9장. 윈도우 프로그래밍

AI 사용 – 코파일럿(copilot)

❖ 생성형 AI 코파일럿

▪ 챗GPT 프롬프트 입력

AI 사용 – 코파일럿(copilot)

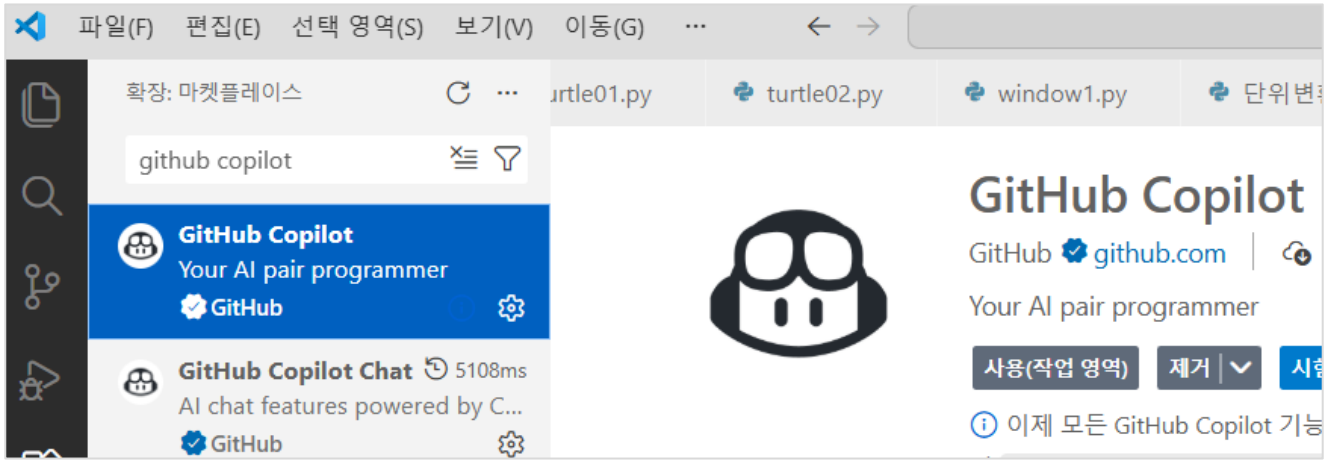
❖ 생성형 AI 코파일럿

▪ 챗GPT 프롬프트 입력

AI 사용 – 코파일럿(copilot)

❖ VS Code에서 GitHub Copilot 사용하기

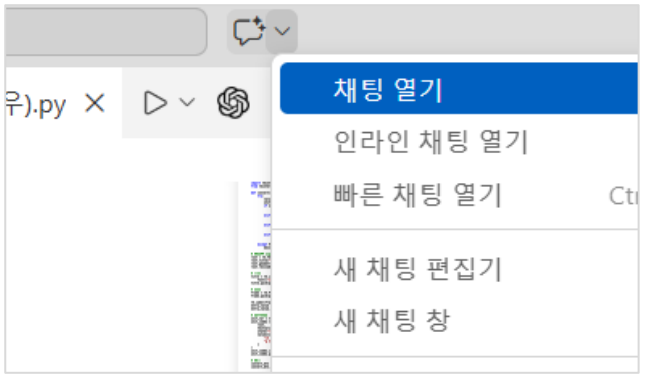
확장 > github copilot 검색 > 설치



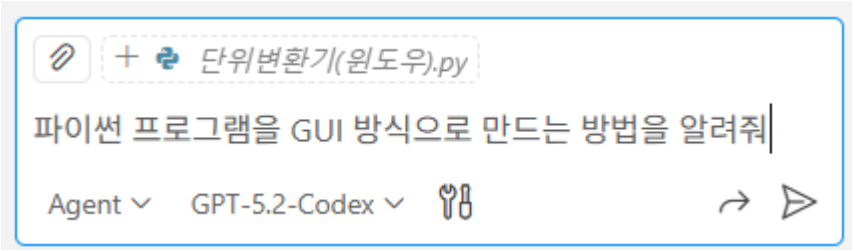
AI 사용 – 코파일럿(copilot)

