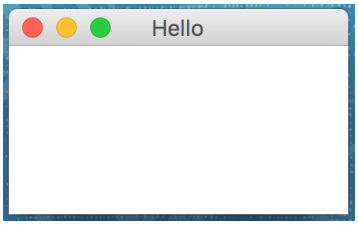
객체지향 프로그래밍
Graphical User Interface (GUI) 활용

소프트웨어와 사용자가 의사소통하는 창구를 **사용자 인터페이스**User Interface(UI)라고 한다. 지금까지 공부한 프로그램은 모두 표준입출력창Standard Input/Output을 통하여 사용자와 의사소통한다. 이와 같이 사용자가 표준입력창에서 키보드에서 문자열로 프로그램에 데이터를 전달하고, 프로그램은 전달받은 문자열을 해석하고 계산하여 결과를 표준출력창에 문자열로 보여 준다. 이렇게 작동하는 사용자 인터페이스를 **명령줄 인터페이스**Command Line Interface(CLI)라고 한다. 이 사용자 인터페이스는 명령창에서 문자열로만 의사소통을 하므로 단조롭고 불편하다. 화면에 표시되는 그래픽을 사용자가 마우스나 손으로 만져서 조작할 수 있도록 하는 새로운 개념의 사용자 인터페이스인 **그래픽 사용자 인터페이스**Graphical User Interface(GUI, 구이로 읽음)가 1970년대에 고안되었으며 1984년 후반 실용화되기 시작하여 그 이후 거의 모든 컴퓨터 환경에서 널리 사용되고 있다. 이 장에서는 Python에서 GUI를 활용하는 프로그램을 만들 수 있는 도구를 모아놓은 표준 라이브러리 모듈인 **Tkinter**(Tk interface의 약자로 '티케이인터'라고 읽음)를 공부한다[1].

빈 창 만들기

먼저 아래 왼쪽과 같은 빈 창은 아래 오른쪽 코드로 만든다.



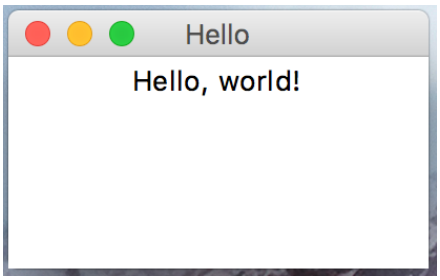

```
1 from tkinter import *
2
3 window = Tk()
4 window.title("Hello")
5 window.geometry("200x100")
6 window.mainloop()
```

- 줄 1과 같이 기술하면 `tkinter` 모듈을 통채로 가져와서 쓸 수 있다. `tkinter` 모듈에는 Tk GUI 프로그램을 만드는데 필요한 도구가 모두 마련되어 있다.
- `tkinter`를 시작하려면 GUI가 활동할 영역인 창window 객체 `Tk`를 먼저 생성해야 한다. 줄 3에서 `Tk` 객체를 생성하여 이름을 `window`라 하였다.
- 창의 간판명은 줄 4와 같이 `title` 메소드를 호출하여 지정할 수 있다.
- 창의 크기는 줄 5와 같이 `geometry` 메소드를 호출하여 지정할 수 있다. 그래픽 화면으로 보는 이미지는 픽셀pixel(화소)이라고 하는 아주 작은 정사각형 점들이 모여서 이루어진다. 픽셀은 그래픽 화면에서 크기를 나타내는 기본 단위로 사용한다. 창의 크기는 `geometry` 메소드의 인수로 `"200x100"`과 같이 하나의 문자열 형식으로 지정하는데, 창의 크기가 가로로 `200` 픽셀, 세로 `100`픽셀 이라는 뜻이다. 가운데 `x`는 영어 알파벳이다.
- 줄 6과 같이 `window.mainloop()`를 호출하면, 준비한 `window`를 화면에 그린다.

