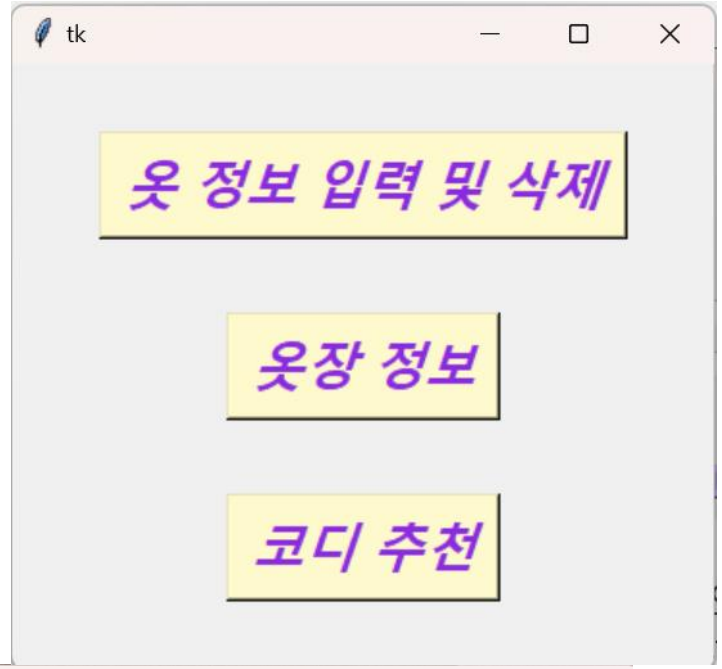
대학기초SW입문
프로젝트 발표 6조

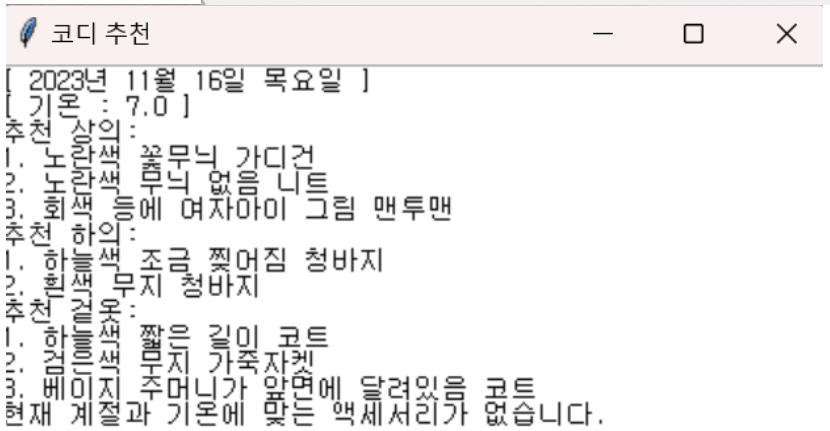
-이은지, 박기쁨, 김시우, 고건우, 윤예원

기온에 따른 의상 코디 추천 프로그램

문제점 : 단순 기온 정보만으로는 어떤 옷을 입어야 할지 직관적으로 와닿지 않고, 또한 옷을 코디할 때 어떻게 해야 할지 어려움을 주로 겪는다.

아이디어: 본인이 가지고 있는 옷의 정보를 프로그램의 옷장에 저장하면 현재 기온에 맞는 의상 코디를 추천해주는 프로그램



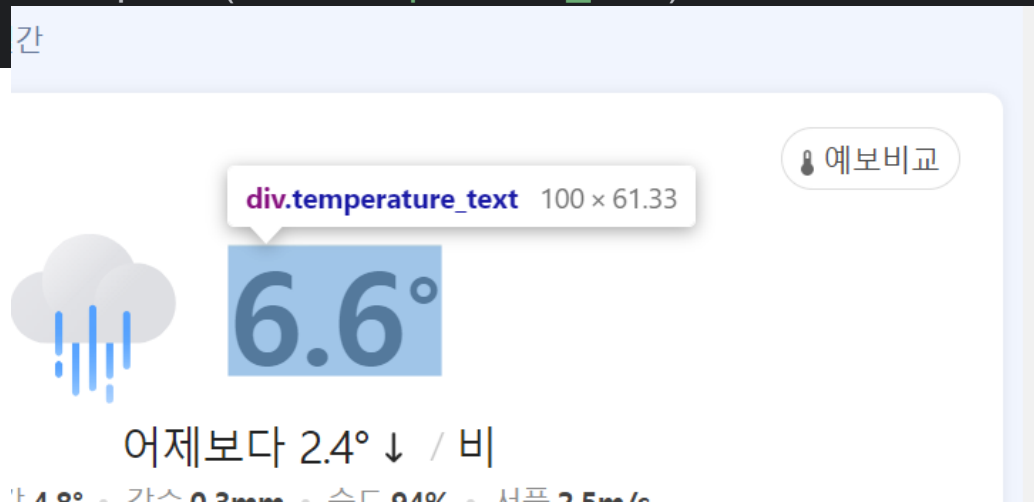


역할 분담

- 1. 현재 기온 가져오기 – 이은지
- 2. 코디 추천 – 박기쁨, 김시우
- 3. 옷 정보 입력 – 고건우
- 4. 전체 코드 통합 – 윤예원

코드 설명 - 이은지

```
Weather = None

# 네이버에서 서울의 기온을 크롤링하는 함수
def GetWeather():
    # https는 필수로 인증서를 체크하게 되어 있기 때문에 검증하지 않은 채 넘어 가기 위해서 context로 ssl.~을 전달한다.
    Context = ssl._create_unverified_context()
    # 네이버의 서울 날씨 웹페이지에 접속한다.
    Webpage = urllib.request.urlopen(
          ' https://search.naver.com/search.naver?sm=top_hty&fbm=0&ie=utf8&query=%EC%84%9C%EC%9A%B8%EB%82%A0%EC%
        context=context)
    # 지정한 url을 파이썬이 대신 열어서 해당 html파일을 복사(파싱)해온다.
    Soup = BeautifulSoup(webpage,  ' html.parser ' )
    # temperature_text클래스이면서 div태그 안에 담겨있는 현재기온 값(temps) 받아오기
     temps = soup.find('div', 'temperature_text')
```



코드설명 - 이은지

```
# 찾은 요소에서 텍스트를 추출하고, 공백을 제거한다.
temps = str(temps.text.strip())
# 현재 기온
global Weather – weather를 전역변수로 선언
# \d+.\d+:숫자들과 숫자들 사이에 어떤 문자가 와 있는 경우를 추출한다.
# re.findall: String에서 패턴에 해당하는 내용을 찾아서 리스트로 리턴한다.
# [0]으로 리스트의 첫 번째 요소, 즉 숫자 패턴의 첫 번째 부분 선택, float로 선택된 숫자를 부동소수점으로 변환.
# 즉, 문자열 temps에서 기온(실수)만 가져온다.
Weather = float(re.findall('\d+.\d+', temps)[0])
```

코드설명 - 김시우

```python
# 상의

def get_recommendation_top(weather, clothing_data):
    recommended_outfits_top = []
    clothing_type = ""
    # 계절에 따른 옷 분류
    if weather == "여름":
        clothing_type = "반팔"
    elif weather == "겨울":
        clothing_type = "두꺼운 긴팔"
    elif weather == "가을":
        clothing_type = "얇은 반팔"
    elif weather == "봄":
        clothing_type = "얇은 반팔"

    # 옷조합 추천 근데 이부분이 지금 제대로 실행이 안되는 거 같음
    for clothing in clothing_data:
        if clothing["category"] == clothing_type:
            recommended_outfits_top.append(clothing)

    # 옷 조합 리턴
    return recommended_outfits_top
# 하의

def get_recommendation_bot(weather, clothing_data):
    recommended_outfits_bot = []
    clothing_type = ""


    if weather == "여름":
        clothing_type = "반바지"
    elif weather == "겨울":
        clothing_type = "두꺼운 긴바지"
    elif weather == "가을":
        clothing_type = "얇은 긴바지"
    elif weather == "봄":
        clothing_type = "얇은 긴바지"


    for clothing in clothing_data:
        if clothing["category"] == clothing_type:
```

상의 하의 추천 코드의 초안

```
    for clothing in clothing_data:
        if clothing["category"] == clothing_type:
            recommended_outfits_bot.append(clothing)


    return recommended_outfits_bot

# 일단 실행 시켜 보려고 넣은 코드 합치면 지울 예정
season = input("계절을 입력하세요 (봄, 여름, 가을, 겨울): ")

# ?
clothing_data = [
    {"color": "파란색", "design": "스트라이프", "category": "반팔"},
    {"color": "빨간색", "design": "체크", "category": "반팔"},
    {"color": "검정색", "design": "플레인", "category": "반팔"},

]

# 위에 코드에서 날씨를 입력받아 상하의 딕셔너리 프린트하고 맞는 옷이 없으면 없다고 출력
recommended_outfits_top = get_recommendation_top(season, clothing_data)
recommended_outfits_bot = get_recommendation_bot(season, clothing_data)
if recommended_outfits_top:
    print("추천 옷 조합:")
    for outfit in recommended_outfits_top:
        print(f"색상: {outfit['color']}, 디자인: {outfit['design']}, 카테고리: {outfit['category']}")
else:
    print("해당 날씨와 계절에 맞는 옷이 없습니다.")

if recommended_outfits_bot:
    print("추천 하의 조합:")
    for outfit in recommended_outfits_bot:
        print(f"색상: {outfit['color']}, 디자인: {outfit['design']}, 카테고리: {outfit['category']}")
else:
    print("해당 날씨와 계절에 맞는 하의가 없습니다.")
```