❖ VS Code에서 GitHub Copilot 사용하기

- **코파일럿 프롬프트 입력**



채팅창에 입력



AI 사용 – 코파일럿(copilot)

❖ GUI 라이브러리 – tkinter

- **GUI 라이브러리 선택**: 표준 `tkinter`(가장 간단), `PyQt/PySide`(기능 풍부), `wxPython` 등
- **기본 흐름**: `창 생성 → 위젯 배치 → 이벤트 연결 → 실행`

예) `tkinter` 최소 구조

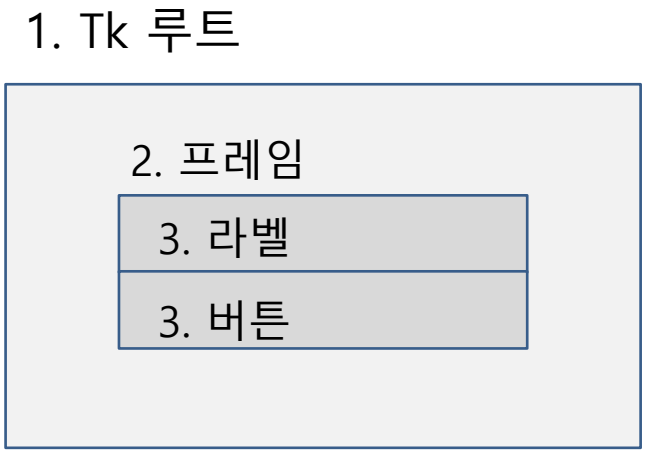
- `Tk()`로 창 생성
- `Label`, `Entry`, `Button` 등 위젯 추가
- `command`로 버튼 클릭 이벤트 연결
- `mainloop()`로 실행

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**



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

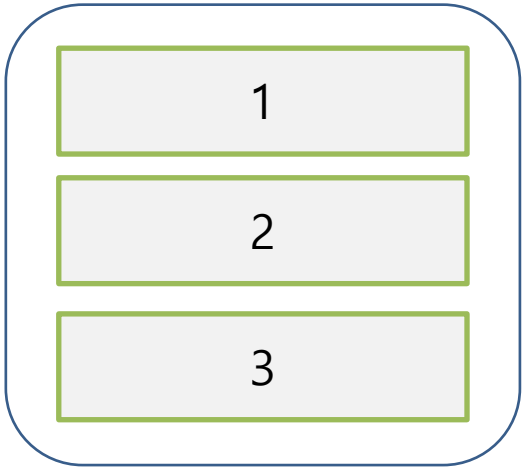
윈도우 프로그래밍

- 창(Window)

모듈	**from tkinter import ***
윈도우 생성	**root = Tk()**
윈도우 제목	**root.title()**
윈도우 구성	**Label, Button**
윈도우 실행	**root.mainloop()**

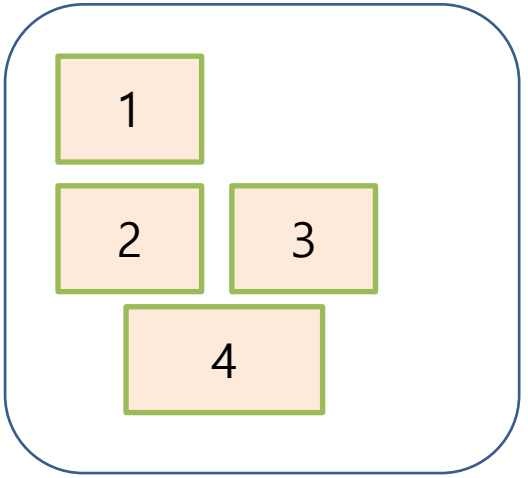
레이아웃(layout)

- 레이아웃 – pack() & grid()



pack()

하나의 컨트롤이 한 줄을 차지함



grid(행, 열)

