20170911 1주차 과제

Python을 활용한 성적 처리 프로그램

Assignment #1

1주차 담당교수 및 조교수:

윤은영, 김은희, 김종구, 정든솔

이름: 박소선

이메일: [parksosun1103@gmail.com](mailto:parksosun1103@gmail.com)

Problem: 성적 처리 프로그램

1. **문제의 개요**

본 프로그램을 간략히 설명하면 다음과 같다.

* 기본적으로 Terminal/Command Prompt을 통해 입력, 명령, 출력한다.
* 기존에 저장된 학생 성적 목록 데이터를 .txt 파일로 입력받는다.
* 그 데이터를 이용해 Average와 Grade 정보를 형성한 후, Student(ID), Name, Midterm, Final, Average, Grade 순으로 새로운 데이터를 정렬 및 출력 (혹은 변수에 저장)한다.
  + 이 때 Average는 (Midterm+Final) /2 의 수식을 사용하고 Grade는 Average 값을 이용해서 A, B, C, D, F 로 등급을 나눈다.
* 새로 형성된 데이터를 7개의 명령어로 접근할 수 있게 한다. (show, search, changescore, searchgrade, add, remove, quit)
  + Show: 새 데이터 목록을 내림차순으로 보여준다. (소수점 이하 첫째자리까지만 표시)
  + Search: 학번을 입력 받아 그에 일치하는 데이터 행을 출력한다.
  + Changescore: 학번, 수정하고자 하는 시험 이름, 새로운 시험 점수를 입력 받는다. 해당 점수, Average, Grade를 업데이트해서 출력한다.
  + Add: 새로운 학생 데이터 row를 입력받고 저장한다.
  + Searchgrade: 특정 grade를 입력 받는다. 그 grade에 해당하는 학생 데이터 row를 출력한다.
  + Remove: 삭제하고자 하는 학생의 학번을 입력받는다. 해당 데이터 row를 삭제한다.
  + Quit: 프로그램의 저장 여부를 묻는다. 저장할 경우 새로운 파일 이름을 입력받고, 내용은 평균을 기준으로 내림차순으로 정렬한 뒤 저장한다. 프로그램을 종료한다.

**2. Flowchart**

Flowchart 에 대한 간략한 설명이다.

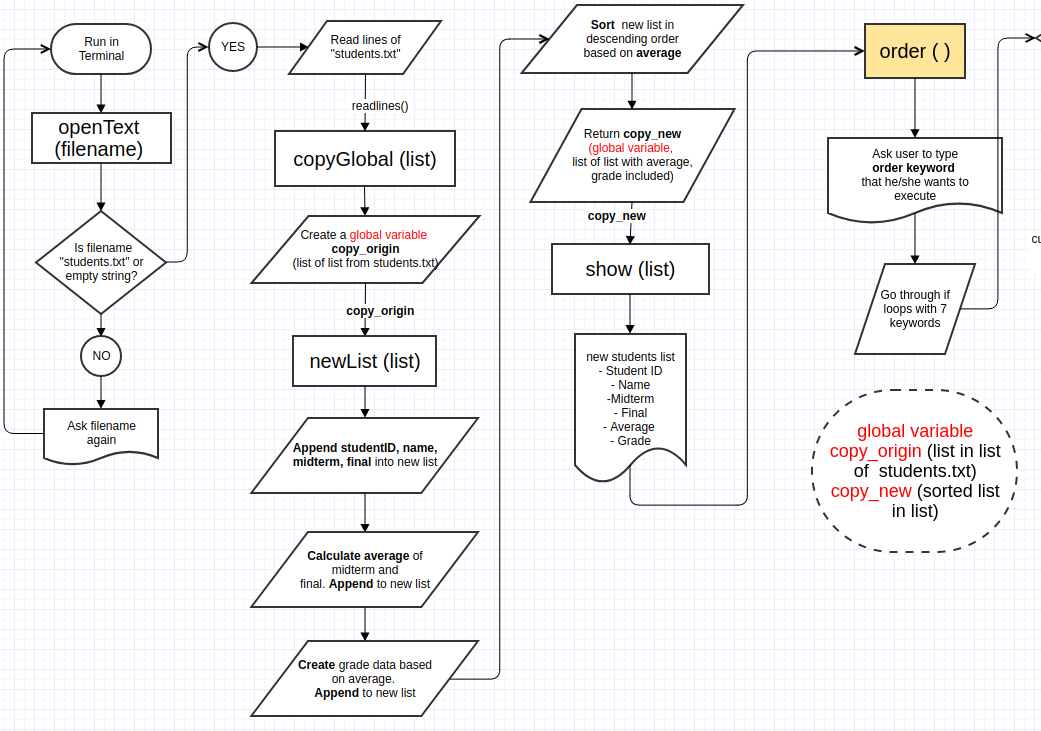
- 공간 부족으로 3 개로 차트를 나누었다. 위 → 아래 순서로 연결된다.

