컴퓨팅적 사고와 프로그래밍 (03)

STUDY HELPER

Kim Dain

1944005 김다인

```
# 시작 모듈
import tkinter
from tkinter import *
from tkinter import ttk
import datetime
from tkinter import messagebox
from time import strftime
```

GUI toolkit 인 Tkinter 패키지를 가져옵니다. 매번 tkinter 를 쓰지 않게 import 합니다. ttk module 도 가져옵니다. Frame2 날짜와 시간에 필요한 Datetime module 을 가져옵니다. Frame1 학점 계산과 학점 별명, Frame2 타이머에 필요한 Tkinter messagebox module 가져옵니다. Frame2 시간에 필요한 time module strftime()을 가져옵니다.

```
# 기본 창 생성
root = Tk()
# 기본 창 설정
root.title("Study Helper")
root.geometry("809x500-500+100")
root.resizable(False, False)
```

루트 윈도를 설정합니다. 제목과 사이즈를 설정합니다. 창의 너비는 809, 높이는 500, 창의 가로 방향 위치는 -500, 세로 방향 위치는 100 으로 설정합니다. Resizable 값을 False 로 지정하여 창의 크기를 고정합니다.

```
# 탭 생성
notebook=ttk.Notebook(root, width=809, height=500)
notebook.pack()
```

Notebook 을 이용해서 페이지를 나눌 수 있는 탭 메뉴를 윈도 창에 생성합니다. Pack 을 이용해 배치합니다.

```
# 학습 탭
frame1 = Frame(root)
notebook.add(frame1, text="학습")
```

다른 위젯을 담을 수 있게 Frame 함수를 이용합니다. "학습"이라고 정해줍니다.

```
# 학점 입력
def myFinal():
    number_of_subject = int(input("과목 개수를 입력해주세요 : "))
    # 리스트
    subject_name = list()
    subject_score = list()
    subject_grade = list()
    # 반복 실행
    while len(subject_name) != number_of_subject:
        subject_name.append(input("과목명 : "))
        subject_score.append(input("성적(알파벳) : "))
        subject_grade.append(input("학점 : "))
```

myFinal() 사용자 정의 함수를 이용합니다. Input 함수를 이용해서 사용자의 입력을 받습니다. 자료형을 바꿔야 하기 때문에 바로 int() 내장 함수를 이용해서 정수로 반환합니다. 데이터를 모아 저장해야 하기 때문에 list 자료형을 이용합니다. Len() 함수를 통해 요소의 개수를 알아냅니다. != 연산으로 같음을 확인합니다. 리스트 안에 있는 과목 이름의 수와 처음에 입력한 과목 수가 같을 때까지 while 반복문을 실행합니다. 과목명, 성적, 학점 input()을 반복해서 보여주고, 리스트에 값을 추가하는 리스트 메서드 append()를 사용합니다.

```
    # 정의
    score = 0
    amount_of_grade = 0

    # for 문 반복
    for index, value in enumerate(subject_score):

        # 학점 합산
        grade = int(subject_grade[index])
        amount_of_grade += grade
```