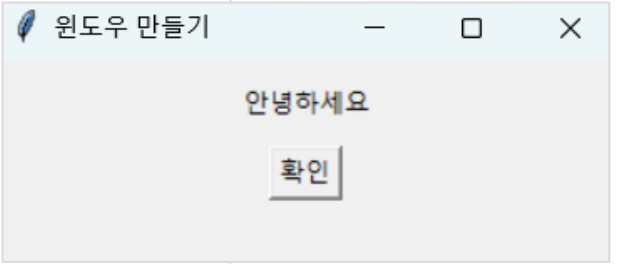
한 줄에 여러 개의 컨트롤을 배치할 수 있음, 셀 병합도 가능

레이아웃(layout)

- pack()

  tkinter 위젯을 부모 컨테이너 안에 배치하는 geometry manager입니다.
  위젯을 상/하/좌/우 방향으로 "쌓아" 배치하며, side, fill, expand, padx, pady 같은 옵션으로 정렬과 여백, 확장을 제어합니다.
  grid()나 place()와는 같은 컨테이너에서 혼용하면 안 됩니다.

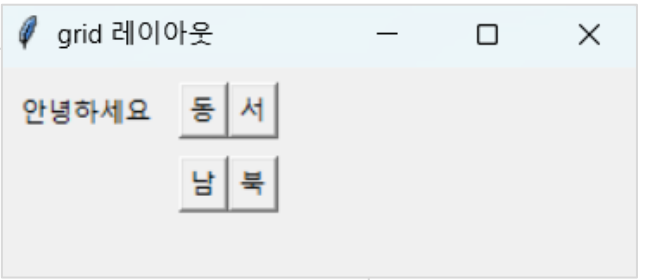
```
from tkinter import *

window = Tk() # 윈도우 객체 생성
window.title("윈도우 만들기")
window.geometry("300x100") # 가로x세로

# 레이블 만들기
label1 = Label(window, text="안녕하세요")
label1.pack(pady=10) # pady: 위아래 여백

# 버튼 만들기
button1 = Button(window, text="확인")
button1.pack()
# button1.pack(side=BOTTOM) # LEFT, RIGHT, TOP, BOTTOM

window.mainloop() # 윈도우 실행
```



레이아웃(layout)

- **grid()**

  tkinter 위젯을 행(row)과 열(column) 좌표에 배치하는 geometry manager입니다. row, column을 기본으로 rowspan, columnspan, sticky, padx, pady로 크기/정렬/여백을 조정합니다. 같은 컨테이너에서 pack()/place()와 혼용하면 안 됩니다.

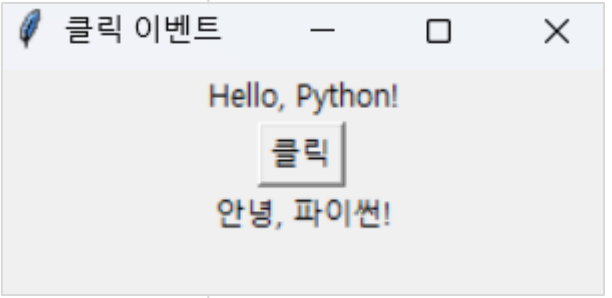



```
from tkinter import *

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

# 레이블 만들기
Label(window, text="안녕하세요").grid(row=0, column=0, padx=10, pady=10)

# 버튼 만들기
# sticky: N, S, E, W (위, 아래, 오른쪽, 왼쪽)
Button(window, text="동").grid(row=0, column=1, sticky=E)
Button(window, text="서").grid(row=0, column=2, sticky=W)
Button(window, text="남").grid(row=1, column=1, sticky=S)
Button(window, text="북").grid(row=1, column=2, sticky=N)

window.mainloop()
```

