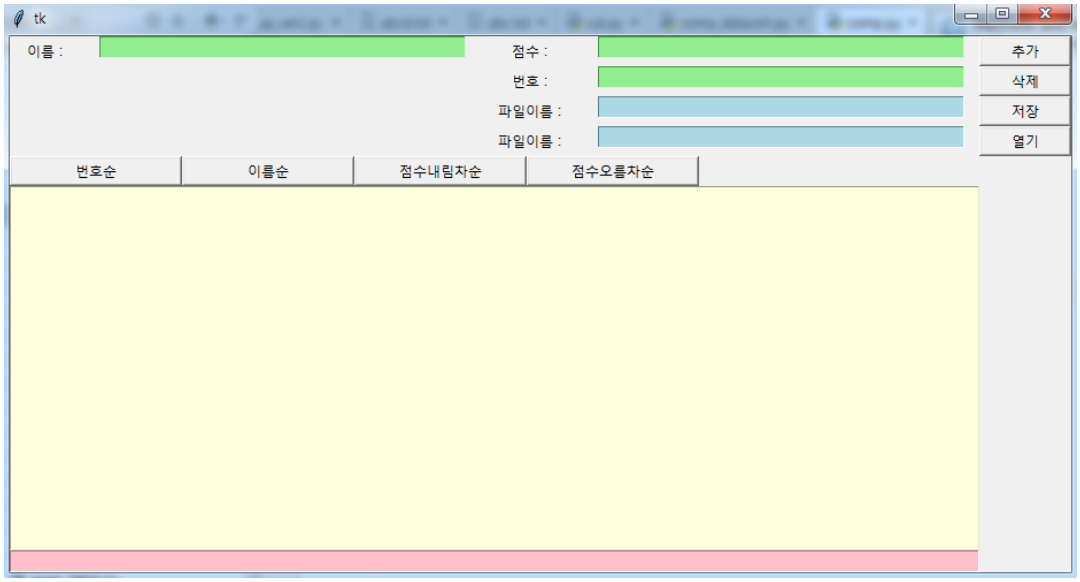
설계 문서 컴퓨터공학부 20113292 안상영

UI



이름, 점수, 번호, 파일이름(1, 2)는 Entry로 구성

```
name = Label(window, text = "이름 : ")
name.grid(row = 0, column = 0)
name_row = Frame(window)
name_row.grid(row = 0, column = 1, sticky = N)
display_name = Entry(name_row, width = 45, bg = "light green")
display_name.grid()

score = Label(window, text = "점수 : ")
score.grid(row = 0, column = 2)
score_row = Frame(window)
score_row.grid(row = 0, column = 3, sticky = N)
display_score = Entry(score_row, width = 45, bg = "light green")
display_score.grid()
```

```
number = Label(window, text = "번호 : ")
number.grid(row = 1, column = 2)
number_row = Frame(window)
number_row.grid(row = 1, column = 3, sticky = N)
display_number = Entry(number_row, width = 45, bg = "light green")
display_number.grid()
```

```
filename1 = Label(window, text = "파일이름 : ")
filename1.grid(row = 2, column = 2)
filename1_row = Frame(window)
filename1_row.grid(row = 2, column = 3,sticky = N)
display_filename1 = Entry(filename1_row, width = 45, bg = "light blue")
display_filename1.grid()

filename2 = Label(window, text = "파일이름 : ")
filename2.grid(row = 3, column = 2)
filename2_row = Frame(window)
filename2_row.grid(row = 3, column = 3, sticky = N)
display_filename2 = Entry(filename2_row, width = 45, bg = "light blue")
display_filename2.grid()
```

데이터 (추가, 삭제, 저장, 열기) 버튼과, 데이터 정렬버튼은 두 개의 버튼 그룹으로 구성.

```
databutton_row = Frame(window)
databutton_row.grid(row = 0, column = 4, rowspan = 4, sticky = E)

datasort_row = Frame(window)
datasort_row.grid(row = 4, column = 0, columnspan = 4, sticky = W)

button_groups = {
    'databutton':{'list':databutton_list, 'window':databutton_row, 'width':10, 'cols':1},
    'datasort':{'list':datasort_list, 'window':datasort_row, 'width':20, 'cols':4},
}
```

➔ databutton_list 와 datasort_list 는 comp_databutton.py 와 comp_datasort.py 에 모듈화 하고, from .. import 로 사용하는 형식을 택함.

