

9장. 윈도우 프로그래밍



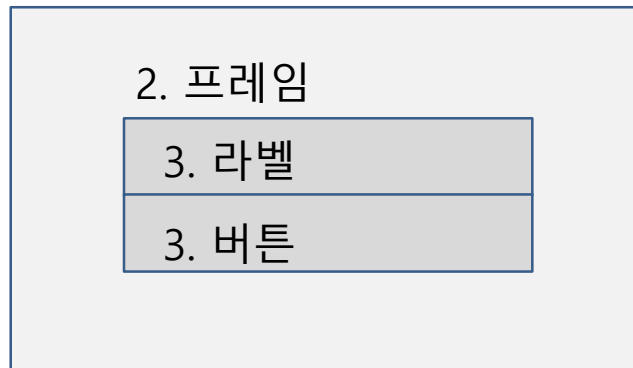
윈도우 프로그래밍

❖ GUI(Graphical User Interface)란?

그래픽 사용자 인터페이스를 줄여서 GUI라고 한다. GUI는 '화면'에 표시된 메뉴나 버튼으로 사용자와 상호 작용을 하는 간단한 프로그램이다.

tkinter 라이브러리를 사용한다. -> **import tkinter**

1. Tk 루트



개체이름	클래스
루트	Tk()
프레임	Frame
레이블	Label
입력상자	Entry
버튼	Button
출력상자	Text

윈도우 프로그래밍

■ 창(Window)

모듈 **from tkinter import ***

윈도우 생성 **root = Tk()**

윈도우 제목 **root.title()**

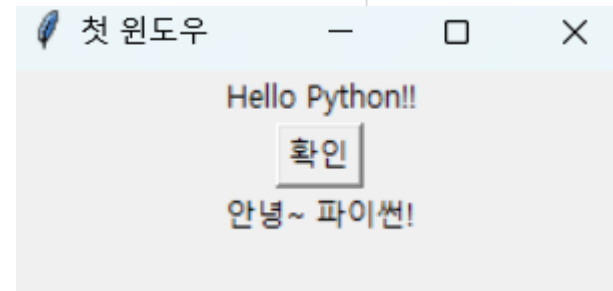
윈도우 구성 **Label, Button**

윈도우 실행 **root.mainloop()**

윈도우 프로그래밍

■ 처음 만드는 윈도우 – pack() 사용

```
def click():  
    # print("안녕~ 파이썬!")  
    demo.config(text="안녕~ 파이썬!")  
  
root = Tk()  
root.title("첫 윈도우")  
root.geometry("250x100+200+100") #너비x높이+x좌표+y좌표  
  
# 라벨과 버튼  
Label(root, text="Hello Python!!").pack()  
Button(root, text="확인", command=click).pack()  
  
# 확인 클릭 후 출력 라벨  
demo = Label(root)  
demo.pack()  
  
root.mainloop()
```



윈도우 프로그래밍

- Button(버튼) – command

Button(frame, text="확인", command=**click**).pack()

※ click에 괄호를 하면 함수 생성시점에서 작동하고, 괄호를 생략하면
클릭이 발생한 때 작동함

1. 콘솔에 출력

```
def click():  
    print("안녕~ 파이썬!")
```

2. 레이블에 출력

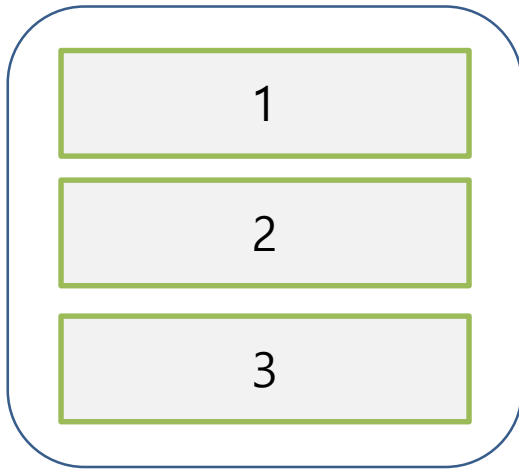
```
def click():  
    demo.config(text="안녕~ 파이썬!")
```

