POSCO BIGDATA ACADEMY 11 기

성적 처리 프로그램

1Week - Assignment

담당교수: 윤은영

학번: B2

학과: 경영학과

이름: 윤민기

LMS ID: s_0716

명예서약(Honor code)

"나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다."

Problem: 성적 처리 프로그램 개발

1. 문제의 개요

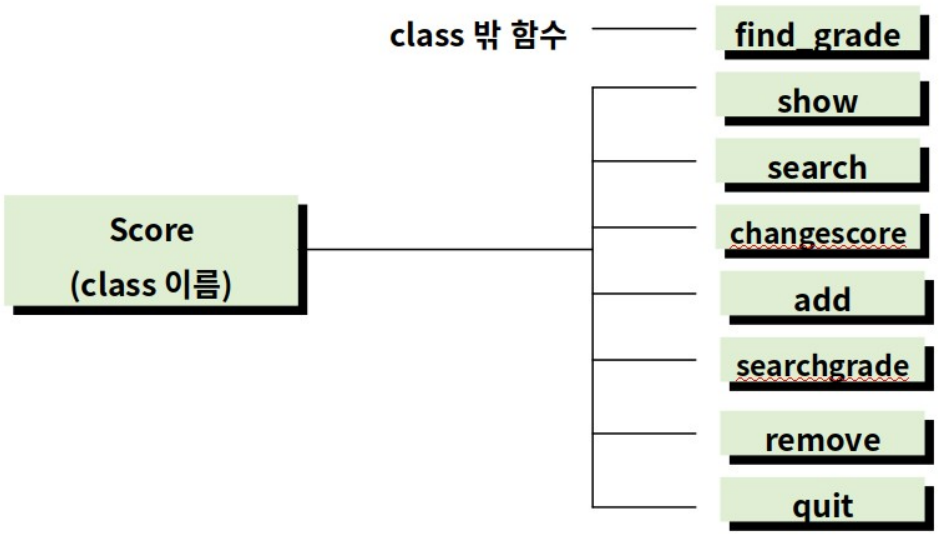
본 프로그램을 간략히 설명하면 다음과 같다.

- erminal 환경에서 project.py 를 students.txt 파일을 읽어 실행시킨다.

- show, search, chagescore, add, searchgrade, remove, quit 총 7 가지의 성적 출력 및 조작 기능을 구현한다.

- 사용자의 선택에 따라 조작된 성적 처리 파일을 newStudents.txt file 로 저장한다.

이 때 사용되는 구상 가능한 구조 차트(structure chart)는 아래와 같이 표현될 수 있다.



- 이 외에 class 내에서 데이터 선언, 저장을 위한 __init__() 함수와 class 전체 실행을 위한 main() 함수가 있다. 위 차트에서는 class 의 기능적 요소 파악에 용이하도록 두 함수를 생략했다. project1.py 의 입출력 처리의 구분은 다음과 같다.

- 입력부: 입력에 관련한 모든 변수를 main() 함수에서 다루도록 설계하였다. Main() 함수에서 주요 7 기능인 show(), search(), changescore(), add(), searchgrade(), remove(), quit() 을 명령어에 맞게 호출한다.

- 처리부: changescore(), add(), remove(), quit() 에서 자료의 조작 및 저장을 구현한다. 이 때 데이터는 앞서 시행된 명령어에 결과에 따른 값으로 하되 함수 서로 간의 영향은 없도록 설계했다.

- 출력부: 출력의 경우 show() 와 search(), searchgrade() 에서 기능한다. changescore() 는 처리부지만 불필요한 코딩을 줄이기 위해 처리 후 바로 출력으로 설계했다.

- 그 외: 처리와 출력 기능 전반에서 성적을 계산하는 경우가 많아 그 때마다 새로 구현 하기는 비효율적이기에 따로 함수를 만들었다. 이 함수는 self 를 받을 필요가 없기 때문에 class 밖에서 기능하며 Score class 내에서 함수들이 필요할 때마다 호출된다.

2. 알고리즘

본 프로그램의 알고리즘은 매우 단순하다. 이를 Pseudo 코드 형태로 나타내면 다음과 같다.

