**빅데이터 AI 아카데미 12기**

**성적 관리 프로그램**

**파이썬 1주차 과제**

**이름: 김준오**

**이메일 : juno1028@naver.com**

|  |
| --- |
| **명예서약(Honor code)**  **“나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.”** |

**Problem 1: 성적 관리 프로그램**

1. 문제의 개요

본 프로그램을 간략히 설명하면 다음과 같다.

(문제)

파일로부터 데이터를 읽어서 성적 목록을 만들어 관리하는 성적 관리 프로그램을 작성한다. 7개의 명령어(show, search, changescore, searchgrade, add, remove, quit)를 입력 받아 각 기능을 수행 하게 된다.

* **Show** (전체 학생 정보 출력) : show 입력 시, 저장되어 있는 전체 목록을 아래와 같이 평균 점수를 기준으로 내림차순으로 출력한다. 평균 점수는 소수점 이하 첫째 자리까지만 표시한다.
* **Search** (특정 학생 검색) : search 입력 시, 아래와 같이 검색하고자 하는 학생의 학번을 요구해 입력 받아 학번, 이름, 중간고사 점수, 기말고사 점수, 평균, 학점을 출력한다.
* **Changescore** (점수 수정) : 목록에 저장된 학생 중 1명의 중간고사(mid) 혹은 기말고사(final)의 점수를 수정한다.
* **Add** (학생 추가) : 학생의 학번, 이름, 중간고사 점수, 기말고사 점수를 차례로 요구해 입력 받는다.
* **Searchgrade** (Grade 검색) : 특정 grade를 입력 받아 그 grade에 해당하는 학생을 모두 출력한다.
* **Remove** (특정 학생 삭제) : 삭제하고자 하는 학생의 학번을 입력 받은 후, 학생이 목록에 있는 경우 삭제한다.
* **Quit** (종료) : 프로그램을 종료한다. 현재까지 편집한 내용의 저장 여부를 묻고, 저장을 선택할 경우 파일명을 입력 받아서 저장하도록 한다.

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

Graphical user interface

Description automatically generated

* 입력부: 학생들의 학번, 이름, 중간고사 점수, 기말고사 점수가 들어있는 텍스트 파일을 입력으로 한다. 파일이 지정되어 있지 않다면, 기본적으로 동일경로 내에 있는 students.txt 파일을 읽어오도록 한다.
* 처리부: show(), search(), change\_score(), add(), search\_grade(), remove(), quit()의 총 7가지의 처리 함수를 가지고 있다. 모두 해당 함수의 이름을 명령어로 입력받아 실행된다.
* 출력부: 처리가 끝난 데이터를 사용자가 원할 시, 파일 형식으로 출력한다. 사용자가 quit 명령어를 입력하면, 파일 저장 여부를 물어보고 실행하도록 한다.
* Readline\_and\_append(), get\_average(), get\_grade(), print\_table\_row()는 처리부 함수 내부에 반복적으로 들어가는 코드를 따로 함수화하여 놓아서 코드의 가독성을 높이고 반복을 줄이는데에 활용한다.

2. 알고리즘

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

|  |  |
| --- | --- |
| **Pseudo-algorithm for main()** | |
|  | *// 프로그램에 필요한 변수들은 미리 선언해놓은 것으로 가정한다.*  while(True):  # 파일 읽어오기 및 데이터 저장하기  if sys.argv’s length == 1: # 사용자가 뒤에 아무것도 입력하지 않았을 때,  file\_name = ""  else:  set file\_name as sys.argv[1] # 사용자가 뒤에 파일명을 입력했을 때,  # stu\_list를 선언해준다. 나중에 stu\_list = [["20180001","Hong Gildong", 84, 73], ["20180001","Hong Gildong", 84, 73], ...]  stu\_list = []  # input 받은 파일명에 따라 실행되는거 분류  if file\_name == "":  open students.txt as fr:  readline\_and\_append(fr)  break  elif file\_name exists in path:  open students.txt as fr:  readline\_and\_append(fr)  break  else:  print 'File not Found'  # 종료 시키기  Quit the program  # stu\_list에 Average, Grade 추가하기  # [["20180001","Hong Gildong", 84, 73] , [~~], ...] --> [["20180001","Hong Gildong", 84, 73, 78.5, "C"], [~~],...]  for student in stu\_list:  set student\_average\_score = get\_average(student\_mid\_score, student\_fina\_score)  student\_grade = get\_grade((mid\_score + final\_score) / 2)  append student\_grade to student  append student to stu\_list  # stu\_list 평균 점수를 기준으로 내림차순 정렬  Sort stu\_list by average\_score(reverse=True)  # 프로그램 실행시키면 출력되는 화면  print\_table\_row # 열과 표를 그려주는 함수  for student in stu\_list:  print student’s info  # 명령어 기다리기  while(True):  set input\_command by input()  make this word upper  if input\_command == "SHOW":  show()  elif input\_command == "SEARCH":  search()  elif input\_command == "CHANGESCORE":  change\_score()  elif input\_command == "ADD":  add()  elif input\_command == "SEARCHGRADE":  search\_grade()  elif input\_command == "REMOVE":  remove()  elif input\_command == "QUIT":  quit()  get out of while # while문을 break하여 빠져나온다.  else:  continue # 계속해서 돌아간다. |
|  |  |

