 그 학점에 해당하는 모든 학생을 출력합니다.

```
def searchgrade_function() :
    stu_grade = input("Grade to search : ")
    glist = []
    for i in range(0, len(data)) :
        glist.append(data[i][5])

    new_list = []
    if stu_grade in glist :
        for i in range(0, len(data)) :
            if data[i][5] == stu_grade :
                new_list.append(data[i])
            else :
                new_list = new_list

        print("%+10s %+15s %10s %10s %10s %10s" % ("Student", "Name", "Midterm", "Final", "Average", "Grade"))
        print("-"*70)
        for j in range(0, len(new_list)) :
            print("%+10s %+15s %8s %11s %9s %10s" % (new_list[j][0], new_list[j][1], new_list[j][2], new_list[j][3], new_list[j][4], new_list[j][5]))

    elif stu_grade not in ["A", "B", "C", "D", "F"] :
        pass
    else :
        return print("NO RESULTS.")
```

searchgrade

Grade to search : E

searchgrade

Grade to search : F

NO RESULTS.

searchgrade

Grade to search : D

Student	Name	Midterm	Final	Average	Grade
20180007	Kim Cheolsu	75	62	68.5	D
20180011	Ha Donghun	58	68	63.0	D

6). remove(특정 학생 삭제)

- remove 입력 시, 삭제하고자 하는 학생의 학번을 입력 받아 목록에서 삭제합니다.

```
def remove_function() :
    if len(data) == 0 :
        print("List is empty.")
        return 0

    stu_id = input("Student ID : ")

    for i in range(0, len(data)) :
        if data[i][0] == stu_id :
            del data[i]
            print("Student removed.")
            return

    print("NO SUCH PERSON.")
```

```
# remove
Student ID : 20180030
NO SUCH PERSON.
# remove
Student ID : 20180011
Student removed.
# show
```

Student	Name	Midterm	Final	Average	Grade
20180021	Lee H	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 S	77	66	71.5	C
20180007	Kim Cheolsu	75	62	68.5	D

7). quit(종료)

- quit 입력 시, 프로그램을 종료한다.

```
def quit_function() :
    command = input("Save data?[yes/no] ")

    if command == "yes" :
        fname = input("File name : ")
        fname.replace(' ', '')

        f = open(fname, "w")
        data.sort(key = lambda e : e[4], reverse = True)

        for i in range(len(data)) :
            dataset = "%+10s %+15s %10s %10s %10s %10s\n" %(data[i][0], data[i][1], data[i][2], data[i][3], data[i][4], data[i][5])
            f.write(dataset)
        print("$")
    else :
        print("$")
    f.close()

# quit
Save data?[yes/no] yes
File name : newStudents.txt
$
```

5. 토론

- class를 활용하면 조금 더 가독성이 높은 코드를 구현할 수 있습니다.
- class 활용을 위해 변수 설정과 객체 간의 상호작용에 대한 이해가 필요합니다.

6. 결론

- 성적 관리 프로그램을 구현하며 함수의 작용 원리에 대해 정확하게 이해할 수 있었습니다.
- 에러를 수정하는 과정에서 깊은 고민과 주변 동료들의 도움을 통해 다양한 생각을 합쳐 더 좋은 방향을 생각하고 실행할 수 있었습니다.

7. 개선 방향

- 전체적인 프로그램 구현을 위한 코드 작성 부분에 있어 오랜 시간이 걸렸던 부분을 개선하기 위해 각 코드에 대한 정확한 이해가 필요합니다. 또한 가독성을 높일 수 있도록 class, OOP 등 1주 동안 진행된 수업에 대한 복습의 시간이 반드시 필요하다고 판단하여 주말을 활용하여 해당 부분의 복습이 필수적입니다.