##line-count

def count\_lines\_in\_file(path\_to\_file):

f = None

try:

f = open(path\_to\_file, 'r')

lines = 0

line = f.readline()

while line != '':

lines += 1

line = f.readline()

return lines

finally:

if f != None:

f.close()

def user\_interface():

while True:

path\_to\_file = input('What file? ').strip()

if path\_to\_file == '':

break

try:

lines\_in\_file = count\_lines\_in\_file(path\_to\_file)

print('{} line(s) in {}'.format(lines\_in\_file, path\_to\_file))

except:

print('Failed')

if \_\_name\_\_ == '\_\_main\_\_':

user\_interface()

##Sum-numbers

def sum\_numbers(numlist):

sum = 0

for sublist in numlist:

for num in sublist:

sum += num

return sum

assert(sum\_numbers([[1, 2], [3, 4], [5, 6, 7]]) == 28)

assert(sum\_numbers([[1], [2], [3, 4, 5]]) == 15)

assert(sum\_numbers([[3], [], [5]]) == 8)

assert(sum\_numbers([[1, 2]]) == 3)

assert(sum\_numbers([[9]]) == 9)

assert(sum\_numbers([]) == 0)

def sum\_numbers(numlist):

sum = 0

for element in numlist:

if type(element) == list:

for num in element:

sum += num

else:

sum += element

return sum

assert(sum\_numbers([[1, 2, 3], 4, [5, 6], 7, 8]) == 36)

assert(sum\_numbers([1, 2, 3]) == 6)

assert(sum\_numbers([[1, 2], [3, 4], [5, 6]]) == 21)

assert(sum\_numbers([[1], [2], [3]]) == 6)

assert(sum\_numbers([[8]]) == 8)

assert(sum\_numbers([[], [], [], []]) == 0)

assert(sum\_numbers([]) == 0)

def nested\_sum(nested\_list):

sum = 0

for element in nested\_list:

if type(element) == list:

sum += nested\_sum(element)

else:

sum += element

return sum

assert(nested\_sum([3, 6, 4]) == 13)

assert(nested\_sum([[[1, 2], 3], 4]) == 10)

assert(nested\_sum([[2, 7], [3, 8], [4, 9]]) == 33)

assert(nested\_sum([1, [2, [3, [4, [5], 6], 7], 8], 9]) == 45)

assert(nested\_sum([]) == 0)

##Server

import socket

if \_\_name\_\_ == '\_\_main\_\_':

listen\_socket = socket.socket()

listen\_address = ('127.0.0.1', 19999)

listen\_socket.bind(listen\_address)

listen\_socket.listen(0)

print('Waiting for connection...')

stream\_socket, from\_address = listen\_socket.accept()

listen\_socket.close()

print('Connection from {}'.format(from\_address))

while True:

incoming\_bytes = stream\_socket.recv(4096)

if len(incoming\_bytes) == 0:

print('Connection from {} closed'.format(from\_address))

break

else:

print(incoming\_bytes.decode(encoding='utf-8'), end='')

stream\_socket.close()

##Server

import socket

if \_\_name\_\_ == '\_\_main\_\_':

listen\_socket = socket.socket()

listen\_socket.bind(('127.0.0.1', 19999))

listen\_socket.listen(0)

print('Waiting for connection...')

connection\_socket, from\_address = listen\_socket.accept()

listen\_socket.close()

print('Connection from {} accepted'.format(from\_address))

connection\_socket\_input = connection\_socket.makefile('r')

line = connection\_socket\_input.readline()

while len(line) > 0:

print(line, end='')

line = connection\_socket\_input.readline()

connection\_socket\_input.close()

connection\_socket.close()

##Client

import socket

import time

if \_\_name\_\_ == '\_\_main\_\_':

connect\_socket = socket.socket()

print('Connecting...')

connect\_socket.connect(('127.0.0.1', 19999))

connect\_socket\_output = connect\_socket.makefile('w')

print('Sending the song lyrics...')

for bottles in range(99, 0, -1):

lyric\_line = '{} bottles of beer on the wall\n'.format(bottles)

connect\_socket\_output.write(lyric\_line)

connect\_socket\_output.flush()

time.sleep(0.2)

print('Done!')

connect\_socket\_output.close()

connect\_socket.close()

##I32tfsp

import socket

import time

if \_\_name\_\_ == '\_\_main\_\_':

connect\_socket = socket.socket()

print('Connecting...')

connect\_socket.connect(('127.0.0.1', 19999))

connect\_socket\_output = connect\_socket.makefile('w')

print('Sending the song lyrics...')

for bottles in range(99, 0, -1):

lyric\_line = '{} bottles of beer on the wall\n'.format(bottles)

connect\_socket\_output.write(lyric\_line)

connect\_socket\_output.flush()

time.sleep(0.2)

print('Done!')

connect\_socket\_output.close()

connect\_socket.close()

