파이썬프로그래밍-프로젝트

Assignment #1

담당교수: 윤은영

조: B3

이름: 김동호

POVIS ID: s_2433

Table of Contents

1.문제 개요

본 프로그램을 간략히 설명하면 다음과 같다.

- 파일로부터 성적 목록 데이터를 읽고 이를 관리하는 프로그램을 만든다.
- 7 개의 명령어(show, search, changescore, searchgrade, add, remove, quit)를 터미널 창에서 입력 받고 각 명령어에 따라 각 기능을 수행한다.
- 명시된 7 가지 명령어 외의 명령어가 입력될 경우 무시하고 다시 명령어 입력을 대기한다.
- 각 동작은 'quit' 명령어를 통해 프로그램을 종료시키기 전까지 계속해서 동작할 수 있도록 설계한다.

2. 알고리즘

본 프로그램 작성을 위한 알고리즘을 Pseudo code 형태로 나타내면 다음과 같다.

Pseudo-algorithm for achievement managing program

// sys 라이브러리를 import 한다.
1. 터미널 창에서 텍스트 파일 이름을 불러온 뒤 해당 파일로부터 학생들의 성적 정보를 불러온다. 불러온 정보로 평균과 학점을 계산한다.
2. 사용자가 원하는 명령을 입력받는다. 명령은 소문자로 바꿔서 저장한다.
3. 사용자가 'quit'을 입력할 때까지 다음 프로그램을 반복한다. 다음 프로그램은 'quit'을 제외하고 6 가지의 명령어로 이루어진다.
    1. 만약 사용자가 'show'를 입력했다면:
    1. 학생들의 성적을 화면을 출력하며 이때 평균 점수를 기준으로 내림차순 정렬한다.
    2. 만약 사용자가 'search'를 입력했다면:
        1. 사용자로부텉 학생의 ID 를 입력받는다.
        2. 찾는 학생이 없을 경우 "NO SUCH PERSON"의 오류 메세지를 출력한다.
    3. 만약 사용자가 'changescore'를 입력했다면:
        1. 사용자로부터 학생의 ID 를 입력받는다.
        2. 사용자로부터 변경할 점수 유형('mid' 또는 'final')을 입력받는다.
        3. 사용자로부터 새로운 점수를 입력받는다.
        4. 해당 학생의 성적을 변경하고 변경된 성적을 평균과 학점을 계산하여 저장한다.
    4. 만약 사용자가 'add'를 입력했다면:
        1. 사용자로부터 새로운 학생의 ID, 이름, 중간고사 점수, 기말고사 점수를 차례대로 입력받는다.
        2. 입력받은 정보를 이용하여 새로운 학생을 성적 정보 딕셔너리에 추가한다.
    5. 만약 사용자가 'searchgrade'를 입력했다면:

1. 사용자로부터 검색할 학점 (A, B, C, D, F) 중에서 입력받는다. 입력한 학점이 포함이 되어있지 않다면 수행할 명령을 다시 입력받는다.
2. 해당 학점의 학생이 있는 경우 해당 학생들의 성적 정보를 화면에 출력한다. 해당 학점의 학생이 없는 경우 “NO RESULT.”를 출력한다.

6. 만약 사용자가 ‘remove’를 입력했다면:
   1. 만약 성적 리스트가 비어있다면 “List is empty.”를 출력한다.
   2. 성적리스트에 정보가 있다면, 사용자로부터 삭제할 학생의 ID 를 입력받는다.
   3. 해당 학생의 정보가 있는지 없는지 확인한다.
   4. 해당 학생의 정보가 없으면 “NO SUCH PERSON”을 출력한다.
   5. 정보가 있을 경우 “Student removed.”을 출력한 후 딕셔너리에서 해당 학생의 성적 정보를 삭제한다.

4. ‘quit’을 입력받은 경우 사용자에게 텍스트 파일 정보를 저장할지 물어본다.
   2. 만약 사용자가 ‘yes’를 입력했다면:
      1. 새로운 파일 이름을 입력받는다.
      2. 학생들의 성적을 새로운 파일에 저장한다.
   3. 사용자가 ‘no’를 입력했다면 그대로 넘어간다.
5. 프로그램을 종료한다.

