

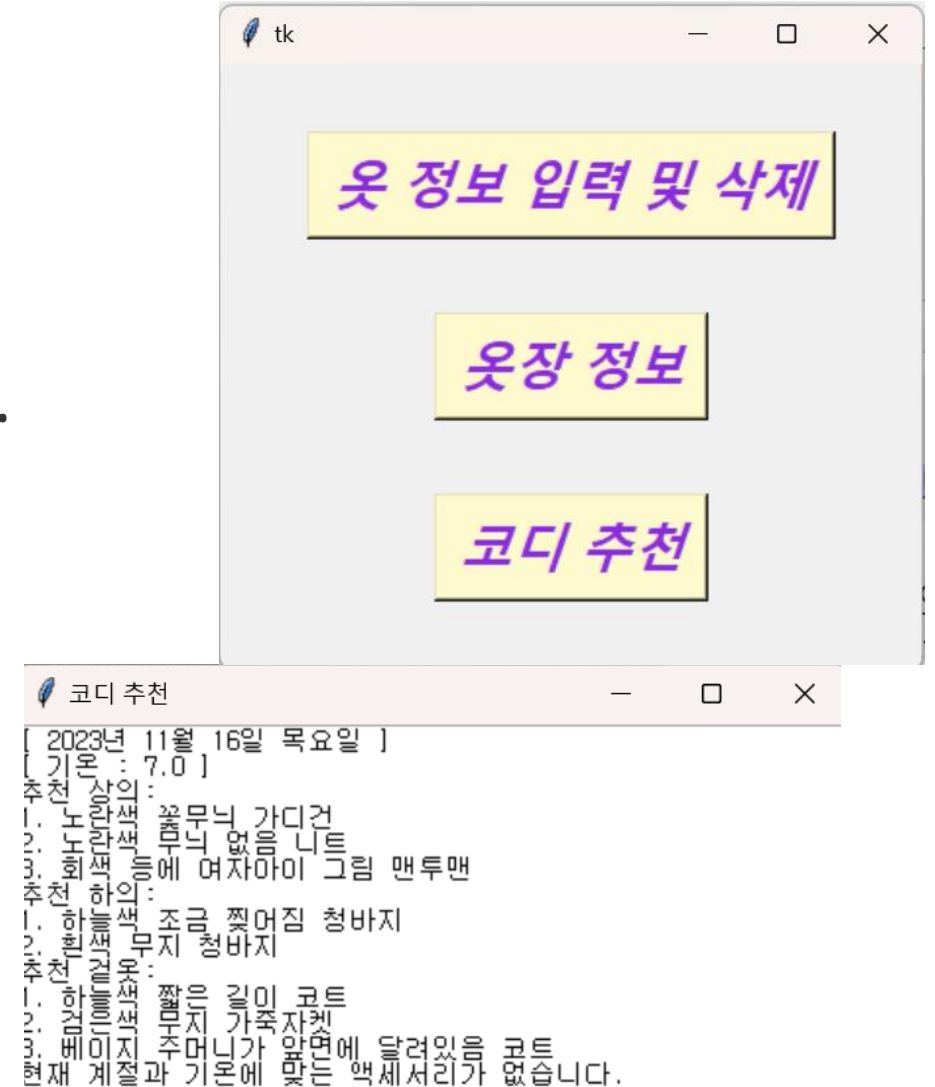
# 대학기초SW입문 프로젝트 발표 6조

- 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\_hyt&fbm=0&ie=utf8&query=%EC%84%9C%EC%9A%B8%EB%82%A0%EC%  
context=context)

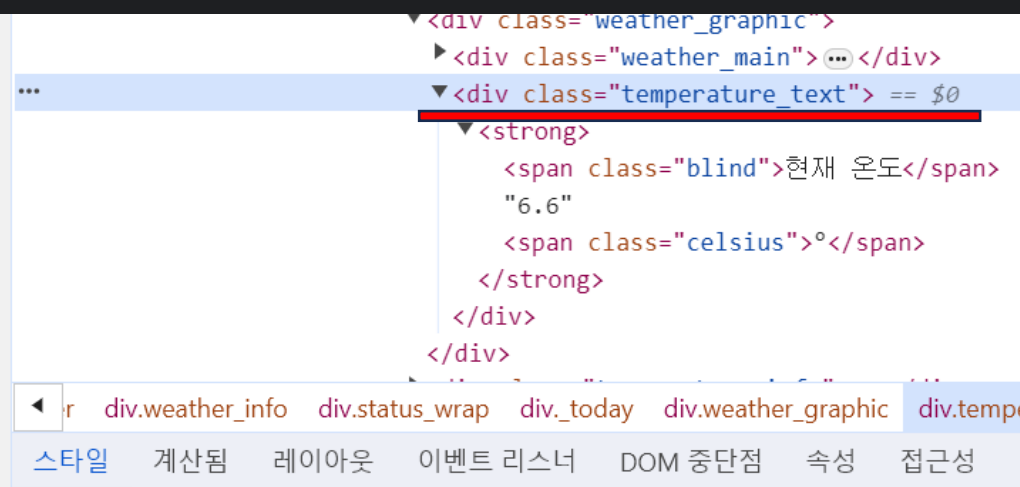
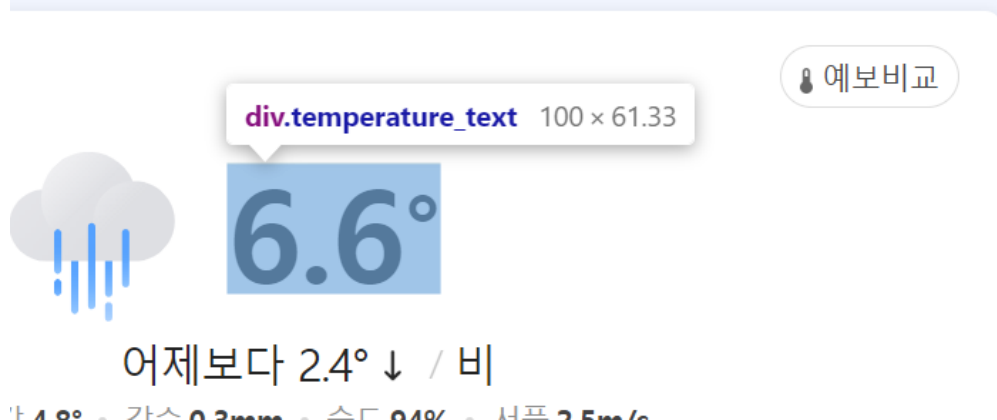
# 지정한 url을 파이썬이 대신 열어서 해당 html파일을 복사(파싱)해온다.

