configobj in Python

WE'LL COVER THE FOLLOWING

- Getting Started

Python comes with a handy module called **ConfigParser**. It's good for creating and reading configuration files (aka INI files). However, Michael Foord (author of IronPython in Action) and Nicola Larosa decided to write their own configuration module called **ConfigObj**. In many ways, it is an improvement over the standard library's module. For example, it will return a dictionary-like object when it reads a config file. ConfigObj can also understand some Python types. Another neat feature is that you can create a configuration spec that ConfigObj will use to validate the config file.

Getting Started

First of all, you need to go and get ConfigObj. This is a good time to use your knowledge from the last chapter on installing packages. Here is how you would get ConfigObj with pip:

```
pip install configobj
```

Once you have it installed, we can move on. To start off, open a text editor and create a file with some contents like this:

```
product = Sony PS3
accessories = controller, eye, memory stick
# This is a comment that will be ignored
retail_price = $400
```

Save it where ever you like. I'm going to call mine *config.ini*. Now let's see how ConfigObj can be used to extract that information:

```
from configobj import ConfigObj
config = ConfigObj(r"path to config.ini")
config["product"]
#'Sony PS3'

config["accessories"]
#['controller', 'eye', 'memory stick']

type(config["accessories"])
#<type 'list'>
```

As you can see, ConfigObj uses Python's **dict** API to access the information it has extracted. All you had to do to get ConfigObj to parse the file was to pass the file's path to ConfigObj. Now, if the information had been under a section (i.e. [Sony]), then you would have had to do pre-pend everything with ["Sony"], like this: **config["Sony"]["product"]**. Also take note that the *accessories* section was returned as a list of strings. ConfigObj will take any valid line with a comma-separated list and return it as a Python list. You can also create multi-line strings in the config file as long as you enclose them with triple single or double quotes.

If you need to create a sub-section in the file, then use extra square brackets. For example, **[Sony]** is the top section, **[[Playstation]]** is the sub-section and **[[[PS3]]]** is the sub-section of the sub-section. You can create sub-sections up to any depth. For more information on the formatting of the file, I recommend reading ConfigObj's documentation.

Now we'll do the reverse and create the config file programmatically.

```
import configobj

def createConfig(path):
    config = configobj.ConfigObj()
    config.filename = path
    config["Sony"] = {}
    config["Sony"]["product"] = "Sony PS3"
    config["Sony"]["accessories"] = ['controller', 'eye', 'memory stick']
    config["Sony"]["retail price"] = "$400"
    config.write()

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

As you can see, all it took was 13 lines of code. In the code above, we create a function and pass it the path for our config file. Then we create a ConfigObj object and set its filename property. To create the section, we create an empty dict with the name “Sony”. Then we pre-pend each line of the sections contents in the same way. Finally, we call our config object’s write method to write the data to the file.