윈도우 프로그래밍

- **클릭 이벤트 – 버튼을 눌렀을때 문자 출력**

```
from tkinter import *

# 버튼 클릭시 호출되는 함수
def click():
    output.config(text="안녕, 파이썬!")

root = Tk() # 윈도우 창 생성
root.title("클릭 이벤트") # 창 제목 설정
root.geometry("250x100+200+100") # 가로x세로+x좌표+y좌표

# 라벨과 버튼 추가
# pack(): 위젯을 창에 추가하는 메서드
Label(root, text="Hello, Python!").pack()
Button(root, text="클릭", command=click).pack()

# 클릭후 출력 라벨
output = Label(root, text="")
output.pack()

# 이벤트 루프 시작
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

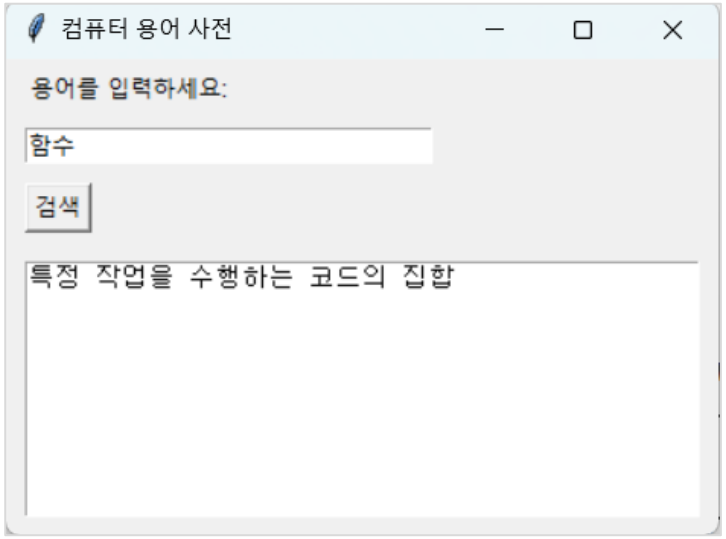
```
def click():
    output.config(text="안녕, 파이썬!")
```

- **창의 크기 및 위치**

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

컴퓨터 용어 사전

★ 컴퓨터 용어 사전 – dictionary 자료구조 이용



App 설명

- 용어를 미리 정의한다. – 딕셔너리 자료 구조
- 단어를 입력하고 검색 버튼을 누르면, 텍스트 상자에 뜻이 출력된다.
- 검색된 단어가 없는 경우 '사전에 없는 용어입니다..'고 출력된다.

컴퓨터 용어 사전

★ 컴퓨터 용어 사전 – dictionary.py

```
from tkinter import *

dict = {
  "알고리즘": "문제를 해결하기 위한 절차나 방법",
  "함수": "특정 작업을 수행하는 코드의 집합",
  "변수": "데이터를 저장하는 메모리 공간의 이름",
  "클래스": "객체 지향 프로그래밍에서 객체를 생성하기 위한 틀",
}