Soup = BeautifulSoup(webpage, 'html.parser')

# temperature\_text클래스이면서 div태그 안에 담겨있는 현재기온 값(temps) 받아오기

temps = soup.find('div', 'temperature\_text')

간



# 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])
```

# 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:
```

# 코드설명 - 김시우

상의 하의 추천 코드의 초안

# 코드설명 - 김시우

```
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.")
```

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

## 코드설명 - 박기쁨

```
1  # 상의 추천 함수
2  def get_recommendation_top(weather, closet):
3      recommended_top = []      # 상의 추천 함수의 리턴 값을 담을 리스트
4      clothing_type = []        # 현재 기온에 맞는 상의 타입(카테고리)을 저장할 리스트
5
6      # 기온에 맞는 상의 분류
7      if weather <= 16:
8          clothing_type = ["맨투맨", "가디건", "후드티", "니트"]
9      elif 16 < weather <= 19:
10         clothing_type = ["맨투맨", "가디건"]
11     elif 19 < weather <= 22:
12         clothing_type = ["긴팔", "셔츠", "블라우스"]
13     elif 22 < weather <= 27:
14         clothing_type = ["반팔"]
15     elif 28 < weather:
16         clothing_type = ["민소매", "반팔"]
17
18     # 기온에 맞는 상의 보유 여부 확인 및 리턴 값(리스트)에 추가
19     for clothing in closet:
20         if clothing["category"] in clothing_type:      # 현재 기온에 맞는 상의 타입과 일치하는 상의를 보유하고 있다면
21             recommended_top.append(clothing)          # 상의 추천 함수의 리턴 값(리스트)에 옷 추가
22
23     # 추천 상의(리스트) 리턴
24     return recommended_top
```



```
27 # 하의 추천 함수
28 def get_recommendation_bot(weather, closet):
29     recommended_bot = []      # 하의 추천 함수의 리턴 값을 담을 리스트
30     clothing_type = []        # 현재 기온에 맞는 하의 타입(카테고리)을 저장할 리스트
31
32     # 기온에 맞는 하의 분류
33     if weather <= 4:
34         clothing_type = ["기모 바지"]
35     elif 4 < weather <= 8:
36         clothing_type = ["청바지"]
37     elif 8 < weather <= 22:
38         clothing_type = ["면바지", "슬랙스", "청바지", "치마"]
39     elif 22 < weather <= 27:
40         clothing_type = ["반바지", "면바지", "치마"]
41     elif 27 < weather:
42         clothing_type = ["반바지", "치마"]
43
44     # 기온에 맞는 하의 보유 여부 확인 및 리턴 값(리스트)에 추가
45     for clothing in closet:
46         if clothing["category"] in clothing_type:      # 현재 기온에 맞는 상의 타입과 일치하는 상의를 보유하고 있다면
47             recommended_bot.append(clothing)          # 하의 추천 함수의 리턴 값(리스트)에 옷 추가
48
49     # 추천 하의(리스트) 리턴
50     return recommended_bot
```

```

53 # 겉옷 추천 함수
54 def get_recommendation_out(weather, closet):
55     recommended_out = []      # 겉옷 추천 함수의 리턴 값을 담을 리스트
56     clothing_type = []        # 현재 기온에 맞는 겉옷 타입(카테고리)을 저장할 리스트
57
58     # 기온에 맞는 겉옷 분류
59     if weather <= 4:
60         clothing_type = ["패딩"]
61     elif 4 < weather <= 19:
62         clothing_type = ["가죽 자켓", "코트"]
63     elif 8 < weather <= 11:
64         clothing_type = ["자켓", "코트", "점퍼"]
65     elif 11 < weather <= 16:
66         clothing_type = ["자켓", "청자켓"]
67     elif 16 < weather <= 19:
68         clothing_type = ["바람막이"]
69
70     # 기온에 맞는 겉옷 보유 여부 확인 및 리턴 값(리스트)에 추가
71     for clothing in closet:
72         if clothing["category"] in clothing_type:      # 현재 기온에 맞는 겉옷 타입과 일치하는 겉옷을 보유하고 있다면
73             recommended_out.append(clothing)          # 겉옷 추천 함수의 리턴 값(리스트)에 옷 추가
74
75     # 추천 겉옷(리스트) 리턴
76     return recommended_out

```

```
79 # 액세서리 추천 함수
80 def get_recommendation_acc(weather, closet):
81     recommended_acc = [] # 액세서리 추천 함수의 리턴 값을 담을 리스트
82     clothing_type = [] # 현재 기온에 맞는 액세서리 타입(카테고리)을 저장할 리스트
83
84     # 기온에 맞는 액세서리 분류
85     if weather <= 4:
86         clothing_type = ["목도리", "장갑"]
87
88     # 기온에 맞는 액세서리 보유 여부 확인 및 리턴 값(리스트)에 추가
89     for clothing in closet:
90         if clothing["category"] in clothing_type: # 현재 기온에 맞는 액세서리 타입과 일치하는 상의를 보유하고 있다면
91             recommended_acc.append(clothing) # 액세서리 추천 함수의 리턴 값(리스트)에 옷 추가
92
93     # 추천 액세서리(리스트) 리턴
94     return recommended_acc
```

```
121 # 상의 추천
122 if recommended_top:
123     print("추천 상의:")
124     i = 1
125     for clothing in recommended_top:
126         print(f"{i}. {clothing['color']} {clothing['design']} {clothing['category']}")
127         i += 1
128 else:
129     print("현재 계절과 기온에 맞는 상의가 없습니다.")
130
131
132 # 하의 추천
133 if recommended_bot:
134     print("추천 하의:")
135     i = 1
136     for clothing in recommended_bot:
137         print(f"{i}. {clothing['color']} {clothing['design']} {clothing['category']}")
138         i += 1
139 else:
140     print("현재 계절과 기온에 맞는 하의가 없습니다.")
```

```
143 # 겉옷 추천
144 if recommended_out:
145     print("추천 겉옷:")
146     i = 1
147     for clothing in recommended_out:
148         print(f"{i}. {clothing['color']} {clothing['design']} {clothing['category']}")
149         i += 1
150 else:
151     print("현재 계절과 기온에 맞는 겉옷이 없습니다.")
152
153
154 # 액세서리 추천
155 if recommended_acc:
156     print("추천 액세서리:")
157     i = 1
158     for clothing in recommended_acc:
159         print(f"{i}. {clothing['color']} {clothing['design']} {clothing['category']}")
160         i += 1
161 else:
162     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

# 코드설명 - 고건우

- 옷을 입력받고 저장하는 함수

```
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()
```

수정한 코드

# 수정한 코드설명 - 김시우

```
# 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()