- 창인 크기 및 위치

root.geometry("250x100+200+100") #너비x높이+x좌표+y좌표

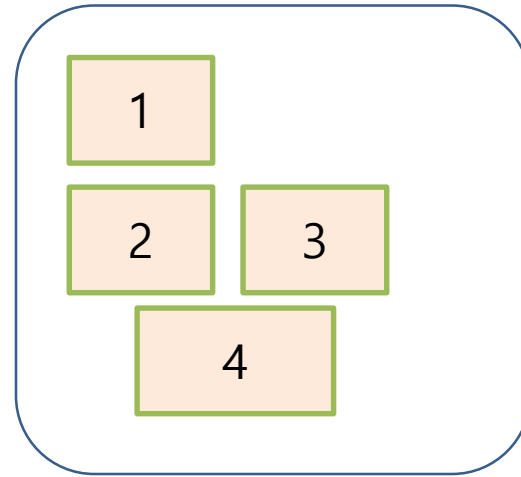
레이아웃(layout)

- 레이아웃 - pack() & grid()



pack()

하나의 컨트롤이 한 줄을 차지함



grid(행, 열)

한 줄에 여러 개의 컨트롤을 배치할 수 있음, 셀 병합도 가능

레이아웃(layout)

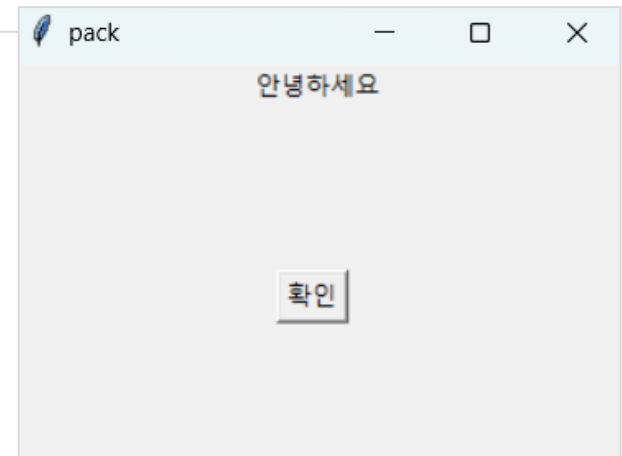
- pack() – 하나의 컨트롤이 한 줄을 차지함
- place(x, y) – 특정 좌표에 배치

```
window = Tk()
window.title('pack')
window.geometry("300x200")

Label(window, text="안녕하세요").pack()
btn = Button(window)
btn.config(text='확인')
# btn.pack()
# btn.pack(side='bottom') #left, right, top, bottom

# 특정 위치에 배치(좌표 사용)
btn.place(x = 130, y = 100)

window.mainloop()
```



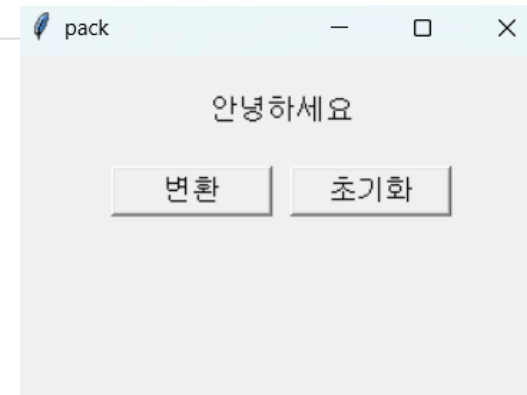
레이아웃(layout)

- `pack(padx=10)` - x축 여백 10픽셀
- `pack(pady=10)` - y축 여백 10픽셀

```
window = Tk()  
window.title('pack')  
window.geometry('300x200')
```

```
frame = Frame(window)  
frame.pack()
```

```
Label(frame, text="안녕하세요").pack(pady=20) #side="top"  
Button(frame, text="변환", width=10).pack(side="left", padx=5)  
Button(frame, text="초기화", width=10).pack(side="left", padx=5)
```



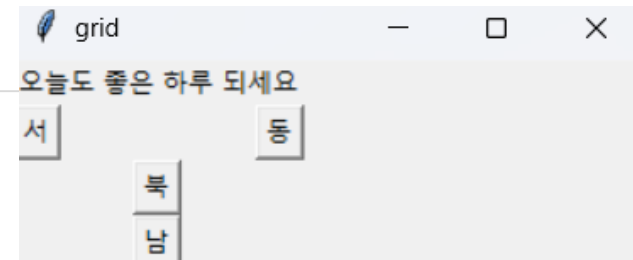
레이아웃(layout)