- 복잡도를 줄이기 위해 처음의 order ( ) 로 모두 화살표를 연결하지 않고 각 order 함수 별로 마지막에 order ( ) 로 연결한다. 사실상 다시 초기의 order ( )로 돌아간다는 의미이다.

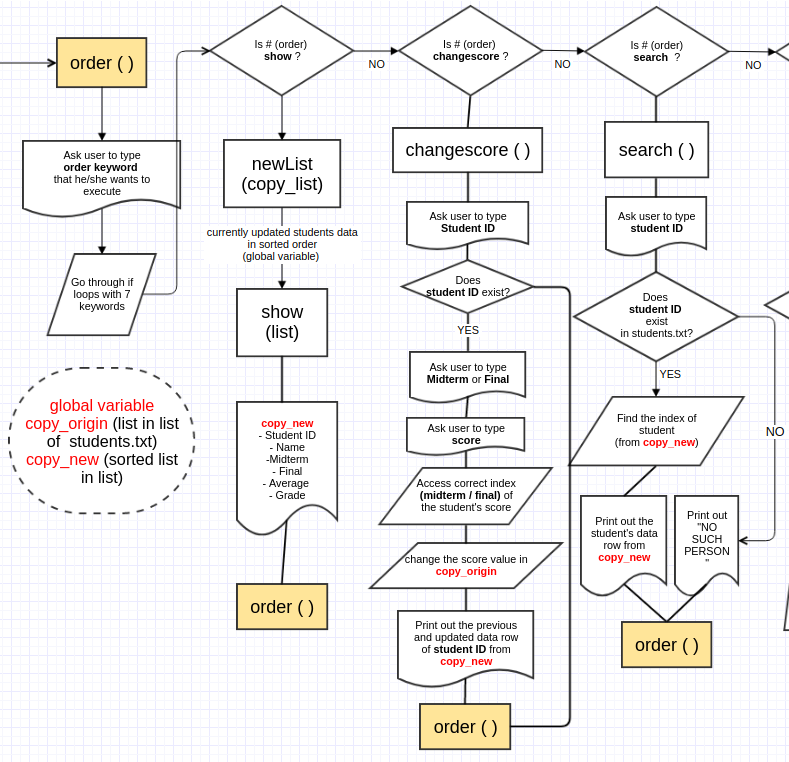
- 함수들에 따라 업데이트 되는 global variable은 copy\_origin과 copy\_new이다.

- copy\_origin은 기존 students.txt를 복사한 후, 중복 리스트로 만들어 저장한다.