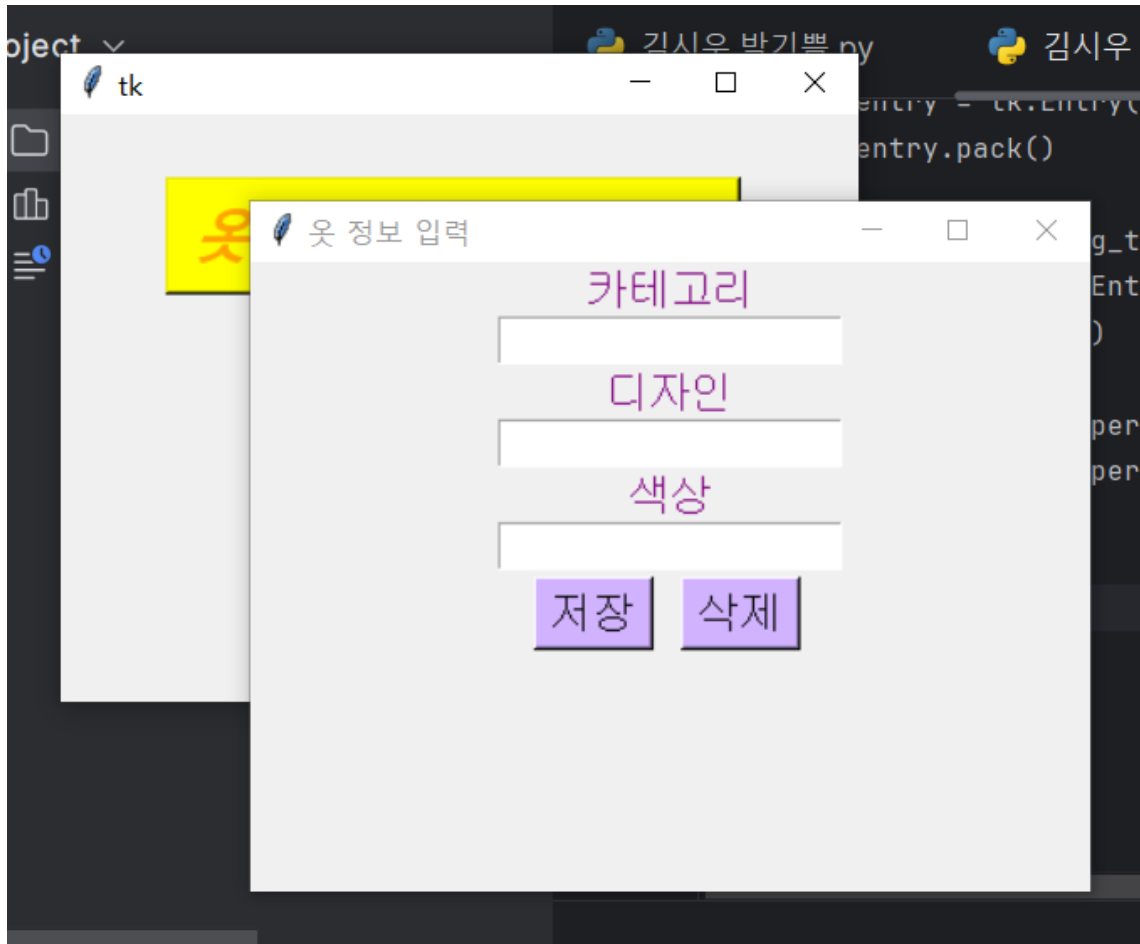
tk.Label(custom_clothing_temperature_window, text="온도", font=(20), fg='purple').pack()
temperature_entry = tk.Entry(custom_clothing_temperature_window)
temperature_entry.pack()

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



반팔? 반팔티?

```
# 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
```

```

1 usage
367 def cloth_enter(): # 옷 정보 입력 및 삭제함수
368     cloth__enter = tk.Toplevel(root) # 새로운 윈도우 창 띄우기
369     cloth__enter.title("옷 정보 입력") # 창 이름
370     cloth__enter.geometry("400x300") # 창 크기
371
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()

```

```

348 # 옷 정보 입력 및 삭제함수
349 S = None # 기존의 category_entry 역할 변수
1 usage
350 def cloth_enter():
351     global S
352     cloth__enter = tk.Toplevel(root) # 새로운 윈도우 창 띄우기
353     cloth__enter.title("옷 정보 입력") # 창 이름
354     cloth__enter.geometry("400x350") # 창 크기
355
356     tk.Label(cloth__enter, text="카테고리", font=(20), fg='purple').pack() # 카테고리 입력받기

```

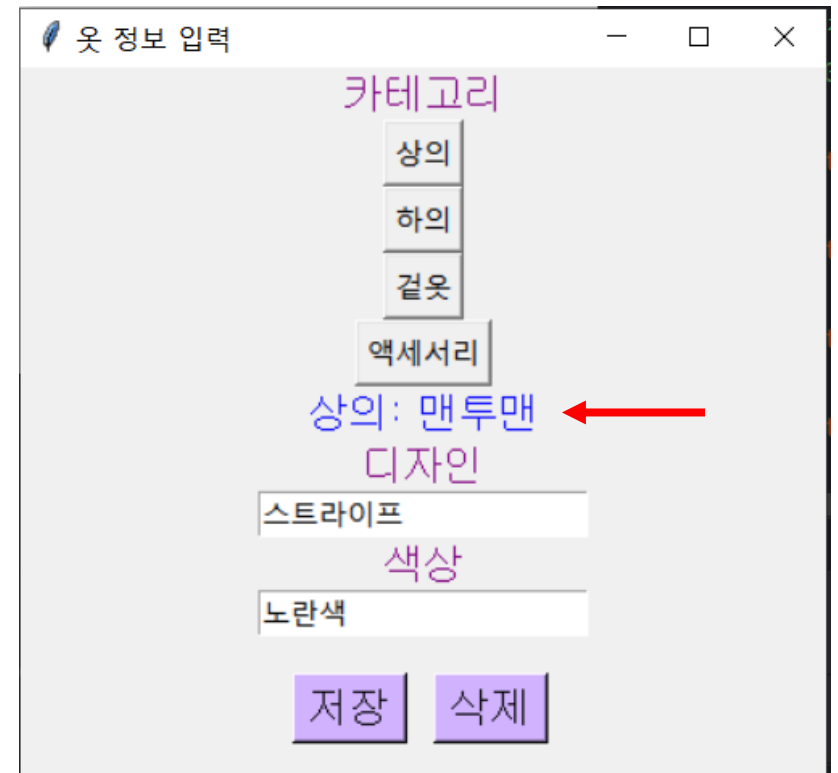
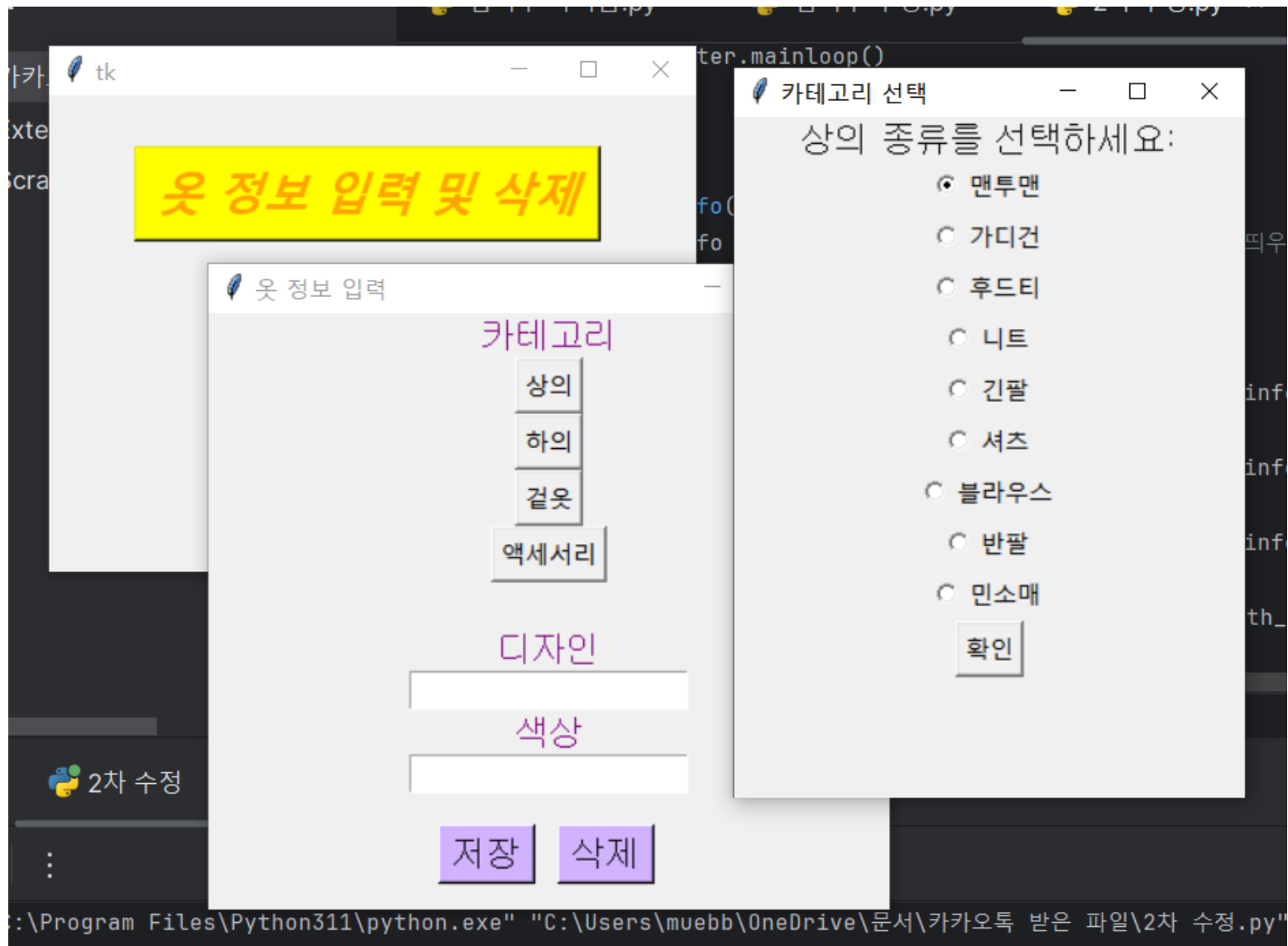
```
372 tk.Label(cloth__enter, text="카테고리", font=(20), fg='purple').pack() # 카테고리 입력받기
373 category_entry = tk.Entry(cloth__enter) ←
374 category_entry.pack()
```

