Using a configspec

ConfigObj also provides a way to validate your configuration files using a **configspec**. When I mentioned that I was going to write on this topic, Steven Sproat (creator of Whyteboard) volunteered his configspec code as an example. I took his specification and used it to create a default config file. In this example, we use Foord's validate module to do the validation. I don't think it's included in your ConfigObj download, so you may need to download it as well. Now, let's take a look at the code:

```
import configobj, validate

cfg = """
bmp_select_transparent = boolean(default=False)
canvas_border = integer(min=10, max=35, default=15)
colour1 = list(min=3, max=3, default=list('280', '0', '0'))
colour2 = list(min=3, max=3, default=list('255', '255', '0'))
colour3 = list(min=3, max=3, default=list('0', '255', '0'))
colour4 = list(min=3, max=3, default=list('255', '0', '0'))
colour5 = list(min=3, max=3, default=list('0', '0', '255'))
colour6 = list(min=3, max=3, default=list('160', '32', '240'))
colour7 = list(min=3, max=3, default=list('0', '255', '255'))
colour8 = list(min=3, max=3, default=list('255', '165', '0'))
colour9 = list(min=3, max=3, default=list('211', '211', '211'))
convert_quality = option('highest', 'high', 'normal', default='normal')
default_font = string
default_width = integer(min=1, max=12000, default=640)
default_height = integer(min=1, max=12000, default=480)
imagemagick_path = string
handle_size = integer(min=3, max=15, default=6)
language = option('English', 'English (United Kingdom)', 'Russian',
                  'Hindi', default='English')
print_title = boolean(default=True)
statusbar = boolean(default=True)
toolbar = boolean(default=True)
toolbox = option('icon', 'text', default='icon')
undo_sheets = integer(min=5, max=50, default=10)
"""
```

```
def createConfig(path):
    """

    Create a config file using a configspec
    and validate it against a Validator object
    """
    spec = cfg.split("\n")
    config = configobj.ConfigObj(path, configspec=spec)
    validator = validate.Validator()
    config.validate(validator, copy=True)
    config.filename = path
    config.write()

if __name__ == "__main__":
    createConfig("config.ini")
```

The configspec allows the programmer the ability to specify what **types** are returned for each line in the configuration file. It also can be used to set a default value and a **min** and **max** values (among other things). If you run the code above, you will see a *config.ini* file generated in the current working directory that has just the default values. If the programmer didn't specify a default, then that line isn't even added to the configuration.

Let's take a closer look at what's going on just to make sure you understand. In the **createConfig** function, we create a ConfigObj instance by passing in the file path and setting the configspec. Note that the configspec can also be a normal text file or a python file rather than the string that is in this example. Next, we create a Validator object. Normal usage is to just call config.validate(validator), but in this code I set the copy argument to True so that I could create a file. Otherwise, all it would do is validate that the file I passed in fit the configspec's rules. Finally I set the config's filename and write the data out.

Wrapping Up

Now you know just enough to get you started on the ins and outs of ConfigObj. I hope you'll find it as helpful as I have. Be sure to go to the module's documentation and read more about what it and **validate** can do.