**청년 AI · Big Data 아카데미 9기**

**프로그래밍과 문제해결**

**Project #1**

**담당교수: 윤은영**

**반: B반 2조**

**이름: 양명철**

**E-mail: jk00085@naver.com**

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

1. 문제의 개요

본 프로그램을 간략히 설명하면 다음과 같다.

* 사용자로부터 다음과 같은 입력을 받는다.

ex) show, search ,changescore, add, searchgrade, remove, quit

* 각 입력을 받으면 각 함수가 실행을 한다.

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

main

add

changescore

quit

remove

searchgrade

search

show

2. 알고리즘

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

|  |  |
| --- | --- |
| **Pseudo-algorithm for Grading** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  12 | def show(), search() ,changescore(), add(), searchgrade(), remove(), quit()  open( ‘data.txt’, “r”)  data.split(‘\t’)  for data  if avg >90:  llist.append(‘A’)  …  While true:  dir = input(“#”)  if dir == “show”:  show()  ….  Elif == ‘quit’: |
|  |  |

위의 의사 알고리즘을 Flowchart를 통해 표현하면 아래와 같다. 이 때 각 함수는 프로그램 구조 및 설명에서 설명한다.

main

show

search

changescore

show

add

searchgrade

remove

quit

3. 프로그램 구조 및 설명

show () 함수 : 학생들의 중간, 기말 평균 점수를 기준으로 내림차순 정렬을 하여 출력한다.

: 람다를 사용하여 sorted 함수를 사용하여 line\_list로 반환을 한다. line\_list[4] (총점을 기준으로) reverse = True로 적용하여 내림차순을 한다. 람다함수의 장점은 코드의 간결함 메모리의 절약이다.

def 함수이름(매개변수): lambda 매개변수 : 결과

return 결과

for문을 사용하여 print 함수를 사용하여 출력을 한다.

|  |  |
| --- | --- |
| **def show()** | |
| 1  2  3  4 | def show(line\_list):  list = sorted(line\_list, lamda, line\_list: line\_list[4], reverse = True)  for i in range( len(line\_list) ):  print(line\_list) |
|  |  |

search() 함수 : 학생 ID를 입력 받아 해당 ID를 출력하는 함수이다. 입력 받은 ID가 없으면 “NO SUCH PERSON”을 출력한다.

line\_list의 길이를 flag로 설정한다. 학생의 ID를 입력 받는다. for문을 돌면서 line[i][0]번째(ID 값을 나타냄)가 ID와 일치하는 경우 해당 line\_list[i](해당 학생 성적 정보)를 출력한다. 그리고 flag의 길이를 1을 감소를 한다.

만약 flag의 길이가 변하지 않았다면(즉, ID를 찾지 못했다면) “NO SUCH PERSON.”을 출력하여 찾지 못했다는 것을 출력한다.

|  |  |
| --- | --- |
| **def** search() | |
| 1  2  3  4  5  6  7  8  9  10 | def search (line\_list):  flag = len(line\_list)  ID = input()  for i in to len(line\_list):  if ID in line\_list[i][0]:  print(line\_list[i]  flag -=1  break  if flag == len(line\_list):  print(“NO SUCH PERSON”) |
|  |  |

changescore() 함수: 학생의 ID를 입력 받는다. 학생의 ID가 없으면(i가 len(line\_lis)t과 같으면) “NO SUCH PERSON”을 출력한다. 그리고 중간고사 혹은 기말고사를 선택한다. mid 혹은 final이 아니면 잘못이 되었다고 출력을 한다. mid 혹은 final을 입력 받는다. 대소문자를 구별하지 않기 때문에 .lower()를 사용하여 소문자로 바꾼다. 점수를 입력 받는다. 점수가 100을 초과하거나 0을 미만이면 잘못이 되었다고 출력을 한다. pint() 함수를 통해서 수정되기 전 리스트를 출력한다. 0에서 100의 숫자를 입력 받으면 line[i][2] (중간고사 점수) 혹은 line[i][3] (기말 고사 점수)를 수정을 한다. 그리고 평균 점수를 구하고 line\_list[i][4] (평균 점수)를 변경한다. 그리고 if문을 사용하여 학점을 변경한다. 그리고 “Score changed”라는 함수를 출력한다.