[1] Tkinter Official Site : https://wiki.python.org/moin/TkInter,
Tkinter Reference : http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html

Label 위젯 만들기

이번엔 이 창 내부에 아래 그림과 같이 Hello, world!를 프린트하는 프로그램을 만들어보자.




```
1  from tkinter import *
2
3  window = Tk()
4  window.title("Hello")
5  window.geometry("200x100")
6  Label(window, text="Hello, world!").pack()
7  window.mainloop()
```

GUI에는 버튼button, 스크롤바scroll bar, 메뉴menu, 입력 상자entry, 텍스트 상자text box와 같은 미리 마련되어 있어 갖다 쓸 수 있는 다양한 종류의 객체들이 있는데, 이를 통틀어 **위젯**widget이라고 한다.

- 줄 6에서 Label 위젯을 하나 만든다. Label 위젯은 텍스트나 이미지를 담을 수 있는 위젯이다.
- Label 위젯의 소속을 첫 인수로 지정해준다. 여기서 이 Label 위젯의 소속은 window 이다.
- 둘째 인수 text="Hello, world!"는 text 옵션을 지정한다. 이 위젯에 담겨있는 텍스트 정보는 "Hello, world!" 로 지정한다.
- 생성한 Label 위젯에 pack() 메소드를 호출하면 window 창에서 텍스트 위치를 적절히 (이 경우 중앙으로) 정해준다.
- 줄 7과 같이 window.mainloop()를 호출하면, 창이 생기면서 지정한 위치에서 Label 객체의 내용을 보여준다.

Frame 위젯 만들기

여러 종류의 위젯이 필요한 경우 이를 담을 수 있는 컨테이너 역할을 하는 Frame 위젯을 먼저 만들고 그 내부에 다양한 위젯을 원하는 위치에 배치하면 편리하다. 이를 구현하려면 아래 코드와 같이 Frame 위젯을 상속받아 기능할 확장하는 클래스를 만들어 쓸 수 있다.

```
1  from tkinter import *
2
3  class App(Frame):
4      def __init__(self, master):
5          super().__init__(master)
6          self.pack()
7          Label(self, text="Hello, world!").pack()
8
9  window = Tk()
10 window.title("Hello")
11 window.geometry("200x100")
12 App(window)
13 window.mainloop()
```

- 줄 3의 `class App(Frame):` 에서 App은 정의하는 클래스의 이름이고, Frame은 상속을 받는 **상위 클래스**super class의 이름이다. 정의하는 App 클래스는 Frame이라는 상위 클래스의 모든 속성과 기능을 상속(물려) 받겠다는 의미이다. 즉, 새로 정의하는 App은 상위 클래스 Frame이 기본적으로 제공하는 Frame 고유의 기능은 그대로 모두 지니고 있다.
- 줄 4에서 self 는 window 안에 속한 App 객체 자신을 가리킨다. App을 장착할 객체를 전달받아 이름을 master라고 하는데, 여기서는 줄 14에서 Tk 객체인 window를 전달받으므로, App의 소속은 window가 된다.
- (self가 자신을 나타내는 대명사이듯이,) 줄 5에서 super는 상위 클래스를 지칭하는 대명사이다. 여기에서 `super().__init__(master)`는 상위 클래스 Frame의 생성메소드를 호출하여 실행하라는 뜻이다.
- 줄 6의 `self.pack()`은 App 객체 자신의 위치를 알아서 적절히 정하라는 명령이다.
- 줄 7은 자기 소속의 Label 위젯을 하나 만들고 text 속성을 "Hello, world!"로 하자는 뜻이다. 뒤에 붙은 `.pack()`은 자신의 위치를 알아서 적절히 정하라는 명령이다.
- 줄 12에서 Frame의 유전자를 상속받은 App 객체를 생성한다. 인수로 객체의 소속이 되는 window를 전달한다.

이 코드를 실행하여 위와 똑같은 창이 생기는지 확인해보자. 다른 점은 window에 Frame 위젯을 상속받은 App 위젯을 컨테이너 위젯으로 만들어 넣고, 그 내부에 Label 위젯을 만들어 담는다.

연습문제

Label 위젯은 다양한 옵션을 제공한다. 웹에서 Tkinter Reference(http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html)를 찾아서 아래 코드에서 추가된 진한 부분이 어떤 의미인지 알아 보자.

