1. Buatlah sebuah program time server dengan ketentuan sebagai berikut
    a. Membuka port di port 45000 dengan transport TCP
    Ubah port tersebut pada file server_thread.py (Buka folder progjar3 lalu buka folder threading_examples). Fungsi tersebut terdapat pada fungsi run.

```
def run(self):
    self.my_socket.bind(('0.0.0.0',45000))
    self.my_socket.listen(1)
    while True:
        self.connection, self.client_address = self.my_socket.accept()
        logging.warning(f"connection from {self.client_address}")

        clt = ProcessTheClient(self.connection, self.client_address)
        clt.start()
        self.the_clients.append(clt)
```

    b. Server harus dapat melayani request yang concurrent
    Ubah variabel ‘svr’ pada fungsi main dari Server() menjadi TimeServer() untuk membuat sebuah time server.

```
def main():
    svr = TimeServer()
    svr.start()
```

    c. Request hanya dilayani dengan ketentuan
        i. Diawali dengan string “TIME dan diakhiri dengan karakter 13 dan karakter 10”

```
data.startswith(b'TIME') and data.endswith(b'\r\n'):
```

        - data.startswith(b'TIME') memeriksa apakah byte string data diawali dengan karakter "TIME, data.endswith(b'\r\n') memeriksa apakah byte string data diakhiri dengan karakter "\r\n".
        - Karakter “\r” menandakan sebagai karakter 13 atau *Carriage Return* dan “\n” sebagai karakter 10 atau *Line Feed*. Jika digabung menjadi “\r\n” maka akan membuat sebuah delimit Bernama CRLF (*Carriage Return Line Feed).*

    d. Server akan merespon dengan jam dengan ketentuan
        i. Dalam bentuk string (UTF-8)

```
self.connection.sendall(action.encode('utf-8'))
```

        ii. Diawali dengan “JAM<spasi><jam>”

```
action = f"JAM {current_time_formatted}\r\n"
```

        iii. <jam> berisikan info jam dalam format “hh:mm:ss” dan diakhiri dengan karakter 13 dan karakter 10
        Format akan diubah menggunakan ‘.strftime’

```
current_time_formatted = time.strftime("%H:%M:%S")
```

        Akhiran karakter 13 dan karakter 10 sudah diformatkan pada variabel “action”.

Berikut hasil akhir dari gabungan code diatas:

```
if data:
    decoded_data = data.decode('utf-8')
    clean_data = decoded_data.strip()
    logging.warning(f"[SERVER] menerima pesan {decoded_data}")
    if data.startswith(b'TIME') and data.endswith(b'\r\n'):
        current_time_formatted = time.strftime("%H:%M:%S")
        respon = f"JAM {current_time_formatted}\r\n"
        logging.warning(f"[SERVER] mengirim respon {respon}")
        self.connection.sendall(respon.encode('utf-8'))
        self.server.count_responses()
    else:
        break
```

2. Jalankan di lab environment
    a. Tuliskan dalam satu file PDF dengan nama TUGAS2.PDF
        i. Link menuju source code anda di github (masing-masing harus punya repository di github)
        Link repository:
        https://github.com/Ichlas02/Tugas2_Progjar_A/tree/main

        ii. Capturelah hasil eksekusi program server Anda

        Hasil eksekusi pada program server:

```
WARNING:root:Jumlah response terkirim: 26691
WARNING:root:connection from ('127.0.0.1', 42196)
WARNING:root:[SERVER] menerima pesan TIME

WARNING:root:[SERVER] mengirim respon JAM 13:34:03

WARNING:root:Jumlah response terkirim: 26692
WARNING:root:connection from ('127.0.0.1', 42198)
WARNING:root:[SERVER] menerima pesan TIME

WARNING:root:[SERVER] mengirim respon JAM 13:34:03

WARNING:root:Jumlah response terkirim: 26693
```

Dapat dilihat delimit CRLF berhasil dijalankan karena setelah mengirim respon yang diakhiri dengan ‘\r\n’, otomatis akan diberi spasi antar baris.