|  |  |
| --- | --- |
| **Pseudo-algorithm show()** | |
|  | *// 프로그램에 필요한 변수들은 미리 선언해놓은 것으로 가정한다.*  def show():  print\_table\_row()  for student in stu\_list:  print student # student 안에는 학번, 이름, 중간고사성적, 기말고사 성적, 평균, 학점이 들어가있다.  return |
|  |  |

|  |  |
| --- | --- |
| **Pseudo-algorithm search()** | |
|  | *// 프로그램에 필요한 변수들은 미리 선언해놓은 것으로 가정한다.*  def search():  set student\_id user\_input  for student in stu\_list:  if student's id == user\_input\_student\_id  print\_table\_row()  print student's information  return # 찾으면 함수를 종료시켜 버린다.  print "NO SUCH PERSON." # for 문을 다 돌때까지 찾지 못하면 "NO SUCH PERSON."을 띄운다.  return |
|  |  |

|  |  |
| --- | --- |
| **Pseudo-algorithm change\_score()** | |
|  | *// 프로그램에 필요한 변수들은 미리 선언해놓은 것으로 가정한다.*  def change\_score():  set student\_id user\_input  for student in stu\_list:  if there is student\_id match to student\_list  ask mid\_or\_final  if answer is mid  ask new\_score  if valid score  # 바뀌기 이전 score 출력  print\_table\_row()  print student information  student's mid\_score = new\_score # 중간고사 점수 수정  student's average score = get\_average(  mid score, final score) # 평균점수 다시 계산  recalculate the grade by new scores # Grade 다시 계산  # 바뀐 후 score 출력  print "Score changed."  print new\_student\_info  # 전체 리스트 다시 정렬  rearrage the list  return  else:  return  elif answer is final: # 기말고사일 때,  ask new score  if valid score:  student's final score = new\_score # 기말고사 점수 수정  student's average score = get\_average(  student\_mid, student\_final) # 평균점수 다시 계산  recalculate the grade # Grade 다시 계산  # 전체 리스트 다시 정렬  rearrange the entire list  return  else:  return  else:  return  print "NO SUCH PERSON." # for 문을 다 돌때까지 찾지 못하면 "NO SUCH PERSON."을 띄운다.  return |
|  |  |

|  |  |
| --- | --- |
| **Pseudo-algorithm add()** | |
|  | *// 프로그램에 필요한 변수들은 미리 선언해놓은 것으로 가정한다.*  def add():  existing\_id\_list  for student in stu\_list:  append id to existing\_id\_list # 이미 존재하는 학번 리스트를 만든다.  set input\_id by user\_input  # 이미 학번 존재하면 함수를 return 해버린다.  if input\_id in existing\_id\_list:  print "ALREADY EXISTS"  return  # 학번이 중복되지 않는 것이 확인되었으므로, 나머지 정보도 입력받는다.  set input\_name by user\_input  set input\_midterm\_score by int(user\_input)  input\_final\_score = int(user\_input)  # 평균과 학점을 계산한다.  calculate averge\_score = get\_average(input\_midterm\_score, input\_final\_score)  grade = get\_grade(averge\_score)  # 새로운 학생의 정보 리스트를 만든다.  set new\_student\_info\_list with  [input\_id, input\_name, input\_midterm\_score, input\_final\_score, averge\_score, grade]  # stu\_list에 추가한다.  stu\_list.append(new\_student\_info\_list)  # 전체 학생 리스트를 평균 순으로 다시 정렬한다.  rearrange\_entire\_stu\_list by average\_score  print "Student added."  return |
|  |  |