1. 줄 7의 bg와 fg 옵션의 의미가 무엇인지 실행하여 확인해보자.

```
1   from tkinter import *
2
3   class App(Frame):
4       def __init__(self, window):
5           super().__init__(window)
6           self.pack()
7           Label(self, text="Hello, world!", bg="yellow", fg="red").pack()
8
9   window = Tk()
10  window.title("Hello")
11  window.geometry("200x100")
12  App(window)
13  window.mainloop()
```

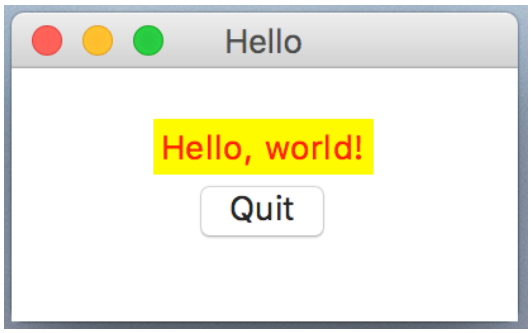
2. 줄 6과 같이 하여 Frame의 주위에 빈 여백을 줄 수도 있다. 여백이 어떻게 생기는지 실행하여 확인해 보자.

```
1  from tkinter import *
2
3  class App(Frame):
4      def __init__(self, window):
5          super().__init__(window)
6          self.pack(padx=20, pady=20)
7          Label(self, text="Hello, world!", bg="yellow", fg="red").pack()
8
9  window = Tk()
10 window.title("Hello")
11 window.geometry("200x100")
12 App(window)
13 window.mainloop()
```

버튼 위젯 만들기

GUI 프로그램은 **이벤트 구동**event-driven 방식으로 작동한다. 생성된 위젯은 window.mainloop()를 호출하는 순간부터 이벤트가 발생하기를 상시 기다리고 있으므로 특정 이벤트가 발생하면 이에 맞는 반응을 하도록 프로그램을 작성해야 한다. Label 위젯은 스스로 이벤트를 발생시켜 내용을 보여주기만 하는 일방형 위젯이므로 바깥에서 생기는 이벤트와 무관하다.

이번엔 사용자가 이벤트를 발생시키면 이에 반응을 하는 위젯을 만들어보자. 버튼 위젯은 클릭하는 이벤트가 발생하면 적절하게 반응하도록 코딩할 수 있는 기능을 제공한다. 위에서 만든 Hello 창에 Quit이라는 간판을 가진 버튼을 다음 그림과 같이 만들고, 이 버튼을 클릭하면 창을 닫도록 프로그램을 확장해보자.



아래 코드의 줄 8과 같이 tkinter 모듈에서 제공하는 Button 위젯을 만든다.

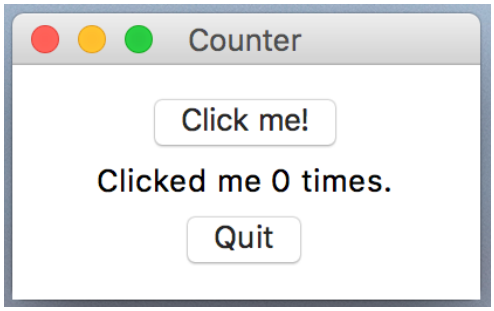
- text 옵션은 Button 위젯의 간판 문자열을 지정한다.
- command 옵션은 이 버튼을 클릭하면 실행할 메소드의 이름을 적는다. quit 메소드는 Frame에 내장되어 있는 메소드로 호출하면 창을 닫아준다. 여기서 self.quit은 메소드 호출이 아님을 주의하자. ()가 없지 않은가! 단순히 메소드 이름만 전달해줄 뿐이다. 언젠가 이 Button을 클릭하는 이벤트가 발생하면 이 메소드가 저절로 호출된다.