- grid()

```
window = Tk()
window.title('grid')
window.geometry("300x100")
```

```
Label(window, text="오늘도 좋은 하루 되세요").grid(row=0, column=0)
Button(window, text='동').grid(row=1, column=0, sticky=E)
Button(window, text='서').grid(row=1, column=0, sticky=W)
Button(window, text='북').grid(row=2, column=0, sticky=N)
Button(window, text='남').grid(row=3, column=0, sticky=S)

window.mainloop()
```



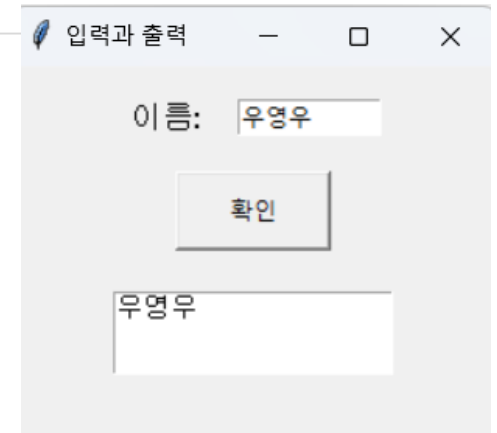
UI 프로그래밍

- 이름을 입력 받고, 화면에 출력하는 윈도우

```
root = Tk()
root.title("입력과 출력")
root.geometry("250x200+200+100")
# root.option_add("*font", "System 12") #글꼴 전체 적용

frame = Frame(root) #프레임 생성
frame.pack()         #가운데 배치

Label(frame, text="이름: ", height=3, font=('System', 12)) \
    .grid(row=0, column=0)
entry = Entry(frame, width=10) #Entry - 입력 상자 클래스
entry.grid(row=0, column=1)
Button(frame, text="확인", command=click, width=10, height=2) \
    .grid(row=1, columnspan=2)
Label(frame, text="").grid(row=2, column=0) #빈 레이블 추가
output = Text(frame, width=20, height=3) #Text - 출력상자 클래스
output.grid(row=3, columnspan=2)
```



UI 프로그래밍

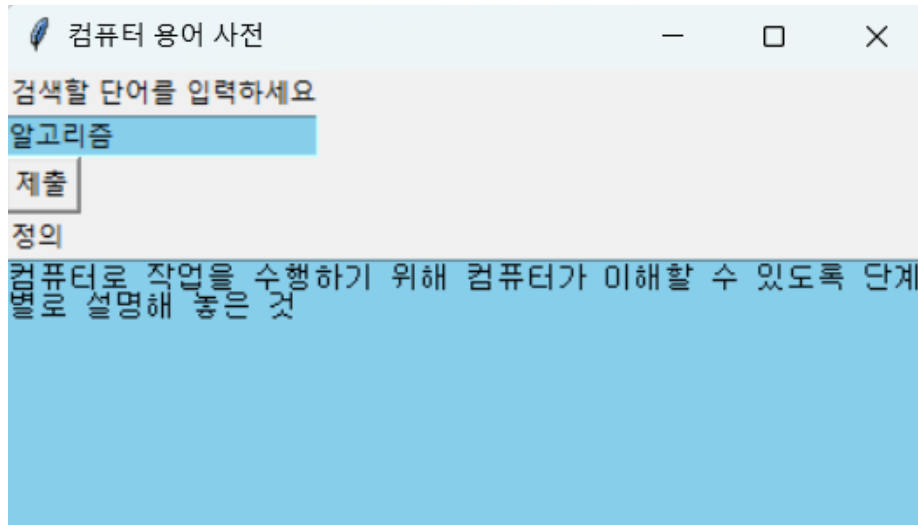
- 이름을 입력 받고, 화면에 출력하는 윈도우

```
def click():  
    text = entry.get()  
    output.delete(0.0, END)  
    output.insert(END, text)
```

#delete(): 0-1행, 0-시작문자, END-끝까지 삭제
#insert(): 끝까지 삭제 후 문자 삽입

컴퓨터 용어 사전

★ 컴퓨터 용어 사전 – dictionary 자료구조 이용



App 설명

- 용어를 미리 정의한다. – 딕셔너리 자료 구조
- 단어를 입력하고 제출 버튼을 누르면, 텍스트 상자에 정의가 출력된다.
- 정의된 단어가 아닌 경우 '단어의 정의를 찾을 수 없습니다.'고 출력된다.

