```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import signal
import sys
import argparse
import array
import time
import math
try:
    import serial
except:
    print('Falta instalar pyserial')
    print('Pruebe con pip3 install pyserial.')

#============================CONSTANTES========================================

# Esperar reporte de stack
ESPERAR_REQ_OPCIONAL_R5 = False

# Esperar reporte de heap
ESPERAR_REQ_OPCIONAL_R6 = False

# Cuántas veces repetir el envío del archivo
REPETICIONES=10

STX = ord('{')
ETX = ord('}')

#==============================================================================

def signal_handler(sig, frame):
    print('Se pulsó Ctrl+C!, saliendo del programa!')
    try:
        puerto.close()
    except:
        pass
    else:
        print('Se cerró el puerto OK.')
    sys.exit(0)

def cartel_de_envio(args):
    if args.operacion == 0:
        str_oper = 'mayusculizar'
    elif args.operacion == 1:
        str_oper = 'minusculizar'
    else:
        str_oper = 'medir perfomance'

    print('------------------------------------------------------------------------------')
    print('--- Enviando {} por {} para {}'.format(
        args.nombre_archivo,
        args.puerto,
        str_oper))
    print('------------------------------------------------------------------------------')

def cartel_de_recepcion():
    print('------------------------------------------------------------------------------')
    print('--- Respuestas recibidas -----------------------------------------------------')
    print('------------------------------------------------------------------------------')

def enviar_archivo(archivo, puerto, operacion):
    '''
    Envía un archivo por partes y retorna el numero de paquetes enviados
    '''
    archivo.seek(0)
```

```
    lineas_enviadas = 0
    for line in archivo:
        if line.strip(' \r\n'): #if line is not empty, or a white space line
            linea = line.strip('\r\n')
            print('Enviando linea', str(lineas_enviadas).rjust(2), end =":")
            enviar_paquete(linea, puerto, operacion)
            lineas_enviadas+=1
    return lineas_enviadas

def enviar_paquete(linea, puerto, operacion):
    packet = '{{{op}{t:02}{datos}}}'.format(op=operacion,
                                            t=len(linea),
                                            datos=linea)
    buf = bytearray()
    buf.extend(bytes(packet, encoding='ASCII'))
    puerto.write(buf)
    puerto.flush()
    print('{}'.format(packet))
    return len(linea)

def mostrar_respuestas(puerto, n_paquetes_esperados):
    eco = bytes()
    paquetes_recibidos = 0
    while True:
        char = puerto.read()
        eco += char
        if ord(char) == ETX: # fin de linea
            paquetes_recibidos += 1 # estamos asumiendo que llego bien...
            linea = eco.decode('ascii').strip('\r\n')
            n_linea = paquetes_recibidos
            print('Recibido {}: {}'.format(n_linea, linea))
            if paquetes_recibidos < n_paquetes_esperados:
                eco = bytes() #nuevo
            else:
                return

def obtener_argumentos_cli():
    parser = argparse.ArgumentParser(
            prog='enviar_texto.py',
            usage='python3 enviar-texto.py <nombre_archivo> <nombre_puerto> <operacion>',
            description='Transmite un <archivo> por <puerto> a 115200bps.')

    parser.add_argument(
            dest='nombre_archivo',
            action='store',
            type=str,
            help='El archivo a abrir.')

    parser.add_argument(
            dest='puerto',
            action='store',
            type=str,
            help='El puerto serie a abrir.')

    parser.add_argument(
            dest='operacion',
            action='store',
            type=int,
            choices=[0, 1],
            help='La operacion 0 (Mayusculizar), 1 (minusculizar)')

    return parser.parse_args()

#===============================================================================
```

```
if __name__ == '__main__':
    signal.signal(signal.SIGINT, signal_handler)

    args = obtener_argumentos_cli()

    try:
        puerto = serial.Serial(args.puerto, 115200)
    except:
        print('No se puede abrir el puerto')
        exit()

    if ESPERAR_REQ_OPCIONAL_R6:
        mostrar_respuestas(puerto, 1) # mostrar reporte de heap

    with open( (args.nombre_archivo), 'rt') as archivo:
        for j in range(REPETICIONES):
            cartel_de_envio(args)
            lineas_enviadas = enviar_archivo(archivo, puerto, args.operacion)
            cartel_de_recepcion()
            if ESPERAR_REQ_OPCIONAL_R5:
                #recibimos lineas más reporte de stack después de cada operacion
                lineas_esperadas = lineas_enviadas * 2
            else:
                lineas_esperadas = lineas_enviadas
            mostrar_respuestas(puerto, lineas_esperadas)

#================================================================================
```