Reading and Writing Files

The file Reading/Writing process

To read/write to a file in Python, you will want to use the `with` statement, which will close the file for you after you are done, managing the available resources for you.

Opening and reading files

The `open` function opens a file and return a corresponding file object.

```
>>> with open('C:\\Users\\your_home_folder\\hi.txt') as hello_file:
...     hello_content = hello_file.read()
...
>>> hello_content
'Hello World!'
```

Alternatively, you can use the *readlines()* method to get a list of string values from the file, one string for each line of text:

```
>>> with open('sonnet29.txt') as sonnet_file:
...     sonnet_file.readlines()
...
# [When, in disgrace with fortune and men's eyes,\n',
# ' I all alone beweep my  outcast state,\n',
# And trouble deaf heaven with my bootless cries,\n', And
# look upon myself and curse my fate,']
```

You can also iterate through the file line by line:

```
>>> with open('sonnet29.txt') as sonnet_file:
...     for line in sonnet_file:
...         print(line, end='')
...
# When, in disgrace with fortune and men's eyes,
# I all alone beweep my outcast state,
# And trouble deaf heaven with my bootless cries,
# And look upon myself and curse my fate,
```

Writing to files

```
>>> with open('bacon.txt', 'w') as bacon_file:
...     bacon_file.write('Hello world!\n')
...
# 13

>>> with open('bacon.txt', 'a') as bacon_file:
...     bacon_file.write('Bacon is not a vegetable.')
...
# 25
```

```
>>> with open('bacon.txt') as bacon_file:
...     content = bacon_file.read()
...
>>> print(content)
# Hello world!
# Bacon is not a vegetable.
```

The shelve module

A “shelf” is a persistent, dictionary-like object. The difference with “dbm” databases is that the values (not the keys!) in a shelf can be essentially arbitrary Python objects — anything that the pickle module can handle. This includes most class instances, recursive data types, and objects containing lots of shared sub-objects. The keys are ordinary strings.

To save variables:

```
>>> import shelve

>>> cats = ['Zophie', 'Pooka', 'Simon']
>>> with shelve.open('mydata') as shelf_file:
...     shelf_file['cats'] = cats
```

To open and read variables:

```
>>> with shelve.open('mydata') as shelf_file:
...     print(type(shelf_file))
...     print(shelf_file['cats'])
...
# <class 'shelve.DbfilenameShelf'>
# ['Zophie', 'Pooka', 'Simon']
```

Just like dictionaries, `shelf` values have `keys()` and `values()` methods that will return list-like values of the keys and values in the shelf. Since these methods return list-like values instead of true lists, you should pass them to the `list()` function to get them in list form.

```
>>> with shelve.open('mydata') as shelf_file:
...     print(list(shelf_file.keys()))
...     print(list(shelf_file.values()))
...
# ['cats']
# [['Zophie', 'Pooka', 'Simon']]
```

Reading ZIP files

```
>>> import zipfile, os

>>> os.chdir('C:\\')    # move to the folder with example.zip
>>> with zipfile.ZipFile('example.zip') as example_zip:
...     print(example_zip.namelist())
```

```
...     spam_info = example_zip.getinfo('spam.txt')
...     print(spam_info.file_size)
...     print(spam_info.compress_size)
...     print('Compressed file is %sx smaller!' % (round(spam_info.file_size /
spam_info.compress_size, 2)))
...
# ['spam.txt', 'cats/', 'cats/catnames.txt', 'cats/zophie.jpg']
# 13908
# 3828
# 'Compressed file is 3.63x smaller!'
```

Extracting from ZIP Files

The `extractall()` method for ZipFile objects extracts all the files and folders from a ZIP file into the current working directory.

```
>>> import zipfile, os

>>> os.chdir('C:\\')    # move to the folder with example.zip

>>> with zipfile.ZipFile('example.zip') as example_zip:
...     example_zip.extractall()
```

The `extract()` method for ZipFile objects will extract a single file from the ZIP file:

```
>>> with zipfile.ZipFile('example.zip') as example_zip:
...     print(example_zip.extract('spam.txt'))
...     print(example_zip.extract('spam.txt', 'C:\\some\\new\\folders'))
...
# 'C:\\spam.txt'
# 'C:\\some\\new\\folders\\spam.txt'
```

Creating and Adding to ZIP Files

```
>>> import zipfile

>>> with zipfile.ZipFile('new.zip', 'w') as new_zip:
...     new_zip.write('spam.txt', compress_type=zipfile.ZIP_DEFLATED)
```

This code will create a new ZIP file named new.zip that has the compressed contents of spam.txt.