Socket Basics

With these challenges we will be communicating with servers using JSON files.
We will need to use the `JSON` package, and the commands loadstring `loads()` and dumpstring `dumps()`.

The first thing we will do is point out what a JSON is.
It is a dictionary object which allows servers to look for options and then access data. It is a common way for servers to detail what they are allowed to to, and respond quickly to requets, or quickly return the necessary error message.

For our challenges here we will ALWAYS send an option, this tells the server what we want to do.

lets say hi to the server connection r:
{"option": "say_hello"}

```
r.send('{"option": "say_hello"}')
```

perfect. we would get a response back.

This won't work with variables

if I wanted to send a value to the server, like my public key, `my_key`. I wouldn't be able to do that.

```
r.send('{"option": "send_key", "key": my_key}')
```

This would produce an error, or at least questionable results.

It would also be unreasonable for you to type in all 2048 bits of a modulus here also, that's a long number, yuck.

So we use the `JSON` package, and dump the string into a json... and that function `dumps()` will package the varialbes into a usable JSON for us.

```
# this is just an example!
# the socket connection is named r

import json

# get key from whatever source it's provided
# here we use an example where it's read from a file
with open("my_key.pem", 'rb') as f:
     my_key = f.read()

# now create a dictionary and add elements
msg_to_server = {}
msg_to_server["option"] = "send_key"
msg_to_server["key"] = my_key

# create json string (encode to make bytes-object too)
msg = json.dumps(msg_to_server).encode()
```

```
# send message to connection r
r.send(msg)
```

There are some basic, common steps.

1. Identify Server (IP Address or URL)
2. Identify Port (value from 0 - 65535)
3. Open Conneection
4. Use send / receive commands to communicate
5. Close the connection

PWN TOOLS

We will work through some simple Socket IO basics.
First we will use the tools from `pwntools` (which you should install if you don't have it)

To create a connection:

```
# import pwntools
import pwn

# 1 - identify server
HOST = 'socket.cryptocat.uk'

# 2 - identify port (one I give in the lab)
PORT = 13375

# 3 - open connection
r = remote(HOST, PORT)

# 4 - use send / receive to communicate
received = r.recvline()
print(received)

sendmsg = b'{"option": "say_hello", "say": "hi"}'
r.send(sendmsg)

received = r.recvline()
print(received)
# this should be an error message

# and so on...

#5 - close connection
r.close()
```

Note This socket should open, and is a vote-for-pedro CTF
Don't try to play this one, without talking to me first
(you're welcome to! but it's quite advanced and you'll need the server code)

SOCKET

We will work through some simple Socket IO basics.
First we will use the tools from `socekt` (which you should install if you don't have it)

To create a connection:

```
# import socket
import socket

# 1 - identify server
HOST = 'socket.cryptocat.uk'

# 2 - identify port (one I give in the lab)
PORT = 13375

# 3 - open connection
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))

# 4 - use send / receive to communicate
buffer = 1024
received = s.recv(1024)
print(received)

sendmsg = b'{"option": "say_hello", "say": "hi"}'
s.send(sendmsg)

received = s.recv(buffer)
print(received)
# this should be an error message

# and so on...

#5 - close connection
s.close()
```

Couple tips and trics

When working with diriect socket connections details are important.

At times you will need to ensure you send a newline character. Pay attention to whether you're doing that already.

Pwntools has lots of commands, like, `r.sendline()` and `r.recvline()`, these parse through lines. Sometimes that can be quite helpful.

The connectinons are usually stable and quick to respond. However you may want to add a `sleep(0.5)` command to rest your process for half a second to give time for communications to complete. Sometimes you may be getting incomplete information. If you feel like that might be you, then you can add a pause.