**청년 AI/BIGDATA 아카데미 12기**

**성적 관리 프로그램**

**파이썬 1주차 과제**

**담당교수: 윤은영**

**반: A반**

**조: 1조**

**이름: 노선호**

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

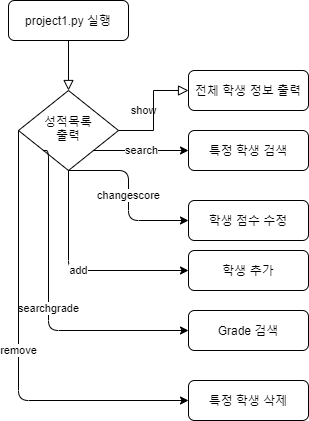
**Problem 1: 성적 관리 프로그램**

1. 문제의 개요

본 프로그램을 간략히 설명하면 다음과 같다.

* .txt 파일로부터 데이터를 읽어서 문자열로 리스트(data)에 입력받는다.
* 읽어온 데이터들로부터 평균과 학점을 계산및 저장하고
* 성적보기,학생 검색, 점수 수정,학생 추가, 성적 검색, 학생삭제의 기능을 갖고 또한 저장할 수 있다.

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



* 입력부: 사용자가 원하는 명령어를 검색시 해당 기능 처리부로 이동.
* 처리부:
  + show: show입력시 data 리스트 저장되어 있는 전체목록을 평균 점수를 기준으로 내림차순 출력. (평균 점수는 소수점 이하 첫째 자리까지 표시)
  + search: search 입력시 검색하고자 하는 학생의 학번을 입력받고 이름, 중간고사 점수, 기말고사 점수, 평균, 학점을 출력한다.
  + changescore: changescore 를 입력시 목록에 저장된 학생 중 1명의 중간고사 or 기말고사의 점수를 수정한다.
  + add: add 입력시 학생의 학번, 이름, 중간고사 점수, 기말고사 점수를 차례로 요구 받고 에러가 안날시 “Student added”를 출력후 종료한다.
  + searchgrade: searchgrade 를 입력시 해당 성적의 학생을 모두 출력해준다.
  + remove: remove를 입력시 해당 학생의 학번을 입력받고 학생의 목록에 존재여부를 확인후 삭제 또는 “Student removed”를 출력한다.
  + quit: quit입력시, data를 저장 여부를 확인하고 프로그램을 종료한다.
* 출력부: 출력부는 맨 위 인덱스와 인덱스및 데이터 구분선을 따로 함수로 정의하고 나머지는 처리부에서 처리후 바로 출력하는 형태로 구현하였다.