|  |  |
| --- | --- |
| **def** changescore () | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  56 | # 점수 수정  def changescore(line\_list):  flag = len(line\_list); ID = input("Student ID: ")  for i in range(len(line\_list)):  if ID in line\_list[i][0]:  se = input("Mid/Final? ").lower()  if se == 'mid':  mscore = int(input("input new score: "))  if mscore > 100 or mscore < 0:  print("wrong")  break  else:  print(line[i])  line\_list[i][2] = mscore  flag -=1  break  elif se == 'final':  fscore = int(input("input new score: "))  if fscore > 100 or fscore < 0:  print("wrong")  break  else:  print(line[i])  line\_list[i][3] = fscore  flag -=1  break  else:  print("wrong ")  break  if flag == len(line\_list) -1:  avg = (int(line\_list[i][2]) + int(line\_list[i][3]))/2  line\_list[i][4]= avg  if avg >= 90:  line\_list[i][5]="A"  elif avg >= 80:  line\_list[i][5]="B"  elif avg >= 70:  line\_list[i][5]="C"  elif avg >= 60:  line\_list[i][5]="D"  else :  line\_list[i][5]="F"  print(line[i])  if flag == len(line\_list):  print("NO SUCH PERSON.") |
|  |  |

add()함수: 학생 ID를 입력받는다. for문을 사용하여 line[i][0] (학생ID)가 존재하면 “ALREADY EXISTS.”을 출력한다. break문을 사용하여 for문을 빠져나온다. ID가 존재하지 않으면 이름과 중간고사, 기말고사 점수를 입력 받는다. 이름과 중간, 기말은 대소문자를 구별하지 않기 때문에 .lower()를 사용하여 소문자로 바꾼다. 그리고 중간고사 점수와 기말고사 점수의 평균을 avg로 받는다. 그리고 ID, 이름, 중간고사, 기말고사, 평균점수, 학점을 list에 append를 한다. 그리고 “Student added.”라는 것을 출력한다. 그리고 break를 사용하여 for문을 출력한다.

|  |  |
| --- | --- |
| **def** add() | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | def add(line\_list):  ID = input("Student ID: ")  for i in range(len(line\_list)):  if ID in line\_list[i][0  print("ALREADY EXISTS.")  break  else :  newName = input("Name: ").lower()  newMScore = input("Midterm Score: ")  newFScore = input("Final Score: ")  avg = (int(newMScore) + int(newFScore)) /2  if avg >= 90:  line\_list.append([ID, newName, newMScore, newFScore, avg,"A"])  elif avg >= 80:  line\_list.append([ID, newName, newMScore, newFScore, avg,"B"])  elif avg >= 70:  line\_list.append([ID, newName, newMScore, newFScore, avg,"C"])  elif avg >= 60:  line\_list.append([ID, newName, newMScore, newFScore, avg,"D"])  else :  line\_list.append([ID, newName, newMScore, newFScore, avg,"F"])  print("Student added.")  break |
|  |  |

searchgrade() 함수: 입력 받아 학점을 찾는다. 대소문자를 구별하지 않기 때문에 .lower()를 사용하여 소문자로 바꾼다. line\_list의 길이를 found로 설정한다. Grade가 A, B, C, D, F 가 아니면 잘못되었다고 출력을 한다. 학점이 A, B, C, D, F이면 해당 line\_list[i]를 출력한다. 만역에 없으면 아니면 “No RESULT”를 출력한다.

|  |  |
| --- | --- |
| **def** searchgrade () | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | # 학점 검색  def searchgrade(line\_list):  grad = input("Grade to search: ").lower()  found = len(line\_list)  for i in range(len(line\_list)):  if grad == 'A' or grad == 'B' or grad == 'C' or grad == 'D' or grad == 'F':  if grad in line\_list[i][5]:  print(line\_list[i])  found = found- 1  else:  found = -1  print("wrong")  break  if found == len(line\_list):  print("No RESULT") |
|  |  |

remove() 함수: 학생ID를 입력 받는다. line\_list의 길이를 flag로 저장을 한다. 만약 line\_list가 비었으면 “List is empty”를 출력한다. for문을 돌면서 ID가 line\_list[i][0] (학생 ID) 있으면 del을 사용하여 삭제를 한다. 그리고 “Student removed.”를 출력한다. 만약 해당 ID가 없으면 “NO SUCH PERSON.”을 출력한다.