코드설명 - 김시우

사용자가 계절을
입력하면(온도로 변경)
이를 추천하는 옷을
딕셔너리에 저장

코드설명 - 박기쁨

```python
# 상의 추천 함수
def get_recommendation_top(weather, closet):
    recommended_top = []      # 상의 추천 함수의 리턴 값을 담을 리스트
    clothing_type = []        # 현재 기온에 맞는 상의 타입(카테고리)을 저장할 리스트

    # 기온에 맞는 상의 분류
    if weather <= 16:
        clothing_type = ["맨투맨", "가디건", "후드티", "니트"]
    elif 16 < weather <= 19:
        clothing_type = ["맨투맨", "가디건"]
    elif 19 < weather <= 22:
        clothing_type = ["긴팔", "셔츠", "블라우스"]
    elif 22 < weather <= 27:
        clothing_type = ["반팔"]
    elif 28 < weather:
        clothing_type = ["민소매", "반팔"]

    # 기온에 맞는 상의 보유 여부 확인 및 리턴 값(리스트)에 추가
    for clothing in closet:
        if clothing["category"] in clothing_type:      # 현재 기온에 맞는 상의 타입과 일치하는 상의를 보유하고 있다면
            recommended_top.append(clothing)           # 상의 추천 함수의 리턴 값(리스트)에 옷 추가

    # 추천 상의(리스트) 리턴
    return recommended_top
```

```python
# 하의 추천 함수
def get_recommendation_bot(weather, closet):
    recommended_bot = []     # 하의 추천 함수의 리턴 값을 담을 리스트
    clothing_type = []       # 현재 기온에 맞는 하의 타입(카테고리)을 저장할 리스트

    # 기온에 맞는 하의 분류
    if weather <= 4:
        clothing_type = ["기모 바지"]
    elif 4 < weather <= 8:
        clothing_type = ["청바지"]
    elif 8 < weather <= 22:
        clothing_type = ["면바지", "슬랙스", "청바지", "치마"]
    elif 22 < weather <= 27:
        clothing_type = ["반바지", "면바지", "치마"]
    elif 27 < weather:
        clothing_type = ["반바지", "치마"]

    # 기온에 맞는 하의 보유 여부 확인 및 리턴 값(리스트)에 추가
    for clothing in closet:
        if clothing["category"] in clothing_type:     # 현재 기온에 맞는 상의 타입과 일치하는 상의를 보유하고 있다면
            recommended_bot.append(clothing)          # 하의 추천 함수의 리턴 값(리스트)에 옷 추가

    # 추천 하의(리스트) 리턴
    return recommended_bot
```

```python
# 겉옷 추천 함수
def get_recommendation_out(weather, closet):
    recommended_out = []      # 겉옷 추천 함수의 리턴 값을 담을 리스트
    clothing_type = []        # 현재 기온에 맞는 겉옷 타입(카테고리)을 저장할 리스트

    # 기온에 맞는 겉옷 분류
    if weather <= 4:
        clothing_type = ["패딩"]
    elif 4 < weather <= 19:
        clothing_type = ["가죽 자켓", "코트"]
    elif 8 < weather <= 11:
        clothing_type = ["자켓", "코트", "점퍼"]
    elif 11 < weather <= 16:
        clothing_type = ["자켓", "청자켓"]
    elif 16 < weather <= 19:
        clothing_type = ["바람막이"]

    # 기온에 맞는 겉옷 보유 여부 확인 및 리턴 값(리스트)에 추가
    for clothing in closet:
        if clothing["category"] in clothing_type:      # 현재 기온에 맞는 겉옷 타입과 일치하는 겉옷을 보유하고 있다면
            recommended_out.append(clothing)           # 겉옷 추천 함수의 리턴 값(리스트)에 옷 추가

    # 추천 겉옷(리스트) 리턴
    return recommended_out
```

```python
# 액세서리 추천 함수
def get_recommendation_acc(weather, closet):
    recommended_acc = []      # 액세서리 추천 함수의 리턴 값을 담을 리스트
    clothing_type = []        # 현재 기온에 맞는 액세서리 타입(카테고리)을 저장할 리스트

    # 기온에 맞는 액세서리 분류
    if weather <= 4:
        clothing_type = ["목도리", "장갑"]

    # 기온에 맞는 액세서리 보유 여부 확인 및 리턴 값(리스트)에 추가
    for clothing in closet:
        if clothing["category"] in clothing_type:      # 현재 기온에 맞는 액세서리 타입과 일치하는 상의를 보유하고 있다면
            recommended_acc.append(clothing)           # 액세서리 추천 함수의 리턴 값(리스트)에 옷 추가

    # 추천 액세서리(리스트) 리턴
    return recommended_acc
```

```python
# 상의 추천
if recommended_top:
    print("추천 상의:")
    i = 1
    for clothing in recommended_top:
        print(f"{i}. {clothing['color']} {clothing['design']} {clothing['category']}")
        i += 1
else:
    print("현재 계절과 기온에 맞는 상의가 없습니다.")


# 하의 추천
if recommended_bot:
    print("추천 하의:")
    i = 1
    for clothing in recommended_bot:
        print(f"{i}. {clothing['color']} {clothing['design']} {clothing['category']}")
        i += 1
else:
    print("현재 계절과 기온에 맞는 하의가 없습니다.")
```

```python
# 겉옷 추천
if recommended_out:
    print("추천 겉옷:")
    i = 1
    for clothing in recommended_out:
        print(f"{i}. {clothing['color']} {clothing['design']} {clothing['category']}")
        i += 1
else:
    print("현재 계절과 기온에 맞는 겉옷이 없습니다.")


# 액세서리 추천
if recommended_acc:
    print("추천 액세서리:")
    i = 1
    for clothing in recommended_acc:
        print(f"{i}. {clothing['color']} {clothing['design']} {clothing['
        i += 1
else:
    print("현재 계절과 기온에 맞는 액세서리가 없습니다.")
```

```
기온을 입력하세요 (*C): 3

추천 상의:
1. 노란색 스트라이프 니트
추천 하의:
1. 베이지색 무지 기모 바지
추천 겉옷:
1. 검은색 무지 패딩
추천 액세서리:
1. 빨간색 체크 목도리

Process finished with exit code 0
```

```python
def cloth_enter():
    root = tk.Tk()
    root.title("옷 정보 입력")
    root.geometry("400x300")

    def submit_clothing_info():
        category = category_entry.get()
        design = design_entry.get()
        color = color_entry.get()

        if not category or not design or not color:
            messagebox.showerror("오류", "카테고리, 디자인, 색상은 모두 입력되어야 합니다.")
        else:
            # 이 정보를 저장하거나 활용하는 로직을 추가할 수 있습니다.
            messagebox.showinfo("성공", "옷 정보가 저장되었습니다.")

    def end_cloth():
        root.destroy()  # Tkinter 창을 종료합니다.

    category_label = tk.Label(root, text="카테고리:")
    category_label.pack()
    category_entry = tk.Entry(root)
    category_entry.pack()

    design_label = tk.Label(root, text="디자인:")
    design_label.pack()
    design_entry = tk.Entry(root)
    design_entry.pack()

    color_label = tk.Label(root, text="색상:")
    color_label.pack()
    color_entry = tk.Entry(root)
    color_entry.pack()

    submit_button = tk.Button(root, text="저장", command=submit_clothing_info, width=20, height=2)
    submit_button.pack()

    end_button = tk.Button(root, text="입력 완료", command=end_cloth, width=20, height=2)
    end_button.pack()
```

코드설명 - 고건우

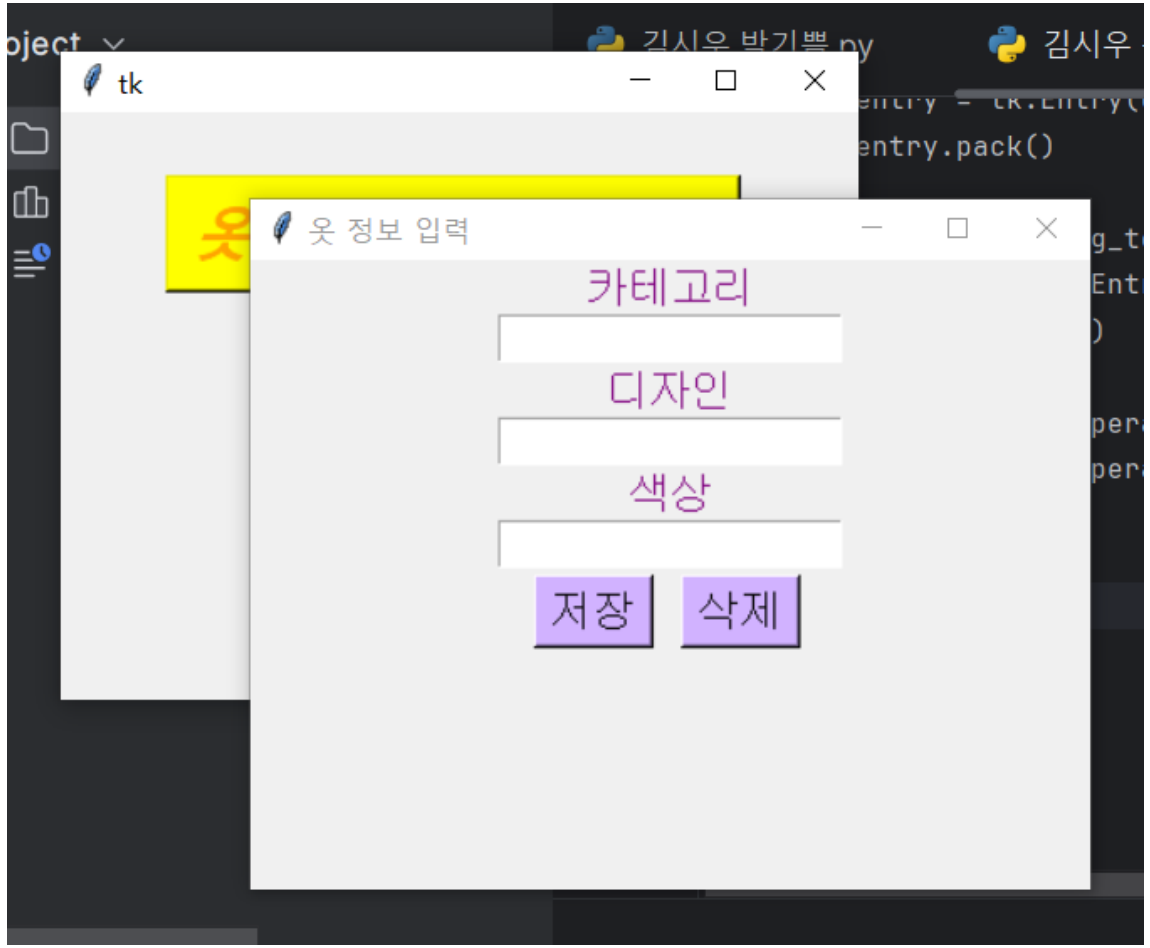
- 옷을 입력받고 저장하는 함수

수정한 코드

수정한 코드설명 - 김시우

```python
# 목록에 없는 옷 종류 어느 온도에 입을 것인지 직접 설정할 수 있는 기능 추가
1 usage
def set_custom_clothing_temperature():
    # 사용자가 입력한 옷 종류를 가져옴
    custom_category = category_entry.get()
    # 사용자가 입력한 온도를 가져옴
    custom_temperature = temperature_entry.get()

    if custom_category and custom_temperature:
        # 사용자가 입력한 옷 정보와 온도를 딕셔너리에 저장
        custom_clothing_temperature[custom_category] = int(custom_temperature)
        messagebox.showinfo("성공", f"{custom_category}은(는) {custom_temperature}도에 어울립니다.")
    else:
        messagebox.showerror("오류", "옷 종류와 온도를 모두 입력해주세요.")



# 온도로 설정하는 창
custom_clothing_temperature_window = tk.Toplevel(root)
custom_clothing_temperature_window.title("옷 종류와 온도 설정")

tk.Label(custom_clothing_temperature_window, text="옷 종류", font=(20), fg='purple').pack()
category_entry = tk.Entry(custom_clothing_temperature_window)
category_entry.pack()

tk.Label(custom_clothing_temperature_window, text="온도", font=(20), fg='purple').pack()
temperature_entry = tk.Entry(custom_clothing_temperature_window)
temperature_entry.pack()

set_custom_clothing_temperature_button = tk.Button(custom_clothing_temperature_window, text="옷 종류와 온도 설정", command=set_custom_clothing_temperature, font=(20),
set_custom_clothing_temperature_button.pack()
```