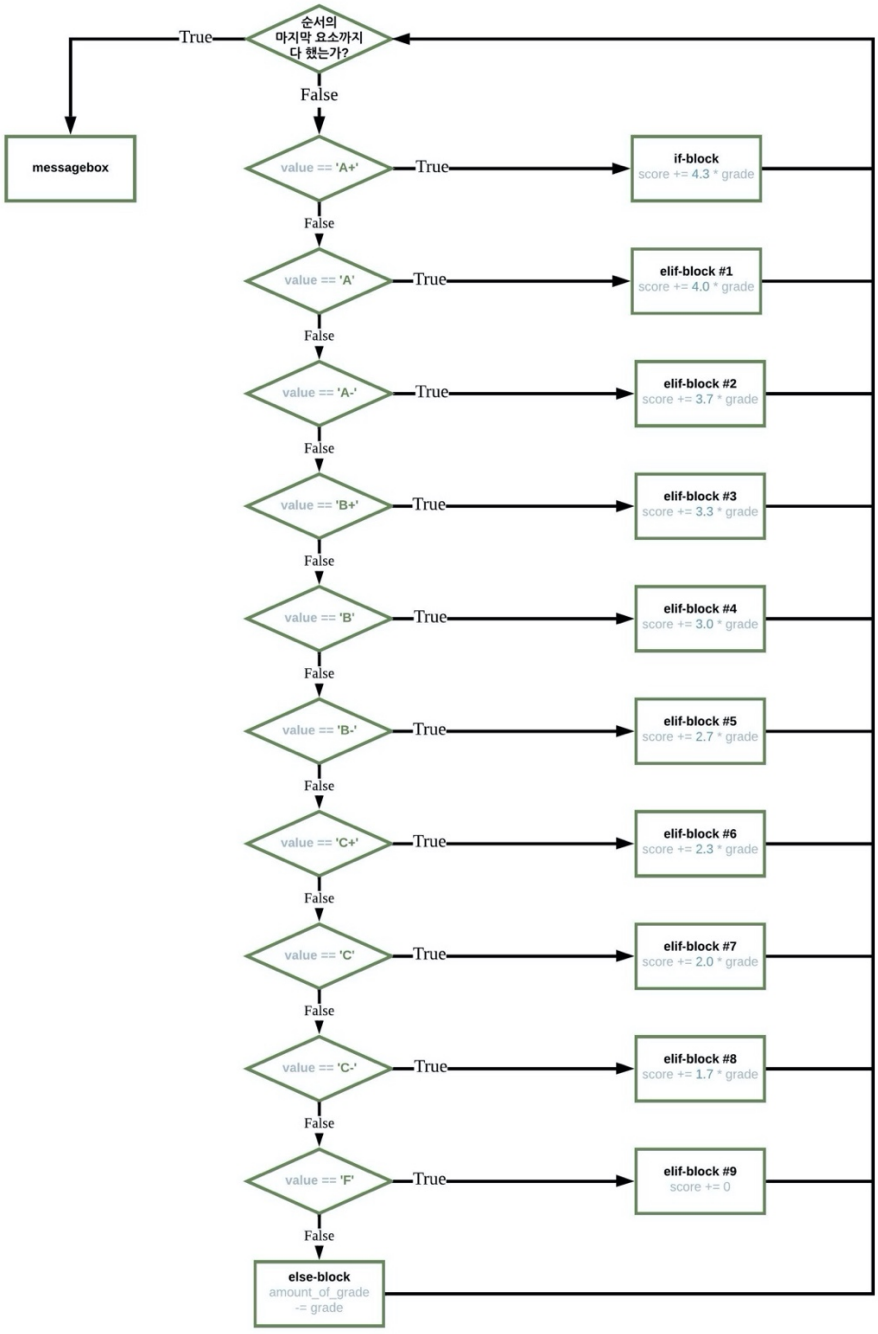
성적, 학점 input()을 반복해서 보여주고, 리스트에 값을 추가하는 리스트 메서드 append()를 사용합니다. score 와 amount_of_grade 를 전역 변수로 정의합니다. enumerate 함수를 써서 몇 번째 반복문인지 확인합니다. (인덱스, 원소)로 value 에서 튜플 형태로 반환한 값을 확인합니다. grade 는 계속 더해져서 amount_of_grade 에 저장됩니다.

```
        # 등급 구간, 계산
        if value == 'A+':
            score += 4.3 * grade
        elif value == 'A':
            score += 4.0 * grade
        elif value == 'A-':
            score += 3.7 * grade
        elif value == 'B+':
            score += 3.3 * grade
        elif value == 'B':
            score += 3.0 * grade
        elif value == 'B-':
            score += 2.7 * grade
        elif value == 'C+':
            score += 2.3 * grade
        elif value == 'C':
            score += 2.0 * grade
        elif value == 'C-':
            score += 1.7 * grade
        elif value == 'F':
            score += 0
        else:
            amount_of_grade -= grade
```