데이터 (추가, 삭제, 저장, 열기) 버튼이 click() 함수에서 조건문의 키값으로 구분될 때 , 앞서 전역 변수로 딕션어리를 선언.

```
insertnumber = 1
dataoutput_dic_number = {}
dataoutput_dic_name = {}
dataoutput_dic_score = {}
dataoutput_dic_nametemp = [0]
dataoutput_dic_scoretemp = [0]
```

➔ Insertnumber는 입력되는 순서에 따른 일련번호이며, dataoutput_dic에 3개의 사전을 선언한다. 이것은 정렬버튼(번호순, 이름순, 점수순)에 해당하는 키값(번호, 이름, 점수)으로 정렬함수sort를 적용한다. 즉 데이터 추가, 삭제버튼을 누를 때마다 같은 내용의 세 개의 사전이 갱신된다. nametemp 와 scoretemp는 갱신하기 위해 사용하는 임시변수이다.

데이터 출력창은 Text로, 상태 메시지 출력창은 Entry로 구성 (과제pdf에서는 상태 메시지 출력창도 Text를 적용한 예를 보여주셨는데, 제 기준으로 Entry가 더 적합한 것 같아서 Entry를 적용했습니다.)

```
dataoutput_row = Frame(window)
dataoutput_row.grid(row = 5, column = 0, columnspan = 4, sticky = W)
display_dataoutput = Text(dataoutput_row, width = 120, bg = "light yellow")
display_dataoutput.grid()

current_messageoutput_row = Frame(window)
current_messageoutput_row.grid(row = 6, column = 0, columnspan = 4, sticky = W)
display_current = Entry(current_messageoutput_row, width = 120, bg = "pink")
display_current.grid()
```

마우스로 버튼을 클릭했을 때, 버튼별로 기능을 수행하기 위한 클릭함수 이다. (상태 메시지 출력창은 새로 버튼을 클릭하기 전까지 이전 수행 작업한 내용을 보여주며, 클릭할 때마다 비워지고 클릭한 버튼에 해당하는 새로운 상태 매시지를 보여준다.)

```
def click(key):
    global insertnumber
    global dataoutput_dic_number
    global dataoutput_dic_name
    global dataoutput_dic_score
    global dataoutput_dic_temp

    display_current.delete(0, END)
```

클릭함수 내에 추가버튼에 해당하는 조건문이다.

```
if key in databutton_list:
    if key == '추가':
        if(str(display_name.get()) in dataoutput_dic_name):
            display_current.insert(END, '[추가 실패] 동일한 이름이 이미 존재합니다.')
        elif(str(display_name.get()) == ""):
            display_current.insert(END, '[추가 실패] 이름이 공란입니다.')
        elif(isnumber_float(display_score.get())):
            display_current.insert(END, '[추가 실패] 점수가 올바른 형태가 아닙니다.')
        else:
            dataoutput_dic_number[str(insertnumber)] = str(str(insertnumber) + '\t' + str(display_name.get()) + '\t'
            dataoutput_dic_name[str(display_name.get())] = str(str(insertnumber) + '\t' + str(display_name.get()) +
            dataoutput_dic_score[str(display_score.get())] = str(str(insertnumber) + '\t' + str(display_name.get())
            dataoutput_dic_nametemp.append(display_name.get())
            dataoutput_dic_scoretemp.append(display_score.get())

            display_dataoutput.insert(END, dataoutput_dic_number[str(insertnumber)])
            insertnumber = insertnumber + 1
            display_current.insert(END, '성공적으로 추가하였습니다.')
        display_name.delete(0, END)
        display_score.delete(0, END)
```

➔ 동일한 이름, 공란, 숫자가 아닌 문자에 대해서 실패를 먼저 처리했으며, 그 외에는 딕션어리에 일련번호와 이름, 점수를 문자열로 합쳐서 추가했다. (문자열의 합으로 구성했기에 열을 맞추어 출력 기능은 포함X)

일련번호는 추가될 때마다 1씩 증가시키며, 이름과 점수의 입력창은 초기화된다.