목록에 없는 옷 종류를 추가 할 수 있는 함수
온도에 따라 옷을 분류해 창을 띄우는 코드



수정한 코드 설명 - 박기쁨

```python
# 기온에 맞는 상의 분류
if Weather <= 16:
    clothing_type = ["맨투맨", "가디건", "후드티", "니트"]
elif 16 < Weather <= 19:
    clothing_type = ["맨투맨", "가디건"]
elif 19 < Weather <= 22:
    clothing_type = ["긴팔", "셔츠", "블라우스"]
elif 22 < Weather <= 27:
    clothing_type = ["반팔"]
elif 28 < Weather:
    clothing_type = ["민소매", "반팔"]

# 기온에 맞는 상의 보유 여부 확인 및 리턴 값(리스트)에 추가
for clothing in closet_Top:
    if clothing["category"] in clothing_type:  # 현재 기온에
        recommended_top.append(clothing)  # 상의 추천 함수의

# 추천 상의(리스트) 리턴
return recommended_top
```

반팔? 반팔티?

```
1 usage
def cloth_enter():  # 옷 정보 입력 및 삭제함수
    cloth__enter = tk.Toplevel(root)  # 새로운 윈도우 창 띄우기
    cloth__enter.title("옷 정보 입력")  # 창 이름
    cloth__enter.geometry("400x300")  # 창 크기

    tk.Label(cloth__enter, text="카테고리", font=(20), fg='purple').pack()  # 카테고리 입력받기
    category_entry = tk.Entry(cloth__enter)
    category_entry.pack()
```

```
# 옷 정보 입력 및 삭제함수
S = None   # 기존의 category_entry 역할 변수
1 usage
def cloth_enter():
    global S
    cloth__enter = tk.Toplevel(root)  # 새로운 윈도우 창 띄우기
    cloth__enter.title("옷 정보 입력")  # 창 이름
    cloth__enter.geometry("400x350")  # 창 크기

    tk.Label(cloth__enter, text="카테고리", font=(20), fg='purple').pack()  # 카테고리 입력받기
```

```
372     tk.Label(cloth__enter, text="카테고리", font=(20), fg='purple').pack()  # 카테고리 입력받기
373     category_entry = tk.Entry(cloth__enter)
374     category_entry.pack()
```

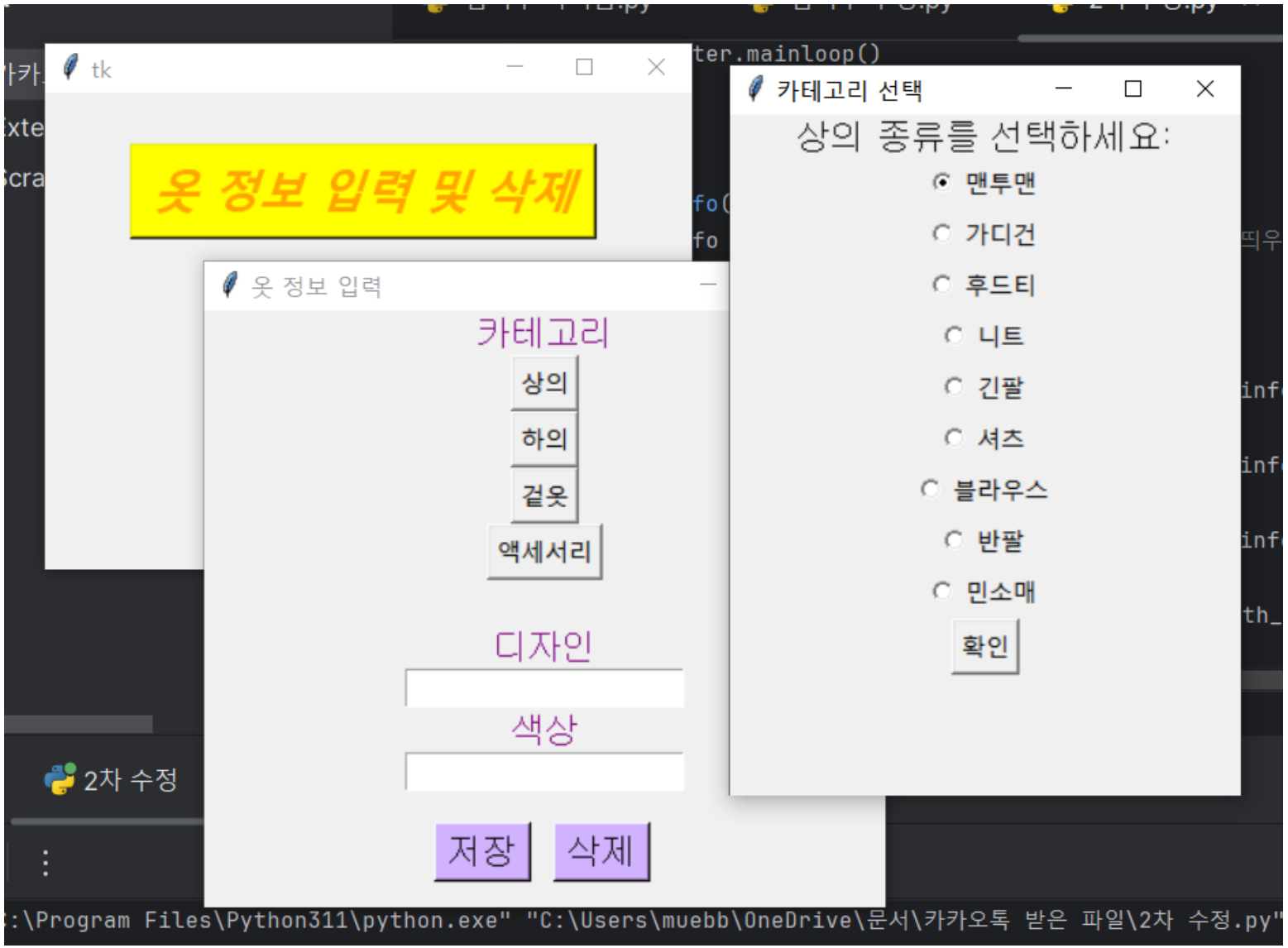
```
358     ##### 버튼과 라디오버튼 코드
359
360     # 상의 버튼
361     button1 = tk.Button(cloth__enter, text="상의", command=lambda: show_radio_buttons_top("상의"))
362     button1.pack()
363     # 하의 버튼
364     button2 = tk.Button(cloth__enter, text="하의", command=lambda: show_radio_buttons_bot("하의"))
365     button2.pack()
366     # 겉옷 버튼
367     button3 = tk.Button(cloth__enter, text="겉옷", command=lambda: show_radio_buttons_out("겉옷"))
368     button3.pack()
369     # 액세서리 버튼
370     button4 = tk.Button(cloth__enter, text="액세서리", command=lambda: show_radio_buttons_acc("액세서리"))
371     button4.pack()
372     # 출력값을 띄워줄 라벨 추가
373     output_label = tk.Label(cloth__enter, text="", font=(20), fg='blue')
374     output_label.pack()
```

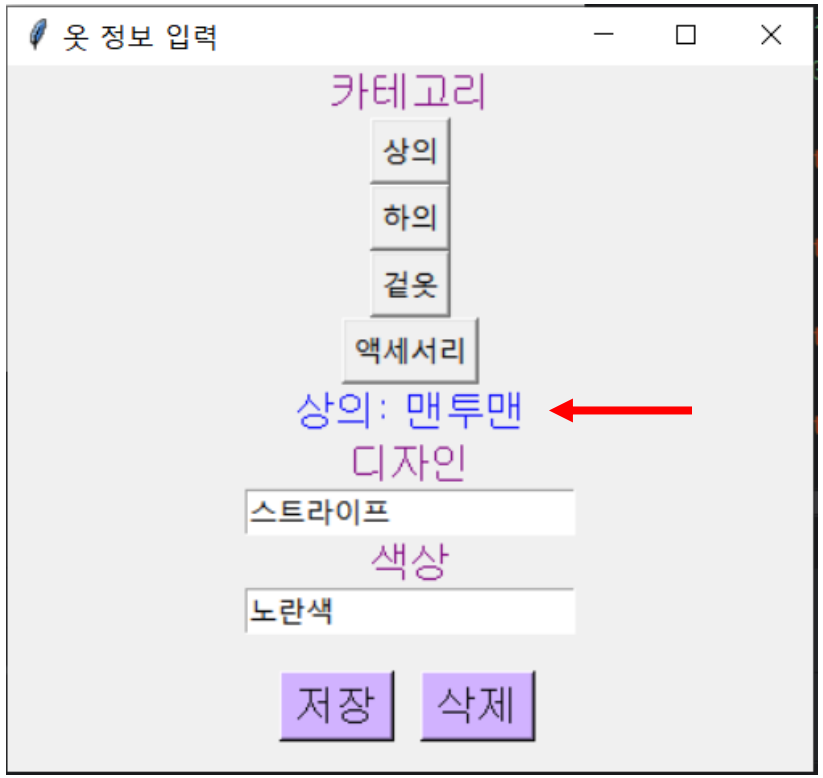
상의 라디오버튼 코드 1

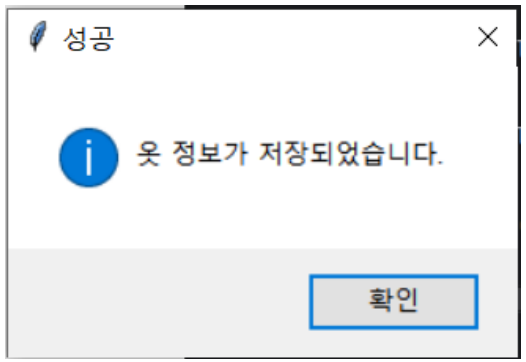
```
376     def show_radio_buttons_top(selected_option):
377         global S
378         # 새로운 창 열기
379         radio_window = tk.Toplevel(root)
380         radio_window.title("카테고리 선택")
381         radio_window.geometry("300x400")
382 
383         tk.Label(radio_window, text="상의 종류를 선택하세요:", font=(16)).pack()
384 
385         # 라디오 버튼 생성
386         selected_radio = tk.StringVar()
387 
388         radio1 = tk.Radiobutton(radio_window, text="맨투맨", variable=selected_radio, value="맨투맨")
389         radio1.pack()
390 
391         radio2 = tk.Radiobutton(radio_window, text="가디건", variable=selected_radio, value="가디건")
392         radio2.pack()
393 
394         radio3 = tk.Radiobutton(radio_window, text="후드티", variable=selected_radio, value="후드티")
395         radio3.pack()
396 
397         radio2 = tk.Radiobutton(radio_window, text="니트", variable=selected_radio, value="니트")
398         radio2.pack()
399 
400         radio2 = tk.Radiobutton(radio_window, text="긴팔", variable=selected_radio, value="긴팔")
401         radio2.pack()
```

