client.py

```
import socket
import sys
import os

def check_magic_no(header):
    """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"""
    try:
        magic_no = ((header[0] << 8) + header[1]).to_bytes(2, 'big')
        if int.from_bytes(magic_no, 'big') != 0x497E:
            sys.exit(1)
        print('Magic number acceptable.\n')

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


def check_packet_type(header):
    try:
        packet_type = header[2]
        if packet_type != 2:
            sys.exit(1)
        print('Packet type accepted.\n')

    except:
        print('packet type was not 2\n')
        sys.exit(1)


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
s
    :param port:
    :return:
    """
    try:
        port = int(port)
        if port in range(1024, 64001):
            print('Port number is valid. Your port number is {}\n'.format(port))
            return port

        else:
            sys.exit(1)

    except:
        print('Unacceptable port number: Must be in range between 1024 to
64000.\n')
        sys.exit(1)
```

```
def check_file_exists(file_name):
    try:
        if os.path.exists(file_name) is True:
            sys.exit()
        else:
            print('Check file: Passed.\n')

    except:
        print('File already exists.\n')
        sys.exit(1)


def check_status_code(header):
    status_code = header[3]
    try:
        if status_code == 1:
            print("Status code: 1 (Processing)\n")

        elif status_code == 0:
            sys.exit(1)
        else:
            sys.exit(1)

        return status_code

    except:
        print('Status code was not 1 (Stopped processing)\n')
        sys.exit(1)


def check_file_length(length_file1, length_file2):
    """
    param length_file1 which is the size of the new_file given from the
file_response len_file from the server
    param lentgth file2 which is the size of the new_file that client has got from
file_response file_Data
    checks the length of the new_file that you get from the server with what
client has processed from the data.
    if the size are equal then we are good
    if nein then raises an error then exit.

    :param length_file1:
    :param length_file2:
    :return:
    """

    try:
        if length_file1 == length_file2:
            print('The file has successfully downloaded.\n')
        else:
            raise OSError
    except:
        print('length of new_file and len data did not match\n')
        print('Expected {} but, got {}.'.format(length_file1, length_file2))
        sys.exit(1)
```

```
def try_connect(s, socket_fd):
    try:
        s.connect(socket_fd)

    except:
        print("Error while trying to connect:")
        s.close()
        sys.exit(1)


def try_receive(s, buffer):

    try:
        print("client has received {} bytes from server".format(buffer))
        return s.recv(buffer)

    except:
        print("Error while trying to receive")
        s.close()
        sys.exit(1)


def try_get_address_info(ip_address, port_number):
    """
    Tries to get a ip address and port number
    if it fails then prints the error and exit
    I go through this process because ip_address could be a name of the host
instead of the
    actual dotted decimal notation

    :param ip_address:
    :param port_number:
    :return ip_address, port_number:
    """
    try:
        return socket.getaddrinfo(ip_address, port_number)[0][4]

    except:
        print("Error while trying to get a ip_address and port number of server")
        sys.exit(1)


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

    except:
        print('Problem occurred while sending')
        sys.exit(1)
```

```
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 4 then print the error and exit
    :return nothing :
    """
    try:
        if len(sys.argv) != 4:
            if len(sys.argv) < 4:
                print('Expected 4 arguments, got only {}'.format(len(sys.argv)))

            else:
                print('Expected 4 arguments, got {}'.format(len(sys.argv)))

            sys.exit(1)

    except:
        print('number of arguments must be 4')
        sys.exit(1)


def try_create_socket():
    try:
        return socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    except:
        print("Error while creating socket")
        sys.exit(1)


def process_command_line():
    """
    checks the ip address and port number and file name
    :return:
    """
    try:
        check_arguments()  # check the number of arguments they should be
client.py ip_address port_number file_name

        ip_address = sys.argv[1]  # no checking at this phase, will check with
port number later om

        port_number = process_port_number(sys.argv[2])  # checks the port

        socket_fd = try_get_address_info(ip_address, port_number)  # gets the
ip_address and port

        print('client log', socket_fd, '\n')  # prints the ip_address and port
number of the client

        file_name = sys.argv[3]  # gets the file name check it later on

        return socket_fd, file_name

    except Exception as e:
        print(e)
```

```
        sys.exit(1)


def contact_server(socket_fd, file_name):
    """
    takes the socket_fd which has ip_address and port number of the server and
name of the file
    tris contact with the server to get the file from server

    :param socket_fd:
    :param file_name:
    :return:
    """
    try:
        check_file_exists(file_name)
        file_len_bytes = len(file_name).to_bytes(2, 'big')
        file_request = bytearray() + 0x497E.to_bytes(2, 'big') + 0x01.to_bytes(1,
'big') + file_len_bytes
        file_request += file_name.encode('utf-8')

    except:
        print("Error while creating a file_request\n")
        sys.exit(1)

    s = try_create_socket()
    s.settimeout(1)
    total_bytes_received = 0
    try:
        try_connect(s, socket_fd)

        try_send(s, file_request)

        header = try_receive(s, 8)
        total_bytes_received += 8

        check_magic_no(header)

        check_packet_type(header)

        status_code = check_status_code(header)

        if status_code == 1:
            new_file = open(file_name, 'wb+')
            try:
                    while True:
                        infile = s.recv(4096)
                        if len(infile) < 4096:
                            total_bytes_received += len(infile)
                            new_file.write(infile)
                            break
                        total_bytes_received += len(infile)
                        new_file.write(infile)

                    new_file.close()

                    data_size_from_server = int.from_bytes(header[4:], 'big')

                    file_that_client_received = open(file_name, 'rb')
```

```
                    content_of_file = file_that_client_received.read()

                    check_file_length(data_size_from_server, len(content_of_file))

                    file_that_client_received.close()

                    print('File has been successfully downloaded!\n')

                    print('total bytes received from server is {}
bytes\n'.format(total_bytes_received))


            except Exception as e:
                print('Problem occurred while processing the file {}\n'.format(e))
                new_file.close()

    except socket.timeout:
        print("client socket timed out\n")

    finally:
        print('client socket closed\n')
        s.close()
        sys.exit()


def main():
    print('')
    socket_fd, file_name = process_command_line()
    print('Arguments accepted\n')
    contact_server(socket_fd, file_name)


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