```
"""Run with 'python client.py <address> <port number> <file name>'"""
from records import FileRequest, FileResponse
from records import FILE_REQUEST_MAGIC_NO, FILE_RESPONSE_MAGIC_NO
import socket
from common import *
import time
import sys
import os


def get_address_portno_filename():
    """Gets the address, port number and file name from the
    command line.  Returns tuple: (address_str, port_num, file_name)"""
    try:
        address_str = sys.argv[1].strip()
        port_num_str = sys.argv[2].strip()
        file_name = sys.argv[3].strip()
    except IndexError:
        error(MISSING_ARG_ERR)

    port_num = convert_portno_str(port_num_str)

    return address_str, port_num, file_name


#def write_bytes_to_file(file_name, byte_data):
    #"""Takes a file_name and a bytearray and writes the byte array
    #to a file.  If there is an IOError then calls
    #error(FILE_ALREADY_EXISTS_ERR)"""
    #try:
        #with open(file_name, 'wb') as outfile:
            #outfile.write(byte_data)

        #print("Wrote?", file_name)
    #except IOError:
        #error(FILE_ALREADY_EXISTS_ERR)
    #finally:
        #outfile.close()



def client():

    # Get command line arguments
    address_str, port_num, file_name = get_address_portno_filename()

    # Get address from address string
    try:
        address = socket.getaddrinfo(address_str, port_num)  # Change to parellel as
signment?
    except socket.gaierror:
        error(CANT_CONVERT_ADRESS_ERR)

    # Check that the requested file doesen't already exist locally
    #if file_exists_locally(file_name):
    if False:  # Temp
        error(FILE_ALREADY_EXISTS_ERR.format(os.path.basename(file_name)))

    # Create a socket
    try:
        sockfd = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sockfd.settimeout(TIMEOUT)
    except socket.gaierror:
        error(COULDNT_CREATE_ERR)

    # Try to connect
    try:
        sockfd.connect(address[0][4])
    except ConnectionRefusedError:
        sockfd.close()
        error(COULDNT_CONNECT_ERR)
```

```
    # Build a FileRequest
    file_request = FileRequest(file_name)

    # Send FileRequest
    n_bytes_sent = sockfd.send(file_request.get_bytearray())
    print(n_bytes_sent, "Bytes sent")

    # Recieve a number of bytes equal to the length of the header
    server_file_response_header = sockfd.recv(FileResponse.header_byte_len())
    print("Response header:", server_file_response_header)
    for byte in server_file_response_header:
        print(byte)

    # Check header validity
    if not FileResponse.is_valid_FileResponse(server_file_response_header):
        error(INVALID_FILE_RESPONSE_ERR)

    # Extract DataLen from header  Do I need this if I'm recieving in blocks?
    status, DataLen = FileResponse.get_status_DataLen(server_file_response_header)
    print("Header DataLen:", DataLen)

    ## Recieve an amount of bytes equal to the length of the file (Temp?)
    #file_bytearray = sockfd.recv(DataLen)

    # Write bytearray to local file
    new_filename = os.path.join(os.path.dirname(file_name), "new_"+os.path.basename(
file_name))
    #write_bytes_to_file(new_filename, file_bytearray)
    if status == 1:
        download_file_from_socket(new_filename, sockfd)
    else:
        print("Server couldn't get the file.")




def download_file_from_socket(file_name, sockfd):
    try:
        outfile = open(file_name, 'wb')

        downloaded_bytes = 0
        reached_EOF = False
        while not reached_EOF:

            data_block = sockfd.recv(BLOCK_SIZE)  #data_block acts like a buffer
            #print("Recieved bytes:", len(data_block))

            if data_block is None:    # Nessesary?
                reached_EOF = True
                break
            if len(data_block) <= 0:
                print("Less than Block", len(data_block))
                reached_EOF = True

            outfile.write(data_block)
            downloaded_bytes += len(data_block)
            print("downloaded {} bytes".format(downloaded_bytes))
            time.sleep(0.01)

    except IOError as e:
        print(e)
        error(COULDNT_WRITE_FILE_ERR)
    finally:
        outfile.close()
        sockfd.close()
```

```
client()
```