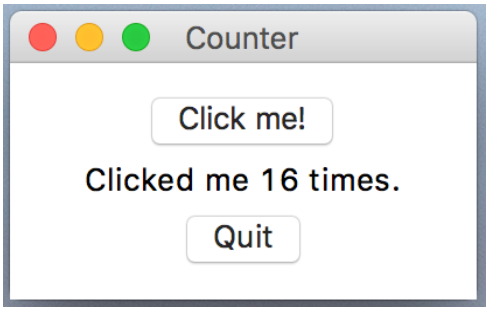
```
1  from tkinter import *
2
3  class App(Frame):
4      def __init__(self, window):
5          super().__init__(window)
6          self.pack(padx=20, pady=20)
7          Label(self, text="Hello, world!", bg="yellow", fg=“ red").pack()
8          Button(self, text="Quit", command=self.quit).pack()
9
10 window = Tk()
11 window.title("Hello")
12 window.geometry("200x100")
13 App(window)
14 window.mainloop()
```

이와 같이 tkinter 모듈은 GUI 프로그램을 작성하는데 필요한 다양한 도구를 갖추고 있다. 여기서 도구를 모두 다 살펴보기는 위젯의 종류도 많고 옵션도 다양하여 대표로 몇 가지만 살펴보았다. GUI의 프로그램을 잘 하기 위해서는 다양한 위젯과 관련 옵션들을 잘 꿰고 있어야 한다. Tkinter Reference(http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html) 에 자세히 기술되어 있으니 찾아서 많이 사용해보는 수 밖에 다른 지름길은 없다.

1. 사례 학습 : Counter

이번에는 카운터를 만들어보자. 다음과 같은 창에서 `Click me!` 단추를 클릭할 때마다 1씩 증가하여 보여주도록 한다. 즉, 아래의 왼쪽 창에서 시작하여 16번 `Click me!` 단추를 클릭하면 오른쪽 창과 같이 되어야 한다.

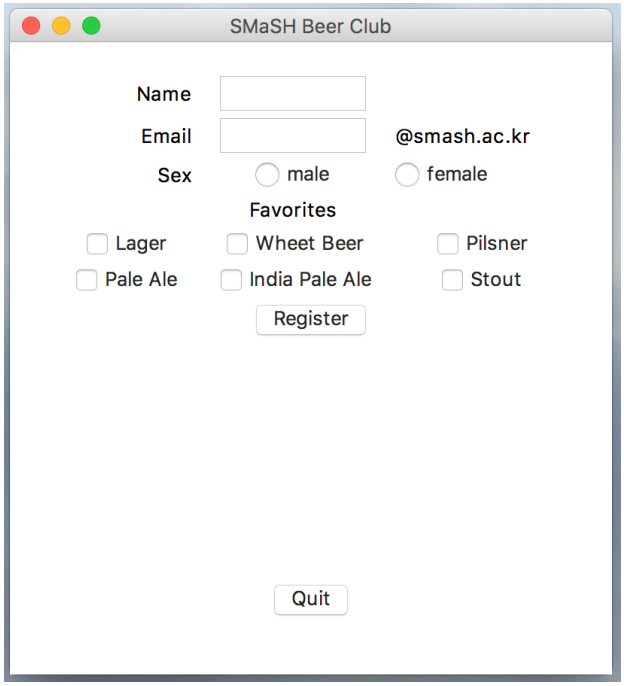





```
1  from tkinter import *
2
3  class App(Frame):
4      def __init__(self, master):
5          super().__init__(master)
6          self.pack(padx = 10, pady = 10)
7          self.button_clicks = 0
8          self.create_widgets()
9
10     def create_widgets(self):
11         Button(self, text="Click me!", command=self.update_count).pack()
12         self.label = Label(self)
13         self.label["text"] = "Clicked me " + str(self.button_clicks) + " times."
14         self.label.pack()
15         Button(self, text="Quit", command=self.quit).pack()
16
17     def update_count(self):
18         self.button_clicks += 1
19         self.label["text"] = "Clicked me " + str(self.button_clicks) + " times."
20
21 root = Tk()
22 root.title("Counter")
23 root.geometry("200x100")
24 app = App(root)
25 root.mainloop()
```

