Section Overview

What You Will Learn

- Reading files
- Writing to files
- Opening and closing files
- File objects
- String methods

What You Will Learn

- Reading files, one line at a time
- File modes
- Line endings (Unix vs Windows)
- Exceptions

Files

Part I

Files

Part II

Input and Output

- input() - Accept standard input
- print() - Write standard output
- Files are great for storage that lasts beyond the execution of a program.

File Input and Output

`open()` – Built-in Function that opens a file and returns a file object.

```
open(path_to_file)
```

```
path_to_file - Can be absolute or relative.
```

```
open('/var/log/messages')
open('log/messages')
```

File Input and Output

```
open('C:\Log\Messages\data.txt')
open('C:/Users/jason/Documents/python-notes.txt')
open('Documents/python-notes.txt')
```

File Objects (Stream Objects)

```
hosts = open('/etc/hosts')
hosts_file_contents = hosts.read()
print(hosts_file_contents)
```

```
127.0.0.1 localhost
```

File Objects (Stream Objects)

```
hosts = open('C:/Windows/System32/drivers/etc/hosts')
hosts_file_contents = hosts.read()
print(hosts_file_contents)
```

```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP
# for Windows.
#
```

File Position

`read()` - Returns the entire file.

`seek(offset)` - Change the current position to offset.

`seek(0)` - Go to the beginning of the file.

`seek(5)` - Go to the 5th byte of the file.

`tell()` - Determine the current position in the file.

```
hosts = open('/etc/hosts')
print('Current position: {}'.format(hosts.tell()))
print(hosts.read())

print('Current position: {}'.format(hosts.tell()))
print(hosts.read())

hosts.seek(0)
print('Current position: {}'.format(hosts.tell()))
print(hosts.read())
```

```
Current position: 0
127.0.0.1 localhost

Current position: 20

Current position: 0
127.0.0.1 localhost
```

```
hosts = open('/etc/hosts')
print(hosts.read(3))
print(hosts.tell())
```

```
127
3
```

```
hosts = open('/etc/hosts')
hosts_file_contents = hosts.read()
print(hosts_file_contents)
hosts.close()
```

```
127.0.0.1 localhost
```

```
hosts = open('/etc/hosts')
hosts_file_contents = hosts.read()
print('File closed? {}'.format(hosts.closed))
if not hosts.closed:
    hosts.close()
print('File closed? {}'.format(hosts.closed))
```

```
File closed? False
File closed? True
```

Automatically Closing a File

```
with open(file_path) as file_object_variable_name:
    # Code block
```

```
print('Started reading the file.')
with open('/etc/hosts') as hosts:
    print('File closed? {}'.format(hosts.closed))
    print(hosts.read())
print('Finished reading the file.')
print('File closed? {}'.format(hosts.closed))
```

```
Started reading the file.
File closed? False
127.0.0.1 localhost
Finished reading the file.
File closed? True
```

This is line one.
This is line two.
Finally, we are on the third and last line of the file.

```
with open('file.txt') as the_file:
    for line in the_file:
        print(line)
```

```
This is line one.

This is line two.

Finally, we are on the third and last line of the file.
```

This is line one.
This is line two.
Finally, we are on the third and last line of the file.

```
with open('file.txt') as the_file:
    for line in the_file:
        print(line.rstrip())
```

```
This is line one.
This is line two.
Finally, we are on the third and last line of the file.
```

File Modes

```
open(path_to_file, mode)
```

Mode	Description
`r`	Open for reading (default)
`w`	Open for writing, truncating first
`x`	Create a new file and open it for writing
`a`	Open for writing, appending to file

File Modes, continued

Mode	Description
`+`	Open a file for updating (read/write)
`b`	Binary mode
`t`	Text mode (default)

```
open('/pics/cat.jpg', rb)
```

Checking the file mode

```
with open('file.txt') as the_file:
    print(the_file.mode)
```

```
r
```

```
with open('file2.txt', 'w') as the_file:
    the_file.write('This text will be written to
the file.')
    the_file.write('Here is more text.')

with open('file2.txt') as the_file:
    print(the_file.read())
```

```
This text will be written to the file.Here
is more text.
```

Carriage Returns and Line Feeds

`\r`	Carriage Return
`\n`	New line

`\n`	Unix/Linux/Mac line endings.
`\r\n`	Windows style line endings.

```
with open('file2.txt', 'w') as the_file:
    the_file.write('This text will be written to
the file.\n')
    the_file.write('Here is more text.\n')

with open('file2.txt') as the_file:
    print(the_file.read())
```

```
This text will be written to the file.
Here is more text.
```

Binary Files

```
with open('cat.jpg', 'rb') as cat_picture:
    cat_picture.seek(2)
    cat_picture.read(4)
    print(cat_picture.tell())
    print(cat_picture.mode)
```

```
6
rb
```

```
# Open a file and assign its contents to a variable.
# If the file is unavailable, create an empty variable.
try:
    contacts = open('contacts.txt').read()
except:
    contacts = ''

print(len(contacts))
```

```
319
```

Section Summary

Summary

- To open a file, use the built-in open() function.

```
open(path_to_file, mode)
```

- If mode is omitted when opening a file it defaults to read-only.
- Forward slashes can be used as directory separators, even in Windows.

Summary

- The `read()` file object method returns the entire contents of the file as a string.
- To close a file, use the `close()` file object method.
- To automatically close a file use the with statement.

```
with open(file_path) as file_object_variable_name:
    # Code Block
```

Summary

- To read a file one line at a time, use a for loop.

```
for line_variable in file_object_variable:
```

- To remove any trailing white space use the `rstrip()` string method.
- Write data to a file using the `write()` file object method.

Summary

- When a file is opened in binary mode, the `read()` method accepts bytes. When a file is opened in text mode, which is the default, `read()` accepts characters.
- In most cases a character is one byte in the length, but this does not hold true in every situation. (UTF-8)

Summary

- Plan for exceptions when working with files. Use try/except blocks.