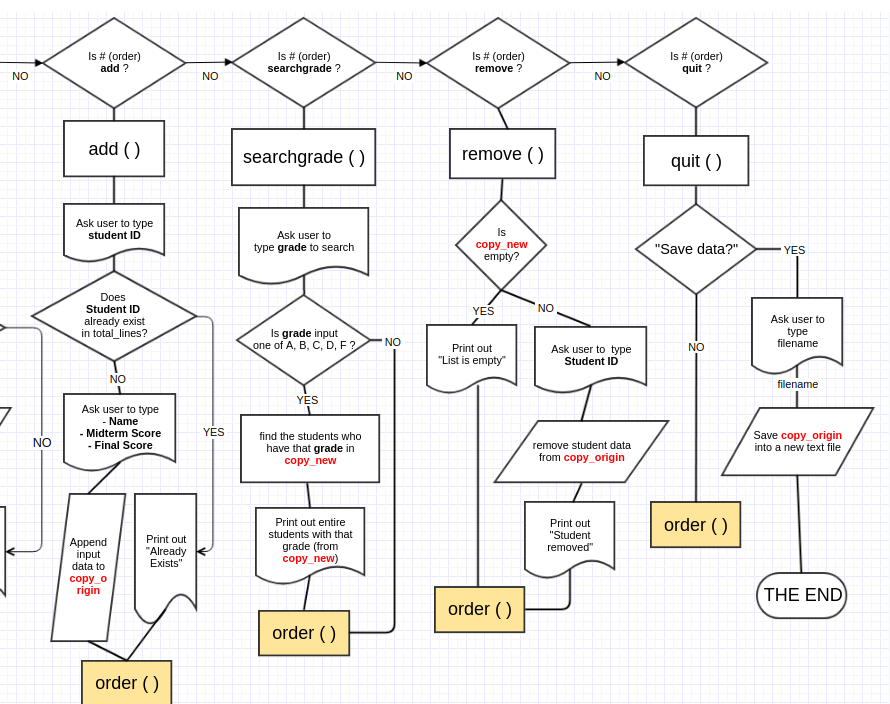
- copy\_new는 copy\_origin을 사용해서 중복 리스트를 만든다. 그 중복 리스트   
 안에 각 학생별로 average, grade를 추가해서 저장한다.

#1

#2



#3



**3. 프로그램 구조 및 설명**

입력 및 글로벌 변수 저장

1. 커맨드 명령으로 터미널에서 실행한다. **(본 과제는 python3 환경에서 제작되었다.)**

e.g.) $ **python3** project1.py students.txt

1. 원하는 성적 데이터가 저장된 텍스트 파일을 입력한다.
2. 프로그램이 본 텍스트 파일을 복사해서 중복 리스트로 만든 후, 글로벌 변수 copy\_origin으로 저장한다.

e.g.) [ [id, name, midterm, final] , [ …] , [...] …..]

1. Copy\_origin으로 저장된 원본 복사 데이터를 활용해서 각 학생별로 average 값을 구한다. 그 average 값을 기준으로 grade를 지정해준다.

average = (midterm + final) / 2

grade 의 경우,

90 <= average : A

80 <= average < 90 : B

70 <= average < 80 : C

60 <= average < 70 : D

Average < 60 : Fx

1. 기존 데이터와 새로운 데이터를 병합해 새로운 글로벌 변수 copy\_new 에 중복 리스트로 저장한다.

e.g.) [ [id, name, midterm, final, average, grade], [....], [...] …]

1. 이 때 lambda 함수를 사용해서 중복리스트를 average 값을 기준으로 내림차순 해준다.

* 여기서 lambda의 쓰임을 간략하게 말하자면, Lambda는 짧은 구문으로 리스트를

접근해서 for loop역할 이상을 해준다.

* ordered\_stu = sorted(total\_stu, key=**lambda** s : float(s[4]), reverse=**True**)

에서와 같이 s 에는 total\_stu 라는 중복 리스트에 저장된 개별 리스트를 훑는다. 그 개별 리스트의 [4] 인덱스 (평균 값의 인덱스) 에 접근해서 서로 비교하여 reverse= True 를 통해 내림차순을 한다.

1. 글로벌 변수 생성 후, show ( ) 함수를 바로 불러서 초기에 자동으로 copy\_new 데이터를 보여준다.
2. 자동으로 order ( ) 함수를 실행시켜 # 에 7가지의 명령어 중 선택하여 입력할 수 있게 한다.