컴퓨터 용어 사전

★ 컴퓨터 용어 사전 – dictionary 자료구조 이용

```
dic = {  
    "이진수": "2진법으로 나타낸 숫자, 0과 1로 구성함",  
    "버그": "프로그램이 적절하게 동작하는데 실패하거나 또는 전혀 동작하지 않는 \  
원인을 제공하는 코드 조각",  
    "함수": "특정한 기능을 수행하는 프로그램으로 재사용 가능한 코드",  
    "알고리즘": "컴퓨터로 작업을 수행하기 위해 컴퓨터가 이해할 수 있도록 \  
단계별로 설명해 놓은 것"  
}  
  
def select():  
    try:  
        word = entry.get() #입력된 단어 가져오기  
        definition = dic[word] #딕셔너리에서 검색  
        output.delete(0.0, END) #초기화  
        output.insert(END, definition) #단어 삽입  
    except:  
        output.delete(0.0, END) #초기화  
        output.insert(END, "단어의 정의를 찾을 수 없습니다.")
```

컴퓨터 용어 사전

★ 컴퓨터 용어 사전

```
root = Tk()
root.title("컴퓨터 용어 사전")

lbl = Label(root, text="검색할 단어를 입력하세요") \
    .grid(row=0, column=0, sticky=W) #sticky=W - 왼쪽에 고정

entry = Entry(root, width=20, bg="skyblue")
entry.grid(row=1, column=0, sticky=W)

Button(root, command=select, text="제출") \
    .grid(row=2, column=0, sticky=W)
Label(root, text="정의").grid(row=3, column=0, sticky=W)

output = Text(root, width=60, height=10, bg="skyblue")
output.grid(row=4, column=0, sticky=W)

root.mainloop()
```

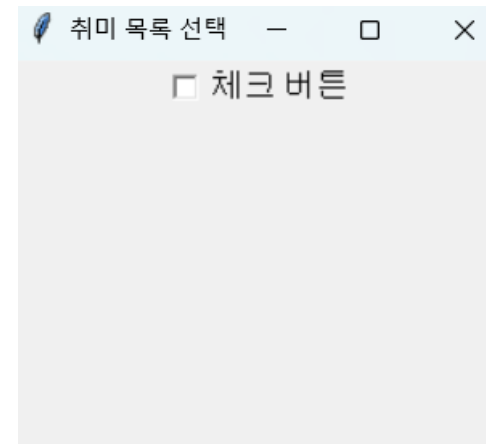
체크 버튼 만들기

■ 체크 버튼 동작 확인

```
import tkinter as tk

def click():
    if ck_val.get() == True:
        print("체크 되었습니다.")
    else:
        print("해제 되었습니다.")

window = tk.Tk()
window.title("취미 목록 선택")
window.geometry("250x200")
```



체크 되었습니다.
해제 되었습니다.

체크 버튼 만들기

- 체크 버튼 동작 확인

```
# BooleanVar() - True/False 속성 가짐
ck_val = tk.BooleanVar()
ck_val.set(False)

# variable 속성 - 변수 역할
tk.Checkbutton(text="체크 버튼", font=("System", 12),
               variable=ck_val, command=click).pack()

window.mainloop()
```

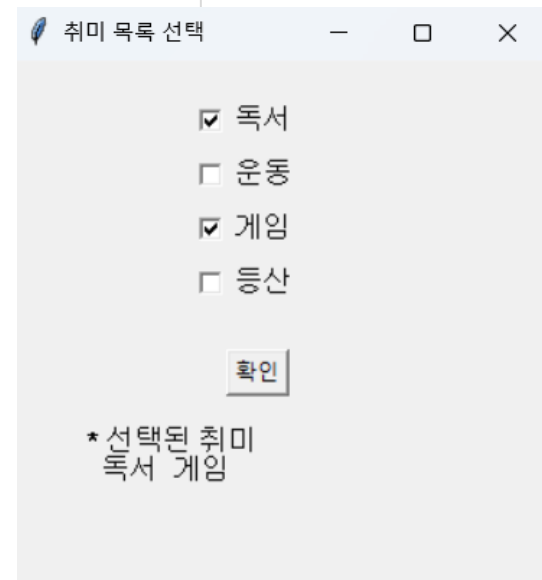

체크 버튼 만들기

■ 취미 목록 만들기

```
def btn_check():  
    result = "* 선택된 취미\n"  
    for i in range(n):  
        if ck_val[i].get():  
            result += f"{hobby[i]} "  
    lbl_result.config(text=result)
```

```
window = tk.Tk()  
window.title("취미 목록 선택")  
window.geometry("300x300")
```

```
hobby = ["독서", "운동", "게임", "등산"]  
n = len(hobby)  
ck_val = [None, None, None, None] #리스트 초기화  
ck_btn = [None]*4
```



