server.py

```
import socket
import sys
import datetime
import os
import traceback

IP = "0.0.0.0"


def check_magic_no(file_request):
    """
    checks the magic_no_from_client which will be give in byte
    tries to confirm that it is in 0x497E
    raises an error  if its not a 0x497E

    :param file_request:
    :return paassed (boolean):
    """
    try:
        magic_no = file_request[:2]
        if int.from_bytes(magic_no, 'big') == 0x497E:
            print('Magic number accepted.\n')
            return True

        print('Unacceptable Magic number: Expected 0x497E got
{}.'.format(magic_no))
        return False

    except:
        print(traceback.format_exc())
        return False


def process_port_number(port):
    """
    checks the port number, returns true if port is in range 1024 to 64000
    if not the print the error and exit

    :param port:
    :return:
    """

    print(socket.getaddrinfo(IP, port))

    try:
        port = int(port)
        if port not in range(1024, 64001):
            print('Unacceptable port number: Must be in range between 1024 to
64000.\n')
            sys.exit(1)
        else:
            print('Port number is valid. Your port number is {}\n'.format(port))
            return port

    except:
        print('Error while checking the port number\n')
        sys.exit(1)


def attempts_bind(s, port_number):
    """tries to bind the socket from client.py, raises an error if something goes
wrong while that"""
    try:
        s.bind((IP, port_number))
        print("The server has been created.\n")

    except:
        s.close()
        print('Error while trying to bind')
        sys.exit(1)


def attempts_accept(s):
    """
    takes a parameter s which stands for a socket and tries to accept the
connection from the client.py
    time out is set to 1 second so if it takes more than 1 second to process then
it will raise an error
    :param s:
    :return connection, address:
    """
    try:
        connection, address = s.accept()

        print('Connection established successfully')
        print(datetime.datetime.now())
        print('Client IP address {} port number
{}\n'.format(socket.getaddrinfo(address[0], address[1])[0][4][0],

socket.getaddrinfo(address[0], address[1])[0][4][1]))
        return connection, address

    except:
        print('Error while accepting \n')
        sys.exit(1)


def check_packet_type(header):
    """
    checks the packet type from header,
    if packet type is 1 then print that packet type is accepted
    if packet type is not 1 then prints the error and returns False.
    :param header:
    :return: passed (boolean)
    """
    try:
        packet_type = header[2]
        if packet_type != 1:
            print('Unacceptable Packet type: Got type {}, expected type
1.\n'.format(packet_type))
```

```
            raise OSError
        print('Packet type accepted.\n')
        return True

    except:
        print("Error while checking the packet type\n")
        return False


def try_get_file_contents(file_name):
    """
    tries to open the given file_name and returns the contents of the file
    if it fails, tell them that it did not work and return the empty byte and
status code as 0
    :param file_name:
    :return: infile, status_code
    """
    try:
        file = open(file_name, 'rb')
        infile = file.read()
        file.close()
        print('File read successfully.')
        return infile, 1

    except:
        print('Error while trying to open the file\n')
        return b'', 0


def attempts_listen(s, num):
    """
    tries to listen, then if it works then prints that socket is listening
    if something else happens then print the error then returns False
    :param s:
    :param num:
    :return: passed (boolean)
    """
    try:
        s.listen(num)
        print('Listening...\n')
        return True
    except:
        print('Error trying to listen.\n')
        return False


def try_create_socket():
    """
    tries to create the socket if creates then prints that socket has been created
    if this causes an error then prints the error then exit
    :return:
    """
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        print('socket has been created')
        return s

    except:
        print('Error while trying to create a socket')
        sys.exit(1)


def try_receive(connection, buffer):
    """
    tries to receive the bytes from client with stated buffer then returns the
received bytes and True
    if this does not work then prints the error and returns False.
