# How program works

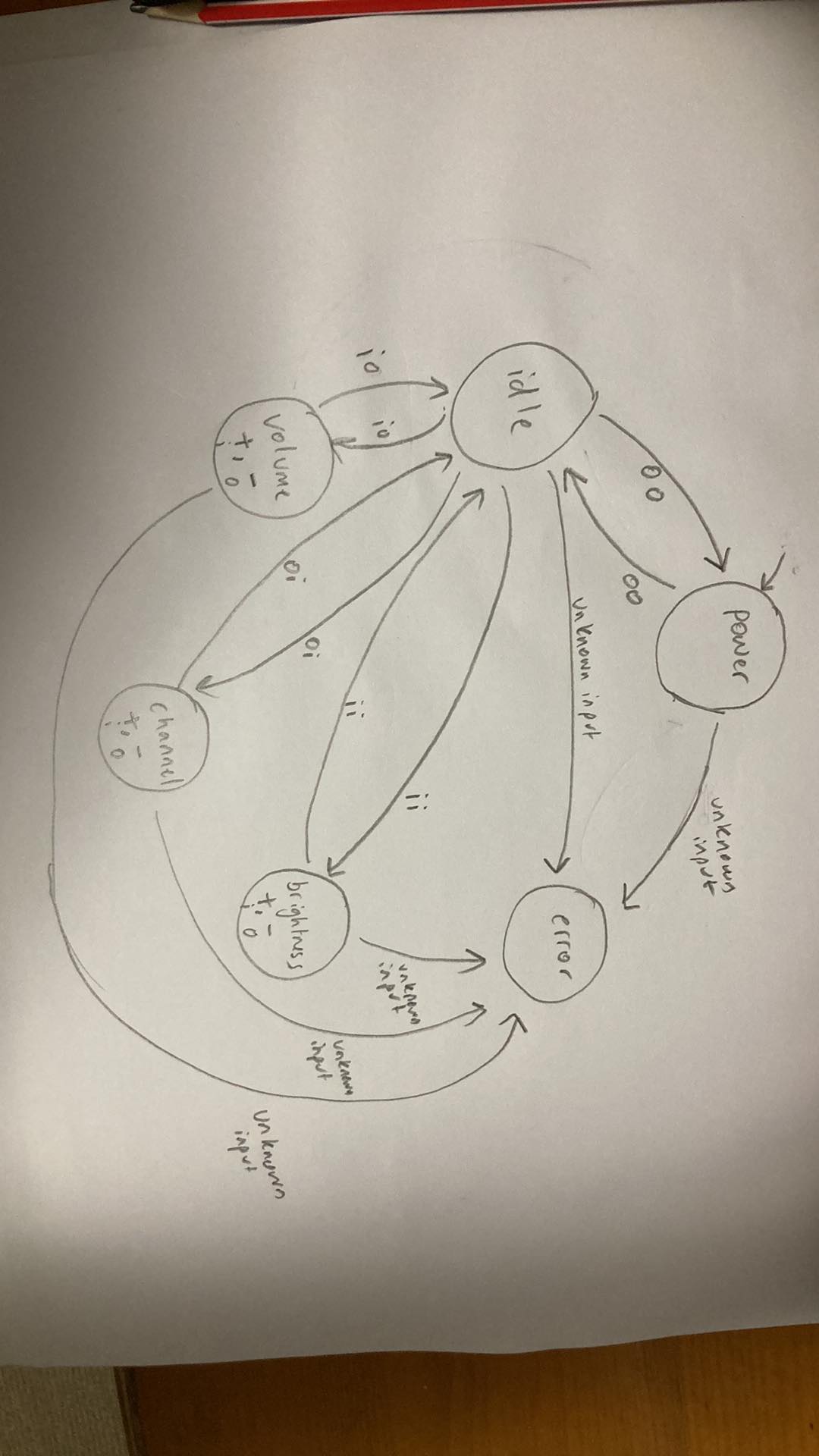
Make sure remote.py and tv.py are in the same folder. Run the terminal from this folder using python remote.py.

The two input buttons are ‘i’ and ‘o’. The TV must be ON for any function to work, such as changing the volume, channel or brightness. If it is not on, the console will say so. All functions can be accessed via this table:

|  |  |  |
| --- | --- | --- |
| key1 | key2 | function |
| o | o | Power on/off |
| o | i | Volume |
| i | o | Channel |
| i | i | Brightness |

Once in the function mode, the volume, channel or brightness can be increased by either using ‘i’ to increase or ‘o’ to decrease. To exit the function, simply enter the correct keypair. After this, the next function can be accessed by entering a different keypair.

# Finite state machine diagram



This is the FSM for the TV. The very first input determines whether the machine is powered on or off.

Description of states:

**Power:** State to show if the TV is on or off, given by a variable (poweron = 0 or 1). Functions such as volume, brightness and channel can only be performed while the tv power is in the on state.

**Idle:** The idle state is when nothing is happening. However, from this state, any other state can be transitioned to, whether that be power, volume, channel or brightness using the function keypairs. Any unknown input will also result the machine going into the error state.

**Volume, channel, brightness:** These states can only be transitioned to when in the idle state. When in these states, the user can choose to increase or decrease the selected function. To return to the idle state, the same keypair that was used to enter the state, must be used to exit the state.

**Error:** This state is entered when an unknown input has been entered by the user. Once the user has been notified, the state transitions immediately to idle.

# Software design principle

I have used a class to describe the TV. This is because you can think of the TV (tv.py) as an object being controlled by a remote (remote.py). The class allows me to give the TV attributes such as the states it can be in. This also allows me to define functions that the TV can do such as increasing/decreasing the volume, channel or brightness.

The advantage of using a class instead of hardcoding the TV is that it is modular. I can easily expand the number of functions and buttons. If there are different types of TVs, then I can easily create different instances of them. Additionally, with inheritance, I can have different TV models that expand the functionality of the existing model.