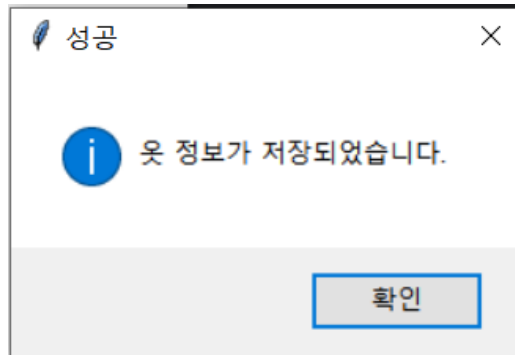
```
358 ##### 버튼과 라디오버튼 코드
359
360 # 상의 버튼
361 button1 = tk.Button(cloth__enter, text="상의", command=lambda: show_radio_buttons_top("상의"))
362 button1.pack()
363 # 하의 버튼
364 button2 = tk.Button(cloth__enter, text="하의", command=lambda: show_radio_buttons_bot("하의"))
365 button2.pack()
366 # 겹옷 버튼
367 button3 = tk.Button(cloth__enter, text="겹옷", command=lambda: show_radio_buttons_out("겹옷"))
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()
```

```
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()
```

## # 상의 라디오버튼 코드 2

```
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()
```





## 수정한 코드 설명 -고건우

```
def center_window(window, width, height):  
    screen_width = window.wininfo_screenwidth()  
    screen_height = window.wininfo_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

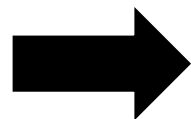
겉옷

액세서리

디자인

색상

저장 삭제



옷 정보 입력

카테고리

상의

하의

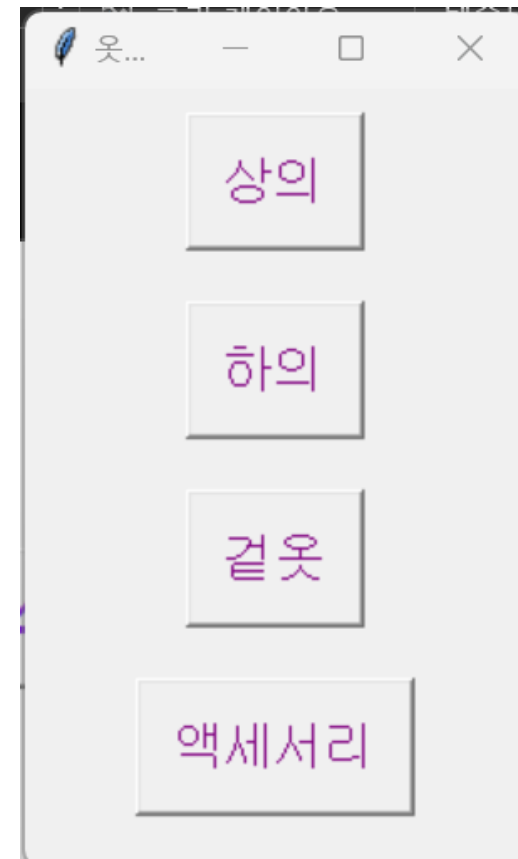
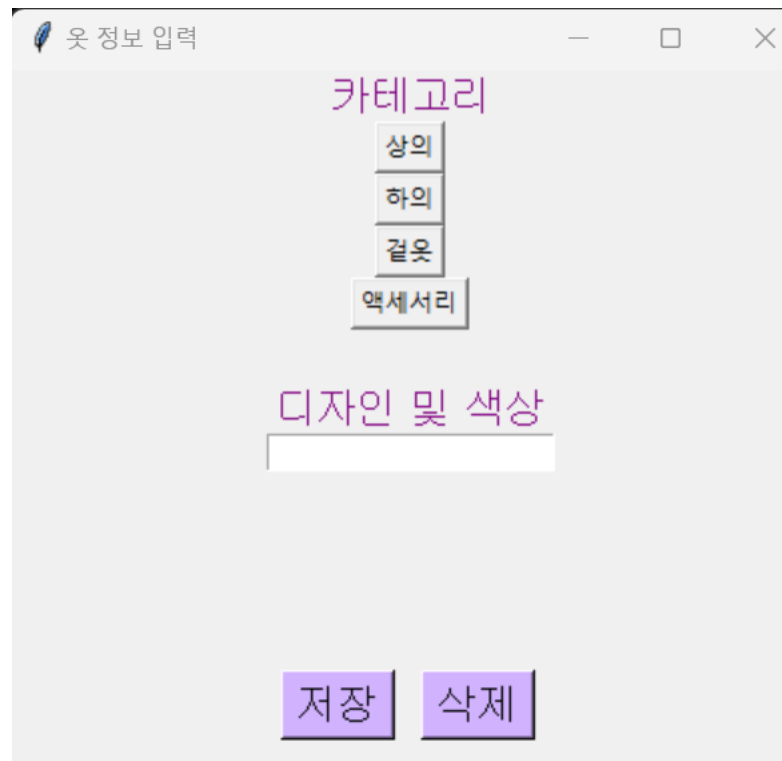
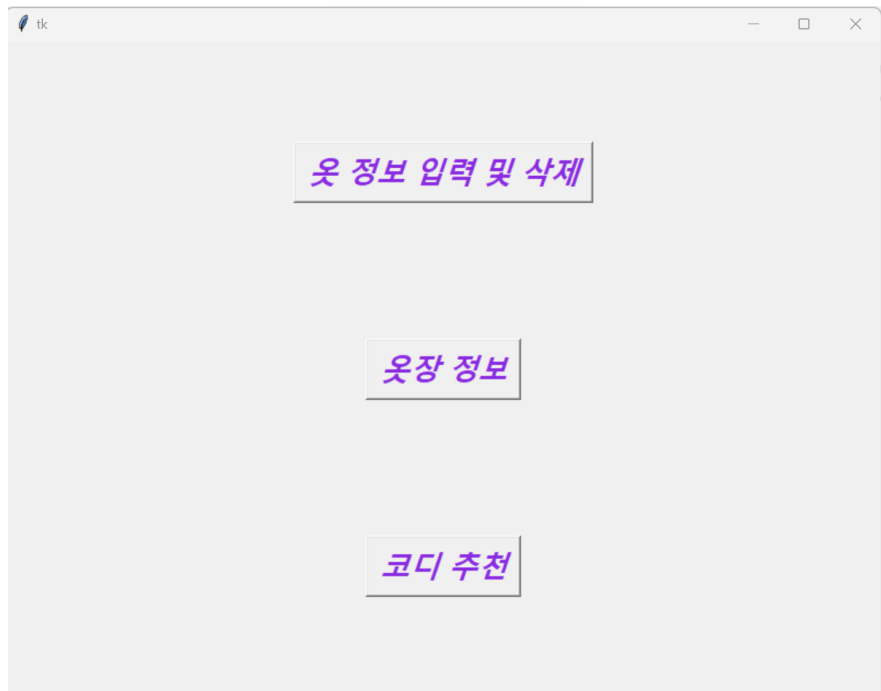
겉옷

액세서리

디자인 및 색상

저장 삭제

디자인과 색상 통합



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_Accessory = [] # 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_Accessory = [] # 카테고리에 없는 악세서리 옷장 리스트
47 custom_return_closet = {}
48
49 year = None # 년, 일, 요일, 월 전역변수
50 day = None
51 day_of_the_week = None
52 Month = None
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674