def search():
  # 입력된 단어로 사전 검색
  try:
    word = entry.get() # 입력창에서 단어 가져오기
    meaning = dict[word] # 사전에서 뜻 찾기
    output.delete(1.0, END) # 1번째 줄, 0번째 문자 = 첫 번째 위치
    output.insert(END, meaning) # 끝 위치 (텍스트의 마지막)
  except KeyError:
    output.delete(1.0, END)
    output.insert(END, "사전에 없는 용어입니다.")
```

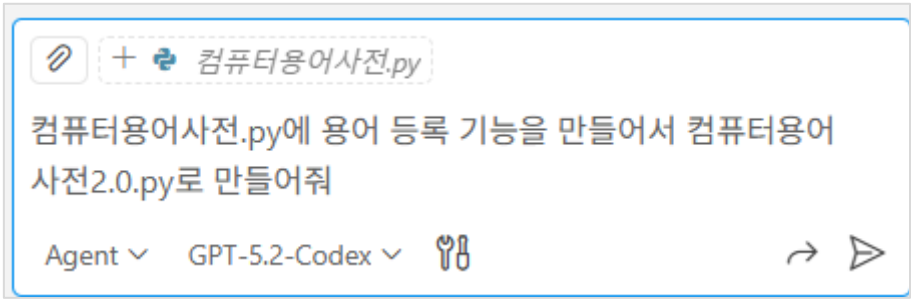
컴퓨터 용어 사전

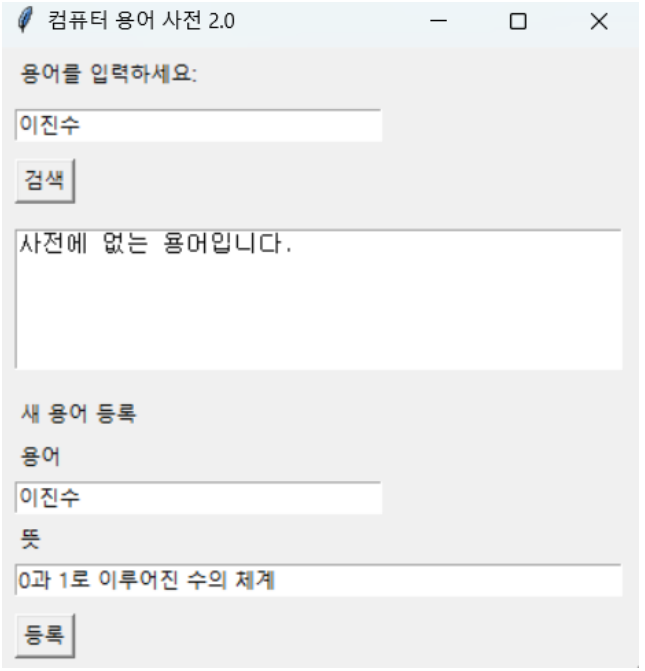
★ 컴퓨터 용어 사전

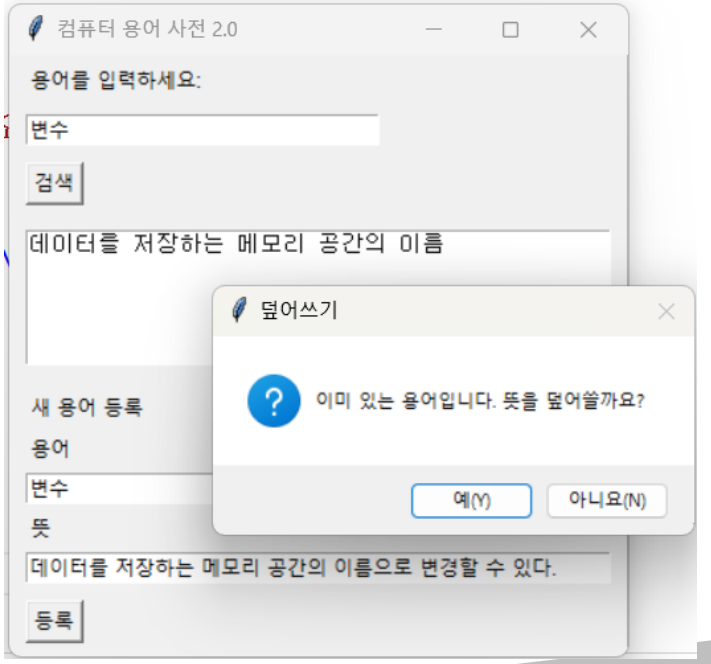
```
root = Tk()
root.title("컴퓨터 용어 사전")

Label(root, text="용어를 입력하세요:") \
  .grid(row=0, column=0, sticky=W, padx=10, pady=5) # 라벨 추가

entry = Entry(root, width=30)
entry.grid(row=1, column=0, sticky=W, padx=10, pady=5) # 입력창 추가

Button(root, text="검색", command=search) \
  .grid(row=2, column=0, sticky=W, padx=10, pady=5) # 버튼 추가

output = Text(root, width=50, height=10)
output.grid(row=3, column=0, sticky=W, padx=10, pady=10) # 출력창 추가

root.mainloop()
```

컴퓨터 용어 사전

- **코파일럿 채팅 입력**







컴퓨터 용어 사전

- **단어 등록 추가**

```
# 새 용어 추가 함수
def add_term():
    word = add_word_entry.get().strip()
    meaning = add_meaning_entry.get().strip()
    if not word or not meaning:
        messagebox.showwarning("입력 오류", "용어와 뜻을 모두 입력하세요.")
        return
    if word in terms:
        if not messagebox.askyesno("덮어쓰기", "이미 있는 용어입니다. \
뜻을 덮어쓸까요?"):
            return
    terms[word] = meaning
    add_word_entry.delete(0, END)
    add_meaning_entry.delete(0, END)
    entry.delete(0, END)
    entry.insert(0, word)
    output.delete(1.0, END)
    output.insert(END, "용어가 등록되었습니다.")
```

컴퓨터 용어 사전

- **단어 등록 추가**