Pseudo-algorithm for dot-line distance	
	# class 밖에서 students.txt 파일로부터 class 의 파라미터로 작동할 리스트 생성
1	make **class Score** and **def find_grade(average)**
2	prompt user and read default list from students.txt
3	input command among show, search, changegrade, searchgrade, add, remove, quit
4	execute main() and be processed by command function
5-1	if input show: print table
5-2	if input search: request input student id and show an individual information
5-3	if input changescore: request input student id and choose between mid and final input modified score and call def find_grade() student's information is changed
5-4	if input searchgrade: request a grade among from A to F except E show student(s) information including the grade
5-5	if input add: request input student id, name, mid, and final call def find_grade() and save new student's information on this table
5-6	if input remove: request input student id and delete the student's information from this table
5-7	if input quit: ask whether user wants to save new table or not, if user chooses yes → save and exit, else → exit

class Score 내의 모든 함수는 상호 독립적이다. 각 함수의 실행이 데이터 값에는 변화를 줄 수 있지만 함수 서로 간에 영향, 선행 조건 등은 없다.

프로그램 알고리즘 설계에서 가장 먼저 신경을 쓴 부분은 데이터의 저장 여부이다. quit 명령어 입력 전까지 무한히 명령어 입력이 가능한 프로그램인 만큼 저장 부분에서 에러가 날 가능성이 높다고 생각했다.

3. 프로그램 구조 및 설명

a) class Score 선언 및 내부 변수, __init__(), main() 선언

- students.txt 파일을 읽어 만들어진 리스트를 파라미터로 받는 __init__(self, table)을 class Score 내에 만든다. 클래스 내에서 광범위 하게 사용할 변수로

- title = ['Student', 'Name', 'Midterm', 'Final', 'Average', 'Grade'] 을 선언해 준다. 그 이유는 클래스 내의 여러 함수에서 self.table 을 통해 리스트가 계속 변환되는데 이 때 학생들의 정보를 담고 있는 다른 리스트들과 구성 원소의 타입이 달라 일일이 처리하기가 번거롭다.

- title = [str, str, str, str, str, str] 인데 다른 학생 정보 리스트들은 student = [str, str, int, int, float, str] 이다.

- main() 함수의 선언을 통해 일곱 개의 명령어 처리를 도맡는다. 클래스 밖에서 인스턴스 변수 선언 후 var.main() 만으로 클래스를 호출할 수 있도록 설계했다.

b) class 밖에서 students.txt 내용을 리스트로 변환, find_grade() 선언

- with open 호출을 통해 students.txt 를 table = [[학생 정보], [학생 정보], …, [학생 정보]] 형태로 만든다.

- 파일 내에는 각 학생 정보가 [id, name, mid, final] 타입으로 5 명이 저장되어 있다. 프로그램은 mid 와 final 의 평균, 그리고 각 점수에 해당하는 등급도 필요로 한다. 프로그램 전반적으로 등급을 불러올 일이 많기 때문에 단순 등급 처리만을 위한 함수를 만들어 평균과 함께 학생 정보에 추가한다.

- 학생 정보 리스트의 타입은 student = [str, str, int, int, float, str] 다. 이를 score = Score(table) 로 인스턴스화 하여 모든 단계는 이제 class 내부로 넘어간다.

c) show, search, searchgrade

- table 의 출력을 담당하는 명령어들로 terminal 환경에서 show, search, searchgrade 를 입력하면 main() 함수의 호출을 통해 각각 show(), search(), searchgrade() 함수를 불러온다.

- show() 는 조작을 거친 현재 상태의 table 을 형식에 맞춰서 출력한다. Search() 는 원하는 학생의 정보만 출력한다. Searchgrade() 는 보고자 하는 등급에 해당하는 모든 학생의 정보를 출력할 수 있다.

d) changescore

- changescore 는 처리와 출력을 모두 하도록 설계했다. 역시 main() 함수의 호출을 통해 changescore() 를 불러온다.