##i32tfsp.py

import collections

import os.path

import socket

class I32TfspError(Exception):

pass

def send\_file(host, port, file\_path, description):

connection = \_initiate\_connection(host, port)

try:

\_send\_hello(connection)

\_receive\_hi(connection)

\_send\_filename\_and\_description(

connection, os.path.basename(file\_path), description)

if \_receive\_yes\_or\_no(connection):

\_send\_file\_contents(connection, file\_path)

\_receive\_got\_it(connection)

finally:

\_close\_connection(connection)

def receive\_file(port, receive\_directory, accept\_file\_function):

connection = \_accept\_connection(port)

try:

\_receive\_hello(connection)

\_send\_hi(connection)

filename, description = \_receive\_filename\_and\_description(connection)

if accept\_file\_function(filename, description):

\_send\_yes(connection)

\_receive\_file\_contents(connection, os.path.join(receive\_directory, filename))

\_send\_got\_it(connection)

else:

\_send\_no(connection)

finally:

\_close\_connection(connection)

\_I32TfspConnection = collections.namedtuple(

'\_I32TfspConnection', ['socket', 'socket\_input', 'socket\_output'])

def \_initiate\_connection(host, port):

connect\_socket = socket.socket()

connect\_socket.connect((host, port))

return \_build\_connection\_object(connect\_socket)

def \_accept\_connection(port):

listen\_socket = socket.socket()

listen\_socket.bind(('', port))

listen\_socket.listen(0)

connect\_socket, from\_address = listen\_socket.accept()

listen\_socket.close()

return \_build\_connection\_object(connect\_socket)

def \_build\_connection\_object(connect\_socket):

connect\_socket\_input = connect\_socket.makefile('r')

connect\_socket\_output = connect\_socket.makefile('w')

return \_I32TfspConnection(

socket=connect\_socket, socket\_input=connect\_socket\_input,

socket\_output=connect\_socket\_output)

def \_readline\_from\_connection(connection):

return connection.socket\_input.readline()[:-1]

def \_writeline\_to\_connection(connection, line):

connection.socket\_output.write(line + '\n')

connection.socket\_output.flush()

def \_close\_connection(connection):

connection.socket\_input.close()

connection.socket\_output.close()

connection.socket.close()

def \_expect\_to\_receive\_message(connection, expected\_message):

if \_readline\_from\_connection(connection) != expected\_message:

raise I32TfspError()

def \_expect\_to\_receive\_message\_with\_parameter(connection, expected\_message):

line = \_readline\_from\_connection(connection)

if line.startswith(expected\_message) and len(line) > len(expected\_message) + 1:

return line[len(expected\_message)+1:]

else:

raise I32TfspError()

def \_send\_hello(connection):

\_writeline\_to\_connection(connection, 'I32TFSP\_HELLO')

def \_receive\_hello(connection):

\_expect\_to\_receive\_message(connection, 'I32TFSP\_HELLO')

def \_send\_hi(connection):

\_writeline\_to\_connection(connection, 'I32TFSP\_HI')

def \_receive\_hi(connection):

\_expect\_to\_receive\_message(connection, 'I32TFSP\_HI')

def \_send\_filename\_and\_description(connection, filename, description):

\_writeline\_to\_connection(connection, 'FILE ' + filename)

\_writeline\_to\_connection(connection, 'DESCRIPTION ' + description)

def \_receive\_filename\_and\_description(connection):

filename = \_expect\_to\_receive\_message\_with\_parameter(connection, 'FILE')

description = \_expect\_to\_receive\_message\_with\_parameter(connection, 'DESCRIPTION')

return filename, description

def \_send\_yes(connection):

\_writeline\_to\_connection(connection, 'YES')

def \_send\_no(connection):

\_writeline\_to\_connection(connection, 'NO')

def \_receive\_yes\_or\_no(connection):

line = \_readline\_from\_connection(connection)

if line == 'YES':

return True

elif line == 'NO':

return False

else:

raise I32TfspError()

def \_send\_file\_contents(connection, file\_path):

file\_input = open(file\_path, 'r')

try:

file\_lines = file\_input.readlines()

\_writeline\_to\_connection(connection, 'LINES {}'.format(len(file\_lines)))

for line in file\_lines:

\_writeline\_to\_connection(connection, line[:-1])

\_writeline\_to\_connection(connection, 'END')

finally:

file\_input.close()

def \_receive\_file\_contents(connection, file\_path):

file\_output = open(file\_path, 'w')

try:

line\_count = int(\_expect\_to\_receive\_message\_with\_parameter(connection, 'LINES'))

for i in range(line\_count):

file\_output.write(\_readline\_from\_connection(connection) + '\n')

\_expect\_to\_receive\_message(connection, 'END')

\_send\_got\_it(connection)

except ValueError:

raise I32TfspError()

finally:

file\_output.close()

def \_send\_got\_it(connection):

\_writeline\_to\_connection(connection, 'GOT\_IT')

def \_receive\_got\_it(connection):

\_expect\_to\_receive\_message(connection, 'GOT\_IT')