|  |  |
| --- | --- |
| **Pseudo-algorithm search\_grade()** | |
|  | *// 프로그램에 필요한 변수들은 미리 선언해놓은 것으로 가정한다.*  def search\_grade():  existing\_grade\_list = []  for student in stu\_list:  if student's id does not exist in in existing\_grade\_list  append student id to existing\_grade\_list # 이미 존재하는 학번 리스트를 중복없이 만든다.  # 찾고있는 학점을 입력받는다.  set grade\_to\_search by user\_input  # "A", "B", "C", "D", "F" 중 하나가 아니라면, return 한다.  if grade\_to\_search not in ["A", "B", "C", "D", "F"]:  return  # 학생들의 학점 중에 없다면, return 한다.  elif grade\_to\_search does not exist in existing\_grade\_list:  print "NO RESULTS."  return  else:  searching\_grade\_student\_list = []  for student in stu\_list:  if student's grade same with grade\_to\_search:  append student to searching\_grade\_student\_list  for student in searching\_grade\_student\_list:  print student info  return  return |
|  |  |

|  |  |
| --- | --- |
| **Pseudo-algorithm remove()** | |
|  | *// 프로그램에 필요한 변수들은 미리 선언해놓은 것으로 가정한다.*  def remove():  # 목록에 아무도 없을 경우, return 한다.  if list is empty:  print("List is empty.")  return  # 학번 리스트를 만든다.  existing\_id\_list = []  for student in stu\_list:  append student id to existing\_id\_list  # 학번을 입력받는다.  set input\_id by user\_input\_id  if input\_id not in existing\_id\_list:  print "NO SUCH PERSON."  return  else:  for student in stu\_list:  if student id same to input\_id:  stu\_list.remove(student)  break  print("Student removed.")  return |
|  |  |

|  |  |
| --- | --- |
| **Pseudo-algorithm quit()** | |
|  | *// 프로그램에 필요한 변수들은 미리 선언해놓은 것으로 가정한다.*  def quit():  ask yes\_or\_no and set save\_data\_yes\_or\_no by user\_input  if save\_data\_yes\_or\_no == "yes":  ask file\_name\_to\_write and set variable by user\_input  open file to write by file name : file\_name\_to\_write  for student in stu\_list:  print student info  write data to writing file  close file  return |
|  |  |

|  |  |
| --- | --- |
| **Pseudo-algorithm readline\_and\_append(readed\_file)** | |
|  | *// 프로그램에 필요한 변수들은 미리 선언해놓은 것으로 가정한다.*  def readline\_and\_append(readed\_file):  for line in readed\_file:  set first name from line  set full name from line  # ["20180001","Hong Gildong", 84, 73]  stu\_number\_name\_mid\_final\_list = [id, full name, mid\_score, final\_score]  # gildong = ["20180001","Hong Gildong", 84, 73] 형태로도 저장됨  first\_name = stu\_number\_name\_mid\_final\_list # 변수값을 리스트의 변수명으로 사용할 예정 -> 찾아봐야할 듯  # 중첩 리스트 형식으로도 저장  append stu\_number\_name\_mid\_final\_list to stu\_list  return |
|  |  |

|  |  |
| --- | --- |
| **Pseudo-algorithm get\_average(mid, final)** | |
|  | *// 프로그램에 필요한 변수들은 미리 선언해놓은 것으로 가정한다.*  def get\_average(mid, final):  return (mid + final)/2 |
|  |  |

|  |  |
| --- | --- |
| **Pseudo-algorithm get\_grade(score)** | |
|  | *// 프로그램에 필요한 변수들은 미리 선언해놓은 것으로 가정한다.*  def get\_grade(score):  if score >= 90:  return 'A'  elif score >= 80 and score < 90:  return 'B'  elif score >= 70 and score < 80:  return 'C'  elif score >= 60 and score < 70:  return 'D'  else:  return "F" |
|  |  |

|  |  |
| --- | --- |
| **Pseudo-algorithm print\_table\_row()** | |
|  | *// 프로그램에 필요한 변수들은 미리 선언해놓은 것으로 가정한다.*  def print\_table\_row():  print("Student" + "\t\t" + "Name" + "\t\t" + "Midterm" + "\t" +  "Final" + "\t" + "Average" + "\t" + "Grade")  print("-" \* 70)  return |
|  |  |