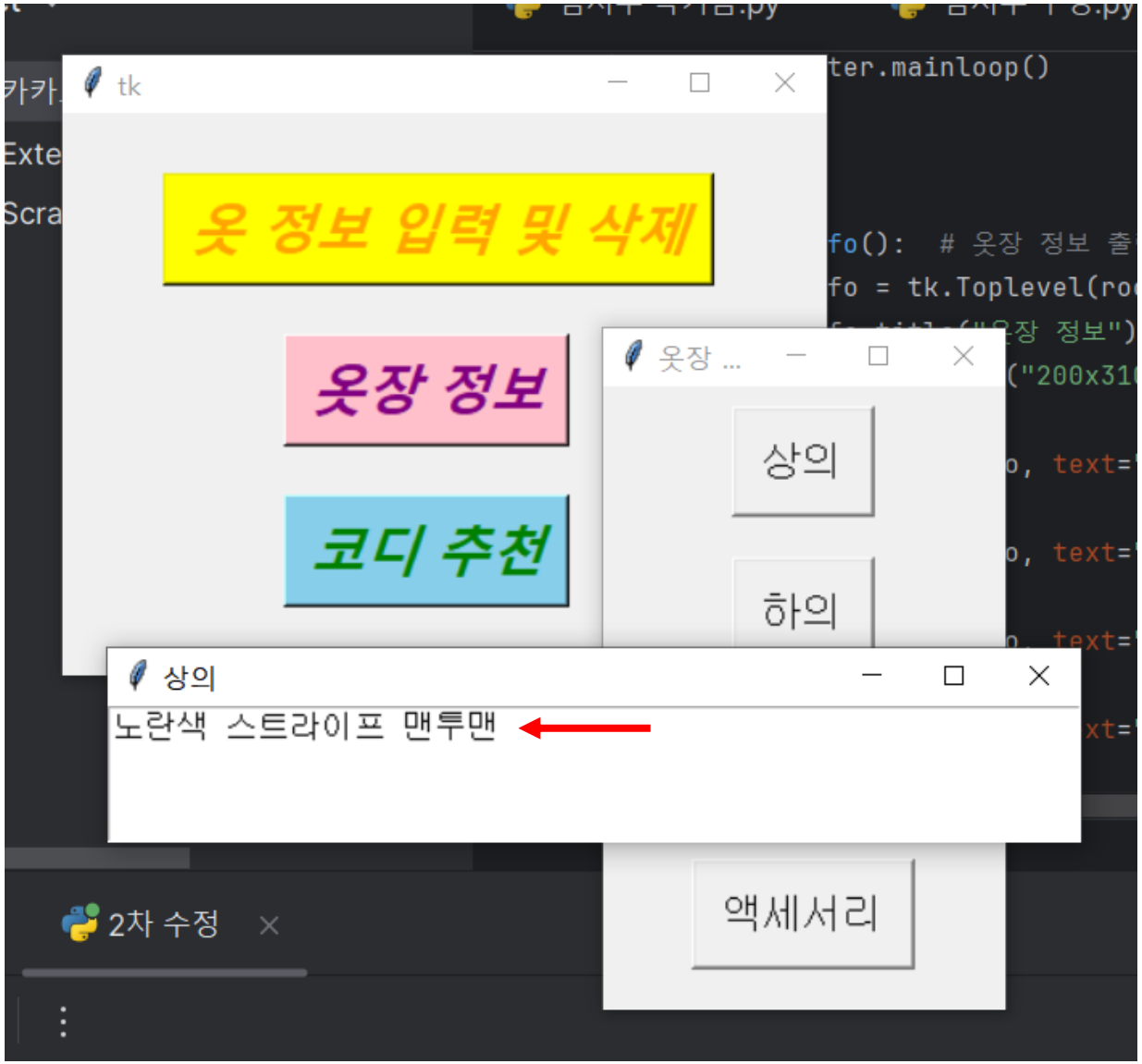
```
403         radio2 = tk.Radiobutton(radio_window, text="셔츠", variable=selected_radio, value="셔츠")
404         radio2.pack()
405 
406         radio2 = tk.Radiobutton(radio_window, text="블라우스", variable=selected_radio, value="블라우스")
407         radio2.pack()
408 
409         radio2 = tk.Radiobutton(radio_window, text="반팔", variable=selected_radio, value="반팔")
410         radio2.pack()
411 
412         radio2 = tk.Radiobutton(radio_window, text="민소매", variable=selected_radio, value="민소매")
413         radio2.pack()
414 
415         # 확인 버튼 생성
416         confirm_button = tk.Button(radio_window, text="확인",
417                                    command=lambda: update_output_label(selected_option, selected_radio.get()))
418         S = selected_radio.get()
419         confirm_button.pack()
420 
421         def update_output_label(selected_option, radio_option):
422             global S
423             # 선택된 옵션을 출력 라벨에 업데이트
424             output_label.config(text=f"{selected_option}: {radio_option}")
425             S = selected_radio.get()
426 
427         # 새로운 창 닫기
428             radio_window.destroy()
```

상의 라디오버튼 코드 2


tk
옷 정보 입력 및 삭제
옷 정보 입력
카테고리
상의
하의
겉옷
액세서리
디자인
색상
저장
삭제
카테고리 선택
상의 종류를 선택하세요:
맨투맨
가디건
후드티
니트
긴팔
셔츠
블라우스
반팔
민소매
확인
2차 수정
ter.mainloop()
:\Program Files\Python311\python.exe" "C:\Users\muebb\OneDrive\문서\카카오톡 받은 파일\2차 수정.py"


옷 정보 입력
카테고리
상의
하의
겉옷
액세서리
상의 : 맨투맨
디자인
스트라이프
색상
노란색
저장
삭제

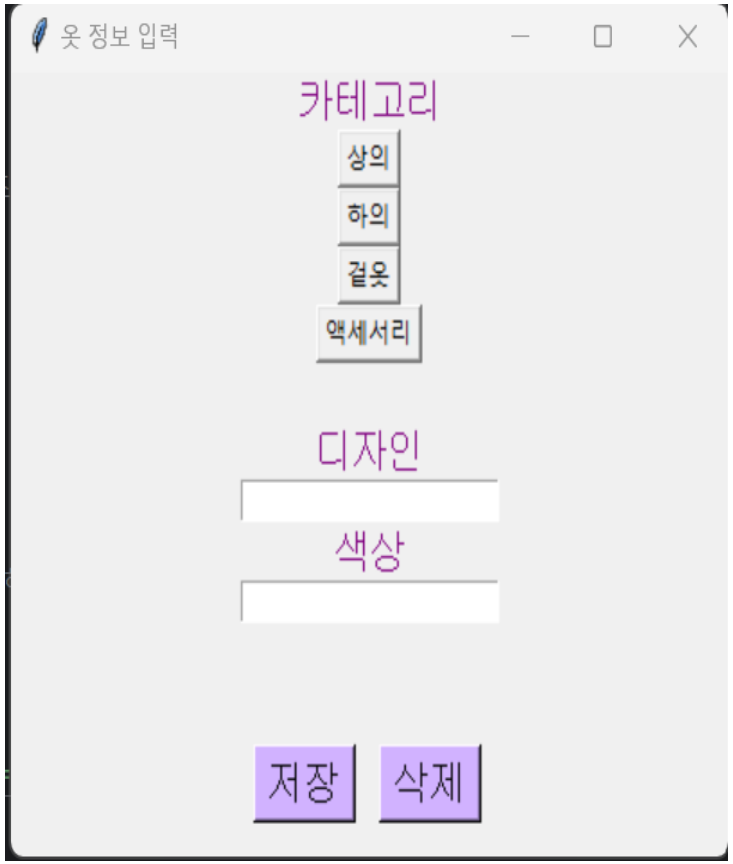

성공
옷 정보가 저장되었습니다.
확인

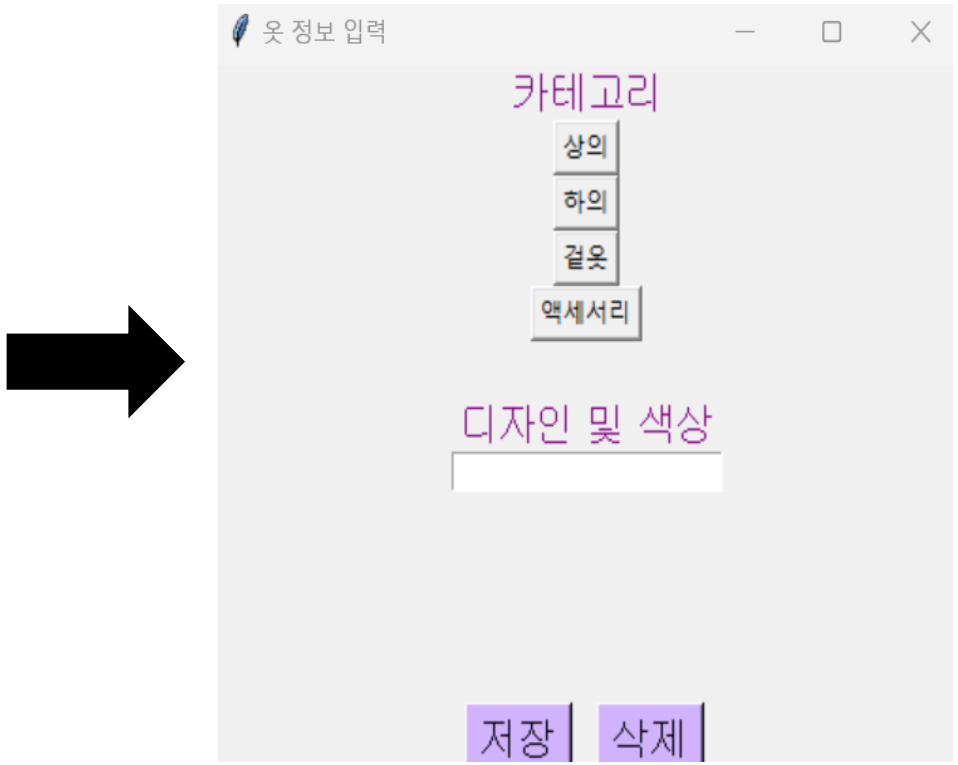

tk
옷 정보 입력 및 삭제
옷장 정보
코디 추천
옷장 ...
상의
하의
액세서리
상의
노란색 스트라이프 맨투맨
2차 수정
ter.mainloop()

수정한 코드 설명 -고건우

```
def center_window(window, width, height):
    screen_width = window.winfo_screenwidth()
    screen_height = window.winfo_screenheight()

    x = (screen_width - width) // 2
    y = (screen_height - height) // 2

    window.geometry(f"{width}x{height}+{x}+{y}")
```