If 문을 이용합니다. 하나만 실행될 수 있는 경우이기에 효율적으로 elif 를 사용합니다. value 값에 따라 성적 구간이 나뉩니다. grade 와 곱해져서 score 에 계속 저장됩니다.

```
    # 결과 메시지 박스
    aver = score/amount_of_grade
    messagebox.showinfo ("학점 계산기","이번 학기에 들은 총 학점 : {0}\n 평균 학점 : {1}"
                         "\n\n 이번 학기도 수고 많으셨어요!"
                         "\n 이제 쉬어요!".format(amount_of_grade, round(aver, 2)))
```

학점 평균을 계산해서 aver 변수에 저장합니다. 팝업을 띄우려면 Messagebox 를 이용합니다. messagebox.showinfo 를 통해 들어갈 내용을 적습니다. 문자열.format() 함수를 이용합니다. 깔끔한 나눗셈을 위해 내장된 round() 함수를 이용합니다. 소수 셋째 자리에서 반올림하게 합니다.



Flowchart 로 구성하면 이렇게 보여집니다. For loop 로 반복하다가 다 끝났으면 최종적으로 messagebox 가 나옵니다. (열심히 만들었는데 정확히 잘 만든 건지 모르겠습니다 ^^)

```
    # 학점 별명 정의
    global mygpa
    mygpa = round(aver, 2)
```

후에 쓰일 mygpa 변수를 global 을 이용해 전역 변수로 정의합니다. 이 방법을 꼭 안 써도 될 것 같은데, aver 지역 변수를 이용하고 있어서 일단 정의해줬습니다.

```
# 학점 별명
def labeling():
    # while 이용해서 오류 잡기
    while True:
        try:
            if mygpa >= 4.1:
                plus = "{0}인 당신! \n 공부의 신!".format(mygpa)
            elif mygpa >= 3.9:
                plus = "{0}인 당신! \n 교수님의 사랑!".format(mygpa)
            elif mygpa >= 3.5:
                plus = "{0}인 당신! \n 교수님의 귀염둥이!".format(mygpa)
            elif mygpa >= 3.2:
                plus = "{0}인 당신! \n 성실한 이대생!".format(mygpa)
            elif mygpa >= 2.9:
                plus = "{0}인 당신! \n 일탈을 꿈꾸는 소시민!".format(mygpa)
            elif mygpa >= 2.3:
                plus = "{0}인 당신! \n 오락문화의 선구자!".format(mygpa)
            elif mygpa >= 1.8:
                plus = "{0}인 당신! \n 영차 달팽이!".format(mygpa)
            elif mygpa >= 1.1:
                plus = "{0}인 당신! \n 귀여운 플라크톤!".format(mygpa)
            else:
                plus = "{0}인 당신! \n 시대를 앞서가는 혁명의 씨앗!".format(mygpa)

            # 결과 메시지 박스
            messagebox.showinfo("학점 별명", plus)
            break

        # 오류 메시지 박스
        except:
            messagebox.showinfo("학점 계산기", "학점 계산부터 해주세요!")
            break
```