3. 프로그램 구조 및 설명

a) main()

- 프로그램을 실행하면, 실행되는 부분으로, show()부터 quit() 함수까지 7가지 함수를 포함한다. 처음 파일을 읽어올 때, sys.argv의 길이가 1, 즉 사용자가 .py 파일 뒤에 아무것도 입력하지 않았을 때에는 file\_name을 빈 문자열로 선언한다. Sys.argv의 길이가 1이 아니라면, 사용자가 입력한 문자열을 file\_name으로 선언한다.

- file\_name이 비어있는 문자열이면, students.txt를 열고, file\_name이 빈 문자열이 아닐 경우에는 os.path.exists()를 통해 경로에 파일이 존재하는지 판단하고 있으면 파일을 읽어온 다음, readline\_and\_append() 함수를 통해, stu\_list에 student 정보를 담는다.

- 사용자가 입력한 파일명을 가진 파일이 없다면, “File not Found”를 출력하고 프로그램을 종료한다.

- stu\_list를 for문을 돌려서 student를 하나씩 가져와서, student\_average\_score를 get\_average() 함수를 활용해 구하고, student\_grade를 get\_grade 함수를 활용해 구한다. 그 다음, student에 student\_average\_score와 student\_grade를 append 해준다.

* Stu\_list를 lambda 함수를 활용해서 student\_average\_score를 key로 써서 내림차순(reverse = True)으로 정렬한다.
* Print\_table\_row()를 통해, 표의 윗부분을 출력하고, for 문을 통해 stu\_list의 student의 정보를 한 줄씩 차례대로 출력한다.
* While(True)를 통해서 실행할 명령어를 받는데, upper\_input\_command로 사용자의 input을 대문자화하여 받는다.
* 각 upper\_input\_command에 매치되는 함수가 실행될 것이고, quit이 입력되면, break를 통해 반복문을 빠져나오게 된다.

b) show()

- stu\_list에 저장되어있는 반복문을 통해 student 요소를 한 줄씩 출력해준다. 0번 인덱스부터 출력하여, 평균이 높은 학생들부터 출력된다.

c) search()

- student\_id를 사용자에게 입력받고, stu\_list를 for문으로 돌면서 student\_id와 일치하는 student를 찾는다. Student\_id는 고유한 값이므로, for문을 돌다가 일치하는 것을 찾게되면 함수 자체를 종료시킨다.

- 일치하는 student\_id를 for문을 다 돌때까지 찾지 못하면, “NO SUCH PERSON”을 출력하면서 함수가 끝난다.

d) change\_score()

e) add()

f) search\_grade()

g) remove()

h) quit()

i) readline\_and\_append()

j) get\_average()

k) get\_grade()

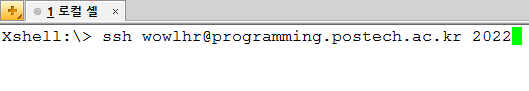
l) print\_table\_row()

* 프로그램을 실행하면 점의 좌표 <x, y>를 입력할 것을 요청한다. 사용자가 공백으로 구분하여 각 좌표를 입력하면 scanf를 통해 정수형 변수 x, y에 저장된다.
* 이 후, 위와 유사하게 직선의 계수 (a, b, c)를 입력할 것을 요청하며, 같은 방식으로 scanf를 통해 정수형 변수 a, b, c에 각각 저장된다.

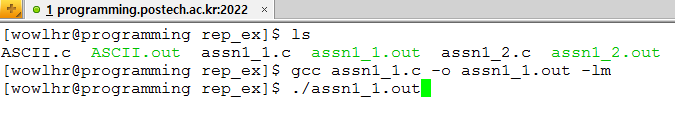
b) 점과 직선 사이의 거리 계산

* 입력을 마친 후 각 변수에 저장된 값을 통해 점과 직선 사이의 거리를 계산하게 된다. 이 때, 점과 직선 사이의 거리 공식을 바탕으로 분자(numerator)와 분모(denominator)로 구분하여 계산한다.
* 분자는 math 헤더에 포함된 절대값 함수 abs()를 활용하여, 아래의 결과를 저장한다.
* 분모는 math 헤더에 포함된 제곱근 함수 sqrt()를 활용하여, 아래의 결과를 저장한다.
* 마지막으로 분자와 분모를 나누어 거리 결과에 저장한다.
* 결과를 출력할 때, “%.2f” 출력포맷을 사용하여 소수점 아래 둘째 자리까지 표현되도록 한다.
* 또는 ASCII code table을 참조하여 각 문자에 해당하는 ASCII 코드를 찾을 수 있다.
* printf(“%c%c%c …”, .ASCII\_1, ASCII\_2, …); 포맷을 활용하여 출력한다.

