```
#select_GUI_client.py
#GUI TCP 클라이언트 프로그램
#버튼 클릭으로 ON/OFF 제어

from socket import *
from tkinter import *
from select import *

#데이터를 수신하고 처리하는 함수
def handle():
    global root, sock, switch_state_label

    #읽기 조사 소켓 설정
    r_sock, w_sock, e_sock = select([sock], [], [], 0)

    #읽기 이벤트가 발생한 소켓에서 데이터를 읽고 메시지를 처리한다
    if r_sock:
        msg = sock.recv(1024).decode()
        print(msg)
        #토글 동작
        if msg.upper() == 'OFF':
            switch_state_label.configure(text='Switch is OFF')
        else:
            switch_state_label.configure(text='Switch is ON')

    #연속적인 동작을 위해 200ms 후에 다시 handle 함수 호출
    root.after(200, handle)

def button_command():
    global sock, btn_text, btn_color

    if btn_text == 'ON':
        btn_text = 'OFF'
        btn_color = 'blue'

    else:
        btn_text = 'ON'
        btn_color = 'red'

    LED_button.configure(text=btn_text, bg=btn_color)#버튼 표시 수정
    sock.send(btn_text.encode()) #LED 상태 전송
```

```
#소켓을 생성하고 연결
sock = socket() #기본으로 TCP 소켓 생성
sock.connect(('localhost', 2500))

#루트 윈도우
root = Tk()
btn_color = 'red'
btn_text = 'ON'

#라벨과 버튼 생성
LED_label = Label(text="LED")
Switch_label = Label(text="SWITCH")
switch_state_label = Label(text="Switch is OFF",fg='blue')
LED_button = Button(text=btn_text, fg='yellow',
                    bg=btn_color, command=button_command)

#라벨과 버튼 배치
LED_label.grid(row=0, column=0)
LED_button.grid(row=0, column=1)
Switch_label.grid(row=1, column=0)
switch_state_label.grid(row=1, column=1, sticky=E)

#데이터를 읽고 처리한다
handle() # mainloop() 이전에 실행해야 한다
root.mainloop() #윈도우 루프 실행
```