[Table. 1] 본 프로그램의 Pseudo Code

3. 프로그램 실행 방법 및 예제

컴파일 및 show (전체 학생 정보 출력) 수행

terminal 창에서 'project.py'와 'students.txt'를 같은 경로에 넣어 두고 해당 경로에서 "python project.py students.txt"와 같은 명령어를 입력한다. 이를 통해 'project.py' 프로그램에 대해 컴파일되며 'students.txt' 파일의 정보를 불러온다.

```
(base) gimdongho@gimdonghoui-MacBookAir fastcampus % python project.py students.txt
find
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
```

[Fig. 1] 컴파일 및 show 실행 예시

이후 "find"와 같이 설정한 명령어가 아닌 다른 명령어를 입력한다. 해당 명령어에 대해서는 아무런 정보가 없으므로 아무 동작도 수행되지 않는다. 이후 "show"를 수행하면 txt 파일의 성적 정보가 정리되어 출력되게 된다.

search (특정 학생 검색)

터미널 창에 “search”를 입력한다. 이후 저장되어 있지 않은 학번에 대해서는 “NO SUCH PERSON.”의 텍스트가 출력된다. 이후 저장되어 있는 학번에 대해 검색을 진행했을 때 해당 학생의 정보가 출력된다.

```
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
```

[Fig. 2] search 수행 예시

changescore (점수 수정)

터미널 창에 “changescore”를 입력한다. 성적 딕셔너리에 학번이 없는 경우 “NO SUCH PERSON.”의 메세지를 출력한다. 이후 성적 딕셔너리에 학번이 있는 경우에도 “miid”와 같이 문자를 잘못 쳤을 때나 score 의 값이 0~100 의 값이 아닌 경우 changescore 가 제대로 저장되지 않는다. 반면에 정상적으로 진행했을 때는 수정

이전의 정보와 수정 이후의 정보가 “Score changed”의 메시지와 함께 같이 출력되는 걸 확인할 수 있다.

```
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.
20180007        Kim Cheolsu     75      62      68.5    D
```

[Fig. 3] changescore 실행 예시

값이 제대로 바뀌었는지 확인하기 위해 “show”를 수행하여 결과를 확인했을 때 정보가 바뀌어 출력이 잘 되는 걸 확인할 수 있다.

```
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
```

[Fig. 4] changescore 실행 이후 성적 정보 출력 예시

add (학생 추가)

```
add
Studnet ID: 20180001
ALREADY EXISTS.

add
Studnet ID: 20180021
Name:  Lee Hyori
Midterm Score: 93
Final Score: 95
Student added.

add
Studnet ID: 20180006
Name:  Lee Sangsun
Midterm Score: 77
Final Score: 66
Student added.

show
Student         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
```

[Fig. 5] add 실행 이미지 예시

터미널 창에 “add”를 입력한다. 이후 기존에 저장되어 있는 학생의 정보를 저장시키려고 할 경우 “ALREADY EXITST.”의 메세지가 출력되면서 제대로 프로그램이 동작되지 않는다. 이후 이름과 midterm, final 의 score 가 정상적으로 입력이 된 경우 “Student added.”라는 메세지 출력과 함께 학생 정보가 추가되는 걸 확인할 수 있다.

```
add
Studnet ID: 20180021
Name:  Lee Hyori
Midterm Score: 101
Wrong Midterm Score.

Midterm Score: 93
Final Score: 101
Wrong Final Score.

Final Score: 95
Student added.
```

[Fig. 6] add 실행 시 잘못된 점수 입력 예시

학생의 정보가 없고 이름이 잘 입력된 경우에도 midterm score 와 final score 가 0~100 사이의 값을 입력하지 않은 경우 'Wrong Score"의 메세지가 출력되며 다시 입력할 수 있도록 해준다.

searchgrade (Grade 검색)

```
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
```

[Fig. 7] searchgrade 실행 예시

터미널 창에 'searchgrade'를 입력한다. 탐색할 수 있는 학점 정보로는 [A, B, C, D, F]가 있다. 입력한 학점 정보가 이 안에 없을 경우에는 명령어를 쳐야 하는 단계로 돌아온다. 이후 탐색할 수 있는 학점을 입력하더라도 해당 학생 정보가 없는 경우에는 "NO RESULTS."와 같은 메세지가 출력되며 다시 명령어를 쳐야 하는 단계로 돌아온다. 탐색할 수 있는 학점 정보 중에서도 해당 학생에 대한 정보가 있는 경우를 입력했을 때에는 해당 학생의 정보가 출력되는 걸 확인할 수 있다.

remove (특정 학생 삭제)

```
remove
Student ID: 20180030
NO SUCH PERSON.

remove
Student ID: 20180011
Student removed

show
Student         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
```