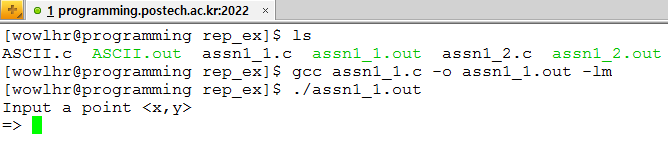
4. 프로그램 실행방법 및 예제



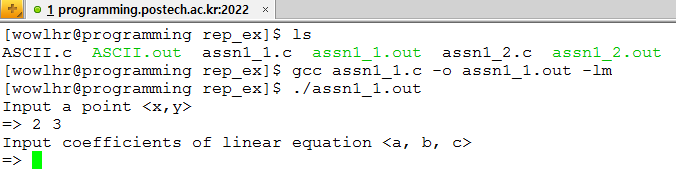
• Xshell 환경에서 위 명령어를 통해 비밀번호를 입력하고, 본인에게 할당된 리눅스 서버에 접속한다. 이 후 본 프로그램이 저장되어 있는 폴더로 이동한다.



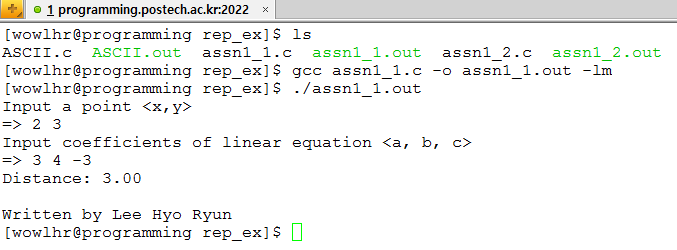
• 이동한 폴더에서 작성한 프로그램의 소스코드가 있는지 확인을 한 후 컴파일을 수행한다. 이 후, 컴파일이 완료된 파일을 실행시킨다.



• 위와 같이 점의 좌표를 요구하게 되고, 각 좌표를 공백으로 구분하여 입력을 완료한다.



• 점의 좌표 입력이 완료된 후, 직선 형성에 필요한 계수들을 요구한다. 점의 좌표 입력과 같은 방식으로 각 계수들을 입력해준다.



• 제시된 문제와 동일한 입력을 완료하였을 경우, 그 결과가 성공적으로 계산되어 실행예제와 동일한 결과를 출력하게 되었다.

• 또한, 프로그램이 종료되기 직전 ASCII 코드를 통한 이름이 정확히 출력됨을 확인할 수 있다.

5. 토론

* 문자의 ASCII 코드를 알아내는 프로그램을 작성할 때, scanf를 연속적으로 사용할 경우 정상적으로 동작하지 않는 문제가 발생하였다.
* 이는 문자를 입력 받을 때 사용한 ‘엔터’가 입력 버퍼에 남아있기 때문에 발생함을 알게 되었다.
* 문제를 해결하기 위해서는 ASCII 코드를 알아내는 프로그램을 필요한 만큼 반복 실행하거나, flushing을 통해 입력버퍼를 비워주는 작업(fflush(stdin))을 매 scanf 앞에서 행하는 방식이 요구된다.

6. 결론

* 본 과제에서는 prinf와 scanf를 사용하는 방법을 익히는데 유용했으며, ASCII 코드를 사용하는 과정을 통해 문자 및 정수 변수의 관계를 학습할 수 있게 되었다.

7. 개선방향

* 본 과제는 요구사항만 적절히 만족시켜주면 완성할 수 있기 때문에 프로그램 기능 자체의 개선보다는 향후 학습하게 될 함수의 사용 또는 반복문/조건문 등을 통해 프로그램이 더욱 유동적이 되도록 할 수 있을 것으로 예상한다.