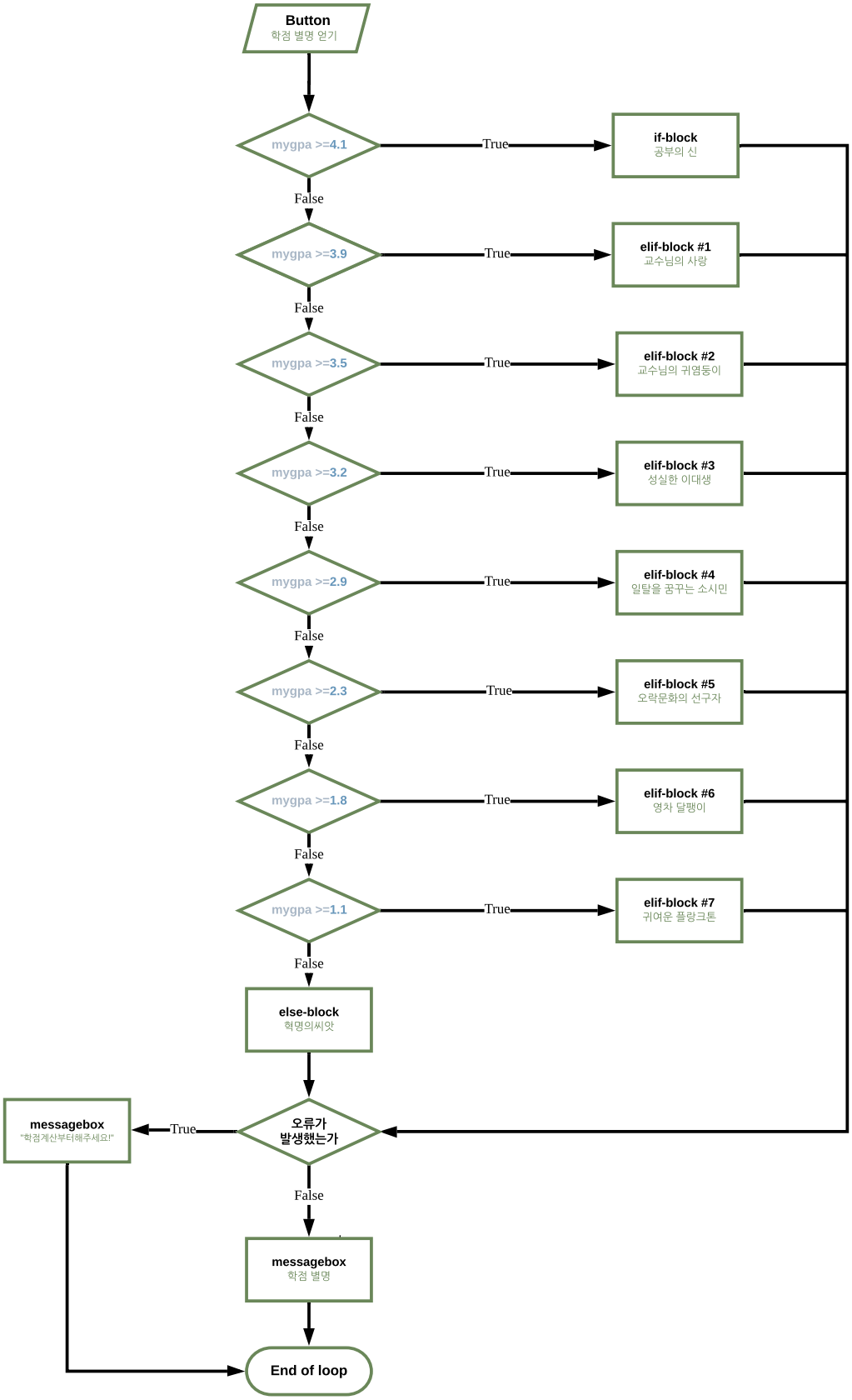
학점을 계산하기 전에 "학점 별명 얻기" 버튼을 누르면 경고 팝업창을 뜨게 하고 싶었습니다. 예외를 처리할 때 try 와 except 을 이용한다는 것 같아서 이렇게 구성해보았습니다. 이게 가장 좋은 방법인지는 모르겠습니다. (분석하다가 생각한 건데, try 에 break 는 없어도 될 것 같습니다.) 학점을 먼저 계산했다면, try 에서 문자열.format()을 이용해서 plus 에 저장된 내용이 messagebox 에서 보여집니다. 하지만 Try 절을 실행했을 때, mygpa 값이 없으면 except 로 가서 messagebox 를 실행하고 break 해서 while 문에서 나옵니다. (플라크톤이 아니라 플랑크톤인데 오타를 지금 발견했습니다...)

```
# 사용법 설명
message = Label(frame1, text="과목명, 성적, 학점을 순서대로 입력하고,"
                            "\n 재밌는 학점 별명까지 얻으세요"
                            "\n(성적은 알파벳 대문자로 입력하고 A0 는 A 로 입력합니다)")
message.pack(side=BOTTOM)

# 학점 별명 위치 설정
labeling_B = Button(frame1, width=15, text = "학점 별명 얻기", font=(None, 20),fg
="green", command = labeling)
labeling_B.pack(pady=35, side=BOTTOM)

#학점 계산 위치 설정
final = Button(frame1, width=15, text = "학점 계산하기", font=(None, 20), fg
="green", command = myFinal)
final.pack(side=BOTTOM)
```