2. 알고리즘

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

|  |  |
| --- | --- |
| **Pseudo-algorithm for dot-line distance** | |
| 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 | *// 프로그램에 필요한 변수들은 미리 선언해놓은 것으로 가정한다.*  def frame() // 데이터 출력을 위한 인덱스와 구분선 출력 함수.  print( “ student name midterm final average grade”)  print(“--------------------------------------------------”)  def show(data):  frame()  data.sort(key=lambda e:e[4],reverse=True) 학생 평균을 오름차순 정렬  for( for(print(data) ) ) //이중포문을 통해 학생데이터 출력  def search(data)  s\_id = input()  for( data[x][0] 전체 탐색):  if data[x][0]==s\_id:  frame()  for(찾은 학생의 데이터 반복)  print()  else: print(“NO SUCH PERSON”)  def changescore\_data(s\_id,new\_score,data,check\_m\_f)  frame()  for(data학생수만큼):  if data[n][0] == s\_id:  점수 변경 전 데이터 출력  for (data학생수만큼):  if check\_m\_f== mid: //mid, Average, Grade 변경  else: check\_m\_f ==final: //final , Average, Grade 변경  for(data학생수만큼):  if data[x][0] ==s\_id:  학생 데이터의 새점수  def changescore(data)  cnt =0  s\_id = input()  for(data[x][0] 전체 탐색):  check\_m\_f = input()  if check\_m\_f ==”mid” or”final”:  new\_score = input()  if 0<=new\_score <=100:  changescore\_data(s\_id,new\_score,data,check\_m\_f)  if cnt==0:  print(“NO SUCH PERSON”)  def searchgrade(data):  find\_grade = []  sc\_grade = input()  if sc\_grade ==”A~F”:  for (학생전체)  if sc\_grade ==학생의 성적:  find\_grade.append(개인학생리스트 인덱스)  if find\_grade ==[]:  print “No results”  else:  frame()  for(find\_grade에 담겨있는 개인학생리스트의 인덱스)  for (해당학생의 정보)  print 학생의 정보  def add\_data(s\_add,data,len\_max):  add\_name,add\_mid,add\_fin =input()  add\_avg = Mean(add\_mid,add\_fin)  add\_grd = grade(add\_avg)  if len(add\_name) > len\_max:  len\_max = len(add\_name) //// 전체학생 데이터 출력시 줄 맞춤을 통한 출력을 위하여  add\_data= [s\_add, add\_name +((len\_max - len(add\_name))\* ‘ ‘),add\_mid,add\_fin,add\_avg,add\_grd]  data.append(add\_data)  print(“Student added)  def add(data,len\_max):  cnt =0  s\_add = input()  for (학생 데이터):  if 학생데이터[0]==s\_add:  print(“ALREADY EXISTS)  cnt+=1  if cnt !=1:  add\_data(s\_add,data,len\_max)  def Remove(data):  cnt = 0  if len(data) ==0:  print(“List is empty”)  else:  s\_id= input()  for(전체학생):  if s\_id in 전체학생:  data.remove(s\_id 데이터)  prin(“Student removed”)  else:  cnt+=1  if cnt == len(data) and len(data) !=0:  print(“No SUCH PERSON.”)  def quit(data)  save\_data=input() //데이터 저장 유무 판단  if save\_data == ‘yes’:  new\_fname = input()  new\_fname으로 데이터 저장  def Mean(mid,final):  mn = (int(mid) +int(final))/2  return round(mn,1)  def grade(a)  if float(a) >= 90:  grd = 'A'  elif float(a) >= 80:  grd = 'B'  elif float(a) >= 70:  grd = 'C'  elif float(a) >= 60:  grd = 'D'  else:  grd = 'F'  return grd  def main()  data = [] //학생 정보담을 리스트 선언  len\_max = 0 // 학생 이름별 길이가 달라서 최대 길이에 맞추기 위하여 선언  with open(“data\_file”,”r)  for( data\_file 라인별)  data.append(파일 한줄씩)  for (전체학생수)  if len(data[x][1] > len\_max)  len\_max = len(data[x][1])  for (전체학생수)  if len(data[x][1] < len\_max)  data[x][1] += ‘ ‘\*(len\_max - len(data[x][1]))  for(전체학생수)  data.append(Mean(mid,final)  data.append(grade(avg))  show()  while(True):  a=input().strip().lower() // 명령어 입력받는 부분  if a == "show":  show(data)  elif a == "search":  search(data)  elif a == "changescore":  changescore(data)  elif a == "add":  add(data,len\_max)  elif a == "remove":  Remove(data)  elif a == "quit":  quit(data)  break  elif a == "searchgrade":  searchgrade(data)  if \_\_name\_\_ == "\_\_main\_\_":  main() |
|  |  |

3. 프로그램 구조 및 설명

a) 학생의 학번 , 이름 , 중간고사 성적, 기말고사 성적 입력 및 평균, 학점 계산(main())

* 프로그램을 실행하면 학생 이름을 저장할 data(리스트) 선언되고 데이터 출력시 줄 정렬을 위해 len\_max(학생이름의 최대길이)를 default(0)로 선언한다.
* python project1.py 로 실행하면 기본 설정인 “studen.txt”로 파일을 읽어온다. 하지만 python project1.py [argv1] 로 실행하면 argv1에 입력한 데이터 파일을 불러온다.
* 읽어온 데이터파일을 한줄씩 \t을 기준으로 map을 이용하여 list로 data.append()를 통해 저장한다.
* 학생 이름이 개개인별 다르기에 최대길이를 반복문을 통해 len\_max에 저장한다.
* 저장된 len\_max를 이용하여 data에 입력된 학생의 이름을 이름+공백으로 정렬해준다.
* mean()과 grade()를 이용하여 평균과 학점을 계산하여 학생데이터에 학생데이터에 추가해준다.
* 마지막으로 전체 학생의 데이터를 출력해주며