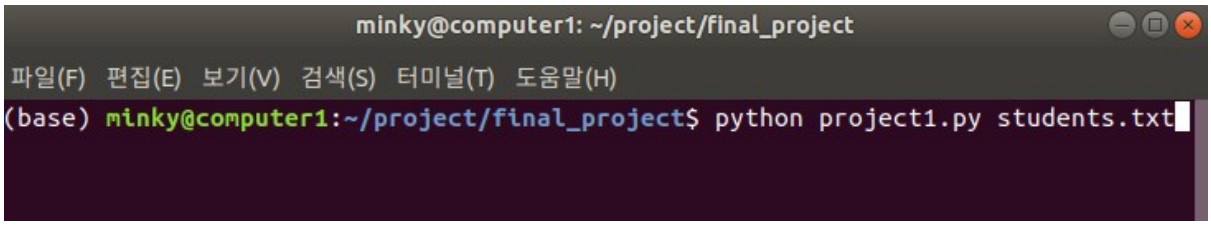
- 처리 부분은 stu_id 테이블 내 존재 유무를 확인 후 존재할 시 mid 와 final 중 선택해 점수를 변환한다. 그 후 find_grade 를 호출해서 바뀐 점수에 따른 평균과 등급을 업데이트 한다.

- 출력 부분의 경우 데이터의 변화를 비교하기 위해 변하기 전에 search() 함수 호출, 변한 후에 search() 함수 호출, 총 두 번의 호출로 표현했다.

e) add, remove, quit

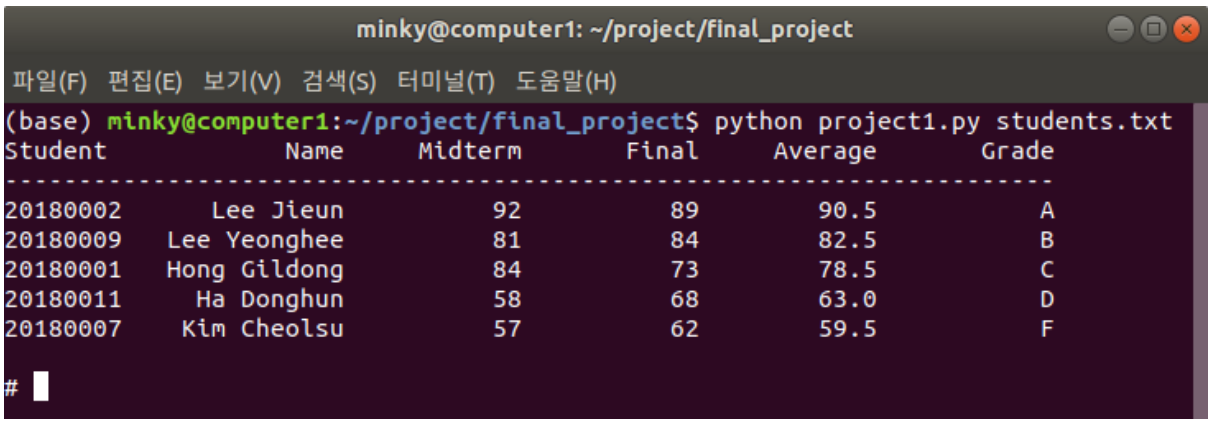
- add 는 추가 학생 정보 입력, remove 는 입력한 id 에 해당한 학생 정보를 테이블에서 삭제, quit 는 조작한 테이블을 텍스트 파일 형식으로 저장할지 여부를 결정 후 exit 를 실행한다. 역시 마찬가지로 각각 main() 함수를 통해 add(), remove(), quit() 함수를 호출한다.