Label 을 이용해서 text 매개변수에 텍스트를 할당해서 frame1 에 설정합니다. Pack() 메서드의 키워드 인수 side 로 label 을 하단에 표시한다. Button 을 이용해 "학점 계산하기"와 "학점 별명 얻기"를 만들고 pack()으로 표시한다. Command 를 이용해 버튼을 누를 때 함수를 호출하도록 하였다. "학점 계산하기"는 myFinal(), "학점 별명 얻기"는 labeling() 함수를 호출한다.

```
# 배경 사진 설정
ewha = PhotoImage(file="ewha.png")
label9 = Label(frame1, image=ewha)
label9.pack(side=BOTTOM)
```

PhotoImage()를 이용해 ewha.png 를 불러온 후, 마찬가지 pack()으로 표시한다. Side = BOTTOM 으로 된 pack()은 먼저 실행한 게 당연히 밑에 위치되기 때문에, 가장 밑에 두고 싶은 사용법 설명부터 가장 위에 들어갈 배경 사진 순으로 코드를 작성했다.



"학점 별명 얻기"를 눌렀을 때의 흐름을 Flowchart 로 만들었습니다. Try 와 except 과정을 이해는 했는데, 순서도에서 보여주는 게 어려운 것 같습니다. 틀릴지도 모르겠습니다.

```
# 날짜
# 날짜 함수 정의
def day():
    string = strftime('\n%Y %B %d')
    lb2.config(text = string)
# 날짜 형식, 표시
lb2 = Label(frame2, fg = 'green', font = (None, 25, 'bold'))
lb2.pack()
day()
# 시간
# 시간 함수 정의
def time():
    string = strftime('%H:%M:%S %p')
    lbl.config(text = string)
    # 실시간 시간
    lbl.after(1000, time)
# 시간 형식, 표시
lbl = Label(frame2, bg = 'green', fg = 'white', font = (None, 25, 'bold'))
lbl.pack()
time()
```

사용자 정의 함수 day()와 time()을 만듭니다. strftime()을 이용해 시간 문자열 반환합니다. %Y, %S, %d 등 알맞은 양식 기호를 이용해서 원하는 출력을 설계한다. config() 메서드를 이용해서 텍스트에 새 값을 할당할 수 있게 합니다. after() 메서드를 이용해서 첫 인자 밀리세컨드에 1000 을 넣어 1 초로 설정하고, 두번째 인자에 time 함수를 실행합니다. 이렇게 1 초 뒤에 time 함수를 실행하도록 합니다. bg, fg, font 등 키워드 인수를 사용하여 label 을 원하는 디자인으로 표기합니다.

```
# 빈 공간
space = Label(frame2, text = "\n", font = (None, 3))
space.pack()
```

초반에 place()의 존재를 몰랐어서 빈 공간을 Label 로 구성했었다.

```
# 타이머
# 정의
Type_tab2 = 0
One = 0
# 타이머 시작 함수
def tab2_submit():
    global Type_tab2, One, Hour, Min, Sec
    if (Type_tab2 == 1):
        return
    if (One == 0):
        One = 1
        Hour = int(timer_1.get())
        Min = int(timer_2.get())
        Sec = int(timer_3.get())
    t = datetime.time(Hour, Min, Sec)
    time_label.configure(text=t, font=(None, 45))
    Sec -= 1
    if (Sec < 0):
        if (Hour + Min + Sec == -1):
            Hour = int(timer_1.get())
            Min = int(timer_2.get())
```

```
            Sec = int(timer_3.get())
            # 완료
            messagebox.showinfo("타이머", "시간이 다 되었습니다!")
            time_label.configure(text=t, font=(None, 45))
            One = 0
            return
        Min -= 1
        Sec = 59
        if (Min < 0):
            Hour -= 1
            Min = 59
    root.after(1000, tab2_submit)
```