체크 버튼 만들기

■ 취미 목록 만들기

```
for i in range(n):
    ck_val[i] = tk.BooleanVar() #True/False
    ck_val[i].set(False) #기본 설정값
    ck_btn[i] = tk.Checkbutton(text=hobby[i], font=("System", 14),
                                variable=ck_val[i])
    # 배치
    ck_btn[i].place(x = 100, y = 20 + (30 * i))

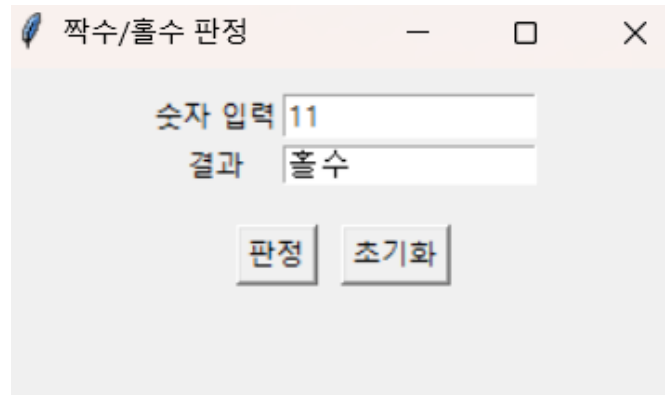
# 확인 버튼
tk.Button(window, text="확인", command=btn_check).place(x=120, y=160)

# 결과 표시 레이블
lbl_result = tk.Label(window, text="", font=("System", 12))
lbl_result.place(x=40, y=200)

window.mainloop()
```

짝수/홀수 판정 프로그램

- 짝수/홀수 판정



App 설명

- 숫자를 입력하면 짝수/홀수를 판정한다.
- 초기화를 수행한다.

짝수/홀수 판정 프로그램

- 짝수/홀수 판정

```
root = Tk()
root.title("짝수/홀수 판정")
root.geometry("300x150+200+200")

# 입력, 출력 프레임
io_frame = Frame(root)
io_frame.pack(pady=10) #가운데 배치

# 입력
Label(io_frame, text="숫자 입력").grid(row=0, column=0)
entry = Entry(io_frame, width=15) #입력
entry.grid(row=0, column=1)

# 출력
Label(io_frame, text="결과").grid(row=1, column=0)
result = Text(io_frame, width=15, height=1) #출력
result.grid(row=1, column=1)
```

짝수/홀수 판정 프로그램

- 짝수/홀수 판정

```
# 버튼 프레임
btn_frame = Frame(root)
btn_frame.pack(pady=5)
Button(btn_frame, text="판정", command=click).pack(side=LEFT, padx=5)
Button(btn_frame, text="초기화", command=reset).pack(side=LEFT, padx=5)

root.mainloop()
```

짝수/홀수 판정 프로그램

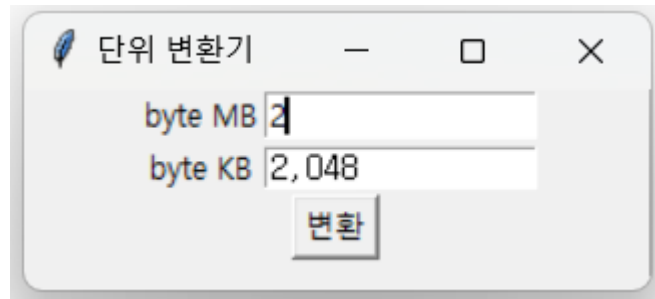
- 짝수/홀수 판정

```
def click():
    try:
        #숫자가 아닌 문자가 입력되면 - ValueError
        num = int(entry.get())
        result.delete(0.0, END) #입력된 숫자 지움
        if num % 2 == 0:
            result.insert(END, "짝수")
        else:
            result.insert(END, "홀수")
    except ValueError:
        #Text는 여러줄(1- 첫째줄, 0- 첫째 열)
        result.delete(1.0, END)
        result.insert(END, '오류')

def reset():
    entry.delete(0, END) #한 줄-0번줄
    result.delete(1.0, END) #여러줄 - 첫째줄, 첫칼럼
```

단위 변환기 1

- 단위 변환기



App 설명

- 메모리 용량을 변환하는 단위 변환기 이다.(MB -> KB)
- 변환 명령은 함수 형태로 코딩되어 있다.