(show,search, changescore,add,remove,searchgrade) 명령어를 계속 입력받고 quit 명령어를 사용시 프로그램 종료를 할 수 있다.

b) 학생 데이터 출력을 위한 틀 ( frame())

* 학생의 데이터를 출력할때 호출하여 사용하는 함수로 데이터의 인덱스와 구분선을 출력해준다.

c) 전체학생 데이터 출력(show())

* show 명령어를 입력하면 실행되는 함수이다. 파라미터로 data(학생데이터)를 받아온다.
* 출력을 위한 기본 인덱스와 구분선을 frame() 함수 호출을 통하여 출력한다.
* 학생 평균(오름차순)으로 출력하기위하여 sort(key=lambda e: e[4], reverse=True)함수로 정렬시켜준다.
* 정렬을 마친후 이중for문을 사용하여 각학생들의 데이터를 \t\t 기준으로 출력해준다.

d) 특정학생 검색(search())

* s\_id에 학생의 학번을 입력받은후 모든데이터에서 s\_id와 일치하는 값을 찾고 일치하면 해당학생의 데이터를 \t\t 기준으로 출력한다.
* s\_id와 일치학생이 없으면 “NO SUCH PERSON” 를 출력하게 된다.

e) 학생점수 수정기능(changescore(),changescore\_data()):

* s\_id(학생학번)을 입력받은후 학생의 일치하는 정보가 있는지 없는지 검정을 하기위해 cnt 값을 0으로 선언한다.
* 학생데이터에서 입력받은 s\_id와 일치하는 학생의 유무를 확인하고 만약 없다면 점수수정 기능은 종료가 되며 명령어를 입력받을 수 있는 단계로 이동한다.
* s\_id와 일치하는 학생이 있다면 mid or final 중 변경할 데이터를 입력받는다. 그후 변경할 점수를 입력받아 점수가 0에서 100사이 값인지 판별후 없다면 기능종료 있다면 changescore\_data() 함수를 호출하여 데이터 변경을 하게된다.
* changescore\_data()함수는 frame()을 통하여 인덱스와 구분선을 출력하고 변경하기전 기존 학생의 점수를 출력한다. 출력을 마치고 해당학생의 리스트로 찾아가 변경할 점수가 ‘mid’인지 ‘final’인지 판단후 저장하게 된다. 이때Mean()과 grade() 함수를 이용하여 평균과 학점도 같이 수정하게 된다.
* 수정을 마치고 “Score changed.” 라는 문장이 출력되며 수정기능이 마무리 되고 명령어 받는 부분으로 돌아가게된다.

f) 특정학점 학생 필터링 기능(searchgrade()):

* 입력하는 학점의 학생의 데이터를 저장하기 위하여 find\_grade라는 빈 리스트를 선언하였다. 검색을 원하는 학점을 입력받는다. 이 때 입력받은 학점을 기준인 학생들의 인덱스를 find\_grade에 저장하게 된다. 만약 저장된 학생의 인덱스가 없다면 “NO RESULTS.”를 출력하고 기능이 종료된다.
* 저장된 학생의 인덱스가 있다면 해당학생 데이터의 출력을 위하여 frame()을 우선적으로 호출하고 find\_grade에 저장되어있는 인덱스의 학생데이터를 data에서 출력해준다.

g) 학생데이터 추가기능(add(),add\_data())

* 해당 학생의 존재여부를 파악하기 위하여 cnt를 선언하였다.
* 입력하고자 하는 학생의 학번을 s\_add로 입력받고 학생데이터에서 s\_add(학번)의 유무를 판단후 존재하면 “ALREADY EXISTS.”를 출력하고 cnt+=1하며 추가기능이 종료된다.
* 만약 존재하는 학생이 없다면add\_data를 호출하고(s\_add,data,len\_max)파라미터를 전달해준다.
* add\_data()함수에서는 이름,중간점수,기말점수를 입력받고 Mean()과 grade()함수를 통하여 평균과 학점을 계산후 학생데이터에 추가해준다. 완료되면 “Student added’가 출력되고 해당 기능은 종료되며 명령어를 칠 수 있는 부분으로 이동한다.

h) 학생데이터 제거기능(Remove())