```

# #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     # 겹옷 버튼
437     button3 = tk.Button(cloth__enter, text="겹옷", command=lambda: show_radio_buttons_out("겹옷"))
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()

```

```

656 tk.Label(cloth__enter, text="디자인 및 색상", font=(20), fg='purple').pack() # 디자인 및 색상 입력받기
657 design_entry = tk.Entry(cloth__enter)
658 design_entry.pack()
659
660 # 옷 정보를 옷장에 저장하는 함수
661 def save_info():
662     global return_closet # return_closet 전역변수(입력받은 옷 정보들을 저장해서 옷장에 분류해서 넣을 때 필요한 변수)
663     global S
664     return_closet = {} # return_closet 초기화
665     category = S # 카테고리 받아오기
666     design = design_entry.get() # 디자인 및 색상 받아오기
667
668     return_closet = {"design": design, "category": category} # return_closet 딕셔너리에 카테고리, 디자인 및 색상 저장
669
670     if not category or not design: # 카테고리, 디자인 및 색상 중 1개라도 입력되지 않았으면 오류 메시지 출력
671         messagebox.showerror( title: "오류", message: "카테고리, 디자인 및 색상은 모두 입력되어야 합니다.")
672     else: #카테고리, 디자인, 색상이 모두 입력되었을 경우 저장되었다는 메시지 출력
673         messagebox.showinfo( title: "성공", message: "옷 정보가 저장되었습니다.")
674         cloth__enter.destroy() #창 지우기
675         categorizing() # 옷을 카테고리에 따라 분류하는 함수 호출

```

## # 카테고리에 있는 옷을 옷 카테고리별로 분류하는 함수

```

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

```



```

677 # 옷 정보를 옷장에서 삭제하는 함수
678 def delete_info():
679     global return_closet # return_closet 전역변수(입력받은 옷 정보들을 저장해서 옷장 속 옷들과 비교할 때 필요한 변수)
680     global S
681     return_closet = {} # return_closet 초기화
682     category = S # 카테고리 받아오기
683     design = design_entry.get() # 디자인 및 색상 받아오기
684
685     return_closet = {"design": design, "category": category} # return_closet 딕셔너리에 카테고리, 디자인 및 색상 저장
686
687     if (return_closet not in closet_Top not in closet_Low not in closet_Outer not in closet_Accessory):
688         messagebox.showinfo( title: " ", message: "삭제할 옷이 존재하지 않습니다.") # 옷 정보와 일치하는 옷이 없을 때는 삭제할 옷이 없다는 메시지 출력
689     if return_closet in closet_Top: # 상의 옷장에 있으면 상의 옷장 속 동일한 요소 삭제
690         closet_Top.remove(return_closet)
691         messagebox.showinfo( title: " ", message: "옷 정보가 삭제되었습니다.") # 옷 정보가 삭제되었다는 메시지 출력
692         cloth__enter.destroy()
693     if return_closet in closet_Low: # 하의 옷장에 있으면 하의 옷장 속 동일한 요소 삭제
694         closet_Low.remove(return_closet)
695         messagebox.showinfo( title: " ", message: "옷 정보가 삭제되었습니다.") # 옷 정보가 삭제되었다는 메시지 출력
696         cloth__enter.destroy()
697     if return_closet in closet_Outer: # 겉옷 옷장에 있으면 겉옷 옷장 속 동일한 요소 삭제
698         closet_Outer.remove(return_closet)
699         messagebox.showinfo( title: " ", message: "옷 정보가 삭제되었습니다.") # 옷 정보가 삭제되었다는 메시지 출력
700         cloth__enter.destroy()
701     if return_closet in closet_Accessory: # 악세서리 옷장에 있으면 악세서리 옷장 속 동일한 요소 삭제
702         closet_Accessory.remove(return_closet)
703         messagebox.showinfo( title: " ", message: "옷 정보가 삭제되었습니다.") # 옷 정보가 삭제되었다는 메시지 출력
704         cloth__enter.destroy()

```

```

700 save_button = tk.Button(cloth__enter, text="저장", command=save_info, font=(20), bg='#D1B2FF') # 저장 버튼
701 save_button.place(x=135, y=300)
702 delete_button = tk.Button(cloth__enter, text="삭제", command=delete_info, font=(20), bg='#D1B2FF') # 삭제 버튼
703 delete_button.place(x=205, y=300)
704
705 cloth__enter.mainloop()

```

## #카테고리에 없는 옷 정보를 받아오는 함수 + 저장함수, 삭제 함수

```
707 # 카테고리에 없는 옷 정보를 받아오는 함수  
708 # 4 usages  
708 def custom_cloth():  
709     custom_clothing_temperature_window = tk.Toplevel(root) #새로운 윈도우 창 띄우기  
710     custom_clothing_temperature_window.title("옷 종류와 온도 설정") #창 이름  
711  
712     type_radio = tk.StringVar() #옷 종류를 받아올 변수 생성  
713     tk.Label(custom_clothing_temperature_window, text="옷 종류를 선택해 주세요").pack()  
714  
715     # 상의 라디오 버튼  
716     button1 = tk.Radiobutton(custom_clothing_temperature_window, text="상의", variable=type_radio, value="상의")  
717     button1.pack()  
718     # 하의 라디오 버튼  
719     button2 = tk.Radiobutton(custom_clothing_temperature_window, text="하의", variable=type_radio, value="하의")  
720     button2.pack()  
721     # 겹옷 라디오 버튼  
722     button3 = tk.Radiobutton(custom_clothing_temperature_window, text="겹옷", variable=type_radio, value="겹옷")  
723     button3.pack()  
724     # 액세서리 라디오 버튼  
725     button4 = tk.Radiobutton(custom_clothing_temperature_window, text="액세서리", variable=type_radio, value="액세서리")  
726     button4.pack()
```

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

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

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