변수를 정의한다. Tab2_sumbit 사용자 정의 함수를 만든다. global 을 이용해 변수를 정의한다. 안 하면 오류가 난다. (초반에 타이머가 아니라 스톱워치를 만들려고 했어서 불필요한 코딩이 조금 섞인 것 같다……) Hour, Min, Sec 을 get()으로 호출한 후 int()함수를 이용해 정수로 정의한다. t 는 datetime.time()을 이용해서 시간, 분, 초를 정의한다. 나중에 configure()에서 text 에서 쓰인다.
If 문을 이용한다. Sec 가 0 보다 작고, 시간, 분, 초가 다 되었을 때 messagebox 를 이용해서 완료되었다는 팝업을 띄운다. return 은 반환하기도 하지만 함수의 중간에서 빠져나올 수도 있어서 함수에서 빠져나오게 된다.
하지만 Sec 가 0 보다는 작지만 시간이 다 된 건 아닐 때, Min 에서 1 을 빼고 Sec 에 다시 59 를 지정한다. (어쩌면 당연하다) Sec 도 0 보다 작고, 시간이 다 된 것도 아니고, Min 도 0 보다 작을 때는 Hour 에서 1 을 빼주고 Min 에 59 를 채워준다. 이때 복합 대입 연산자를 이용한다. Min = Min – 1 은 Min -=1 로 표현하면 편리하다. after() 함수를 이용해서 1000 밀리세컨드 뒤에 tab2_submit 함수를 계속 실행한다.

```
# 정의
timer_1 = StringVar()
timer_2 = StringVar()
timer_3 = StringVar()
```

불변형을 가변적으로 사용하기 위해 StringVar() 클래스를 사용해서 정의한다. 변수의 값이 바뀔 때 자동으로 화면에서 바뀌기를 원하기 때문이다. 더이상 고정적인 문자열 변수가 아니기 때문에 set() 메서드를 이용해야 한다.

```
# hour 위치 설정
labelframe1 = LabelFrame(frame2, text="Hour")
labelframe1.pack(pady=5)
scale_1 = ttk.Scale(labelframe1, orient=HORIZONTAL, length=300, variable=timer_1,
                    command=lambda x: timer_1.set('%d' % (float(x) + 0.5)),
from_=0, to=23)
scale_1.grid()
spinbox_1 = Spinbox(labelframe1, from_=0, to=23, textvariable=timer_1, width=5)
spinbox_1.grid(row=0, column=1)
# min 위치 설정
labelframe2 = LabelFrame(frame2, text="Min")
labelframe2.pack(pady=5)
scale_2 = ttk.Scale(labelframe2, orient=HORIZONTAL, length=300, variable=timer_2,
                    command=lambda x: timer_2.set('%d' % (float(x) + 0.5)),
from_=0, to=59)
scale_2.grid()
spinbox_2 = Spinbox(labelframe2, from_=0, to=59, textvariable=timer_2, width=5)
```

```
spinbox_2.grid(row=0, column=1)
# sec 위치 설정
labelframe3 = LabelFrame(frame2, text="Sec")
labelframe3.pack(pady=5)

scale_3 = ttk.Scale(labelframe3, orient=HORIZONTAL, length=300, variable=timer_3,
                    command=lambda x: timer_3.set('%d' % (float(x) + 0.5)),
from_=0, to=59)
scale_3.grid()
spinbox_3 = Spinbox(labelframe3, from_=0, to=59, textvariable=timer_3, width=5)
spinbox_3.grid(row=0, column=1)
```

