Python files

Reuven M. Lerner, PhD
reuven@lerner.co.il

1

File I/O

```
f = open("/etc/passwd")

f = file("/etc/passwd")

f = file("/etc/passwd", "r")
```

2

Modes

```
f = file("/etc/passwd", “r”) # Read-only

f = file("/etc/passwd", “w”) # Write-only

f = file("/etc/passwd", “a”) # Append

f = file("/etc/passwd", “r+”) # Read/write
```

3

Reading from a file

```
f.seek(0)      # start of file

f.read()       # get the file

f.read(100)    # read 100 bytes

f.readline()   # read one line

f.readlines()  # read all lines
```

Seeking

- You can move in the file by passing seek two parameters:
- #1: Which byte
- #2: Relative to where?

```
f.seek(100)  # 100 bytes from the start

f.seek(-100, 1) # 100 bytes before the end

f.seek(100, 2) # 100 bytes before current
position
```

5

Text vs. binary

- On Unix machines, all files are equal
- On Windows, you can open files in “text” or “binary” mode.
    - In binary mode, files are opened verbatim.
    - In text mode, newlines are translated
- Add a “b” to any mode for it to work in binary mode

Closing a file

- You can close a file with:

  ```
  f.close()
  ```

- You don’t need to save, since files are closed when Python exits or the variable falls out of scope
- You might, however, need to flush its buffer:

  ```
  f.flush()
  ```

Universal newlines

- Want to read from files in a cross-platform way?
- Use “U” as the mode when reading from files
- This means, “Open for reading with universal newline support”.
- You can check the “newlines” attribute, to see what string(s) are considered newlines after reading:

```
f.newlines
```

Reading DOS on Unix

```
for line in open('dostext.txt'):
    print len(line)
5
5
5
9
10
22
19
20
```

9

Reading Unix on Unix

```
for line in open('unixtext.txt'):
    print len(line)
4
4
4
8
9
21
18
19
```

10

Reading DOS with “U”

```
for line in open('dostext.txt', 'U'):
    print len(line)
4
4
4
8
9
21
18
19
```

File buffering

- When you write to disk, the data isn’t stored right away
- Rather, it is buffered; only when the buffer fills up, is the data actually stored to disk.
- Open a file in unbuffered mode by passing a third (optional) parameter set to False:

```
open(FILENAME, MODE, False)
```

12

Flushing

- If a file is open in buffered (i.e., usual) mode, you can force Python to flush the buffer:

```
f.flush()
```

- The file remains open.

Printing a file

```
for line in f.readlines():
    print line
```

Even better: f is an iterator

```
for line in f:
    print line
```

Using with

```
with open('/etc/passwd') as f:
    for line in f:
        print line
```

16

Writing to a file

```
outfile = open("/tmp/squares.log", "w")

for num in range(10):

    outfile.write("{}{}\n".format(num, num*num))

outfile.close()
```

Types of writing

- ‘w’
- ‘a’
- ‘r+’

stdout, stderr, stdin

- In sys, Unix standards for input and output

```
stdout # Standard console output

stderr # Error console output

stdin  # Input from the user
```

19

Standard files

```
import sys

sys.stdout.write("foo")

sys.stderr.write("foo")
```

Replacing stdout

```
f = open('/tmp/output', 'w')
import sys
old_stdout = sys.stdout
sys.stdout = f
```

Directories

```
import os

files = os.listdir("/etc/")
```

Or use globbing

```
import glob

# get a list of matching files

glob.glob(‘/etc/*~’)

# get an iterator of matching files

glob.iglob(‘/etc/*~’)
```

23