```
# 메인 윈도우 설정
root = Tk()
root.title("컴퓨터 용어 사전 2.0")

# 검색 섹션
Label(root, text="용어를 입력하세요:") \
  .grid(row=0, column=0, sticky=W, padx=10, pady=5)

entry = Entry(root, width=30)
entry.grid(row=1, column=0, sticky=W, padx=10, pady=5)

Button(root, text="검색", command=search) \
  .grid(row=2, column=0, sticky=W, padx=10, pady=5)

output = Text(root, width=50, height=6)
output.grid(row=3, column=0, sticky=W, padx=10, pady=10)
```

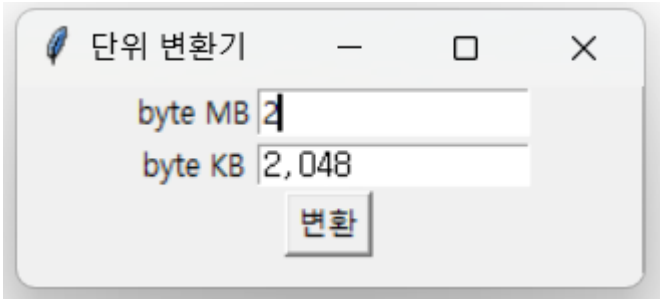
컴퓨터 용어 사전

- **단어 등록 추가**

```
# 새 용어 등록 섹션
Label(root, text="새 용어 등록") \
  .grid(row=4, column=0, sticky=W, padx=10, pady=(5, 2))

Label(root, text="용어") \
  .grid(row=5, column=0, sticky=W, padx=10, pady=2)

add_word_entry = Entry(root, width=30)
add_word_entry.grid(row=6, column=0, sticky=W, padx=10, pady=2)

Label(root, text="뜻") \
  .grid(row=7, column=0, sticky=W, padx=10, pady=2)

add_meaning_entry = Entry(root, width=50)
add_meaning_entry.grid(row=8, column=0, sticky=W, padx=10, pady=2)

Button(root, text="등록", command=add_term) \
  .grid(row=9, column=0, sticky=W, padx=10, pady=8)
```

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")
```

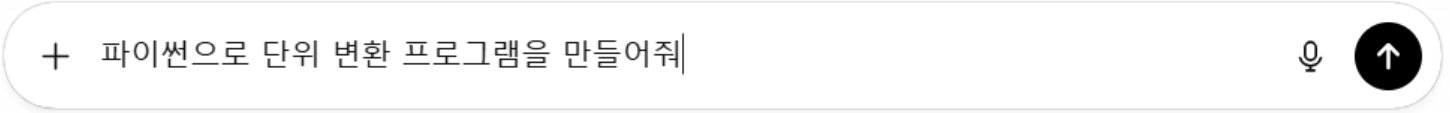
단위 변환기 1

- 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

- **챗GPT 프롬프트 입력**




```
===== 단위 변환 프로그램 =====
1. 길이 변환
2. 무게 변환
0. 종료
메뉴 선택: 1

[길이 변환]
1. cm → m
2. m → cm
선택: 2
값 입력: 10
10.0 m = 1000.0 cm
```

단위 변환기

- **단위 변환기(콘솔)**

```
def length_converter():
    print("\n[길이 변환]")
    print("1. cm → m")
    print("2. m → cm")
    choice = input("선택: ")

    value = float(input("값 입력: "))
    if choice == "1":
        print(f"{value} cm = {value / 100} m")
    elif choice == "2":
        print(f"{value} m = {value * 100} cm")
    else:
        print("잘못된 선택입니다.")
```

