Config files in Python

Configuration files are well suited to specify configuration data to your program. Within each config file, values are grouped into different sections (e.g., “installation”, “debug” and “server”).

Each section then has a specific value for various variables in that section. For the same purpose, there are some prominent differences between a config file and using a Python source file.

How to read a config file in Python

Call configparser.ConfigParser() to create a ConfigParser object.

Call ConfigParser.read(filename) to read the config data from filename.

Use ConfigParser.get(section, option) to retrieve the value for option in the config section.

SAMPLE_CONFIG.TXT

```
[config]
option1=a
option2=b
option3=c
```

```
parser = configparser.ConfigParser()
parser.read("sample_config.txt")
```

```
print(parser.get("config", "option1"))
```

OUTPUT

```
a
```

```
print(parser.get("config", "option2"))
```

OUTPUT

```
b
```

```
print(parser.get("config", "option3"))
```

OUTPUT

```
c
```

Read configuration files written in the common .ini configuration file format.

Code #1 : Configuration File

abc.ini

```
; Sample configuration file
[installation]
library = %(prefix)s/lib
include = %(prefix)s/include
bin = %(prefix)s/bin
prefix = /usr/local

# Setting related to debug configuration
[debug]
pid-file = /tmp/spam.pid
show_warnings = False
log_errors = true

[server]
nworkers: 32
port: 8080
root = /www/root
signature:
```

Code #2 : Reading the file and extracting values.

```
from configparser import ConfigParser

configur = ConfigParser()
print (configur.read('config.ini'))

print ("Sections : ", configur.sections())
print ("Installation Library : ", configur.get('installation','library'))
print ("Log Errors debugged ? : ", configur.getboolean('debug','log_errors'))
print ("Port Server : ", configur.getint('server','port'))
print ("Worker Server : ", configur.getint('server','nworkers'))
```

Output :

```
['config.ini']
Sections : ['installation', 'debug', 'server']
Installation Library : '/usr/local/lib'
Log Errors debugged ? : True
Port Server : 8080
Worker Server : 32
```

One can also modify the configuration and write it back to a file using the <u>cfg.write()</u> method.

Code #3 :

```
configur.set('server','port','9000')
configur.set('debug','log_errors','False')

import sys
configur.write(sys.stdout)
```

Output :

```
[installation]
library = %(prefix)s/lib
include = %(prefix)s/include
bin = %(prefix)s/bin
prefix = /usr/local

[debug]
log_errors = False
show_warnings = False

[server]
port = 9000
nworkers = 32
```

```
pid-file = /tmp/spam.pid
root = /www/root
```

Names used in a config file are also assumed to be case-insensitive as shown in the code below –

```
configur.get('installation','PREFIX')
configur.get('installation','prefix')
```

```
Output :
'/usr/local'
'/usr/local'
```

Multiple configuration files can be read together and their results can be merged into a single configuration using ConfigParser, which makes it so special to use.

Example – A user made their own configuration file that looks as.

```
; ~/.config.ini

[installation]
prefix = /Users/beazley/test
[debug]
log_errors = False
```

This file can be merged with the previous configuration by reading it separately

Code #4 :

```
import os
# Previously read configuration
print (configur.get('installation', 'prefix'))

# Merge in user-specific configuration
print (configur.read(os.path.expanduser('~/.config.ini')))
print (configur.get('installation', 'prefix'))
print (configur.get('installation', 'library'))
print (configur.getboolean('debug', 'log_errors'))
```

Output :

```
'/usr/local'
['/Users/HP/.config.ini']
'/Users/HP/test'
'/Users/HP/test/lib'
False
```