```
747 # 목록에 없는 옷 종류 어느 온도에 입을 것인지 직접 설정할 수 있는 기능 추가
```

```
748 #카테고리에 없는 옷 정보를 저장하는 함수
```

```
749 def set_custom_clothing_temperature():
```

```
750     global custom_return_closet #custom_return_closet 전역변수(입력받은 옷 정보들을 저장해서 옷장에 분류해서 넣을 때 필요한 변수)
```

```
751     # 사용자가 입력한 옷 카테고리를 가져옴
```

```
752     category = custom_category.get()
```

```
753     # 사용자가 입력한 온도를 가져옴
```

```
754     temperature = temperature_entry.get()
```

```
755     type = type_radio.get() #옷 종류 받아오기
```

```
756     design = custom_design.get() #디자인 및 색상 받아오기
```

```
759 # 사용자가 입력한 옷 정보와 온도를 딕셔너리에 저장
```

```
760 custom_return_closet = {"type": type, "category": category, "design": design, "temperature": temperature}
```

```
762 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
```

```
763     messagebox.showerror( title: "오류", message: "종류, 카테고리, 디자인 및 색상, 온도를 모두 입력해주세요.")
```

```
764 else: #모든 정보가 입력되었을 경우 저장되었다는 메시지 출력
```

```
765     messagebox.showinfo( title: "성공", message: "옷 정보가 저장되었습니다.")
```

```
766     custom_clothing_temperature_window.destroy() # 창 지우기
```

```
767     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_Accessory.append(custom_return_closet)
```

```

769 #카테고리에 없는 옷 정보를 삭제하는 함수
770 def delete_custom_clothing_temperature() :
771     global custom_return_closet #custom_return_closet 전역변수(입력받은 옷 정보들을 저장해서 옷장 속 옷들과 비교할 때 필요한 변수)
772     category = custom_category.get() #카테고리 받아오기
773     temperature = temperature_entry.get() #온도 받아오기
774     type = type_radio.get() #옷 종류 받아오기
775     design = custom_design.get() #디자인 및 색상 받아오기
776
777     #사용자가 입력한 옷 정보와 온도를 딕셔너리에 저장
778     custom_return_closet = {"type": type, "category": category, "design": design, "temperature": temperature}
779
780     if(custom_return_closet not in custom_closet_Top not in custom_closet_Low not in custom_closet_Outer not in custom_closet_Accessory):
781         #옷 정보와 일치하는 옷이 없을 때는 삭제할 옷이 없다는 메시지 출력
782         messagebox.showinfo( title: " ", message: "삭제할 옷이 존재하지 않습니다.")
783     if return_closet in custom_closet_Top : #상의 옷장에 있으면 상의 옷장 속 동일한 요소 삭제
784         custom_closet_Top.remove(custom_return_closet)
785         messagebox.showinfo( title: " ", message: "옷 정보가 삭제되었습니다.") #삭제되었다는 메시지 출력
786         custom_clothing_temperature_window.destroy() #창 지우기
787     if return_closet in custom_closet_Low: #하의 옷장에 있으면 하의 옷장 속 동일한 요소 삭제
788         custom_closet_Low.remove(custom_return_closet)
789         messagebox.showinfo( title: " ", message: "옷 정보가 삭제되었습니다.") #삭제되었다는 메시지 출력
790         custom_clothing_temperature_window.destroy() #창 지우기
791     if return_closet in custom_closet_Outer: #겉옷 옷장에 있으면 겉옷 옷장 속 동일한 요소 삭제
792         custom_closet_Outer.remove(custom_return_closet)
793         messagebox.showinfo( title: " ", message: "옷 정보가 삭제되었습니다.") #삭제되었다는 메시지 출력
794         custom_clothing_temperature_window.destroy() #창 지우기
795     if return_closet in custom_closet_Accessory: #액세서리 옷장에 있으면 액세서리 옷장 속 동일한 요소 삭제
796         custom_closet_Accessory.remove(custom_return_closet)
797         messagebox.showinfo( title: " ", message: "옷 정보가 삭제되었습니다.") #삭제되었다는 메시지 출력
798         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
807 def center_window(window, width, height):
808     screen_width = window.winfo_screenwidth()
809     screen_height = window.winfo_screenheight()
810
811     x = (screen_width - width) // 2
812     y = (screen_height - height) // 2
813
814     window.geometry(f"{width}x{height}+{x}+{y}")
815
816     root = tk.Tk() # 메인 윈도우 창
817     root.geometry("800x600") # 창 크기를 크게 조절
818
819     cloth_enter_btn = tk.Button(root, text="옷 정보 입력 및 삭제", command=cloth_enter,
820                                font="Helvetica 20 bold italic", fg="#8A2BE2") # 옷 정보 입력 및 삭제 버튼
821     cloth_enter_btn.place(relx=0.5, rely=0.2, anchor="center") # 가로 중앙 정렬
822
823     cloth_info_btn = tk.Button(root, text="옷장 정보", command=closet_info, font="Helvetica 20 bold italic", fg="#8A2BE2") # 옷장 정보 출력 버튼
824     cloth_info_btn.place(relx=0.5, rely=0.5, anchor="center") # 가로 중앙 정렬
825
826     cloth_recommend_btn = tk.Button(root, text="코디 추천", command=recommend_coordi,
827                                     font="Helvetica 20 bold italic", fg="#8A2BE2") # 코디 추천 버튼
828     cloth_recommend_btn.place(relx=0.5, rely=0.8, anchor="center") # 가로 중앙 정렬
829
830     # 창을 화면 중앙에 배치
831     center_window(root, width=800, height=600)
832
833     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_Accessory, font=(20), fg="purple").pack(pady=10, ipadx=7, ipady=10) # 액세서리 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_Accessory():
382     cloth_info_print__Accessory = tk.Toplevel(root) # 새로운 윈도우 창 띄우기
383     cloth_info_print__Accessory.title("악세서리") #창 이름
384     T = tk.Text(cloth_info_print__Accessory, height=(3 + len(closet_Accessory)), width=60) # 텍스트 위젯 생성
385
386     if (closet_Accessory == [] and custom_closet_Accessory == []): # 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_Accessory)):
390             T.insert(tk.END, '%s %s\n' % (
391                 closet_Accessory[i]["design"], closet_Accessory[i]["category"]))
392             for i in range(len(custom_closet_Top)):
393                 T.insert(tk.END, '%s %s\n' % (
394                     custom_closet_Accessory[i]["design"], custom_closet_Accessory[i]["category"]))
395
396     T.pack()
397     cloth_info_print__Accessory.mainloop()
```

# #코디 추천 버튼