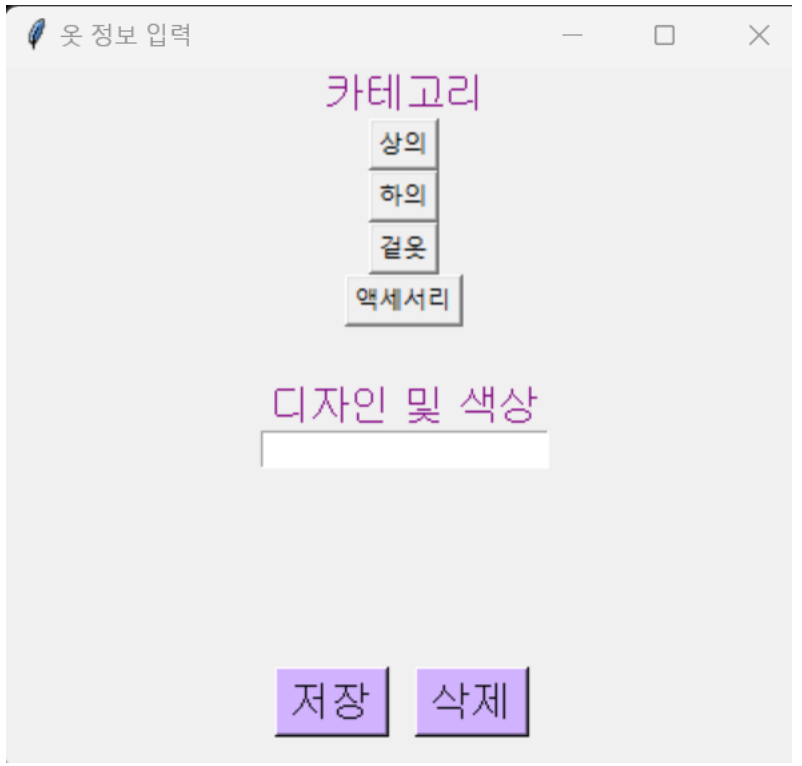
- screenwidth 및 screenheight 메서드를 사용해서 화면 가로 세로 길이를 구한 후 메인창의 크기를 받아 메인창을 화면 중앙에 고정하는 함수


옷 정보 입력
카테고리
상의
하의
겉옷
액세서리
디자인
색상
저장
삭제


옷 정보 입력
카테고리
상의
하의
겉옷
액세서리
디자인 및 색상
저장
삭제

디자인과 색상 통합


tk
옷 정보 입력 및 삭제
옷장 정보
코디 추천


옷 정보 입력
카테고리
상의
하의
겉옷
액세서리
디자인 및 색상
저장
삭제


옷...
상의
하의
겉옷
액세서리

최종 코드

#메인 윈도우 창

```
38      closet_Top = []  # 상의 옷장 리스트
39      closet_Low = []  # 하의 옷장 리스트
40      closet_Outer = []  # 겉옷 옷장 리스트
41      closet_Axcessory = []  # 악세서리 옷장 리스트
42      return_closet = {}
43      custom_closet_Top = []  # 카테고리에 없는 상의 옷장 리스트
44      custom_closet_Low = []  # 카테고리에 없는 하의 옷장 리스트
45      custom_closet_Outer = []  # 카테고리에 없는 겉옷 옷장 리스트
46      custom_closet_Axcessory = []  # 카테고리에 없는 악세서리 옷장 리스트
47      custom_return_closet = {}
48
                                                A2BE2")  # 옷장 정보 출력 버튼
49      year = None  # 년, 일, 요일, 월 전역변수
50      day = None
51      day_of_the_week = None
52      Month = None
831     center_window(root, width: 800, height: 600)
832
833     root.mainloop()
```

#옷 정보를 받아오는 함수 + 저장 함수, 삭제함수

```
418  # 옷 정보 입력 및 삭제함수
419  S = None   # 기존의 category_entry 역할 변수
     1 usage
420  def cloth_enter():
421      global S
422      cloth__enter = tk.Toplevel(root)  # 새로운 윈도우 창 띄우기
423      cloth__enter.title("옷 정보 입력")  # 창 이름
424      cloth__enter.geometry("400x350")  # 창 크기
425
426      tk.Label(cloth__enter, text="카테고리", font=(20), fg='purple').pack()  # 카테고리 입력받기
427
428      ##### 버튼과 라디오버튼 코드
429
430      # 상의 버튼
431      button1 = tk.Button(cloth__enter, text="상의", command=lambda: show_radio_buttons_top("상의"))
432      button1.pack()
433      # 하의 버튼
434      button2 = tk.Button(cloth__enter, text="하의", command=lambda: show_radio_buttons_bot("하의"))
435      button2.pack()
436      # 겉옷 버튼
437      button3 = tk.Button(cloth__enter, text="겉옷", command=lambda: show_radio_buttons_out("겉옷"))
438      button3.pack()
439      # 액세서리 버튼
440      button4 = tk.Button(cloth__enter, text="액세서리", command=lambda: show_radio_buttons_acc("액세서리"))
441      button4.pack()
442      # 출력값을 띄워줄 라벨 추가
443      output_label = tk.Label(cloth__enter, text="", font=(20), fg='blue')
444      output_label.pack()
```

```
439
440     def show_radio_buttons_top(selected_option):
441         global S
442         # 새로운 창 열기
443         radio_window = tk.Toplevel(root)
444         radio_window.title("카테고리 선택")
445         radio_window.geometry("300x400")
446
447         tk.Label(radio_window, text="상의 종류를 선택하세요:", font=(16)).pack()
448
449         # 라디오 버튼 생성
450         selected_radio = tk.StringVar()
451
452         radio1 = tk.Radiobutton(radio_window, text="맨투맨", variable=selected_radio, value="맨투맨")
453         radio1.pack()
454
455         radio2 = tk.Radiobutton(radio_window, text="가디건", variable=selected_radio, value="가디건")
456         radio2.pack()
457
458         radio3 = tk.Radiobutton(radio_window, text="후드티", variable=selected_radio, value="후드티")
459         radio3.pack()
460
461         radio4 = tk.Radiobutton(radio_window, text="니트", variable=selected_radio, value="니트")
462         radio4.pack()
463
464         radio5 = tk.Radiobutton(radio_window, text="긴팔", variable=selected_radio, value="긴팔")
465         radio5.pack()
```

```
466
467         radio6 = tk.Radiobutton(radio_window, text="셔츠", variable=selected_radio, value="셔츠")
468         radio6.pack()
469
470         radio7 = tk.Radiobutton(radio_window, text="블라우스", variable=selected_radio, value="블라우스")
471         radio7.pack()
472
473         radio8 = tk.Radiobutton(radio_window, text="반팔", variable=selected_radio, value="반팔")
474         radio8.pack()
475
476         radio9 = tk.Radiobutton(radio_window, text="민소매", variable=selected_radio, value="민소매")
477         radio9.pack()
478
479         radio10 = tk.Radiobutton(radio_window, text= "카테고리에 없는 옷", variable = selected_radio, value = "기타")
480         radio10.pack()
481
482         # 확인 버튼 생성
483         confirm_button = tk.Button(radio_window, text="확인",
484                                    command=lambda: update_output_label(selected_option, selected_radio.get()))
485         S = selected_radio.get()
486         confirm_button.pack()
487
488         def update_output_label(selected_option, radio_option):
489             global S
490             # 선택된 옵션을 출력 라벨에 업데이트
491             output_label.config(text=f"{selected_option}: {radio_option}")
492             S = selected_radio.get()
493             if (S == "기타"): #'카테고리에 없는 옷' 버튼을 클릭했을 경우 기존 창들을 모두 지우고 카테고리에 없는 옷을 입력받을 함수를 불러온다.
494                 radio_window.destroy()
495                 cloth__enter.destroy()
496                 custom_cloth()
497             else : #카테고리를 입력받았을 경우 창을 지운다.
498                 radio_window.destroy()
```

```
    tk.Label(cloth__enter, text="디자인 및 색상", font=(20), fg='purple').pack()  # 디자인 및 색상 입력받기
    design_entry = tk.Entry(cloth__enter)
    design_entry.pack()

    # 옷 정보를 옷장에 저장하는 함수
    def save_info():
        global return_closet  # return_closet 전역변수(입력받은 옷 정보들을 저장해서 옷장에 분류해서 넣을 때 필요한 변수)
        global S
        return_closet = {}  # return_closet 초기화
        category = S  # 카테고리 받아오기
        design = design_entry.get()  # 디자인 및 색상 받아오기

        return_closet = {"design": design, "category": category}  # return_closet 딕셔너리에 카테고리, 디자인 및 색상 저장

        if not category or not design:  # 카테고리, 디자인 및 색상 중 1개라도 입력되지 않았으면 오류 메시지 출력
            messagebox.showerror( title: "오류",  message: "카테고리, 디자인 및 색상은 모두 입력되어야 합니다.")
        else: #카테고리, 디자인, 색상이 모두 입력되었을 경우 저장되었다는 메시지 출력
            messagebox.showinfo( title: "성공",  message: "옷 정보가 저장되었습니다.")
            cloth__enter.destroy() #창 지우기
            categorizing()  # 옷을 카테고리에 따라 분류하는 함수 호출
```

카테고리에 있는 옷을 옷 카테고리별로 분류하는 함수

```
1 usage
def categorizing():  # cloth_enter함수에서 변경한 전역 변수 return_closet값을 받아 카테고리에 따라 옷장 별로 분류
    global return_closet  # return_closet 전역변수
    if return_closet["category"] in ["맨투맨", "가디건", "후드티", "니트", "긴팔", "셔츠", "블라우스", "반팔", "민소매"]: closet_Top.append(
        return_closet)
    if return_closet["category"] in ["기모바지", "청바지", "면바지", "슬랙스", "치마", "반바지"]: closet_Low.append(return_closet)
    if return_closet["category"] in ["패딩", "가죽자켓", "코트", "자켓", "점퍼", "청자켓", "바람막이"]: closet_Outer.append(return_closet)
    if return_closet["category"] in ["목도리", "장갑"]: closet_Axcessory.append(return_closet)
```

```python
    # 옷 정보를 옷장에서 삭제하는 함수
    def delete_info():
        global return_closet  # return_closet 전역변수(입력받은 옷 정보들을 저장해서 옷장 속 옷들과 비교할 때 필요한 변수)
        global S
        return_closet = {}  # return_closet 초기화
        category = S  # 카테고리 받아오기
        design = design_entry.get()  # 디자인 및 색상 받아오기

        return_closet = {"design": design, "category": category}  # return_closet 딕셔너리에 카테고리, 디자인 및 색상 저장

        if (return_closet not in closet_Top not in closet_Low not in closet_Outer not in closet_Axcessory):
            messagebox.showinfo( title: " ",  message: "삭제할 옷이 존재하지 않습니다.")  # 옷 정보와 일치하는 옷이 없을 때는 삭제할 옷이 없다는 메시지 출력
        if return_closet in closet_Top:  # 상의 옷장에 있으면 상의 옷장 속 동일한 요소 삭제
            closet_Top.remove(return_closet)
            messagebox.showinfo( title: " ",  message: "옷 정보가 삭제되었습니다.")  # 옷 정보가 삭제되었다는 메시지 출력
            cloth__enter.destroy()
        if return_closet in closet_Low:  # 하의 옷장에 있으면 하의 옷장 속 동일한 요소 삭제
            closet_Low.remove(return_closet)
            messagebox.showinfo( title: " ",  message: "옷 정보가 삭제되었습니다.")  # 옷 정보가 삭제되었다는 메시지 출력
            cloth__enter.destroy()
        if return_closet in closet_Outer:  # 겉옷 옷장에 있으면 겉옷 옷장 속 동일한 요소 삭제
            closet_Outer.remove(return_closet)
            messagebox.showinfo( title: " ",  message: "옷 정보가 삭제되었습니다.")  # 옷 정보가 삭제되었다는 메시지 출력
            cloth__enter.destroy()
        if return_closet in closet_Axcessory:  # 악세서리 옷장에 있으면 악세서리 옷장 속 동일한 요소 삭제
            closet_Axcessory.remove(return_closet)
            messagebox.showinfo( title: " ",  message: "옷 정보가 삭제되었습니다.")  # 옷 정보가 삭제되었다는 메시지 출력
            cloth__enter.destroy()
```

```python
    save_button = tk.Button(cloth__enter, text="저장", command=save_info, font=(20), bg='#D1B2FF')  # 저장 버튼
    save_button.place(x=135, y=300)
    delete_button = tk.Button(cloth__enter, text="삭제", command=delete_info, font=(20), bg='#D1B2FF')  # 삭제 버튼
    delete_button.place(x=205, y=300)

    cloth__enter.mainloop()
```