키워드 명령 (project1.py 에 입력한 순서대로)

1. show ( list ) : 카테고리를 출력한다. 인풋으로 글로벌 변수 copy\_new를 입력받아 그때 그때 업데이트 된 데이터를 출력해준다.
2. changescore( ) : 학생 ID, 시험 (중간/ 기말), 점수 를 input 함수들로 입력받는다. copy\_origin에 업데이트 시킨 후, newList 에 copy\_origin을 입력값으로 주고 호출한다. 이렇게 copy\_origin, copy\_new 를 동시에 업데이트 시킨다. (copy\_origin은 원본 txt 파일을 복사한 것이므로 기존 파일 자체를 변경시키지 않는다.)
3. add( ): 학생 ID, 이름, 시험 성적 (중간 & 기말) 을 Input 함수들로 입력받는다. copy\_origin에 정보를 업데이트 시킨다. 마찬가지로 newList를 호출해서 copy\_new 를 업데이트 한다.
4. searchgrade( ): grade 알파벳을 input 함수로 입력받는다. Copy\_new 에 존재하지 않을 시 에러 메세지를 띄우며, 존재한다면 copy\_new 리스트 중 해당 grade를 가진 학생들의 리스트를 문자열로 출력한다.
5. search( ): 학생 ID를 input 함수로 입력받는다. 존재 여부를 copy\_new 에서 확인한다. (이유는 터미널 상에서 quit 이전에 add했을 수 있기 때문이다. ) 존재한다면 copy\_new 에서 해당 학생 리스트를 찾아서 문자열로 출력한다.
6. remove( ): 학생 ID를 input 함수로 입력받고, copy\_new에서 존재 여부를 확인한다. 존재한다면 해당 학생 데이터 행을 copy\_origin 에서 지운다. 그리고 newList(copy\_origin)을 호출해서 copy\_new 또한 업데이트시킨다.
7. quit( ): 데이터를 저장할지 여부를 묻는다. 저장한다면 최종까지 업데이트 된 copy\_origin 을 보여주고 새로운 파일 이름을 입력받는다. 새로운 파일 이름인 텍스트 파일을 만들어서 copy\_origin 을 write한다.

**4. 실행 방법 및 예제**

$ python3 project1.py students.txt

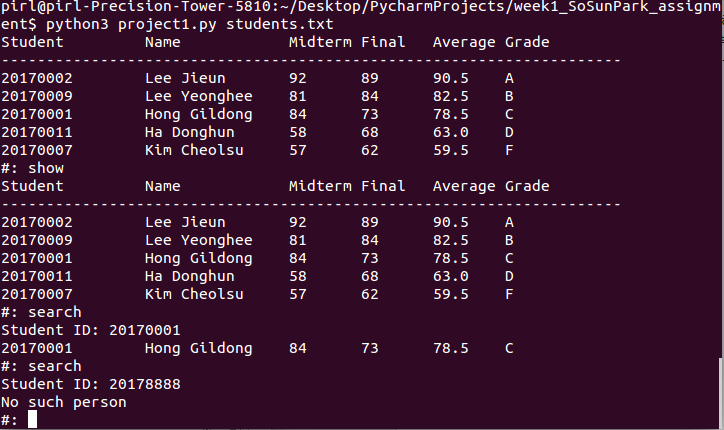
처음에 show 자동실행.