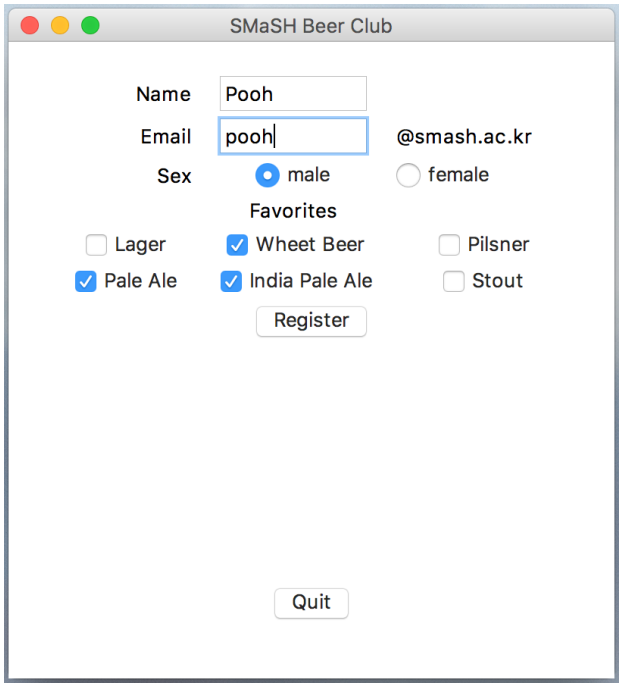
2. 사례 학습 : Beer Club Membership Registration

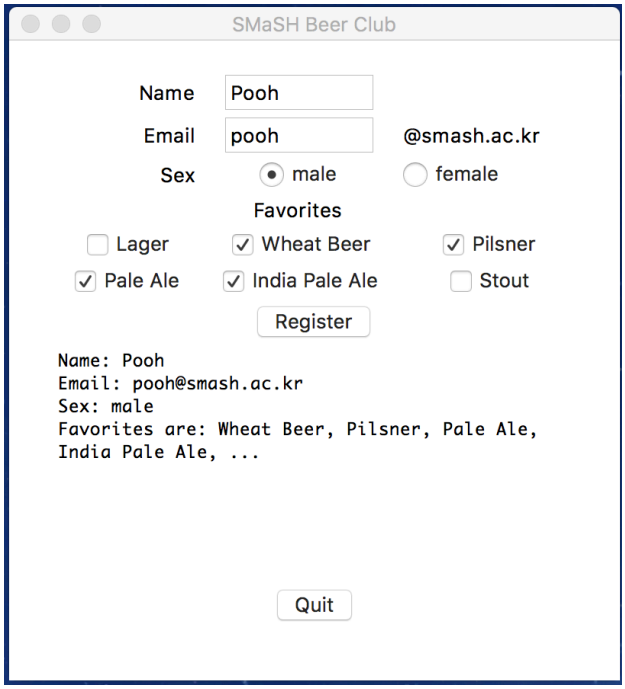
수제 맥주 동아리 클럽의 회원등록창을 만들어보자. (회원 목록을 저장하는 기능은 만들지 않고 그림과 같은 입력 및 출력 인터페이스만 만든다.)
오른쪽 창은 초기에 보여주는 창이다. 이 창을 잘 관찰해보고 이 창에 있는 다양한 위젯이 어떤 종류의 위젯인지 공부해보자.



- Name, Email, Sex, @smash.ac.kr, Favorites는 모두 Label 위젯이다.
- 비어있는 직사각형 두개는 Entry 위젯이다. 사용자가 입력을 문자열로 받아들이는 상자이다.
- male과 female은 각각의 왼쪽에 위치한 원과 함께 Radiobutton 위젯이다. 이 위젯은 둘 중 하나만 선택하게 하는 위젯이다.
- 6가지 맥주의 종류가 나열되어 있는 부분 전체는 Checkbutton 위젯이다. 각 맥주 종류 왼쪽에 있는 네모는 체크박스로 몇개 든 선택할 수 있다.
- Register와 Quit은 Button 위젯이다.
- Register와 Quit은사이의 빈 공백은 보이지는 않지만 무엇이든 여러 줄에 걸쳐서 문자열을 프린트할 수 있는 Text 위젯이다.