#카테고리에 없는 옷 정보를 받아오는 함수 + 저장함수, 삭제 함수

```
# 카테고리에 없는 옷 정보를 받아오는 함수
4 usages
def custom_cloth():
    custom_clothing_temperature_window = tk.Toplevel(root) #새로운 윈도우 창 띄우기
    custom_clothing_temperature_window.title("옷 종류와 온도 설정") #창 이름

    type_radio = tk.StringVar()  #옷 종류를 받아올 변수 생성
    tk.Label(custom_clothing_temperature_window, text="옷 종류를 선택해 주세요").pack()

    # 상의 라디오 버튼
    button1 = tk.Radiobutton(custom_clothing_temperature_window, text="상의", variable=type_radio,value="상의")
    button1.pack()
    # 하의 라디오 버튼
    button2 = tk.Radiobutton(custom_clothing_temperature_window, text="하의", variable=type_radio,value="하의")
    button2.pack()
    # 겉옷 라디오 버튼
    button3 = tk.Radiobutton(custom_clothing_temperature_window, text="겉옷", variable=type_radio,value="겉옷")
    button3.pack()
    # 액세서리 라디오 버튼
    button4 = tk.Radiobutton(custom_clothing_temperature_window, text="액세서리", variable=type_radio,value="액세서리")
    button4.pack()
```

```
    tk.Label(custom_clothing_temperature_window, text="카테고리", font=(20), fg='purple').pack() #카테고리 받아오기
    custom_category = tk.Entry(custom_clothing_temperature_window)
    custom_category.pack()

    tk.Label(custom_clothing_temperature_window, text="온도", font=(20), fg='purple').pack() #온도 받아오기
    temperature_entry = tk.Entry(custom_clothing_temperature_window)
    temperature_entry.pack()

    tk.Label(custom_clothing_temperature_window, text="디자인 및 색상").pack() #디자인 및 색상 받아오기
    custom_design = tk.Entry(custom_clothing_temperature_window)

# 목록에 없는 옷 종류 어느 온도에 입을 것인지 직접 설정할 수 있는 기능 추가
#카테고리에 없는 옷 정보를 저장하는 함수
def set_custom_clothing_temperature():
    global custom_return_closet #custom_return_closet 전역변수(입력받은 옷 정보들을 저장해서 옷장에 분류해서 넣을 때 필요한 변수)
    # 사용자가 입력한 옷 카테고리를 가져옴
    category = custom_category.get()
    # 사용자가 입력한 온도를 가져옴
    temperature = temperature_entry.get()
    type = type_radio.get() #옷 종류 받아오기
    design = custom_design.get() #디자인 및 색상 받아오기


    # 사용자가 입력한 옷 정보와 온도를 딕셔너리에 저장
    custom_return_closet = {"type": type, "category": category, "design": design, "temperature": temperature}

    if not type or not category or not temperature or not design: #종류,카테고리,디자인 및 색상,온도 중 한 개라도 입력되지 않았을 경우 오류 메시지 출력
        messagebox.showerror( title: "오류",  message: "종류, 카테고리, 디자인 및 색상, 온도를 모두 입력해주세요.")
    else: #모든 정보가 입력되었을 경우 저장되었다는 메시지 출력
        messagebox.showinfo( title: "성공",  message: "옷 정보가 저장되었습니다.")
        custom_clothing_temperature_window.destroy()  # 창 지우기
        custom_categorizing() #옷 분류 함수 호출
```

#카테고리에 없는 옷을 옷 종류별로 분류하는 함수

```
1 usage
319  def custom_categorizing() : # custom_cloth함수에서 옷 정보 입력 및 삭제 함수에서 변경한 전역 변수 custom_return_closet값을 받아 카테고리에 따라 옷장 별로 분류
320      if custom_return_closet["type"] == "상의" : custom_closet_Top.append(custom_return_closet)
321      if custom_return_closet["type"] == "하의": custom_closet_Low.append(custom_return_closet)
322      if custom_return_closet["type"] == "겉옷": custom_closet_Outer.append(custom_return_closet)
323      if custom_return_closet["type"] == "액세서리": custom_closet_Axcessory.append(custom_return_closet)
```

```python
    #카테고리에 없는 옷 정보를 삭제하는 함수
    def delete_custom_clothing_temperature() :
        global custom_return_closet #custon_return_closet 전역변수(입력받은 옷 정보들을 저장해서 옷장 속 옷들과 비교할 때 필요한 변수)
        category = custom_category.get() #카테고리 받아오기
        temperature = temperature_entry.get() #온도 받아오기
        type = type_radio.get() #옷 종류 받아오기
        design = custom_design.get() #디자인 및 색상 받아오기

        #사용자가 입력한 옷 정보와 온도를 딕셔너리에 저장
        custom_return_closet = {"type" : type, "category": category, "design": design, "temperature" : temperature}

        if(custom_return_closet not in custom_closet_Top not in custom_closet_Low not in custom_closet_Outer not in custom_closet_Axcessory):
            #옷 정보와 일치하는 옷이 없을 때는 삭제할 옷이 없다는 메시지 출력
            messagebox.showinfo( title: " ", message: "삭제할 옷이 존재하지 않습니다.")
        if return_closet in custom_closet_Top : #상의 옷장에 있으면 상의 옷장 속 동일한 요소 삭제
            custom_closet_Top.remove(custom_return_closet)
            messagebox.showinfo( title: " ", message: "옷 정보가 삭제되었습니다.") #삭제되었다는 메시지 출력
            custom_clothing_temperature_window.destroy() #창 지우기
        if return_closet in custom_closet_Low: #하의 옷장에 있으면 하의 옷장 속 동일한 요소 삭제
            custom_closet_Low.remove(custom_return_closet)
            messagebox.showinfo( title: " ", message: "옷 정보가 삭제되었습니다.") #삭제되었다는 메시지 출력
            custom_clothing_temperature_window.destroy() #창 지우기
        if return_closet in custom_closet_Outer: #겉옷 옷장에 있으면 겉옷 옷장 속 동일한 요소 삭제
            custom_closet_Outer.remove(custom_return_closet)
            messagebox.showinfo( title: " ", message: "옷 정보가 삭제되었습니다.") #삭제되었다는 메시지 출력
            custom_clothing_temperature_window.destroy() #창 지우기
        if return_closet in custom_closet_Axcessory: #액세서리 옷장에 있으면 액세서리 옷장 속 동일한 요소 삭제
            custom_closet_Axcessory.remove(custom_return_closet)
            messagebox.showinfo( title: " ", message: "옷 정보가 삭제되었습니다.") #삭제되었다는 메시지 출력
            custom_clothing_temperature_window.destroy() #창 지우기
```

```
793
794     set_custom_clothing_temperature_button = tk.Button(custom_clothing_temperature_window, text="저장",
795                                                        command=set_custom_clothing_temperature, font=(20), bg='#D1B2FF') #저장 버튼
796     set_custom_clothing_temperature_button.pack()
797     delete_custom_clothing_temperature_button = tk.Button(custom_clothing_temperature_window, text = "삭제", command = delete_custom_clothing_temperature, font = (20), bg = '#D1B2FF') #삭제 버튼
798     delete_custom_clothing_temperature_button.pack()
799     custom_clothing_temperature_window.mainloop()
800
```

#옷장 정보 출력 버튼

```
1 usage
def center_window(window, width, height):
    screen_width = window.winfo_screenwidth()
    screen_height = window.winfo_screenheight()

    x = (screen_width - width) // 2
    y = (screen_height - height) // 2

    window.geometry(f"{width}x{height}+{x}+{y}")

root = tk.Tk()  # 메인 윈도우 창
root.geometry("800x600")  # 창 크기를 크게 조절

cloth_enter_btn = tk.Button(root, text="옷 정보 입력 및 삭제", command=cloth_enter,
                            font="Helvetica 20 bold italic", fg="#8A2BE2")  # 옷 정보 입력 및 삭제 버튼
cloth_enter_btn.place(relx=0.5, rely=0.2, anchor="center")  # 가로 중앙 정렬

cloth_info_btn = tk.Button(root, text="옷장 정보", command=closet_info, font="Helvetica 20 bold italic", fg="#8A2BE2")  # 옷장 정보 출력 버튼
cloth_info_btn.place(relx=0.5, rely=0.5, anchor="center")  # 가로 중앙 정렬

cloth_recommend_btn = tk.Button(root, text="코디 추천", command=recommend_coordi,
                                font="Helvetica 20 bold italic", fg="#8A2BE2")  # 코디 추천 버튼
cloth_recommend_btn.place(relx=0.5, rely=0.8, anchor="center")  # 가로 중앙 정렬

# 창을 화면 중앙에 배치
center_window(root, width: 800, height: 600)

root.mainloop()
```

#옷장 속 옷들을 출력하는 버튼

```
399     # 옷장 정보 출력 버튼함수
        1 usage
400     def closet_info():
401         cloth_info = tk.Toplevel(root)  # 새로운 윈도우 창 띄우기
402         cloth_info.title("옷장 정보")  # 창 이름
403         cloth_info.geometry("200x310")  # 창 크기
404
405         tk.Button(cloth_info, text="상의", command=cloth_info_print_Top, font=(20), fg="purple").pack(pady=10, ipadx=7, ipady=10)  # 상의 옷장 출력 버튼
406         tk.Button(cloth_info, text="하의", command=cloth_info_print_Low, font=(20), fg="purple").pack(pady=10, ipadx=7, ipady=10)  # 하의 옷장 출력 버튼
407         tk.Button(cloth_info, text="겉옷", command=cloth_info_print_Outer, font=(20), fg="purple").pack(pady=10, ipadx=7, ipady=10)  # 겉옷 옷장 출력 버튼
408         tk.Button(cloth_info, text="액세서리", command=cloth_info_print_Axcessory, font=(20), fg="purple").pack(pady=10, ipadx=7, ipady=10)  # 악세서리 옷장 출력 버튼
409
410         cloth_info.mainloop()
```