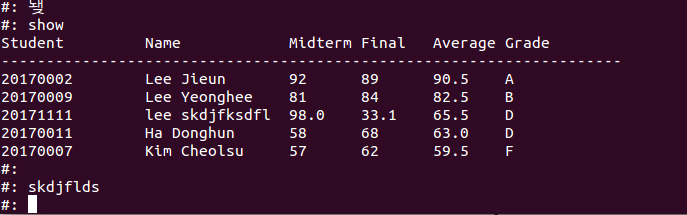
# show

# search - 정상 작동줌

# search - 에러

# show - 7개의 이외의 명령어 입력시 그냥 # 다시 보여줌



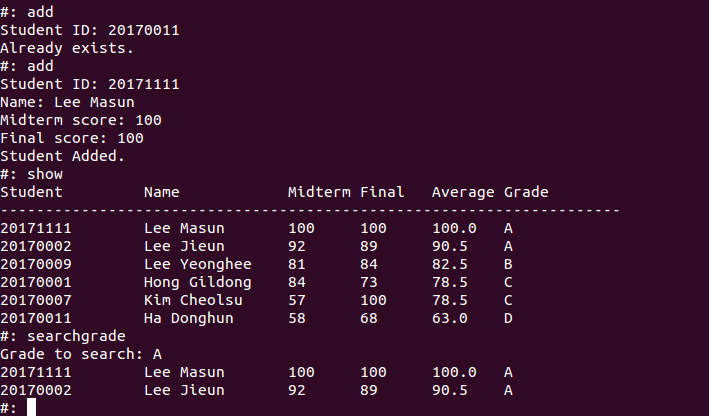


#add - 이미 존재

#add - 정상 작동

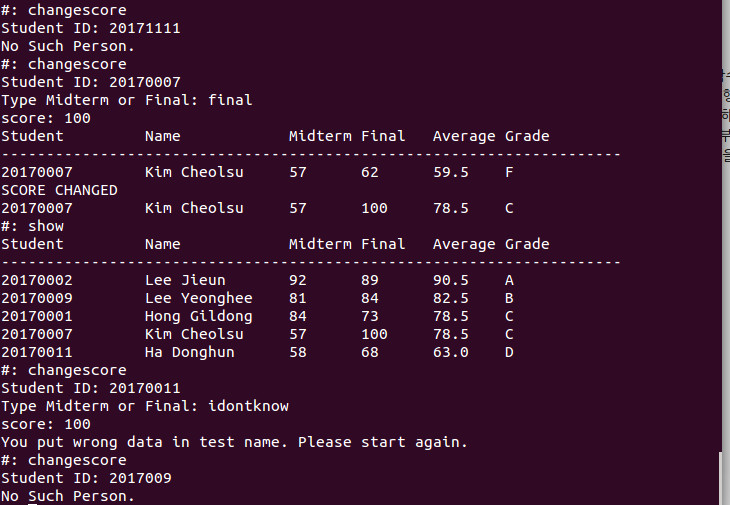
#show - add 이후 업데이트 된 값을 볼 수 있음.

#searchgrade 에도 방금 add 된 데이터가 나옴.