[Fig. 8] remove 실행 예시

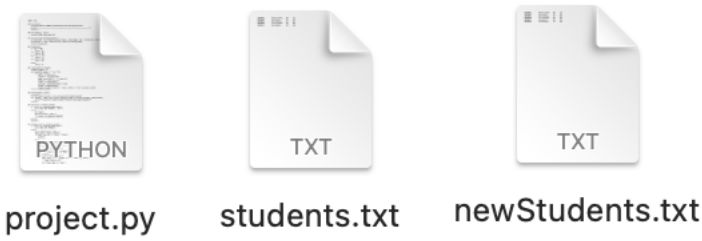
터미널 창에 “remove”를 입력한다. 학생이 목록에 없는 경우에 “NO SUCH PERSON.”과 같이 오류 메시지가 출력된다. 이후 입력한 정보가 포함되어 있는 경우 “Student removed.”의 메시지와 함께 학생 정보가 사라지며 이후 “show”의 명령어와 함께 학생 정보를 확인해보면 이전 정보에서 제거하고자 하는 학생 정보가 사라진 걸 확인할 수 있다.

quit (종료)

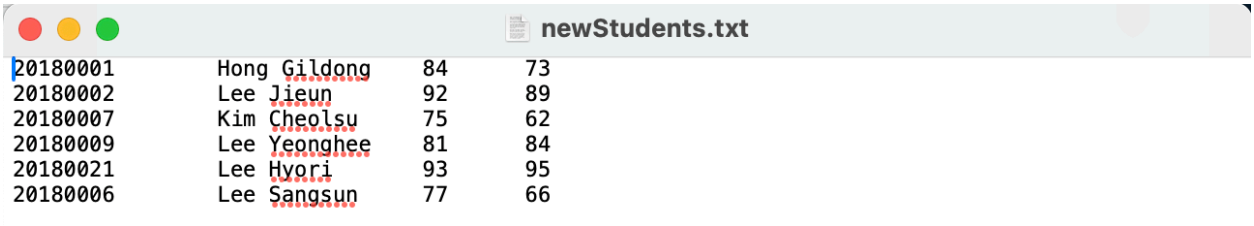
```
quit
Save data?[yes/no] yes
File name: newStudents.txt
```

[Fig. 9] quit 실행 예시

터미널 창에 “quit” 입력 시, 프로그램을 종료한다. 편집한 내용을 새로운 파일로 저장할지 여부를 물어보고 파일명을 입력 받아 저장하도록 한다.



[Fig. 10] 새롭게 생성된 txt 파일 이미지



[Fig. 11] 새롭게 생성된 txt 파일의 내용

편집한 내용을 새롭게 정의한 파일 이름으로 저장한다. 편집한 내용에 따라 알맞게 ‘₩t’ 간격으로 잘 저장되어 있는 걸 확인할 수 있고 project.py 와 students.txt 파일과 같은 경로에 함께 저장된 걸 확인할 수 있다.

4. 토론

- 성적의 평균값을 기준으로 내림차순 정리를 할 때에 딕셔너리 형태에서 특정 값에 대해 정렬을 수행하는 데에 있어 어려움을 겪었다. 이는 lamda 함수를 이용하여 딕셔너리의 item 에서 해당 값들에 대해 sorting 하는 방식으로 해결 가능했다.

- 파일로부터 성적 정보를 바로 불러와 딕셔너리에 집어 넣는 데에 어려움을 겪어다. 모든 데이터를 한꺼번에 읽어와 '₩t' 혹은 '₩n'을 처리하는 게 곤혹스러웠다. 이를 해결하기 위해 파일로부터 한 줄 씩 불러오며 각각의 정보들에 대해 선처리를 한 이후 원하고자 하는 딕셔너리에 넣는 형태로 진행하여 문제를 해결했다.

5. 결론

- 본 과제에서 명령어에 따라 다양한 동작을 수행하는 프로그램을 만드는 방법을 만들었다. 각 동작에 대해 정상적으로 동작하며 'quit'이 나오기 전까지 반복적으로 여러 동작들을 수행할 수 있음을 검증했다. 오류 없이 정상적으로 동작하며 예외처리의 경우까지 모두 고려하여 프로그램 작성 완료했다.

6. 개선방향

- 본 과제는 요구사항에 맞춰 각 7 가지의 기능을 수행할 수 있는 프로그램을 만들었다. 구현된 기능 이외에도 표준편차, 중앙값 등 다양한 통계값을 구할 수 있다. 구현된 프로그램을 좀 더 모듈화하여 가시화한다면 더욱 유동적으로 이용 가능한 프로그램이 될 것으로 기대한다.