Adding and Removing Permissions Shorthand

Shorthand yes, but long brain!

To understand how this shorthand method works we first need a little background in number systems. Our typical number system is decimal. It is a base 10 number system and as such has 10 symbols (0 - 9) used. Another number system is octal which is base 8 (0-7). Now it just so happens that with 3 permissions and each being on or off, we have 8 possible combinations (2^3). Now we can also represent our numbers using binary which only has 2 symbols (0 and 1). The mapping of octal to binary is in the table below.

```
Octal    Binary
  0       000
  1       001
  2       010
  3       011
  4       100
  5       101
  6       110
  7       111
```

So if you can imagine, that each set of three binary numbers, corresponds with the three rights per Owner, Group and User. Then you can use Octal numbers to set the binary equivalent rights.
Wait what? I don't get it!

Remember this:

1. rw-r–r– 1 pi pi 4452 Jul 25 18:01 example.py

```
Could be written:
-110 100 100 (a one where that attribute is on, a zero where it is off)
- (minus sign) is the file/directory, we don't change that
-110 100 100 binary, in Octal would be 644
```

Let's look at a file with all rights:

```
chmod 777 examply.py
ls -l
-rwxrwxrwx 1 pi pi 4452 Jul 25 18:01 example.py
```

```
File has FULL permissions
-rwxrwxrwx is -111111111 or in octal 777
```

```
chmod 644 example.py
ls -l
-rw-r--r-- 1 pi pi 4452 Jul 25 18:01 example.py
Now our file is back to its original permissions.
```