#옷장 속 옷들을 출력하는 함수

```
380     # 옷장 속 악세서리 출력 함수
        1 usage
381     def cloth_info_print_Axcessory():
382         cloth_info_print__Axcessory = tk.Toplevel(root)  # 새로운 윈도우 창 띄우기
383         cloth_info_print__Axcessory.title("악세서리") #창 이름
384         T = tk.Text(cloth_info_print__Axcessory, height=(3 + len(closet_Axcessory)), width=60)  # 텍스트 위젯 생성
385
386         if (closet_Axcessory == [] and custom_closet_Axcessory == []):  # 옷장에 악세서리가 1개도 없을 경우
387             T.insert(tk.END, chars: "옷장에 악세서리가 없습니다.")
388         else:  # 옷장에 악세서리가 있을 경우 → 리스트의 길이만큼 옷장 속 악세서리 딕셔너리들 출력
389             for i in range(len(closet_Axcessory)):
390                 T.insert(tk.END, '%s %s\n' % (
391                 closet_Axcessory[i]["design"], closet_Axcessory[i]["category"]))
392                 for i in range(len(custom_closet_Top)):
393                     T.insert(tk.END, '%s %s\n' % (
394                     custom_closet_Axcessory[i]["design"], custom_closet_Axcessory[i]["category"]))
395
396         T.pack()
397         cloth_info_print__Axcessory.mainloop()
```

#코디 추천 버튼

```
1 usage
def center_window(window, width, height):
    screen_width = window.winfo_screenwidth()
    screen_height = window.winfo_screenheight()

    x = (screen_width - width) // 2
    y = (screen_height - height) // 2

    window.geometry(f"{width}x{height}+{x}+{y}")

root = tk.Tk()  # 메인 윈도우 창
root.geometry("800x600")  # 창 크기를 크게 조절

cloth_enter_btn = tk.Button(root, text="옷 정보 입력 및 삭제", command=cloth_enter,
                            font="Helvetica 20 bold italic", fg="#8A2BE2")  # 옷 정보 입력 및 삭제 버튼
cloth_enter_btn.place(relx=0.5, rely=0.2, anchor="center")  # 가로 중앙 정렬

cloth_info_btn = tk.Button(root, text="옷장 정보", command=closet_info, font="Helvetica 20 bold italic", fg="#8A2BE2")  # 옷장 정보 출력 버튼
cloth_info_btn.place(relx=0.5, rely=0.5, anchor="center")  # 가로 중앙 정렬

cloth_recommend_btn = tk.Button(root, text="코디 추천", command=recommend_coordi,
                                font="Helvetica 20 bold italic", fg="#8A2BE2")  # 코디 추천 버튼
cloth_recommend_btn.place(relx=0.5, rely=0.8, anchor="center")  # 가로 중앙 정렬

# 창을 화면 중앙에 배치
center_window(root, width: 800, height: 600)

root.mainloop()
```

#추천한 코디를 출력하는 함수(1)

```
151  def recommend_coordi():
152      global year  # 년도 전역변수
153      global day  # 일 전역변수
154      global day_of_the_week  # 요일 전역변수
155      global Month  # 달 전역변수
156
157      recommend__coordi = tk.Toplevel(root)  # 새로운 윈도우 창 띄우기
158      recommend__coordi.title("코디 추천")  # 창 이름
159      T = tk.Text(recommend__coordi, height=(10 + len(closet_Top) + len(closet_Low) + len(closet_Outer) + len(closet_Axcessory)+ len(custom_closet_Top)+len(custom_closet_Low)+
160                                             +len(custom_closet_Low)+len(custom_closet_Outer)+len(custom_closet_Axcessory)),width=60)  # 텍스트 위젯 생성
161      GetWeather()  # 기온 함수 호출
162      recommended_top = get_recommendation_top()
163      recommended_bot = get_recommendation_bot()
164      recommended_out = get_recommendation_out()
165      recommended_acc = get_recommendation_acc()
166      custom_recommended_Top = get_recommendation_custom_Top() #카테고리에 없는 상의 추천 리스트
167      custom_recommended_Low = get_recommendation_custom_Low() #카테고리에 없는 하의 추천 리스트
168      custom_recommended_Outer = get_recommendation_custom_Outer() #카테고리에 없는 겉옷 추천 리스트
169      custom_recommended_Axcessory = get_recommendation_custom_Axcessory() #카테고리에 없는 액세서리 추천 리스트
170
171      calendar()  # 날짜 함수 호출
172      date = "[ " + year + "년" + Month + day + "일" + day_of_the_week + " ]"
173      T.insert(tk.END, date)  # 날짜 출력
174      T.insert(tk.END, chars: "\n")
175      temp = "[ 기온 : " + str(Weather) + " ]"
176      T.insert(tk.END, temp)  # 기온 출력
177      T.insert(tk.END, chars: "\n")
178
```

#날씨 정보를 가져오는 함수

```
14      Weather = None
15
16      # 네이버에서 서울의 기온을 크롤링
        1 usage
17      def GetWeather():
18          # https에 필요한 ssl 인증서 확인을 무시하기 위해 SSL 컨텍스트를 생성해 https문서를 열람할 수 있게 한다.
19          context = ssl._create_unverified_context()
20          # 네이버의 서울 날씨 웹페이지에 접속한다.
21          webpage = urllib.request.urlopen(
22              url: 'https://search.naver.com/search.naver?sm=top_hty&fbm=0&ie=utf8&query=%EC%84%9C%EC%9A%B8%EB%82%A0%EC%94%A8',
23              context=context)
24          # 지정한 url을 파이썬이 대신 열어서 해당 html파일을 복사(파싱)해온다.
25          soup = BeautifulSoup(webpage, features: 'html.parser')
26          # temperature_text클래스이면서 div태그 안에 담겨있는 현재기온 값(temps) 받아오기
27          temps = soup.find( name: 'div', attrs: 'temperature_text')
28          # 찾은 요소에서 텍스트를 추출하고, 공백을 제거한다.
29          temps = str(temps.text.strip())
30          # 현재 기온
31          global Weather
32          # \d+.\d+:숫자들과 숫자들 사이에 어떤 문자가 와 있는 경우를 추출한다.
33          # re.findall: String에서 패턴에 해당하는 내용을 찾아서 리스트로 리턴한다.
34          # [0]으로 리스트의 첫 번째 요소, 즉 숫자 패턴의 첫 번째 부분 선택, float로 선택된 숫자를 부동소수점으로 변환.
35          # 즉, 문자열 temps에서 기온(실수)만 가져온다.
36          Weather = float(re.findall( pattern: '\d+.\d+', temps)[0])
37
```

#날짜를 저장하는 함수

```
279  def calendar():
280      global year  # 년도 전역변수
281      global day  # 일 전역변수
282      global day_of_the_week  # 요일 전역변수
283      global Month  # 월 전역변수
284
285      t1 = time.asctime()  # 현재 날짜와 시간을 문자열 형태로 가져오기
286      t2 = t1.split()  # 문자열 나누기
287      year = t2[4]  # 년도
288      day = t2[2]  # 날짜
289
290      if (t2[0] == 'Mon'): day_of_the_week = " 월요일"
291      if (t2[0] == 'Tue'): day_of_the_week = " 화요일"
292      if (t2[0] == 'Wed'): day_of_the_week = " 수요일"
293      if (t2[0] == 'Thu'): day_of_the_week = " 목요일"
294      if (t2[0] == 'Fri'): day_of_the_week = " 금요일"
295      if (t2[0] == 'Sat'): day_of_the_week = " 토요일"
296      if (t2[0] == 'Sun'): day_of_the_week = " 일요일"  # 요일 변환
297
298      if (t2[1] == 'Jan'): Month = " 1월 "
299      if (t2[1] == 'Feb'): Month = " 2월 "
300      if (t2[1] == 'Mar'): Month = " 3월 "
301      if (t2[1] == 'Apr'): Month = " 4월 "
302      if (t2[1] == 'May'): Month = " 5월 "
303      if (t2[1] == 'Jun'): Month = " 6월 "
304      if (t2[1] == 'Jul'): Month = " 7월 "
305      if (t2[1] == 'Aug'): Month = " 8월 "
306      if (t2[1] == 'Sep'): Month = " 9월 "
307      if (t2[1] == 'Oct'): Month = " 10월 "
308      if (t2[1] == 'Nov'): Month = " 11월 "
309      if (t2[1] == 'Dec'): Month = " 12월 "  # 월 변환
```

#카테고리에 있는 옷 추천하는 함수

```
133  def get_recommendation_acc():
134      global Weather
135      recommended_acc = []  # 액세서리 추천 함수의 리턴 값을 담을 리스트
136      clothing_type = []  # 현재 기온에 맞는 액세서리 타입(카테고리)을 저장할 리스트
137
138      # 기온에 맞는 액세서리 분류
139      if Weather <= 4:
140          clothing_type = ["목도리", "장갑"]
141
142      # 기온에 맞는 액세서리 보유 여부 확인 및 리턴 값(리스트)에 추가
143      for clothing in closet_Axcessory:
144          if clothing["category"] in clothing_type:  # 현재 기온에 맞는 액세서리 타입과 일치하는 상의를 보유하고 있다면
145              recommended_acc.append(clothing)  # 액세서리 추천 함수의 리턴 값(리스트)에 옷 추가
146
147      # 추천 액세서리(리스트) 리턴
148      return recommended_acc
     for clothing in closet_outer:
126      if clothing["category"] in clothing_type:  # 현재 기온에 맞는 겉옷 타입과 일치하는 겉옷을 보유하고 있다면
127          recommended_out.append(clothing)  # 겉옷 추천 함수의 리턴 값(리스트)에 옷 추가
128
129  # 추천 겉옷(리스트) 리턴
130  return recommended_out
104  return recommended_bot
```

