# Text-adventure MUD's

## MUD-Pi

https://github.com/Frimkron/mud-pi/

> What is a MUD?
>
> MUD is short for Multi-User Dungeon. A MUD is a text-based online role-playing game. MUDs were popular in the early 80s and were the precursor to the graphical Massively-Multiplayer Online Role-Playing Games we have today, like World of Warcraft. http://www.mudconnect.com is a great site for learning more about MUDs.

In [ ]:

In [ ]:

# telnet

To connect to the game: `$ telnet <ip address> 1234`

> Telnet (short for "teletype network")[1][2] is a client/server application protocol that provides access to virtual terminals of remote systems on local area networks or the Internet.[3]

https://en.wikipedia.org/wiki/Telnet

> Telnet is simple text-based network communication protocol that was invented in 1969 and has since been superseded by other, more secure protocols. It does remain popular for a few specialised uses however, MUD games being one of these uses. A long (and boring) history of the telnet protocol can be found here: http://www.cs.utexas.edu/users/chris/think/ARPANET/Telnet/Telnet.shtml

https://github.com/Frimkron/mud-pi/

# run a game as a service

To keep the game running as a background service on the server, we can use systemd *service files*.

It makes it possible to run commands like: `$ sudo service mygame status`, to see if the game is still running.

Or restart it with `$ sudo service mygame restart`.

See: https://pzwiki.wdka.nl/mediadesign/Service_files

In [ ]:

# Class()

> One of the goals of object-oriented programming is to **create reusable code**. Once you have written the code for a class, you can create as many objects from that class as you need. It is worth mentioning at this point that classes are usually saved in a separate file, and then imported into the program you are working on. So you can build a library of classes, and use those classes over and over again in different programs. Once you know a class works well, you can leave it alone and know that the objects you create in a new program are going to work as they always have.

http://introtopython.org/classes.html

- rocket example: http://introtopython.org/classes.html#Making-multiple-objects-from-a-class
- dogs example: https://docs.python.org/3/tutorial/classes.html#class-and-instance-variables

# Sockets

> Sockets and the socket API are used to send messages across a network. They provide a form of inter-process communication (IPC). The network can be a logical, local network to the computer, or one that's physically connected to an external network, with its own connections to other networks.

If you want to explore sockets in Python: https://realpython.com/python-sockets/

# JSON

**writing** to a json file

In [3]:
```
#remember to import the library
import json

#the data, in this case a dictionary
messages = {
    "room1": "this is room one's message",
    "room2": "this is room two's message",
    "room3": "this is room three's message"
}

# Serializing json
json_messages = json.dumps(messages, indent=4)

# Writing to sample.json
with open("sample.json", "w") as outfile:
    outfile.write(json_messages)

# Close the file
outfile.close()
```

**opening** a json file

In [7]:
```
import json

with open('sample.json', 'r') as f:
  data = json.load(f)

print(data)
```

```
{'room1': "this is room one's message", 'room2':
"this is room two's message", 'room3': "this is room
three's message"}
```