    :param connection:
    :param buffer:
    :return: received bytes passed (boolean)
    """
    try:
        print('server received {} bytes\n'.format(buffer))
        return connection.recv(buffer), True

    except:
        print('Error while tyring to receive file request {}\n')
        return b'', False


def process_file_name_len(header):
    """
    tries the get a length of file name then returns it
    if this causes an error then prints the error and returns False

    :param header:
    :return: length of file name, processed
    """
    try:
        file_name_len = header[3] + header[4]
        if 1 > file_name_len or file_name_len > 1024:
            sys.exit(1)
        return file_name_len, True

    except:
        print("Error while trying to read the file name length from header")
        return 0, False


def check_file_exists(file_name):
    """
    returns if the file actually exists or not
    if yes then True
    else False
    if something goes when checking then print the error then return False
    :param file_name:
    :return:
    """
    try:
        return os.path.exists(file_name)

    except:
        print("Error while checking the existence of file")
        return False
```

```
def try_send(s, packet):
    """
    tries to send the packet if this process works then return True
    if not then print the error then return False
    :param s:
    :param packet:
    :return True or False:
    """
    try:
        s.sendall(packet)
        return True

    except:
        print('Problem occurred while sending')
        return False


def check_arguments():
    """
    checks the arguments that I was given
    it should be just server.py and port_number but you never know!

    checks the number of arguments if not 2 then print the error and exit
    :return nothing :
    """
    try:
        if len(sys.argv) != 2:
            if len(sys.argv) < 2:
                print('Expected 2 arguments, got only {}\n'.format(len(sys.argv)))

            else:
                print('Expected 2 arguments, got {}\n'.format(len(sys.argv)))
            sys.exit(1)
    except:
        print("Error while checking number of arguments")
        sys.exit(1)


def run_server(server_socket, port_number):
    """
    takes the server socket and port number and runs the main functions for server

    :param server_socket:
    :param port_number:
    :return:
    """
    with server_socket:
        attempts_bind(server_socket, port_number)
        attempts_listen(server_socket, 1)
        while True:
            print('Awaiting the clients to connect...\n')

            connection, address = attempts_accept(server_socket)  # gets client
socket and its address
            try:
                connection.settimeout(1)  # set a time out to check that socket
process will not take more than 1 sec
```

```
                header_array = bytearray()
                file_request_header, passed = try_receive(connection, 5)  # tries
to get a header of file request header
                if not passed:  # if receiving did not work then pass must be
False
                    connection.close()  # since you did not get a header you close
the socket
                    continue
                header_array += file_request_header  # add the header bytes into
bytes

                passed = check_magic_no(header_array)  # checks if magic_no is
what I want
                if not passed:  # when it is not 0x497E
                    connection.close()
                    continue

                passed = check_packet_type(header_array)  # checks if packet type
is what I want
                if not passed:  # when it is not 1
                    connection.close()
                    continue

                file_name_len, processed = process_file_name_len(header_array)   #
checks the length
                if not processed:  # when you could not get a length
                    connection.close()
                    continue

                file_name, passed = try_receive(connection, file_name_len)  #
buffer will be length that I got before
                if not passed:  # when you could not get a file name
                    connection.close()
                    continue

                file_exists = check_file_exists(file_name)  # checks if server has
the file that client want
                if file_exists:
                    infile, status_code = try_get_file_contents(file_name)  #
status code should be 1

                else:
                    print(file_name, ' does not exist\n')  # if file does not
exist then do the following
                    infile = b''  # empty it
                    status_code = 0  # 0 means not oh my good it did not work!
                if infile == b'':
                    file_response = bytearray() + 0x497E.to_bytes(2, 'big') +
0x02.to_bytes(1, 'big')
                    file_response += status_code.to_bytes(1, 'big') +
len(infile).to_bytes(4, 'big')
                else:
                    file_response = bytearray() + 0x497E.to_bytes(2, 'big') +
0x02.to_bytes(1, 'big')
                    file_response += status_code.to_bytes(1, 'big') +
len(infile).to_bytes(4, 'big') + infile

                passed = try_send(connection, file_response)  # this one should
```

```
work just fine
                if not passed:
                    connection.close()
                    continue

                print("{} bytes has been transferred.\n".format(len(infile) + 8))

            except socket.timeout:  # oh no! this whole process took more than 1
second!?
                print('connection timed out\n')

            except:  # not sure if I get this but just in case if something goes
wrong
                print(traceback.format_exc())  # print the error
            connection.close()


def main():
    check_arguments()  # server gotta get 2 arguments which are server.py and
port_number
    port_number = process_port_number(sys.argv[1])  # chuck it in the processor
    server_socket = try_create_socket()
    run_server(server_socket, port_number)  # goes to the main process


if __name__ == "__main__":
    main()
```