Python project

국민대학교 컴퓨터공학부

20123375 성주혁

UI

모든 라벨과 버튼의 이름은 먼저 리스트를 작성한 후 반복문으로 생성하였다.

버튼의 command 로 연결되는 기능들 또한 계산기의 수식과는 다르게 각자 고유한 기능을

가졌으므로 하나의 click 함수를 만들어 if문으로 기능을 구분하는 것이 아닌 함수를 리스트에 넣어 (리스트에는 모든 타입이 들어갈 수 있다.!!) for 문으로 command를 구별해 주었다.

데이터 관리

데이터의 저장은 기본적으로 StudentData Class에 의해서 이루어지나 실제로 하나의 객체 생성 후 그 안의 info 라는 리스트에 학생의 정보가 저장되도록 하였다.

StudentData Class의 funcdtion은 크게 insertInfo / deleteInfo / PrintData 나뉘어 있는데

StudentData Class 클래스의 리스트에 정보를 삽입하고 삭제하는 기능을한다. printData의 경우 실제로 tkinter window에 출력되는 것이 아닌 쉘에 출력되는데 이는 단위 테스트를 목적으로 정의된 함수이다.

insertInfo function은 리스트의 append 함수를 사용해서 index / name / score 을 순서대로 추가하는 기능을 가지고 있다.

deleteInfo function은 주어진 번호의 학생정보를 리스트에서 삭제하는데 삭제 후 번호가 자동정렬 되게 하기 위해서 제거하려는 번호 이전의 번호를 1씩 증가시킨 후 리스트의 pop함수를 사용해서 index / name / score를 제거하였다. 그 후 모든 번호를 1감소시킴으로써 중간의 번호를 제거하더라도 번호가 일정하게 나열되도록 구현하였다.

위의 두 함수는 StudentData의 info (list) 값에 insert / delete 하는 기능을 가진 함수들로 이러한 값의 tkinter window 표시는 view.py 파일에서 구현하였다.

View.py에서 학생정보를 저장한 StudentData 클래스의 인스턴스 이름은 allStudent 이다.

View.py 파일은 tkinter / StudentData / MakeError 을 import 하는데 StudentData는 학생정보를 저장하는 info list 를 가진 클래스를 가지고 있고 MakeError 는 사용자가 정의한 예외 클래스를 가지고 있다.

함수 소개

Insert / delete / sava / openf / numsort / namesort / downsort / upsort

Button command에 함수의 리스트를 반복문으로 전달한 8개의 함수이다.

Insert 함수는 2개의 버튼으로부터 인자를 받아 AllStudent.insertInfo의 인자로 전달하는데 각각의 버튼 또한 리스트로 만들었다.

entInput[0] => 이름을 입력 받는 버튼 / entInput[1] => 점수를 입력 받는 버튼

Allstudent.info에 인자를 전달하기에 앞서 이름과 점수가 공백인지 / 이미 동일 이름이 존재하는지를 검사한다.

검사 결과 이상이 없으면 scoreCheck함수를 통해서 entInput[1].get()의 값이 int 형으로 변환 가능한지 (점수의 값으로 숫자가 입력되었는지 검사한다. 또한 문법적으로는 이상이 없으나 점수가 양수가 아닌 경우에도 raise ValueError을 통해서 예외가 발생하도록 구현하였다.

이름과 점수가 정상적으로 입력 되었을 때 두 값을 allStudent.insertInfo의 인자로 전달한다.

리스트의 추가와는 별개로 이를 tkinter window에 출력해주어야하는데 allStudent.index , entInput[0].get() , entInput[1].get()을 output.insert의 인자로 전달하여 window에 출력한다.

Delete 함수는 번호를 입력받아 그 번호의 학생정보를 지운다.

entInput[2].get()에 의해서 인자를 전달받으면 우선 받은 인자가 “”(empty)인지 검사하고 공백이 아닐경우 try / except문을 사용해서 받은 인자가 수가 아닐 경우와 입력받은 학생번호가 존재하는지를 검사한다. 이는 (search data) in list 의 리스트 문법을 활용하여 제거하려는 번호가 존재하지 않을 경우 직접 정의한 not\_exist\_error이 발생하도록 하였다.

제거하려는 번호가 정상적으로 존재할 경우 그 값을 allStudent.deleteInfo의 인자로 전달하고

allStudent.info 의 정보를 모두 출력해준다. 리스트의 정렬은 deleteInfo function에서 담당한다.

Save 함수는 우선 저장하고자 하는 파일명을 입력 받는데 저장은 오직 txt파일에만 가능하도록 하였다. 이는 문자열의 슬라이싱 기능을 사용하여 입력 받은 파일명의 확장자가 txt인지 확인한다.

후에 파일을 불러올 때 학생 정보를 나누기 위해서 파일 저장 시 임의로 : 를 삽입하는데 이는 list의 split기능을 사용하기 위함이다.

Openf 함수 또한 열 파일 이름을 입력 받아 그 파일의 확장자를 검사한다.

Openf 함수는 save와는 다르게 이미 존재하는 파일을 열어야 하므로 예외처리를 해 주었는데

파일이 존재하지 않을 때는 FileNotFound / 파일은 존재하지만 열고자 하는 format이 아닐 때는

IndexError에 대해 처리해주었다.

바른 format을 가진 txt파일의 이름을 인자로 전달할 경우 우선 allStudent.info()를 빈 리스트로 만들어준다. 이는 새로운 파일을 열 때 이전의 저장하지 않은 정보는 처리하지 않는 설계의도에 의해서 구현되었다. 그 후 파일 정보를 read함수로 읽어 들여 : 키워드를 기준으로 스플릿한다. 그렇게 학생 별로 나누어진 정보를 다시 ‘ ‘으로 스플릿 해주는데 이로 인하여 학생별 / 학생의 index , name , score별로 나누기가 가능해진다. 이 정보들을 insertInfo함수를 통해 빈 리스트에 추가해주고 output.insert함수를 통해 windwow에 표시해준다.

모든 정렬은 데이터의 추가 / 변경 / 삭제가 이루어지지 않으므로 allStudent.info를 복사해서 사용하는데 같은 리스트를 참조하는 것을 막기 위해서 SortList = allStudent.info[:]을 사용했다.

이렇게 복사된 리스트는 내부에 index / name / score / index / name / score …..의 반복된 구조를 가진다.

모든 sorting은 Bubble sort를 사용한다.

Numsort의 경우 index / name / score에서 index만이 정렬의 기준이 되므로 반복문의 I 와 j 값이 3으로 나누어떨어지지 않는 경우 즉 name / score의 경우 continue 키워드를 사용해서 sorting시 고려하지 않는다. Index의 값을 비교하여 정렬시에 index 뒤에 연달아 저장되어 있는 name / score를 같이 sorting 해주기 위해서 3번 반복하는 for문을 실행하여 정렬해준다.

Namesort는 위에서 정렬의 대상이 index 에서 name으로 바뀐 것을 제외하면 동일하다.

Downsort는 위에서 정렬의 대상이 index 에서 score으로 바뀐 것을 제외하면 동일하다.

Upsort는 위에서 정렬의 대상이 index에서 score으로 바뀐 것을 제외하면 동일하다.