단위 변환기 1

- 단위 변환기 – 함수로 구현

```
def convert():  
    try:  
        # 입력 상자의 자료형은 문자형 -> 숫자로 변환  
        byte_mb = int(entry.get())  
        output.delete(0.0, END)      #이전 입력문자 삭제  
        byte_kb = byte_mb * 1024     #변환 상수 곱함  
        output.insert(END, byte_kb)  #변환된 kb값 저장  
    except ValueError: #숫자가 아닌 문자 입력된 경우 오류 처리  
        output.delete(0.0, END)  
        output.insert(END, "오류")
```


단위 변환기 1

- 단위 변환기 – 함수로 구현

```
window = Tk()
window.title("단위 변환기")
window.geometry("250x100+200+200")
window.option_add("*font", "돋움 13")

frame = Frame(window) #프레임 생성

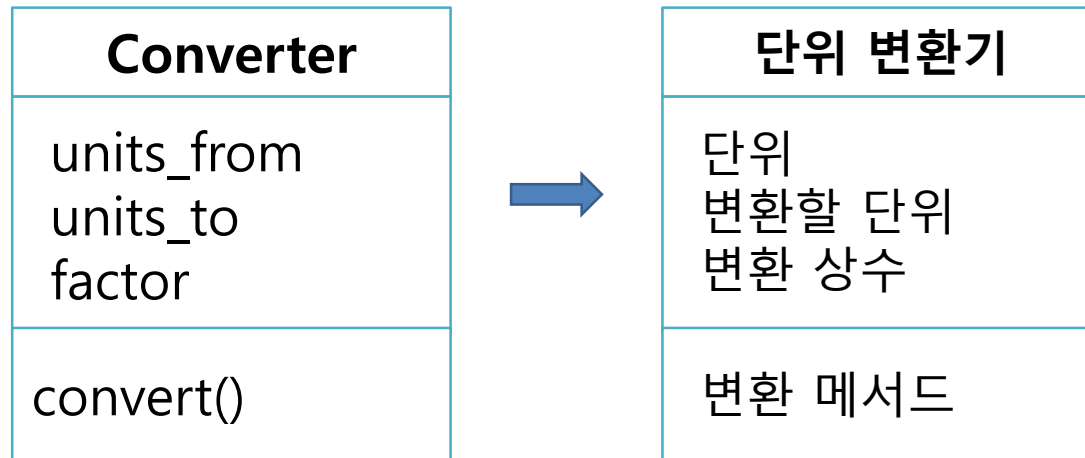
Label(frame, text="byte MB").grid(row=0, column=0)
entry = Entry(frame, width=15)
entry.grid(row=0, column=1)

Label(frame, text="byte KB").grid(row=1, column=0)
output = Text(frame, width=15, height=1)
output.grid(row=1, column=1)

# 변환 버튼
Button(frame, text="변환", command=convert) \
    .grid(row=2, columnspan=2)
```

단위 변환기 클래스

▪ 단위 변환 클래스



1 MB = 1024 KB

1 inch = 25mm

단위 변환기

★ 단위 변환 클래스 - converters.py

```
class Converter:
    def __init__(self, units_from, units_to, factor):
        self.units_from = units_from # 단위
        self.units_to = units_to     # 변환할 단위
        self.factor = factor         # 변환 상수

    def convert(self, value):
        return self.factor * value

if __name__ == "__main__":
    # 메가바이트를 킬로바이트로 변환 : 1MB = 1024KB
    con1 = Converter('MB', 'KB', 1024)
    print(f'1MB = {con1.convert(1)}KB')
    print(f'10MB = {con1.convert(10):,}KB')

    # 인치를 cm로 변환 - 1인치 = 2.54cm
    con2 = Converter('inch', 'cm', 2.54)
    print(f'1inch = {con2.convert(1)}cm')
    print(f'10inch = {con2.convert(10)}cm')
```