* 학생의 데이터유무 판단을 위하여 cnt를 선언하였다. 만약 학생의 데이터 길이가 0 이면 “List is empty”를 출력하고 해당 기능을 종료하며 명령어 입력 부분으로 간다.
* 만약 학생의 데이터가 존재한다면 s\_id에 학생의 학번을 입력받고 해당학생의 데이터로 이동하여 해당데이터를 remove를 통하여 제거하게 된다. 제거후 “Student removed”를 출력한다.
* 만약 학생의 데이터가 존재하지만 입력받은 학생의 학번이 데이터에 존재하지 않다면 “NO SUCH PERSON”이 출력되고 해당 기능이 종료되며 명령어 입력부분으로 돌아간다.

i) 프로그램 종료부분(quit())

* quit 명령어를 입력하면 실행되고 현재까지 변동된 데이터의 저장유무를 입력받는다.
* ‘yes’를 입력했을시 저장할 파일 이름을 입력받고 학생의 데이터가 저장되며 해당기능을 종료하게된다.
* ‘no’를 입력했을시 해당 프로그램이 명령어 입력 부분으로 가지 않고 바로 종료된다.

j) 학생의 평균계산(Mean())

* 학생의 데이터를 입력받아올 때 혹은 변동사항이 있을 때 호출되는 함수로 중간점수(mid)와 기말점수(final)을 파라미터로 입력받아 계산후 소수점 한자리로 리턴시켜준다.

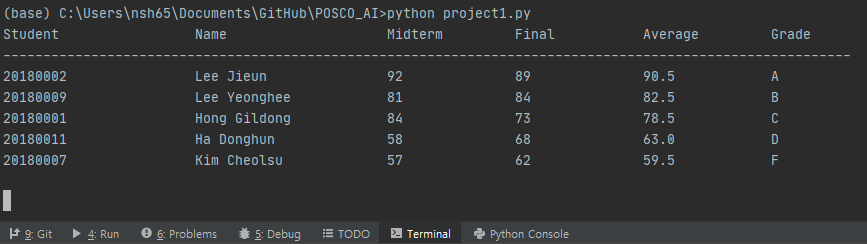
k) 학생의 학점계산(grade())

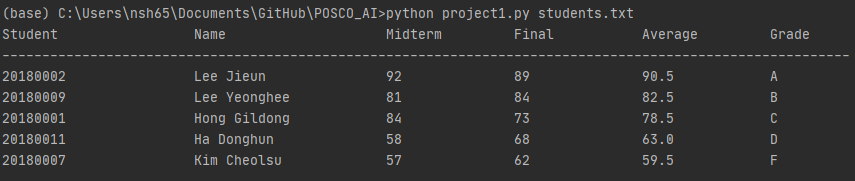
* 학생의 평균을 기준으로 90점 이상시 “A” 80점이상시 “B” 70점이상시”C” 60점 이상시 “D” 그 이하 점수는 “F”로 학점을 문자로 리턴해주는 함수이다.

|  |  |
| --- | --- |
|  | |
|  |  |

4. 프로그램 실행방법 및 예제

1. 프로그램 실행방법 및 실행시 상태

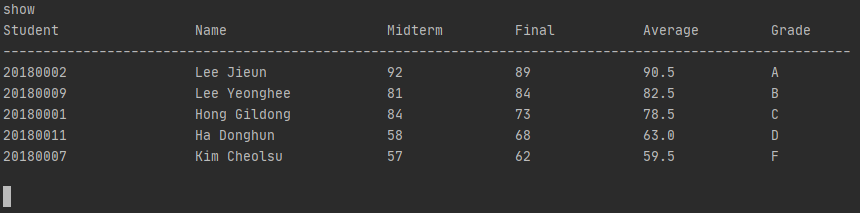




• 해당 환경은 WINDOWS 10 PyCharm 터미널로 진행되었다.

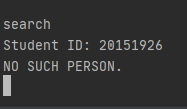
python이 깔려있는 환경에서 python project1.py 혹은 python project1.py [파일명]을 입력하여 실핸한다. 실행되면 바로 명령어를 받는것을 확인할 수 있다.

1. 전체학생 정보출력 기능

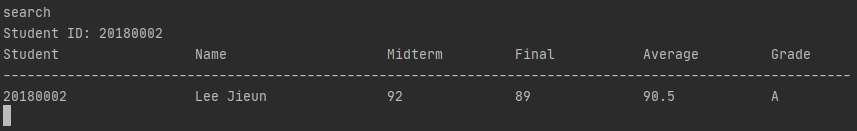


위 사진과 같이 show 명령어를 입력하면 전체학생 정보를 확인 할 수 있으며 바로 명령어 입력을 받을 수 있다.