클릭함수 내에 삭제버튼에 해당하는 조건문이다.

```
elif key == '삭제':
    if(isnumber_int(display_number.get())):
        display_current.insert(END, '[추가 실패] 번호가 올바른 형태가 아닙니다.')
    elif(str(display_number.get()) in dataoutput_dic_number):
        del dataoutput_dic_number[display_number.get()]
        del dataoutput_dic_name[dataoutput_dic_nametemp[int(display_number.get())]]
        del dataoutput_dic_score[dataoutput_dic_scoretemp[int(display_number.get())]]

        display_current.insert(END, '성공적으로 삭제하였습니다.')
        display_dataoutput.delete("0.0", "end")
        for i in dataoutput_dic_number.keys():
            display_dataoutput.insert(END, dataoutput_dic_number[i])
    else:
        display_current.insert(END, '[추가 실패] 존재하지 않는 번호를 입력했습니다.')
    display_number.delete(0, END)
```

➔ 번호가 숫자가 아닌 경우, 존재하지 않는 번호를 입력하는 경우 실패로 처리했다. 그 외에는 세 개의 딕션어리를 모두 갱신하면서, 데이터 출력 창에는 모두 지우고, 번호가 키 값인 딕션어리에 포함된 정보를 새로 출력하는 형식을 택했다.

추가, 삭제 버튼에서 점수와 번호에 숫자가 아닌 경우를 검출하는 함수를 구현했다. (일반적으로와 다르게 if문을 통과하기 위해서 true, false를 역으로 사용했다.)

```
def isnumber_float(x):
    try:
        float(x)
        return False;
    except ValueError:
        return True;
def isnumber_int(x):
    try:
        int(x)
        return False;
    except ValueError:
        return True;
```

클릭 함수 내에 저장 버튼에 해당하는 조건문이다.

```
elif key == '저장':
    f = open(str(display_filename1.get()) + str('.txt'), 'w')
    for i in dataoutput_dic_number.keys():
        f.write(dataoutput_dic_number[i])
    f.close()
    display_current.insert(END, str('성공적으로 저장하였습니다. (파일이름 : ') + str(display_filename1.get()) + str(' )'))
    display_filename1.delete(0, END)
```

➔ 입력창에 입력한 파일명에 텍스트파일 확장자를 합해서 텍스트파일을 만들고, 번호 딕션어리에 있는 순서대로 파일에 쓴다. 상태 표시줄에 저장한 파일 이름을 표시하고, 파일 이름 입력창은 초기화한다.

클릭 함수 내에 열기 버튼에 해당하는 조건문이다.

```
elif key == '열기':
    display_dataoutput.delete("0.0", "end")
    try:
        f = open(str(display_filename2.get()) + str('.txt'), 'r')
        while True:
            line = f.readline()
            if not line: break
            display_dataoutput.insert(END, line)
        f.close()
        display_current.insert(END, str('성공적으로 파일을 읽었습니다. (파일이름 : ') + str(display_filename2.get()) + str(' )'))
    except FileNotFoundError:
        display_current.insert(END, '[열기 실패] 파일이 존재하지 않습니다.')
    display_filename2.delete(0, END)
```

➔ 파일이 존재하지 않는 경우에 try except로 처리하였고, 존재하는 경우에는 한 줄씩 읽어서 데이터 출력 창에 한 줄씩 출력하는 방식을 택했다. 상태 표시줄에 읽어 들인 파일 이름을 표시하고, 파일 이름 입력창은 초기화한다.

클릭 함수 내에 번호순 정렬과 이름순 정렬에 해당하는 조건문이다.

```
elif key in datasort_list:
    if key == '번호순':
        display_dataoutput.delete("0.0", "end")
        temp = list(dataoutput_dic_number.keys())
        temp.sort()
        for i in temp:
            display_dataoutput.insert(END, dataoutput_dic_number[i])
    elif key == '이름순':
        display_dataoutput.delete("0.0", "end")
        temp = list(dataoutput_dic_name.keys())
        temp.sort()
        for i in temp:
            display_dataoutput.insert(END, dataoutput_dic_name[i])
```

➔ 번호순 정렬은 번호딕션어리를 사용하여 번호를 기준으로 sort() 함수를 사용했고, 이름순 정렬은 이름을 기준으로 sort() 함수를 사용했다.

클릭함수 내에 점수내림차순과 점수오름차순 정렬에 해당하는 조건문이다.(sort함수는 오름차순으로 정렬되기 때문에, 내림차순에서는 reverse()함수를 추가한다)

```
elif key == '점수내림차순':
    display_dataoutput.delete("0.0", "end")
    temp = list(dataoutput_dic_score.keys())
    temp.sort()
    temp.reverse()
    for i in temp:
        display_dataoutput.insert(END, dataoutput_dic_score[i])
elif key == '점수오름차순':
    display_dataoutput.delete("0.0", "end")
    temp = list(dataoutput_dic_score.keys())
    temp.sort()
    for i in temp:
        display_dataoutput.insert(END, dataoutput_dic_score[i])
```