단위 변환기2

- 단위 변환기 – 클래스로 구현

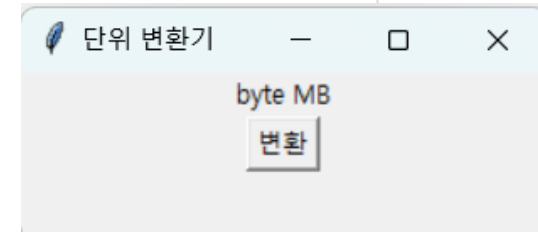
- 계산이 구현되지 않은 단위 변환기 만들기

```
class App:
    def __init__(self, root):
        frame = Frame(root)
        frame.pack()

        Label(frame, text="byte MB").grid(row=0, column=0)
        Button(frame, text="변환", command=self.convert) \
            .grid(row=1, column=0)

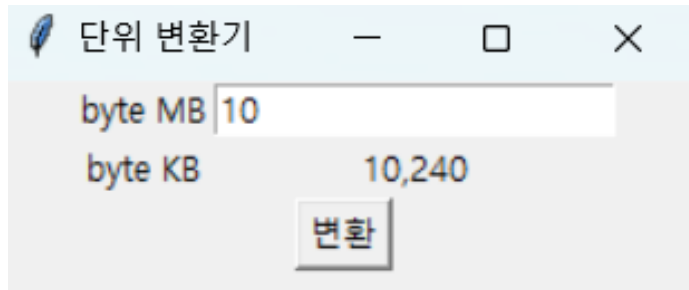
    def convert(self):
        print("아직 구현안됨")

root = Tk()
root.title("단위 변환기")
root.geometry("250x80+200+200")
app = App(root)
```

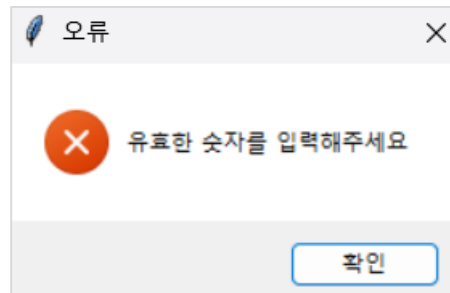


단위 변환기

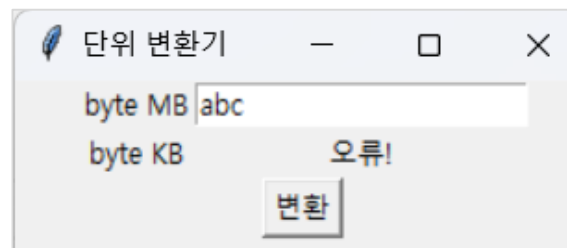
- 단위 변환기 – 클래스로 구현



A screenshot of a '단위 변환기' (Unit Converter) window. The title bar includes a feather icon, the text '단위 변환기', and standard window controls. The main area shows 'byte MB' with a text input field containing '10'. Below it, 'byte KB' is displayed with the value '10,240'. A '변환' (Convert) button is at the bottom.



An '오류' (Error) dialog box with a title bar containing a feather icon and a close button. It features a red circle with a white 'X' icon and the text '유효한 숫자를 입력해주세요' (Please enter a valid number). A '확인' (OK) button is at the bottom.



A screenshot of the '단위 변환기' (Unit Converter) window. The title bar is the same. The main area shows 'byte MB' with a text input field containing 'abc'. Below it, 'byte KB' is displayed with the text '오류!' (Error!). A '변환' (Convert) button is at the bottom.

단위 변환기

● 단위 변환기 – 클래스로 구현

```
from tkinter import *
from tkinter import messagebox
from class_lib.converter import Converter

class App:
    def __init__(self, root):
        # con 객체 생성
        self.con = Converter('MB', 'KB', 1024)
        frame = Frame(root)
        frame.pack()

        Label(frame, text="byte MB").grid(row=0, column=0)
        self.mb = DoubleVar() #실수값을 mb에 저장
        Entry(frame, textvariable=self.mb).grid(row=0, column=1)

        Label(frame, text="byte KB").grid(row=1, column=0)
        self.kb = StringVar() #문자값을 kb에 저장
        Label(frame, textvariable=self.kb).grid(row=1, column=1)

        Button(frame, text="변환", command=self.convert) \
            .grid(row=2, columnspan=2)
```

단위 변환기

● 단위 변환기 – 클래스로 구현

```
def convert(self): #변환 함수 정의
    try:
        mb = self.mb.get()
        if mb < 0:
            messagebox.showerror("오류", "양수를 입력해주세요")
            return
        conv_kb = self.con.convert(mb) #변환 결과값
        conv_kb = f'{self.con.convert(mb):,}' #천단위 구분기호 설정
        self.kb.set(conv_kb) #결과값 저장
    except TclError: #숫자가 아닌 문자가 입력된 경우
        messagebox.showerror("오류", "유효한 숫자를 입력해주세요")
        self.kb.set("오류!")

root = Tk()
root.title("단위 변환기")
root.geometry("250x80+200+200")
app = App(root)
```