```
1 usage
807 def center_window(window, width, height):
808     screen_width = window.winfo_screenwidth()
809     screen_height = window.winfo_screenheight()
810
811     x = (screen_width - width) // 2
812     y = (screen_height - height) // 2
813
814     window.geometry(f"{width}x{height}+{x}+{y}")
815
816 root = tk.Tk() # 메인 윈도우 창
817 root.geometry("800x600") # 창 크기를 크게 조절
818
819 cloth_enter_btn = tk.Button(root, text="옷 정보 입력 및 삭제", command=cloth_enter,
820                             font="Helvetica 20 bold italic", fg="#8A2BE2") # 옷 정보 입력 및 삭제 버튼
821 cloth_enter_btn.place(relx=0.5, rely=0.2, anchor="center") # 가로 중앙 정렬
822
823 cloth_info_btn = tk.Button(root, text="옷장 정보", command=closet_info, font="Helvetica 20 bold italic", fg="#8A2BE2") # 옷장 정보 출력 버튼
824 cloth_info_btn.place(relx=0.5, rely=0.5, anchor="center") # 가로 중앙 정렬
825
826 cloth_recommend_btn = tk.Button(root, text="코디 추천", command=recommend_coordi,
827                                 font="Helvetica 20 bold italic", fg="#8A2BE2") # 코디 추천 버튼
828 cloth_recommend_btn.place(relx=0.5, rely=0.8, anchor="center") # 가로 중앙 정렬
829
830 # 창을 화면 중앙에 배치
831 center_window(root, width=800, height=600)
832
833 root.mainloop()
```

# #추천한 코드를 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_Accessory) + len(custom_closet_Top) + len(custom_closet_Low) +
160                                                         +len(custom_closet_Low) + len(custom_closet_Outer) + len(custom_closet_Accessory)), 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_Accessory = get_recommendation_custom_Accessory() #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
17 1 usage
18 def GetWeather():
19     # https에 필요한 ssl 인증서 확인을 무시하기 위해 SSL 컨텍스트를 생성해 https문서를 열람할 수 있게 한다.
20     context = ssl._create_unverified_context()
21     # 네이버의 서울 날씨 웹페이지에 접속한다.
22     webpage = urllib.request.urlopen(
23         url: 'https://search.naver.com/search.naver?sm=top_hyt&fbm=0&ie=utf8&query=%EC%84%9C%EC%9A%B8%EB%82%A0%EC%94%A8',
24         context=context)
25     # 지정한 url을 파이썬이 대신 열어서 해당 html파일을 복사(파싱)해온다.
26     soup = BeautifulSoup(webpage, features: 'html.parser')
27     # temperature_text클래스이면서 div태그 안에 담겨있는 현재기온 값(temps) 받아오기
28     temps = soup.find(name: 'div', attrs: 'temperature_text')
29     # 찾은 요소에서 텍스트를 추출하고, 공백을 제거한다.
30     temps = str(temps.text.strip())
31     # 현재 기온
32     global Weather
33     # \d+.\d+:숫자들과 숫자들 사이에 어떤 문자가 와 있는 경우를 추출한다.
34     # re.findall: String에서 패턴에 해당하는 내용을 찾아서 리스트로 리턴한다.
35     # [0]으로 리스트의 첫 번째 요소, 즉 숫자 패턴의 첫 번째 부분 선택, float로 선택된 숫자를 부동소수점으로 변환.
36     # 즉, 문자열 temps에서 기온(실수)만 가져온다.
37     Weather = float(re.findall(pattern: '\d+.\d+', temps)[0])
```

# #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_Accessory:
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
149
150 for clothing in closet_outlet:
151     if clothing["category"] in clothing_type: # 현재 기온에 맞는 겹옷 타입과 일치하는 겹옷을 보유하고 있다면
152         recommended_out.append(clothing) # 겹옷 추천 함수의 리턴 값(리스트)에 옷 추가
153
154 # 추천 겹옷(리스트) 리턴
155 return recommended_out
156 return recommended_bot
```

## #카테고리에 없는 옷 추천하는 함수

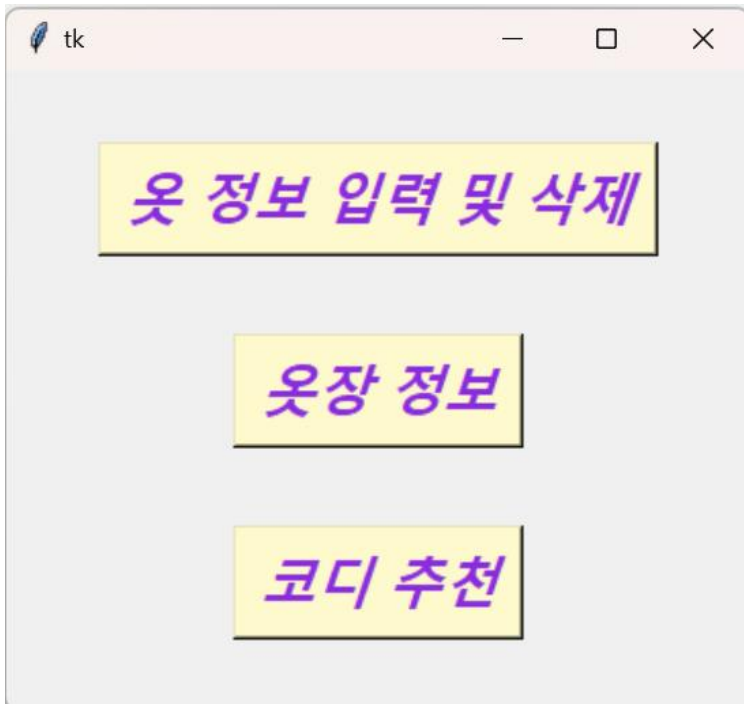
```
267 #카테고리에 없는 액세서리 추천 함수  
    1 usage  
268 def get_recommendation_custom_Accessory():  
269     global Weather #기온 전역변수  
270     custom_recommended_Accessory = [] #기온 범위에 들어가는 카테고리에 없는 액세서리들을 저장할 리스트 카테고리에 없는 액세서리 추천 리스트  
271  
272     for i in range(len(custom_closet_Accessory)): #반복문으로 카테고리에 없는 액세서리들을 저장한 리스트의  
273         if (Weather - 2) <= float(custom_closet_Accessory[i]["temperature"]) <= (Weather + 2): #테  
274             custom_recommended_Accessory.append(custom_closet_Accessory[i]) #그 요소를 카테고리에 없는  
275  
276     return custom_recommended_Accessory #카테고리에 없는 액세서리 추천 리스트를 리턴  
277
```

## #추천한 코드를 출력하는 함수(2)

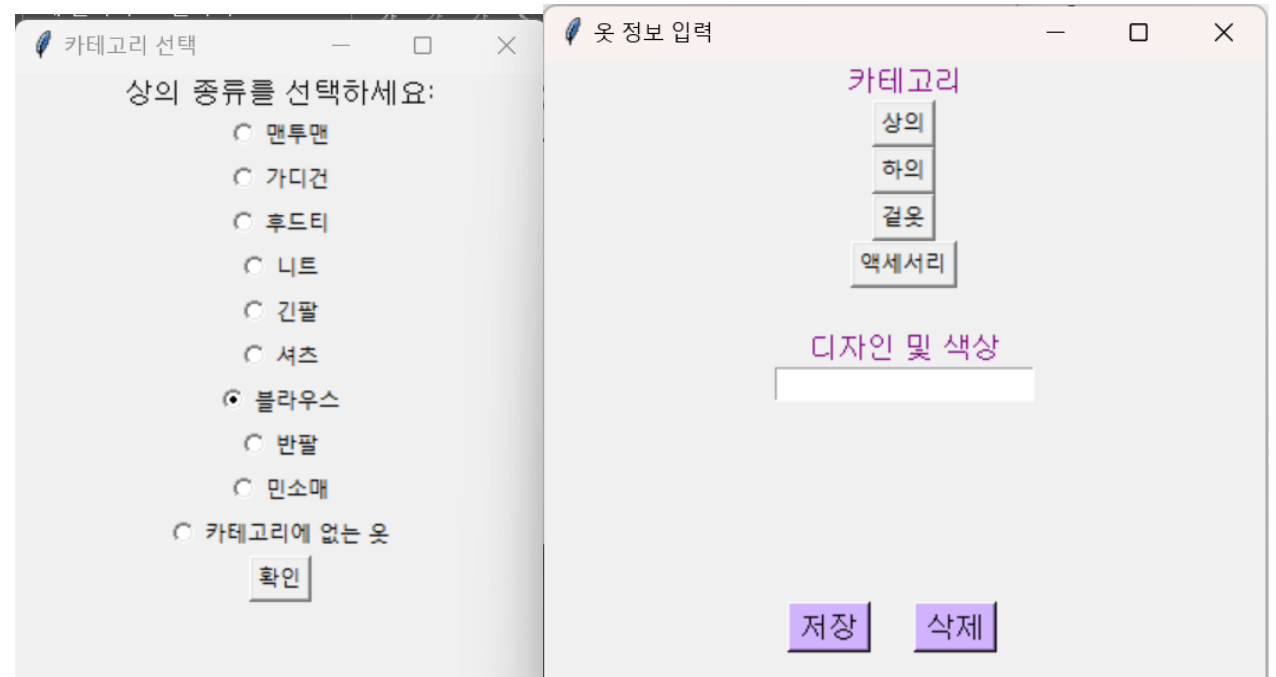
```
218 # 액세서리 추천
219 if recommended_acc or (custom_recommended_Accessory != []):
220     T.insert(tk.END, chars: "추천 액세서리:\n")
221     i = 1
222     for clothing in recommended_acc:
223         T.insert(tk.END, chars: f"{i}. {clothing['design']} {clothing['category']}")
224         T.insert(tk.END, chars: "\n")
225         i += 1
226     for n in range(len(custom_recommended_Accessory)) : #반복문으로 카테고리에 없는 액세서리 추천 리스트의 요소들 출력
227         T.insert(tk.END, '현재 기온에는 %s %s도 적합합니다.\n' % (custom_recommended_Accessory[n]["design"], custom_recommended_Accessory[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