|  |  |
| --- | --- |
| **def** remove () | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | def remove(line\_list):  if line\_list == []:  print("List is empty")  else:  ID = input("Student ID: ")  flag = len(line\_list)  if line\_list == []:  print("List is empty")  for i in range(len(line\_list)):  if ID in line\_list[i][0]:  del line\_list[i]  print("Student removed.")  flag -=2  break  if flag == len(line\_list):  print("NO SUCH PERSON.") |
|  |  |

quit() 함수 : line\_list를 전달받는다. 람다함수를 사용하여 sorted함수를 사용하여 line\_list로 반환을 한다. 그리고 Filename을 입력받는다. Open( filename, “w”)를 사용하여 write를 사용한다. for문을 사용하여 line\_list를 저장을 한다. f.close()로 닫는다.

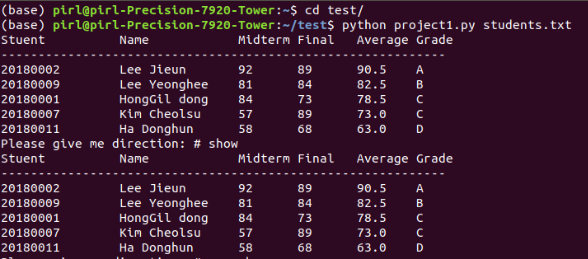
|  |  |
| --- | --- |
| **def** quit () | |
| 1  2  3  4  5  6  7  8 | def quit(line\_list):  # 파일 쓰기  line\_list = sorted(line\_list, key=lambda line\_list: line\_list[4], reverse = True)  newFilename = input("File name:")  f = open( newFilename, "w")  for i in range(len(line\_list)):  f.write("%s\t%s\t%s\t%s\n" %tuple(line\_list[i][0:4]))  f.close() |
|  |  |

main 함수: grade.txt를 address로 저장을 한다. Open(address “r”)을 활용하여 score 점수에 저장을 한다. Tab을 기준으로 split한다. line\_list를 빈 리스트를 만든다. 중간고사 점수와 기말고사 점수의 평균을 avg에 구한다. 그리고 avg를 append를 한다. 그리고 점수에 맞게 학점을 부여한다. 처음 show함수를 사용하여 출력한다. while True를 사용하여 무한 루프를 돕니다. 사용자로부터 명령을 입력 받는다. 대소문자를 구별하지 않기 때문에 .lower()를 사용하여 소문자로 바꾼다. Show를 입력받으면 show함수를 호출하여 출력한다. Search를 입력 받으면 search함수를 호출한다. changescore를 입력받으면 changescore를 호출한다. add를 입력 받으면 add함수를 호출한다. searchgrade함수를 입력받으면 searchgrade함수를 호출한다. Remove를 입력 받으면 remove함수를 호출한다. quit함수를 입력받으면 save를 할 것인지 물어본다. yes이면 quit함수를 호출한다. 그리고 “File Saved”를 출력한다. 만약 그렇지 않으면 저장하지 않고 종료를 한다.

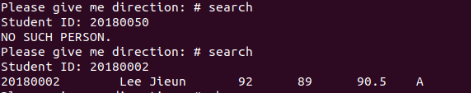
|  |  |
| --- | --- |
| **def** main () | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  56  57 | import sys  def main():  address = 'last.txt'  score = open( address, "r")  data = score.read().split("\n")  line\_list = []  for line in data[:-1]:  line\_list.append(line.split("\t"))  for i in range(0, len(line\_list)):  avg = (int(line\_list[i][2]) + int(line\_list[i][3]))/2  line\_list[i].append(avg)  if avg >= 90:  line\_list[i].append("A")  elif avg >= 80:  line\_list[i].append("B")  elif avg >= 70:  line\_list[i].append("C")  elif avg >= 60:  line\_list[i].append("D")  else :  line\_list[i].append("F")  show(line\_list)  while True:  dir = input("Please give me direction: # ").lower()  if dir == "show":  show(line\_list)  elif dir == 'search':  search(line\_list)  elif dir == 'changescore':  changescore(line\_list)  elif dir == 'add':  add(line\_list)  elif dir == 'searchgrade':  searchgrade(line\_list)  elif dir == 'remove':  remove(line\_list)  elif dir == 'quit':  save = input("Save data?[yes/no] ")  if save == 'yes':  quit(line\_list)  print("File Saved")  break  else:  print("Not saved")  break  score.close() |
|  |  |