이 위젯들을 만들어 Frame에 배치한 다음, 아래 왼쪽 그림과 같이 입력하고 선택하고 Register 단추를 클릭했을 때, 아래 오른쪽 그림과 같이 입력 및 선택 사항을 정리하여 보여주는 프로그램을 만들어보자.

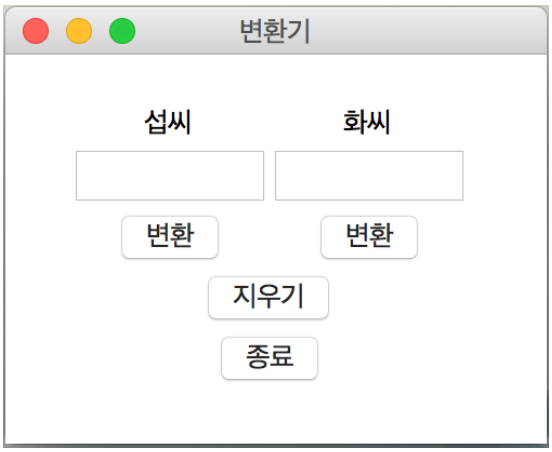




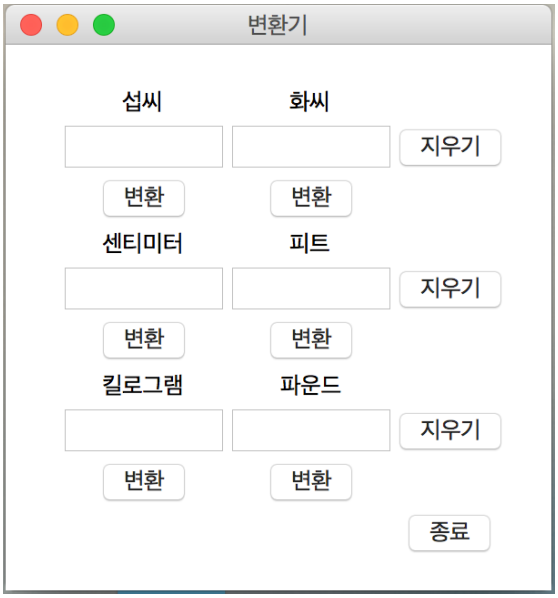
```
from tkinter import *

class App(Frame):
    def __init__(self, master):
        super().__init__(master)
        self.pack(padx=20, pady=20)
        self.create_widgets()

    def create_widgets(self):
        Label(self, text="Name").grid(row=0, column=0, sticky=E)
        self.name = Entry(self, width=10)
        self.name.grid(row=0, column=1)
        Label(self, text="Email").grid(row=1, column=0, sticky=E)
        self.email = Entry(self, width=10)
        self.email.grid(row=1, column=1)
        Label(self, text="@smash.ac.kr").grid(row=1, column=2, sticky=W)
        self.sex = StringVar()
        self.sex.set(None)
        Label(self, text="Sex").grid(row=2, column=0, sticky=E)
        Radiobutton(self, text='male',
                    variable=self.sex, value='male'
                    ).grid(row=2, column=1)
        Radiobutton(self, text='female',
                    variable=self.sex, value='female'
                    ).grid(row=2, column=2, sticky=W)
        Label(self, text="Favorites").grid(row=3, column=1)
        self.lagers = BooleanVar()
        Checkbutton(self, text="Lager", variable=self.lagers
                    ).grid(row=4, column=0)
        self.wheetbeer = BooleanVar()
        Checkbutton(self, text="Wheet Beer", variable=self.wheetbeer
                    ).grid(row=4, column=1)
        self.pilsners = BooleanVar()
        Checkbutton(self, text="Pilsner", variable=self.pilsners
                    ).grid(row=4, column=2)
        self.paleales = BooleanVar()
        Checkbutton(self, text="Pale Ale", variable=self.paleales
                    ).grid(row=5, column=0)
        self.indiapaleales = BooleanVar()
        Checkbutton(self, text="India Pale Ale", variable=self.indiapaleales
                    ).grid(row=5, column=1)
        self.stouts = BooleanVar()
        Checkbutton(self, text="Stout", variable=self.stouts
                    ).grid(row=5, column=2)
```