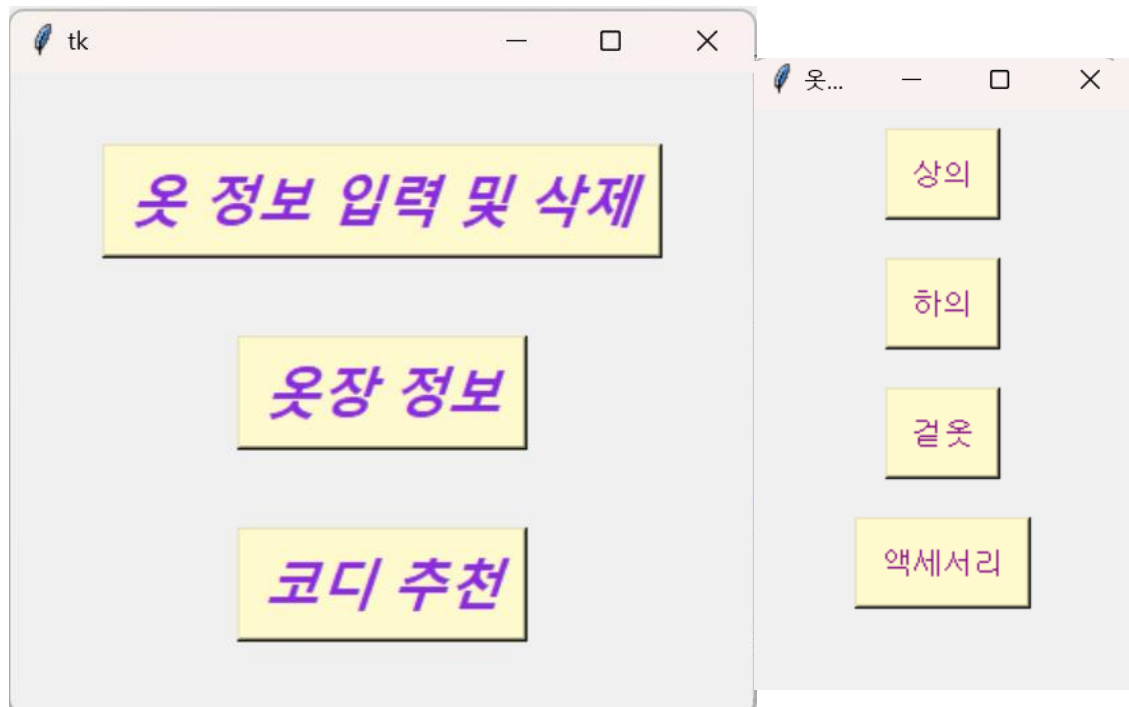


## 실행 시 첫 화면

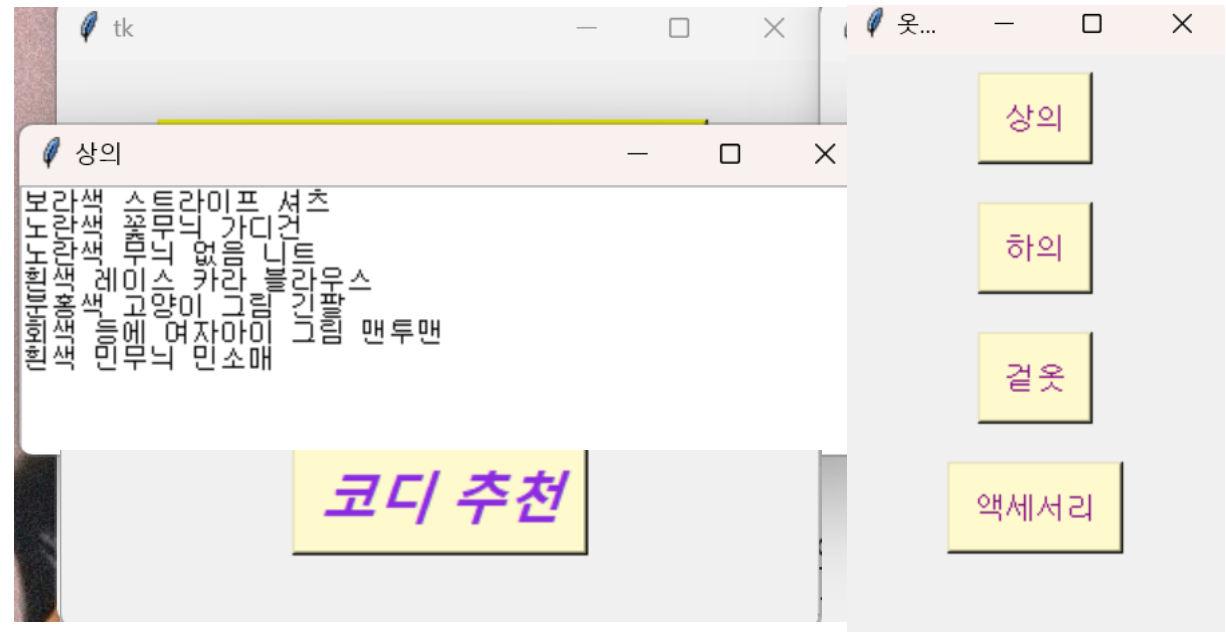


옷 정보 입력 및 삭제 : 상의, 하의, 겉옷, 액세서리로 나눠서 정보를 입력하고 삭제할 수 있다.

# 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, 겹옷, 액세서리가  
출력된다.