GUI 위젯

★ 컨트롤 도구(control tools)

```
from tkinter import *
```

```
class App:
```

```
    def __init__(self, master):
```

```
        frame = Frame(master)
```

```
        frame.pack()
```

```
        Label(frame, text="제목").grid(row=0, column=0)
```

```
        Entry(frame, width=20).grid(row=0, column=1)
```

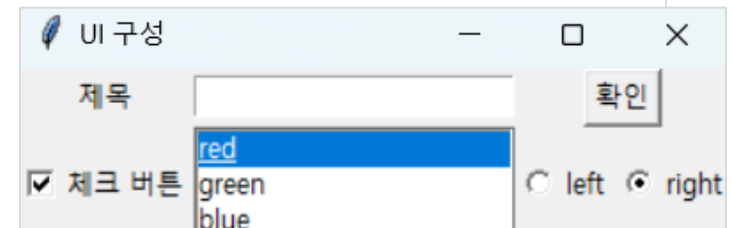
```
        Button(frame, text="확인").grid(row=0, column=2)
```

```
        # 체크버튼
```

```
        check_var = StringVar()
```

```
        check = Checkbutton(frame, text="체크 버튼", variable=check_var,  
                             onvalue='Y', offvalue='N')
```

```
        check.grid(row=1, column=0)
```



GUI 위젯

★ 컨트롤 도구(control tools)

```
# 리스트 목록
listbox = Listbox(frame, height=3, selectmode=SINGLE)
for item in ['red', 'green', 'blue', 'yellow']:
    listbox.insert(END, item)
listbox.grid(row=1, column=1)
# radio 버튼
radio_frame = Frame(frame)
radio_selection = StringVar()
b1 = Radiobutton(radio_frame, text="left",
    variable=radio_selection, value='L')
b1.pack(side=LEFT)
b2 = Radiobutton(radio_frame, text="right",
    variable=radio_selection, value='R')
b2.pack(side=LEFT)
radio_frame.grid(row=1, column=2)

root = Tk()
root.title("UI 구성")
app = App(root) #App 클래스의 객체 생성
```

쿠폰 추첨기 - 실습 예제

- COUPON 추첨



- 추첨 버튼을 누르면 3명의 이름이 랜덤하게 출력됨.
- 이름은 중복되지 않도록 함

쿠폰 추첨기

- 쿠폰 추첨기

```
window = Tk()
window.title("쿠폰 추첨기")
window.option_add('*font', '맑은고딕 13')

lbl_img = Label(window)
#이미지 파일 경로
img = PhotoImage(file = "bronx.png")
lbl_img.config(image = img)
lbl_img.grid(row=0, column=0, sticky=W)

# 추첨 버튼
Button(window, text='추첨', command=click) \
    .grid(row=1, column=0, sticky=W)

# 이름 출력
output = Text(window, width=41, height=4, bg='orange')
output.grid(row=2, column=0, sticky=W)
```

쿠폰 추첨기

- 쿠폰 추첨기

```
from tkinter import *
import random

def click():
    namelist = ['강감찬', '고담덕', '이순신', '장영실',
                '이도', '김구', '신사임당', '유관순',
                '서희', '이산']
    winners = [] #당첨자
    while len(winners) < 3:
        winner = random.choice(namelist) #1명 랜덤 선택
        if winner not in winners: #중복되지 않은 이름
            winners.append(winner) #당첨자 추가

    output.delete(0.0, END)
    output.insert(END, winners)
```

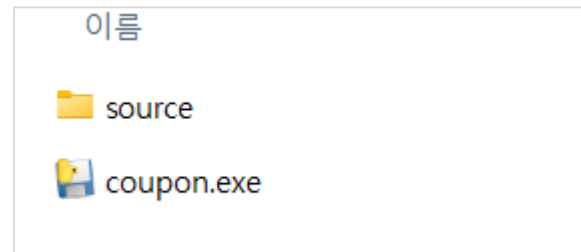
실행 파일(.exe) 만들기

- ✓ 파일(스크립트)이 1개인 경우

```
C:\Wpyworks/gui-tkinter>pyinstaller --noconfirm --onefile  
--windowed 단위변환기.py
```

- ✓ 파일이 2개인 경우 – 이미지 파일 포함

```
C:\Wpyworks/쿠폰추첨기>pyinstaller --noconfirm --onefile  
--windowed --add-data=source/bronx.png:쿠폰추첨기 coupon.py
```



<dist> 폴더