단위 변환기

- **단위 변환기(콘솔)**

```
def weight_converter():
    print("\n[무게 변환]")
    print("1. g → kg")
    print("2. kg → g")
    choice = input("선택: ")

    value = float(input("값 입력: "))
    if choice == "1":
        print(f"{value} g = {value / 1000} kg")
    elif choice == "2":
        print(f"{value} kg = {value * 1000} g")
    else:
        print("잘못된 선택입니다.")
```

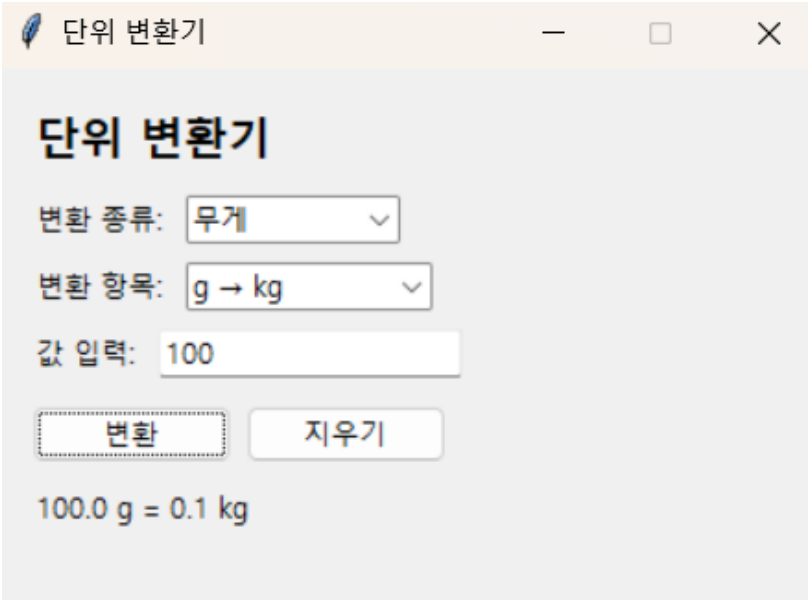
단위 변환기

- **단위 변환기(콘솔)**

```
def main():
    while True:
        print("\n===== 단위 변환 프로그램 =====")
        print("1. 길이 변환")
        print("2. 무게 변환")
        print("0. 종료")

        menu = input("메뉴 선택: ")
        if menu == "1":
            length_converter()
        elif menu == "2":
            weight_converter()
        elif menu == "0":
            print("프로그램 종료")
            break
        else:
            print("잘못된 입력입니다.")

if __name__ == "__main__":
    main()
```

단위 변환기

- **코파일럿 채팅 입력**



단위 변환기

- 단위 변환기(윈도우)

```
import tkinter as tk
from tkinter import ttk, messagebox

# 단위 변환 옵션 정의
LENGTH_OPTIONS = [
    ("cm → m", "cm_to_m"),
    ("m → cm", "m_to_cm"),
]

# 무게 변환 옵션 정의
WEIGHT_OPTIONS = [
    ("g → kg", "g_to_kg"),
    ("kg → g", "kg_to_g"),
]
```

단위 변환기

- 단위 변환기(윈도우)

```
def convert():
    category = category_var.get()
    option = option_var.get()
    value_text = value_var.get().strip()

    if not value_text:
        result_var.set("값을 입력하세요.")
        return

    try:
        value = float(value_text)
    except ValueError:
        result_var.set("숫자만 입력하세요.")
        return
```

단위 변환기

- 단위 변환기(윈도우)

```
if category == "길이":
    if option == "cm_to_m":
        result = value / 100
        result_var.set(f"{value} cm = {result} m")
    elif option == "m_to_cm":
        result = value * 100
        result_var.set(f"{value} m = {result} cm")
    else:
        result_var.set("변환 항목을 선택하세요.")
elif category == "무게":
    if option == "g_to_kg":
        result = value / 1000
        result_var.set(f"{value} g = {result} kg")
    elif option == "kg_to_g":
        result = value * 1000
        result_var.set(f"{value} kg = {result} g")
    else:
        result_var.set("변환 항목을 선택하세요.")
else:
    result_var.set("변환 종류를 선택하세요.")
```

