/home/cosc/student/rvo16/Documents/Cosc264/Assignment - Socket/Server/server.py

```
 1  import socket
 2  import datetime
 3  import sys
 4  import os
 5
 6  HOST = '0.0.0.0'
 7
 8  def get_port():
 9      PORT = int(sys.argv[1])
10      if PORT < 1024 or PORT > 64000:
11          sys.exit("ERROR: PORT NUMBER MUST BE BETWEEN 1024 AND 64000 (INCLUSIVE)")
12      else:
13          print('PORT IS VALID') #just to check
14          return PORT
15
16  def create_and_bind(PORT):
17      s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
18      try:
19          s.bind((HOST, PORT))
20      except Exception as e:
21          print("Error {}".format(e))
22          sys.exit("ERROR: SOCKET CREATION IS BAD")
23      print("CREATE AND BIND SUCCESS") #just to check
24      return s
25
26  def listen(s):
27      try:
28          s.listen()
29      except socket.error:
30          s.close()
31          sys.exit("ERROR: LISTENING FAILURE")
32      print("LISTENING...")  #just to check
33
34      while True:
35          connection_socket, address = s.accept()
36          now = datetime.datetime.now()
37          print(now.strftime("Time: %H:%M:%S")) #current time
38          ip_address, port_number = address
```

```
39          print('Connected by IP adress:{} and Port number:{}'.format(ip_address, port_number))
40
41          #=========READ FIXED HEADER==============
42          connection_socket.settimeout(1)
43          try:
44              data = connection_socket.recv(5)
45          except socket.timeout:
46              print("ERROR: CONNECTION TIMEOUT. RESTARTING LOOP")
47              connection_socket.close()
48              continue #goes back to the start of the loop
49
50          #==============VALIDATING DATA============
51
52          MagicNo = data[0] << 8 | data[1]
53          Type = data[2]
54          FilenameLen = data[3] << 8 | data[4]
55          if MagicNo == 0x497E and Type == 1 and FilenameLen > 1 and FilenameLen < 1024:
56              print("CONDITIONS ARE CORRECT")
57              pass
58          else:
59              print("ERROR: FILE REQUEST IS ERRONEOUS")
60              connection_socket.close()
61              continue
62
63          #========READING MORE BYTES FOR FILENAME============
64
65          connection_socket.settimeout(1)
66          try:
67              filename_data = connection_socket.recv(FilenameLen)
68          except socket.timeout:
69              print("ERROR: CONNECTION TIMEOUT. RESTARTING LOOP")
70              connection_socket.close()
71              continue #goes back to the start of the loop
```

```
72
73          #======OPEN FILE FOR READING===========
74          requested_filename = filename_data.decode('utf-8')
75
76          try:
77              f = open(requested_filename, 'rb') #rb means to 'read bytes'
78              print("FILE EXISTS AND CAN BE OPENED") #just to check
79
80              MagicNo_Response = (0x497E).to_bytes(2, byteorder='big')
81              Type_Response = (2).to_bytes(1, byteorder='big')
82              StatusCode = (1).to_bytes(1, byteorder='big')
83
84              cwd = os.getcwd()
85              DataLength = os.path.getsize(cwd + '/' + str(requested_filename))
86              DataLength = DataLength.to_bytes(4, byteorder='big')
87
88              header = bytearray(MagicNo_Response + Type_Response + StatusCode + DataLength)
89              connection_socket.send(header)
90              DataLength_sent = 0
91
92          except IOError:
93              MagicNo_Response = (0x497E).to_bytes(2, byteorder='big')
94              Type_Response = (2).to_bytes(1, byteorder='big')
95              StatusCode = (0).to_bytes(1, byteorder='big')
96              DataLength = (0).to_bytes(0, byteorder='big')
97              # StatusCode is 0 so FileData field contains no bytes.
98
99              header = bytearray(MagicNo_Response +
```

```
            Type_Response + StatusCode + DataLength)
100             connection_socket.send(header)
101
102             print("ERROR: FILE DOES NOT EXIST OR CANNOT BE OPENED")
103             connection_socket.close()
104             continue
105
106         while True:
107             f_data = f.read(4096)
108             connection_socket.send(f_data)
109             if len(f_data) == 0:
110                 break
111             DataLength_sent += len(f_data)
112         print("THE NUMBER OF BYTES TRANSFERED IS: {}".format(DataLength_sent))
113         connection_socket.close()
114         continue
115
116
117 def main():
118     port = get_port()
119     s = create_and_bind(port)
120     listen(s)
121
122
123 main()
```