1. 특정 학생 검색

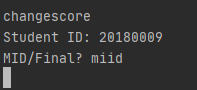


• 명령어 search를 입력받고 바로 학생의 학번을 입력받는다. 하지만 데이터에 20151926이라는 학번이 없기때문에 NO SUCH PERSON. 을 출력후 다시 명령어 받는 부분으로 돌아온 상태이다.

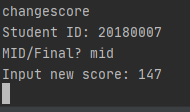


명령어 search를 입력받고 학생의 학번을 입력받은후 데이터에 존재하는 학번일 경우 위와 같이 특정학생의 데이터가 출력된다.

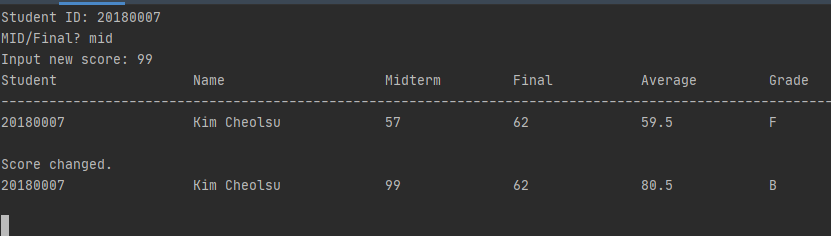
1. 점수 수정



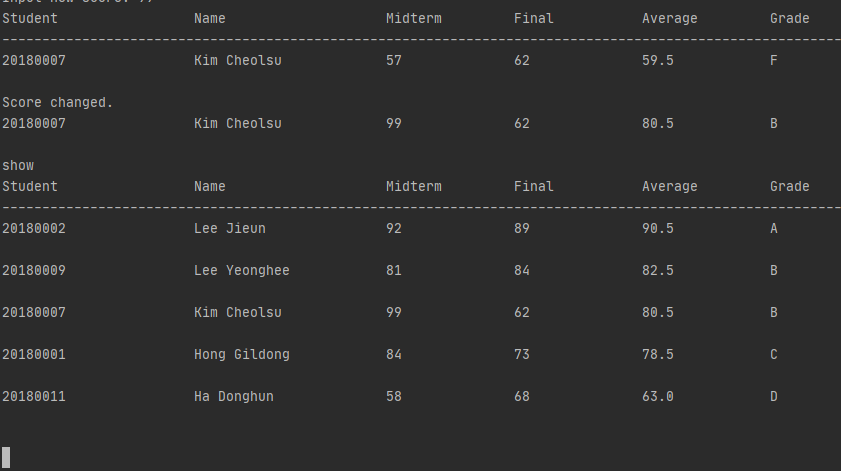
mid/final 외의 값이 입력된 경우 점수수정 기능이 종료되고 명령어 치는 부분으로 넘어간다.



mid/final을 입력하면 점수 입력을 받는데 새로운 점수가 0에서 100사이 이면 기능이 종료되고 명령어 치는 부분으로 넘어간다.

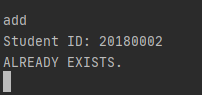


정상적으로 변경이 되면 이전 점수가 출력되고 변경되었다는 “Score changed”가 출력되며 변경된 학생의 데이터가 출력된다.



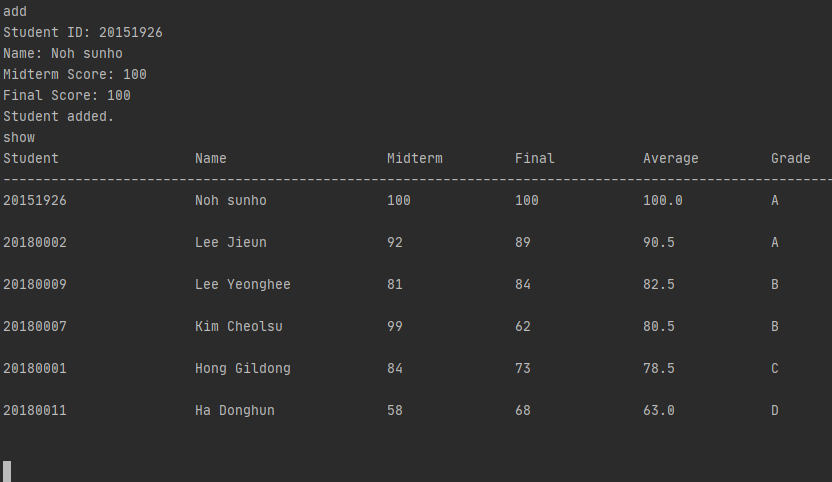
변경된 후 show 명령어를 이용하여 데이터가 잘 저장되어있는지 확인하는 사진이다.