Scale 과 Spinbox 를 이용해서 사용자의 입력을 받는다. 텍스트, 높이, 길이, 방향 등 디자인적 요소를 설정한다. LabelFrame 을 설정해서 그 안에 scale 과 spinbox 를 같이 넣는다. Scale 에서는 variable, spinbox 에서는 textvariable 를 같은 변수로 설정하여 연동되어 보여지게 한다. Scale 함수 안에 일시적으로 이용하는 lambda 함수를 이용해서 button 안에 command 키워드 인수에서도 인수를 넘겨줄 수 있게 한다. 정수를 넣기 위해 문자열 포맷 코드 %d 를 사용한다. Hour 는 0~23, Min 은 0~59, Sec 는 0~59 로 설정한다. Grid()에서 column 을 1 로 설정하여 spinbox 를 scale 우측에 위치하게 한다.

```
# 시작 위치 설정
button_1 = ttk.Button(frame2, text="시작", width = 10, command=tab2_submit)
button_1.pack(pady=10)

# 카운트다운 표시
time_label = Label(frame2, text="00:00:00", fg= "green", font=(None, 45))
time_label.pack()

# 사용법 설명
message = Label(frame2, text="시간을 확인하거나, 시간을 설정하여 타이머를 이용하세요")
message.pack(side=BOTTOM)
```

Button() 함수를 이용해서 시작 버튼을 만든다. command 키워드 인수 = tab2_sumit 을 작성해서 클릭했을 때 tab2_sumbit 함수를 실행한다. 카운트다운에 text 로 시간을 표기한 후 pack()으로 화면에 표시한다. 창 하단에 사용법 설명 text 를 넣는다.

```
# 메모 탭
frame3 = Frame(root)
notebook.add(frame3, text="메모")
```

마지막 3 번째 메모 탭을 생성한다.

```
# 텍스트 메모
textbox = Text(frame3, width=105, height=3, font=(None, 20), highlightbackground =
"green")
textbox.pack()
```

Text()를 이용해서 여러줄의 문자열을 출력할 수 있게 하면서, 타이핑 할 수 있는 곳을 만든다. 높이를 3 으로 설정해 3 줄로 한다.

```
# 그리는 메모
# 함수 정의
def paint(event):
    x1, y1 = (event.x - 5), (event.y - 5)
    x2, y2 = (event.x + 5), (event.y + 5)
    w.create_oval(x1, y1, x2, y2, fill= "green", outline= "green", width=10)
```

paint() 사용자 정의 함수를 만든다. 그림을 그리기 위해 create_oval 메소드를 이용한다. (x1, y1)에서 (x2, y2)의 크기를 갖는 원을 생성한다. 수를 조절하여 적당한 크기의 oval 을 만든다. Oval 을 계속 그리면서 선을 만들 계획이다.

```
# 캔버스 설정
w = Canvas(frame3, width=500, height=150, background='beige', cursor="pencil")
w.pack(expand=YES, fill=BOTH)
w.bind("<B1-Motion>", paint)
```

다각형, 선, 원을 그릴 수 있는 Canvas 를 만든다. 배경색을 'beige'로 하고 cursor 를 'pencil'로 해서 자신이 원하는 대로 매개변수를 설정한다. Pack()에서 'expand=YES', 'fill=BOTH'를 해서 화면에 캔버스가 꽉 차게 해준다. bind()를 이용해서 위젯의 이벤트와 함수를 설정한다. <B1-Motion>은 마우스 왼쪽을 누르고 움직일 때 실행되는 모션을 뜻한다. bind()를 이용해서, 이 모션을 할 때 paint 사용자 정의 함수를 실행하게 한다.

```
# 지우기 함수
def Click():
    delete2 = w.delete(ALL)

# 지우기 버튼 설정
button1 = Button(frame3, text = "지우기", padx=10, pady=5, command=Click, fg='red')
button1.pack(side=BOTTOM)
```

텍스트는 키보드로 지울 수 있지만 그림은 지울 수 없어서 "지우기" 버튼을 하단에 만들어서 위치시킨다. 지우기 위해 delete() 함수를 이용해서 Click()에 사용자 정의 함수를 만든다. (초반엔 delete2 를 적었었는데 나중에 수정하면서 필요가 없어졌다.) "지우기" 버튼을 생성하고 side 를 이용해서 하단에 위치시킨다.

```
# 최종 실행
mainloop()
```

mainloop() 메서드가 GUI 의 모든 이벤트를 실행하기 때문에 필수적으로 있어야 한다.