4. 프로그램 실행방법 및 예제

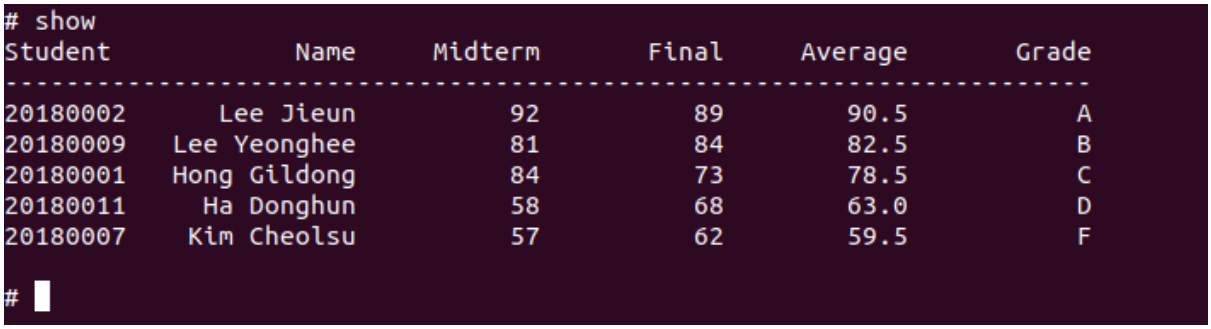


- ubuntu 18.04 LTS OS terminal 에서 project1.py 가 있는 파일로 이동 후

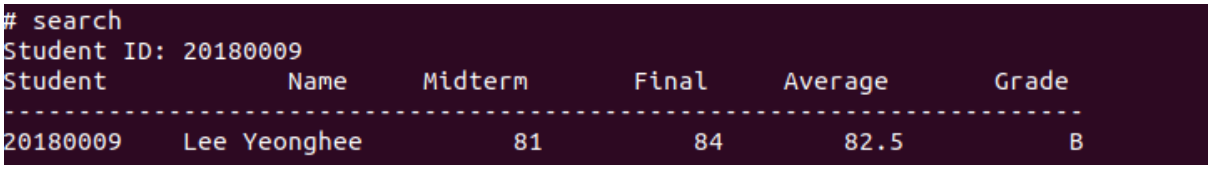
python project1.py [students.txt] 를 입력해 파일을 실행한다.



- 실행하면 먼저 students.txt 파일의 내용이 형식에 맞춰 출력된다. 실행과 동시에 텍스트 파일에 있던 내용이 list 로 class Score 에 들어간다.



- show 를 실행하면 명령어를 통해 조작된 list 형식에 맞춰 출력된다. 현재 show 를 제외한 명령어가 입력되지 않았기 때문에 변동 사항은 없다.



- search 를 실행하면 “Student ID: Input value“ 가 뜨고, 만약 그 ID 가 list 에 있으면 해당 ID 에 해당하는 정보를 출력한다.

```
# search
Student ID: 201312005
NO SUCH PERSON
```

- search 실행 시 "Student ID: Input value" 에 list 에 없는 값을 입력하면 "NO SUCH PERSON"을 반환한다.

```
# CHANGESCORE
Student ID: 20180002
Mid/Final? mid
Input new score: 100
Student            Name      Midterm       Final      Average       Grade
------------------------------------------------------------------------
20180002       Lee Jieun           92          89         90.5           A

Score Changed
Student            Name      Midterm       Final      Average       Grade
------------------------------------------------------------------------
20180002       Lee Jieun          100          89         94.5           A
```

- changescore 를 대소문자 상관없이(모든 명령어가 그러하다) 실행시키면 "Student ID: Input value" 가 뜨고 list 에 있는 id 를 입력하면 mid 와 final 중에 선택이 가능하다. 그 후 변환시키고 싶은 점수를 입력하면 바뀌기 전과 후를 비교해 그 학생의 정보가 출력 된다. search 와 마찬가지로 list 에 없는 id 입력 시 "NO SUCH PERSON"을 반환한다.

```
# add
Student ID: 201312007
Name: minky2
Midterm Score: 100
Final Score: 90
```

```
# add
Student ID: 201312007
ALREADY EXISTS
```

- add 를 실행시키면 "Student ID: Input value" 가 뜨고 list 에 해당 id 가 없으면 name, mid, final 을 모두 입력한 후 find_grade() 를 호출해 추가할 정보를 완성한다. 만약 해당 id 가 list 에 있으면 "ALREADY EXISTS" 를 반환한다.

```
# SEARCHGRADE
Grade to Search: A
Student            Name      Midterm       Final      Average       Grade
------------------------------------------------------------------------
201312005         minky           90         100         95.0           A
201312007        minky2          100          90         95.0           A
20180002       Lee Jieun          100          89         94.5           A
```

```
# searchgrade
Grade to Search: E
NO RESULTS
```

- searchgrade 를 실행시키면 "Grade to Search: "에 확인하고 싶은 성적을 입력한다. 그 성적이 리스트 안에 있으면 그 성적에 해당하는 학생 정보를 반환한다. 만약 해당 성적이 리스트 안에 없으면 결과 없음을 반환한다.

```
#  remove
Student ID: 201312007
```

```
#  remove
Student ID: 201312007
NO SUCH PERSON
```