1. 새로운 학생 데이터 추가



add 명령어를 입력하고 이미 있는 학생의 테이터를 입력하면 위와같이

ALREADY EXISTS. 가 출력된다.

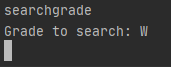


새로운 학번을 입력하고 이름과 중간고사/기말고사 점수를 입력하면

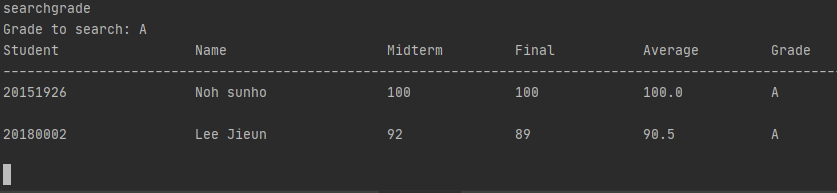
위 사진과 동일하게 Student added.가 출력되며 show로 확인결과

데이터가 제대로 들어가 있음을 확인할 수 있다.

1. 특정 학점 학생 검색기능

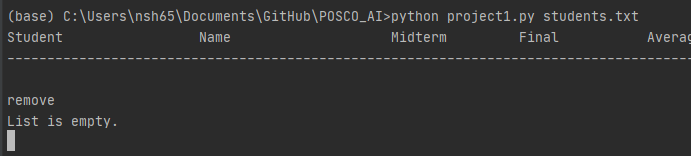


위와 같이 searchgrade 명령어를 입력하면 학점을 입력받는데 여기서 A,B,C,D,E,F 이외의 값이 들어가면 위 사진과 동일하게 종료되고 명령어를 받는다.

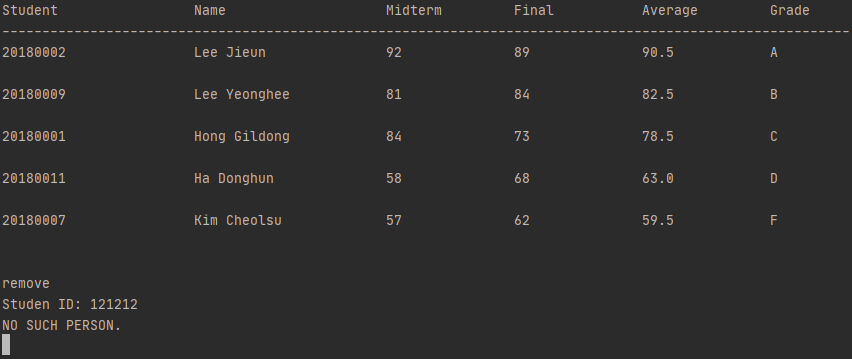


위와 같이 정상적인 학점을 입력하면 특정학점의 학생들의 정보가 성적기준 오름차순으로 출력된다.

