Files

Relevant for: COMP5711M

Brandon Bennett

School of Computing
University of Leeds

B.Bennett@leeds.ac.uk

Last Updated: August 2019

Overview

These slides cover basic operations on files, including:

- Opening and closing files.
- Creating a new file.
- Writing information to a file.
- Reading information from a file.
- The relation between file objects and files on disk.

Opening and Writing to a File

Here is a simple example of code for writing something to a file:

```
f = open( 'my_newfile', 'w' )
f.write( 'This is my new file.\n' )
f.close()
```

What it means:
`f = open( 'my_newfile', 'w' )` means assign variable `f` to the 'file object' created by the `open` function.

The string `'my_newfile'` is the name of the file (and may be a full path rather than just the file name).

The string `'w'` sets the *mode* in which the file will be used — in this case *write* mode.

...

...

`f.write( 'This is my new file.\n' )` calls the `write` method of the file object refered to by `f`.
It tells Python to write the given string to that file.

(`\n` is used to refer to the *newline* character.)

...

`f.write( 'This is my new file.\n' )` calls the `write` method of the file object refered to by `f`.
It tells Python to write the given string to that file.

(`\n` is used to refer to the *newline* character.)

`f.close()` closes the file (so you can no longer change it) and writes the file contents to the computer's hard disk).

File Objects and Files on Hard Disks

Reading or changing a file on disk is very slow compared to changing things in the computer's memory.

Thus Python (and most other programming languages) work with 'file objects' that are stored in memory and are only syncronised at certain times with the actual files on disk.

Thus, the `write` command does not normally write directly to the disk but only changes the 'file object' stored in memory.

A file object is copied to disk when a `close` command is executed.

(You can also use the command `f.flush()` to explicitly ask Python to copy the file object referred to by `f` to the corresponding file on disk, without closing the file. You probably won't need to do this.)

Another File Writing Example

Can you work out what this does and how it works?

```
squares_file = open( 'squares.txt', 'w' )

for k in range( 1, 101 ):
      output_string = str(k) + " squared is " \
                      + str(k*k) + "\n"
      squares_file.write( output_string )

squares_file.close()
```

Notes: `\` at the end of a line is used to 'wrap' a code line in the code without Python treating it as a new line of code.

In an expression of the form `s1 + s2`, where `s1` and `s2` are strings, the `+` operator joins the strings together.

Modes of File Use

Write mode is one of several ways in which you may want to operate on a file:

- `f = open( 'filename', 'w' )` opens `filename` for *writing*.
  This will create a new file if `filename` does not exist.
  If `filename` already exists, its contents will be deleted and over-written.

- `f = open( 'filename', 'r' )` opens `filename` for *reading*.

- `f = open( 'filename', 'a' )` opens `filename` for *appendi*
  (i.e. adding stuff to the end of the file).

- `f = open( 'filename', 'r+' )` opens `filename` for both *reading and writing*.

Example Code for Reading and Displaying File Contents

```
input_file = open( 'squares.txt', 'r')

while True:
    line = input_file.readline()
    if line == "":
        break
    else:
        print( line )
```

Notes: The `while True` construct is used to create a loop which will keep on running until a `break` command is executed.
After a `break` execution continues from immediately after the smallest enclosing loop (either `while` or `for`).

Notes (continued):

`readline` is a method that can be applied to any file object that has been opened in read or read+write mode. It will return the next line of text as a string.

Srings returned by `readline` will normally end in a newline character (referred to as `\n`) — except possibly the final line in a file.

The file object automatically keeps track of where it is up to in reading the file.

If the end of the file has been reached, an empty string (`""`) will be returned.

Another File Reading Example

Define a function to count lines in a file:

```
def number_of_lines_in_file( filename ):
    input_file = open( filename, 'r')
    line_list = input_file.readlines()
    n = len( line_list )
    input_file.close()
    return n
```

The `readlines()` method returns a list of all lines in the file.

Can then call the defined function to get line count for a file:

```
>>> number_of_lines_in_file( 'maze-adventure.py' )
50
```

Using the `with ... as` Construct

For cases like opening files, where one will in nearly all cases want to eventually close the file, Python provides a general construct that will automatically take care of this:

```
with open('my_newfile','r') as fileObject:
    while True:
        line = fileObject.readline()
        if line == "" :
            break
        else:
            print(line)
```

Conclusion

Reading and writing files is an important operation within many computer applications.

Modern languages like Python abstract away from most of the fiddly details of what goes on the in the computer.

It is relatively easy to read and write strings from/to a file.

You should remember to `close()` your files when you have finished with them.