- remove 를 실행시키면 student id 를 받고, 해당 아이디가 리스트 안에 있으면 아무런 출력 없이 해당 아이디에 해당하는 정보를 삭제한다. 만일 해당 아이디가 리스트 안에 없으면 "NO SUCH PERSON"을 반환한다.

```
# SHOW
Student            Name     Midterm       Final     Average       Grade
-------------------------------------------------------------------------
201312005         minky          90         100        95.0           A
20180002      Lee Jieun         100          89        94.5           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
```

- 현재까지 조작한 정보를 show 하면 다음과 같다.

```
# quit
Save data?[yes/no]: yes
(base) minky@computer1:~/project/final_project$
```

- quit 를 입력하면 지금까지 조작한 데이터를 새로운 텍스트 파일에 저장할 것인지 여부를 묻는다. yes 를 선택하면 저장 후 exit 를, no 를 선택하면 저장 없이 exit 를 반환한다.

5. 토론

- 이 과제를 해결하기까지 정말 많은 시행착오를 겪었다.

- 어떻게 하면 terminal 환경에서 예쁘게 출력할까? python 에 terminal 환경 출력을 위한 여러 module 들이 있다. 이를 활용할까도 했지만 print() 를 통해 직접 구현하는 방향을 택했다. 만일 과제가 아니었다면 주저없이 module 을 이용하는 것을 추천한다.

- 가장 쉽게 table 내에 data 변환을 반영할 방법은 무엇인지에 관해 처음에는 class 가 필요할 것이라 생각하지 못해 .py 파일 안에 함수만 개별적으로 선언하다가 data 의 저장이 번거로워진다는 것을 깨달았다. 뿐만 아니라 class 가 받을 매개변수로 텍스트 파일로부터 생성된 테이블 하나만 받을지, 혹은 학생 정보만 포함한 테이블과 제목까지 포함한 테이블을 분리해 두 종류의 테이블을 함께 매개변수를 받을지 선택했다. 두 개의 테이블을 모두 받는 방법을 택하면 테이블 내의 타입에 관해 무신경해도 된다는 장점이 있지만 필요 없는 변수 할당이 늘어난다는 단점이 커서 조금 더 정교하게 코딩을 하고 학생 정보만을 포함한 하나의 테이블을 받는 방향으로 선택했다.

- 입력, 처리, 출력 부분의 구분에 관해 어떻게 하면 더 효율적일까를 생각하다가 list 로는 현재 내 수준으로는 방법이 없어 보인다는 결론이 나왔다. Dict() 로 구현하는 편이 훨씬 효율적이라는 결론에 도달했다.

6. 결론

- 본 과제를 통해 python class 를 통한 프로그램 모듈 통제에 대해 보다 더 깊이 이해할 수 있게 되었다. 뿐만 아니라 Python 을 통한 텍스트 파일 조작 및 처리에 관해 심도 있게 생각할 수 있는 시간을 가질 수 있었다.

- 성적 처리 같은 비교적 간단한 프로그램을 만들 때에도 세밀한 프로그래밍 설계가 필요하다는 것을 알 수 있었다. 다음에 다시 만들게 되면 더 잘할 수 있을 것 같다는 용기를 얻었다.

7. 개선방향

- 처음 프로그램을 설계할 때 코드 전반에 걸쳐 사용할 자료형으로 list() 가 편리해 보여 채택하였다. 그러나 실제로 코드를 구현한 결과 student_id 라는 key 값으로 호출이 필요한 경우가 많았으며 list() 는 이 때 효율적인 도구가 아니다. 당장의 편리함에 눈이 멀어 list() 를 선택한 결과 나무를 보지 못한 결과를 낳았다. 다음에 비슷한 유형의 프로그램을 설계하게 된다면 dict() 를 우선순위에 둔다.

- 불필요하게 코드가 너무 길다는 인상을 받았다. 현재 185 줄로 몇 번의 검토를 거친 끝에 많이 줄였지만 여전히 깔끔함과는 거리가 멀어 보인다. 더 직관적이면서 길지 않게 구현하는 방법을 찾아야 한다.

-