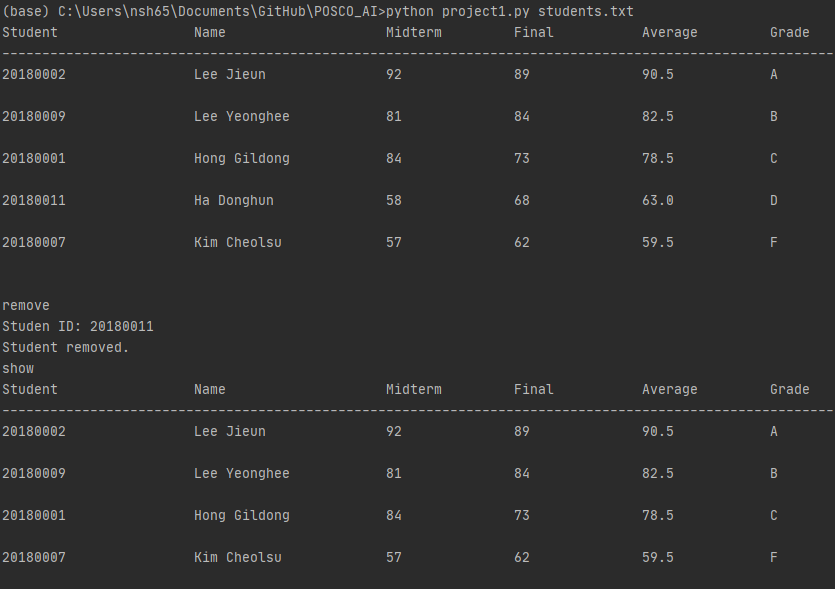
1. 특정 학생 데이터 삭제



위 사진은 학생데이터가 아무도 없는 것을 가정하기위하여 students.txt 파일을 빈 파일로 놓고 실행한 사진이다. remove 명령어를 입력하면 List is empty라고 뜨는 것을 볼 수 있다.

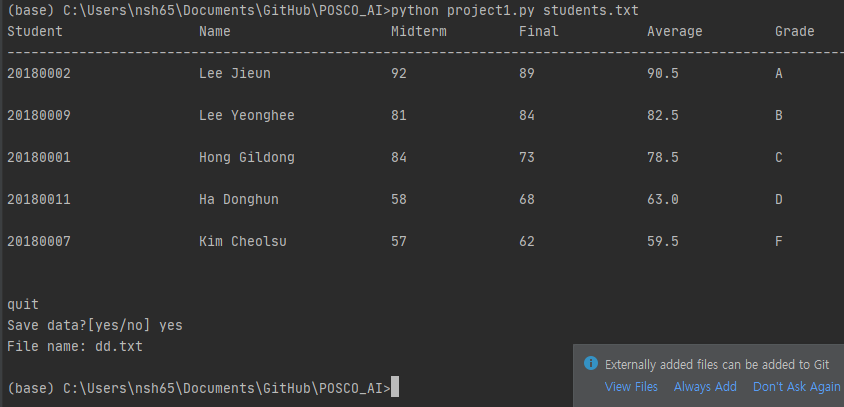


위사진은 입력한 학생의 학번이 데이터에 없을 경우 “NO SUCH PERSON.”를 출력하는 모습이다.

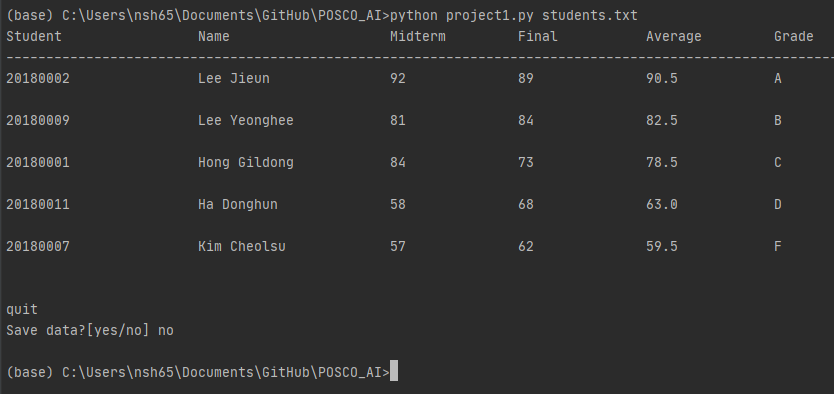


위 사진은 학생의 학번이 데이터에 존재 했을 때 remove 명령어로 학생의 학번을 받고 삭제시 “Student removed.”를 출력하고 show를 했을 때 데이터에 학생의 정보가 삭제 됨을 알려준다.

1. 프로그램 종료



위 사진은 quit 명령어를 입력했을 때 yes라고 하고 새로운 파일의 이름을 입력받고 프로그램 종료한 모습이다. 오른쪽 밑 알림으로 새로운 파일이 추가를 알 수 있다.



위 사진은 quit 명령어를 받고 저장하지 않는 no를 입력하였을 때 저장하지 않고 바로 프로그램이 종료됨을 알수 있다.