면

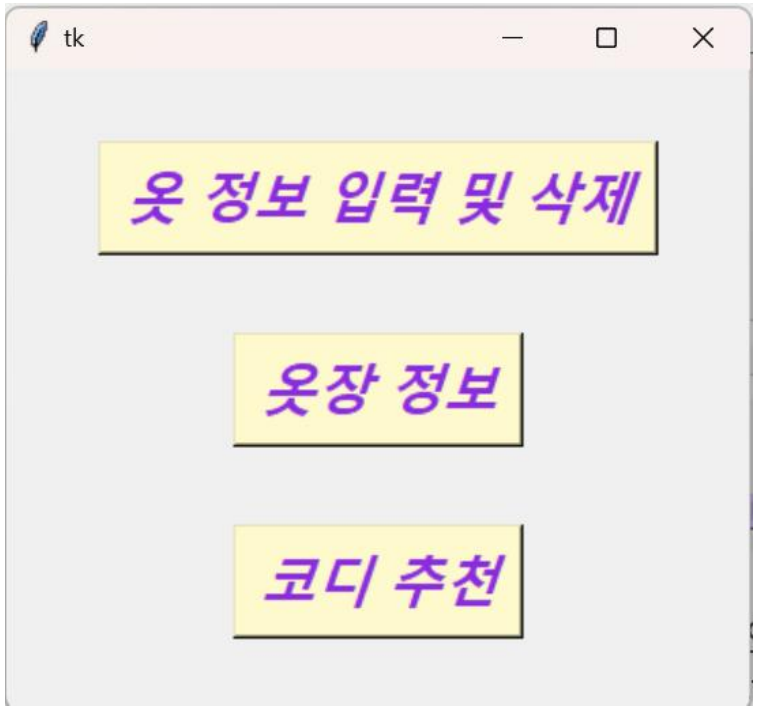
#카테고리에 없는 옷 추천하는 함수

```
#카테고리에 없는 액세서리 추천 함수
1 usage
def get_recommendation_custom_Axcessory():
    global Weather #기온 전역변수
    custom_recommended_Axcessory = [] #기온 범위에 들어가는 카테고리에 없는 액세서리들을 저장할 리스트

    for i in range(len(custom_closet_Axcessory)): #반복문으로 카테고리에 없는 액세서리들을 저장한 리스트의
        if (Weather - 2) <= float(custom_closet_Axcessory[i]["temperature"]) <= (Weather + 2): #te
            custom_recommended_Axcessory.append(custom_closet_Axcessory[i]) #그 요소를 카테고리에 없는

    return custom_recommended_Axcessory #카테고리에 없는 액세서리 추천 리스트를 리턴
```

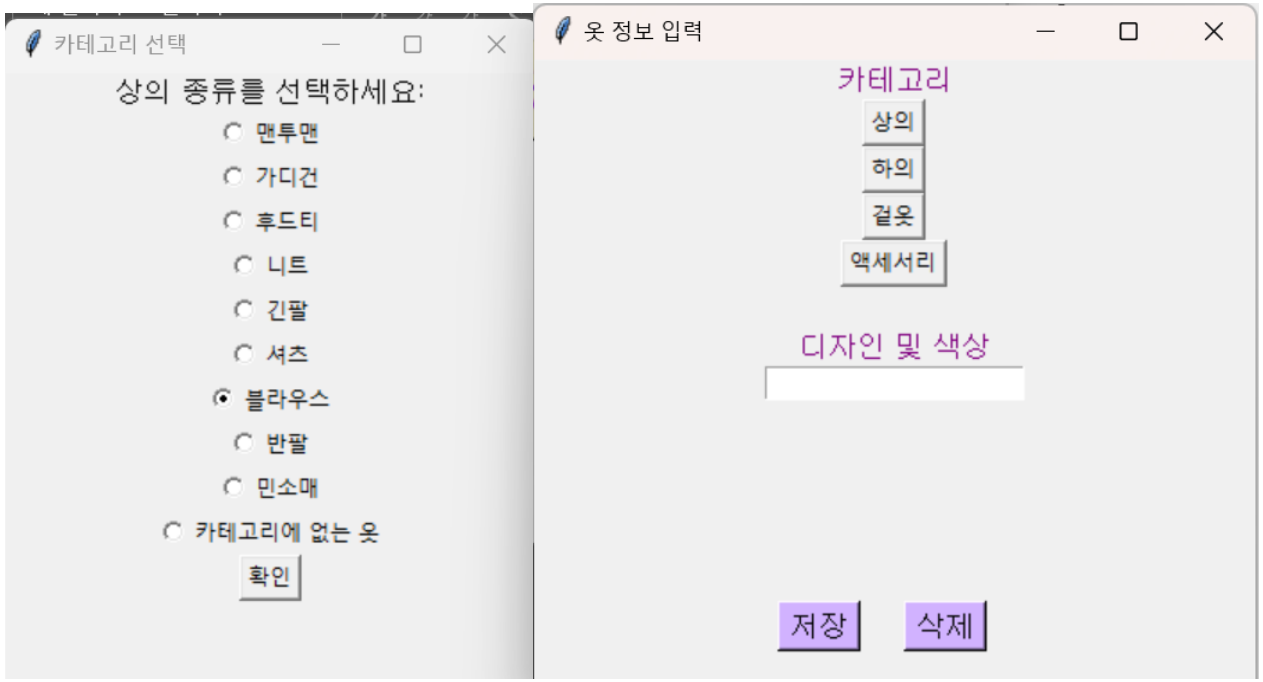
#추천한 코디를 출력하는 함수(2)

```
218     # 액세서리 추천
219     if recommended_acc or (custom_recommended_Axcessory != []):
220         T.insert(tk.END, chars: "추천 액세서리:\n")
221         i = 1
222         for clothing in recommended_acc:
223             T.insert(tk.END, chars: f"{i}. {clothing['design']} {clothing['category']}")
224             T.insert(tk.END, chars: "\n")
225             i += 1
226         for n in range(len(custom_recommended_Axcessory)) : #반복문으로 카테고리에 없는 액세서리 추천 리스트의 요소들 출력
227             T.insert(tk.END, '현재 기온에는 %s %s도 적합합니다.\n' % (custom_recommended_Axcessory[n]["design"],custom_recommended_Axcessory[n]["category"]))
228     else:
229         T.insert(tk.END, chars: "현재 계절과 기온에 맞는 액세서리가 없습니다.\n")
230
231     T.pack()
232     recommend__coordi.mainloop()
```

실행방법

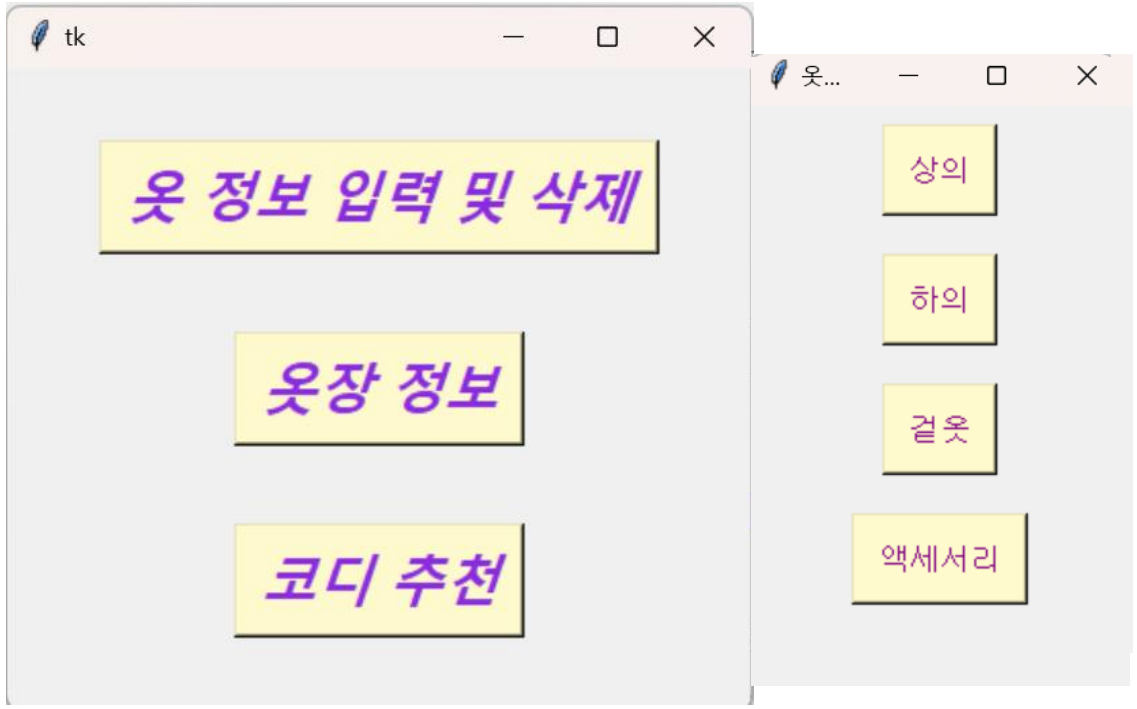


실행 시 첫 화면

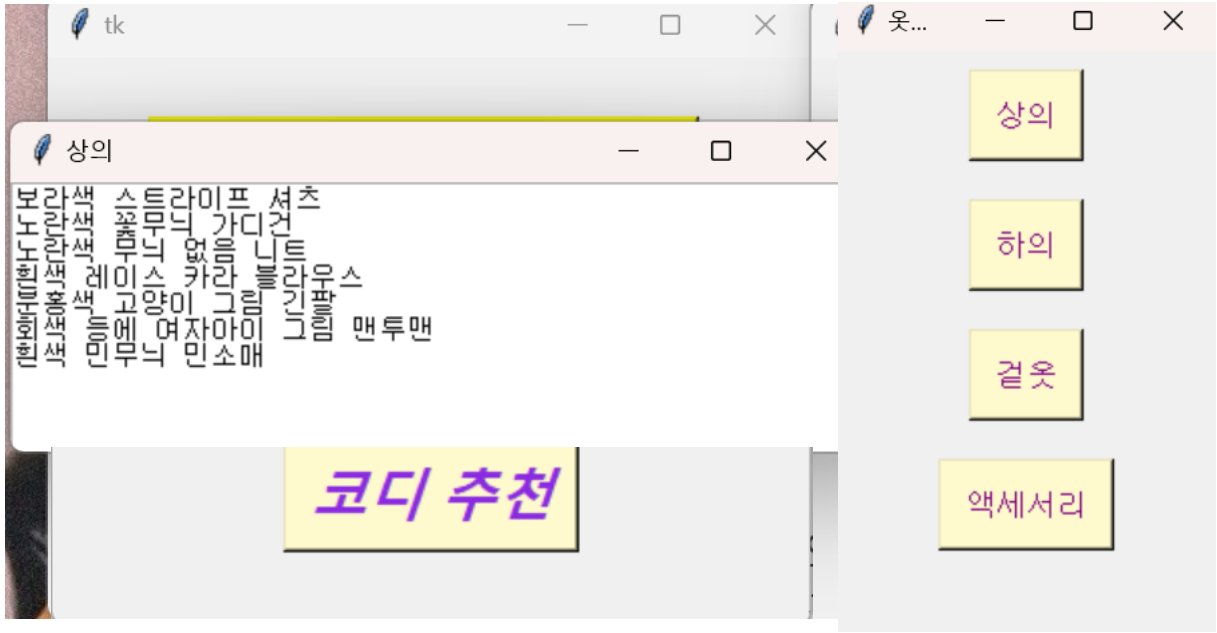


옷 정보 입력 및 삭제 : 상의, 하의, 겉옷, 액세서리로 나눠서 정보를 입력하고 삭제할 수 있다.

실행방법



옷장 정보 – 입력한 옷 정보 확인



상의 옷장 정보 확인

결과 화면

옷 정보를 입력한 뒤
코디 추천 버튼을 누르면
현재 날짜, 기온과 함께
현재 기온에 따른 추천 상의,
하의, 겉옷, 액세서리가
출력된다.