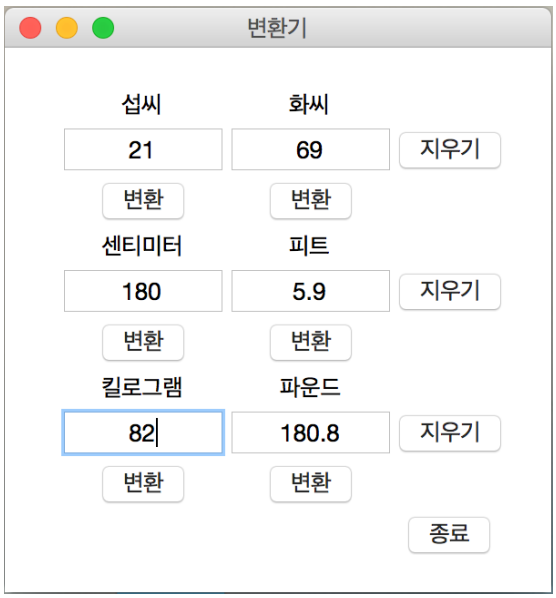
```
45          Button(self, text="Register",
46                 command=self.write_summary
47                 ).grid(row=6, column=0, columnspan=3, sticky=S)
48          self.summary = Text(self, width=48, height=10, wrap=WORD)
49          self.summary.grid(row=7, column=0, columnspan=3, sticky=S)
50          Button(self, text="Quit", command=self.quit
51                 ).grid(row=8, column=0, columnspan=3)
52
53      def write_summary(self):
54          summary = "Name: " + self.name.get() + "\n"
55          summary += "Email: " + self.email.get() + "@smash.ac.kr\n"
56          summary += "Sex: " + self.sex.get() + "\n"
57          summary += "Favorites are: "
58          if self.lager.get():
59              summary += "Lager, "
60          if self.wheetbeer.get():
61              summary += "Wheet Beer, "
62          if self.pilsner.get():
63              summary += "Pilsner, "
64          if self.paleale.get():
65              summary += "Pale Ale, "
66          if self.indiapaleale.get():
67              summary += "India Pale Ale, "
68          if self.stout.get():
69              summary += "Stout, "
70          summary += "..."
71          self.summary.delete(0.0, END)
72          self.summary.insert(0.0, summary)
73
74  # main
75  root = Tk()
76  root.title("SMaSH Beer Club")
77  root.geometry("400x420")
78  App(root)
79  root.mainloop()
```

실습 1 : 계량 변환 애플리케이션



conversion.py 파일에 위의 창과 같이 섭씨 온도를 화씨로 화씨 온도를 섭씨로 변환하는 애플리케이션이 들어있다. 변환하고 싶은 온도를 정수로 입력하고 바로 밑의 변환버튼을 누르면 변환한 값이 정수로 옆 창에 나타난다. 지우기버튼을 누르면 온도데이터가 모두 지워지고, 종료버튼을 누르면 창이 닫힌다. 이 파일의 코드를 모두 이해하고 이를 확장하여 다음 창과 같은 기능을 추가로 갖추도록 애플리케이션을 확장하자.





추가로 구현해야 하는 기능은 다음과 같다.

- 센티미터를 피트로 변환하거나, 피트를 센티미터로 변환 (입력은 정수만 허용, 출력은 소수점 첫째자리 미만에서 반올림)
- 킬로그램을 파운드로 변환하거나, 파운드를 킬로그램으로 변환 (입력은 정수만 허용, 출력은 소수점 첫째자리 미만에서 반올림)
- 해당 입력창에 정수를 입력하고 바로 아래 버튼을 누르면 변환한 값이 옆 창에 나타난다.
- 지우기버튼을 누르면 해당 행의 데이터가 모두 지워진다.

- 종료버튼을 누르면 창이 닫힌다.

변환 공식은 다음과 같다.

centimeters	feets
1	0.03281
30.48	1

kilograms	pounds
1	2.2046
0.4536	1