4. 프로그램 실행방법 및 예제

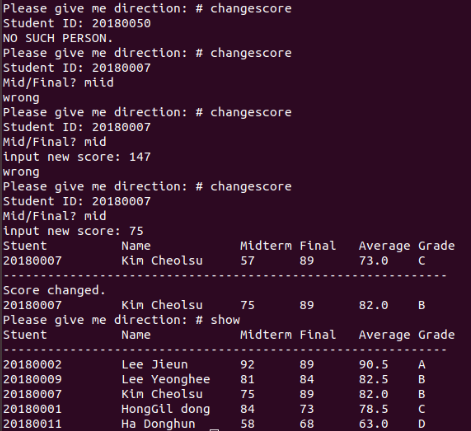
프로그램 시작 및 show() 함수



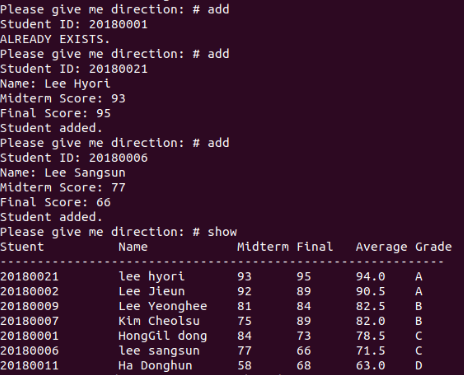
search() 함수



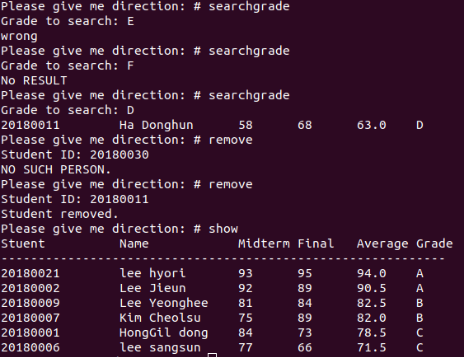
changscore() 함수



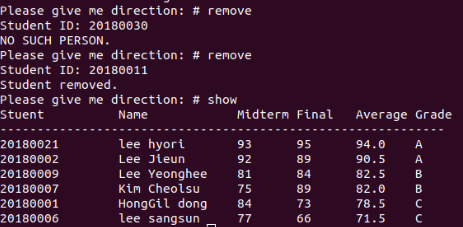
add() 함수



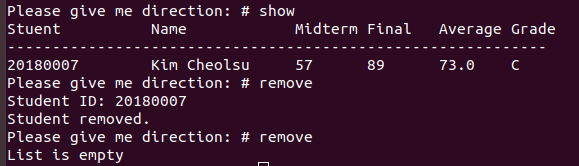
searchgrade()



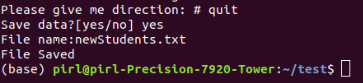
remove() 함수



리스트가 비어 있는 경우



quit() 함수



5. 토론

- 프로그램의 예외 처리하는 부분이 어려웠다. search함수를 구현하는데 for문을 사용하여 if, else를 사용하였는데, for문이 돌아가면서 발견을 하였지만, else문에 들어가서 예외처리 출력을 하였다.

- 이를 통해 flag 혹은 found를 사용하여 if문이 들어갔을 때 감소하는 방향을 사용하였다.

그 밖에 다른 예외처리를 하는 부분에 고민을 하였다. 잘못 입력을 받았을 때, ‘wrong’을 출력하였다. 대소문자를 상관없이 표시하는 부분도 고민을 하였다. 대소문자를 소문자로 바꾸어서 소문자에 대해서 입력 받도록 하였다. (.lower()함수 사용)

6. 결론

본 과제는 리스트 또는 딕셔너리를 사용하여 구현하였다. 리스트의 인덱스를 하는 방법, 함수를 구현하여 인자를 전달하는 방법, for문, if/elif/else문, while문 등 다양한 파이썬 문법을 익힐 수 있었다. 또한 파일 입출력과 파일 쓰기에 대해서 공부하였다.

7. 개선 방향

본 과제는 요구사항만 만족시키면 완성이 될 수 있었다. 요구 조건은 리스트 또는 딕셔너리 타입으로 구현을 하는 것이었다. 다음 번 과제를 하게 되면 딕셔너리 타입을 통해서 구현하고 싶다. 또한, 객체지향형 Class를 통해서 구현을 하는 방법도 적절할 것같다.