단위 변환기

- **단위 변환기(윈도우)**

```
# 필드 초기화 함수
def reset_fields():
    value_var.set("")
    result_var.set("")
    value_entry.focus_set()

# 카테고리 변경 시 옵션 업데이트
def on_category_change(*_args):
    if category_var.get() == "길이":
        option_menu["values"] = [text for text, _ in LENGTH_OPTIONS]
        option_var.set(LENGTH_OPTIONS[0][1])
        option_display_var.set(LENGTH_OPTIONS[0][0])
    elif category_var.get() == "무게":
        option_menu["values"] = [text for text, _ in WEIGHT_OPTIONS]
        option_var.set(WEIGHT_OPTIONS[0][1])
        option_display_var.set(WEIGHT_OPTIONS[0][0])
    else:
        option_menu["values"] = []
        option_var.set("")
        option_display_var.set("")
```

단위 변환기

- 단위 변환기(윈도우)

```
# 옵션 표시 변경 시 실제 옵션 값 업데이트
def on_option_display_change(*_args):
    display = option_display_var.get()
    for text, key in LENGTH_OPTIONS + WEIGHT_OPTIONS:
        if text == display:
            option_var.set(key)
            return

root = tk.Tk()
root.title("단위 변환기")
root.geometry("360x240")
root.resizable(False, False)

main_frame = ttk.Frame(root, padding=16)
main_frame.pack(fill="both", expand=True)

header = ttk.Label(main_frame, text="단위 변환기",
    font=("맑은 고딕", 14, "bold"))
header.pack(anchor="w")
```

단위 변환기

- **단위 변환기(윈도우)**

```
# 카테고리 선택
category_frame = ttk.Frame(main_frame)
category_frame.pack(fill="x", pady=(10, 4))

category_label = ttk.Label(category_frame, text="변환 종류:")
category_label.pack(side="left")

category_var = tk.StringVar(value="길이")
category_menu = ttk.Combobox(category_frame,
    textvariable=category_var, state="readonly", width=10)
category_menu["values"] = ["길이", "무게"]
category_menu.pack(side="left", padx=8)

# 옵션 선택
option_frame = ttk.Frame(main_frame)
option_frame.pack(fill="x", pady=4)

option_label = ttk.Label(option_frame, text="변환 항목:")
option_label.pack(side="left")
```

단위 변환기

- 단위 변환기(윈도우)

```
option_display_var = tk.StringVar(value=LENGTH_OPTIONS[0][0])
option_var = tk.StringVar(value=LENGTH_OPTIONS[0][1])
option_menu = ttk.Combobox(option_frame,
  textvariable=option_display_var, state="readonly", width=12)
option_menu["values"] = [text for text, _ in LENGTH_OPTIONS]
option_menu.pack(side="left", padx=8)

# 값 입력
value_frame = ttk.Frame(main_frame)
value_frame.pack(fill="x", pady=4)

value_label = ttk.Label(value_frame, text="값 입력:")
value_label.pack(side="left")

value_var = tk.StringVar()
value_entry = ttk.Entry(value_frame, textvariable=value_var, width=18)
value_entry.pack(side="left", padx=8)
```

단위 변환기

- 단위 변환기(윈도우)

```
# 변환 및 결과 표시
button_frame = ttk.Frame(main_frame)
button_frame.pack(fill="x", pady=(8, 4))

convert_button = ttk.Button(button_frame, text="변환",
    command=convert)
convert_button.pack(side="left")

reset_button = ttk.Button(button_frame, text="지우기",
  command=reset_fields)
reset_button.pack(side="left", padx=6)

result_var = tk.StringVar(value="")
result_label = ttk.Label(main_frame,
  textvariable=result_var, wraplength=320)
result_label.pack(fill="x", pady=(6, 0))

category_var.trace_add("write", on_category_change)
option_display_var.trace_add("write", on_option_display_change)

value_entry.bind("<Return>", lambda _e: convert())
value_entry.focus_set()
```

실행 파일(.exe) 만들기

✓ 실행 파일로 배포하기 (.py -> .exe)

C:₩pyworks/단위변환기>pyinstaller --onefile
--windowed 단위변환기(윈도우).py