#changescore

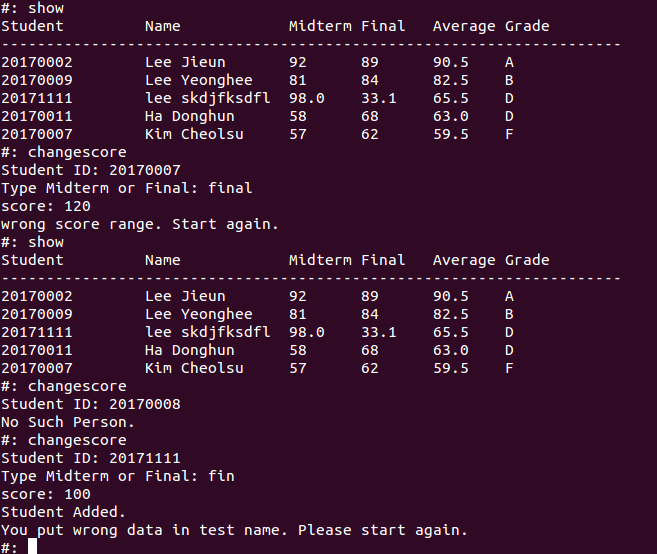
#changescore - 잘못된 시험 이름 작성시 #로 돌아감

#changescore - 존재하지 않는 아이디 기입시 에러 메시지

#changescore - 점수 입력 범위 에러 메시지

#changescore - 존재하지 않는 아이디 에러 메시지

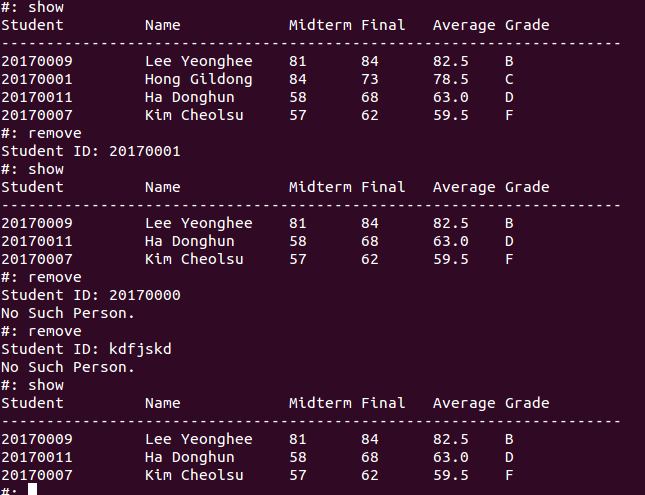
#changescore - 잘못된 시험 이름 기입 에러 메시지

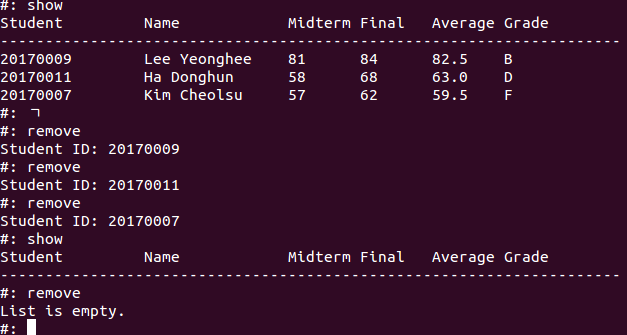


#remove - 정상 작동

#remove - 존재 하지 않는 사람은 No such person 에러 메시지

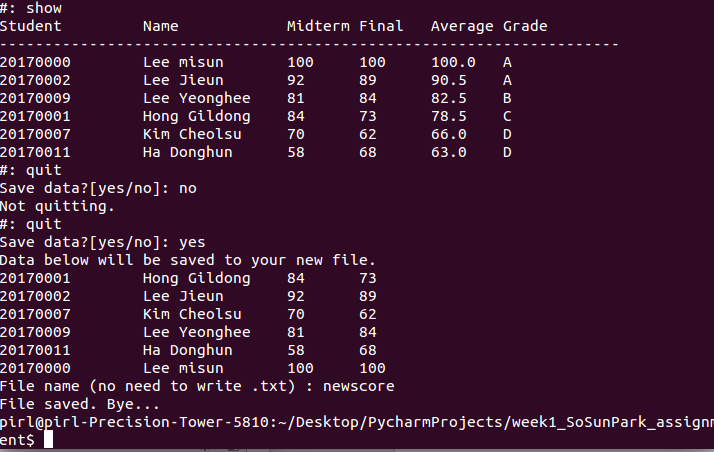
#remove - 다 지운 경우, list is empty 에러 메시지



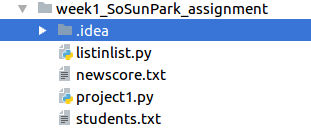


#quit - no 이면 #로 돌아감

#quit - yes 이면 파일 이름값 받음. 데이타 보여줌. 저장.



위 이름으로 같은 디렉토리에 저장되었음.



을 해줄 수 있다.

**5. 결론 및 개선 방향**

* 테스트 결과 에러는 없지만 더욱 효율적으로, 반복되는 부분을 줄일 수 있을 듯 하다.
* changescore()함수를 작성하던 초반에, 변경되기 전 데이터와 변경 후의 데이터가 제대로 출력되지 않았다. 여러 번의 시도 결과, 함수가 끝나기 전에 newList를 호출해서 글로벌 변수를 업데이트해줘야 한다는 것을 알았다.
  + 다른 6개의 키워드 명령어 중 add, remove 처럼 데이터를 변경하는 경우에 위와 같이 작동하게 해야 한다.
* changescore() 에서 미드텀, 파이널 시험 이름 입력이 잘못되어도 우선 점수를 입력하고 나서 오류를 알려주는 한계가 있다.
* 글로벌 변수가 총 3개이며 2개를 지속적으로 업데이트 하고 있다. 글로벌 변수를 사용하지 않는 방법이 있는지 알아봐야 한다.
* 코드의 메인 부분에 original\_lines = openText(f\_name) 불필요한 글로벌 변수를 설정하지만 copyGlobal(openTexT(f\_name)) 처럼 바로 집어넣을 경우 quit함수에서 yes → 파일 이름 지정 이후 order()로 다시 돌아오기 때문에 다른 방법을 찾지 못했다.
* 성적 원본 텍스트 파일은 openText() 안에서 부르기 때문에 메인 에서 f.close() 와 같은 코드를 넣지 못했다. 에러는 없지만 이후의 다른 작업을 추가할 때 문제 발생의 여지가 있을 수도 있다.