Working with Files in Python

MSc. Ivan A. Escobar Broitman

Introduction

- File access is one of the most important aspects of a language once you are confortable with the syntax.
- It is vital to be able to import data into your code in order to perform operations with it.
- Imagine if you had to type 1000 values of data one by one to perform some scientific operations on them?

Reading from Text Files

- **Plain text file:** File made up of only ASCII characters
- Easy to read strings from plain text files
- Text files good choice for simple information
  - Easy to edit
  - Cross-platform
  - Human readable!

Opening a Data File

- To open a data file you have to options:
  - **open()** or **file()**.
- File() is a backward compatibility since python 2.2.
- Syntax:
  - Handle = open (file_name, access_mode)

Opening and Closing a Text File

```
text_file = open("read_it.txt", "r")
```

- Must open before read (or write)
- `open()` function
  - Must pass string filename as first argument, can include path info
  - Pass access mode as second argument
  - Returns file object
- `"r"` opens file for reading
- Can open a file for reading, writing, or both

Opening and Closing a Text File (continued)

TABLE 7.1 SELECTED FILE ACCESS MODES

Mode	Description
"r"	Read from a file. If the file doesn't exist, Python will complain with an error.
"w"	Write to a file. If the file exists, its contents are overwritten. If the file doesn't exist, it's created.
"a"	Append a file. If the file exists, new data is appended to it. If the file doesn't exist, it's created.
"r+"	Read from and write to a file. If the file doesn't exist, Python will complain with an error.
"w+"	Write to and read from a file. If the file exists, its contents are overwritten. If the file doesn't exist, it's created.
"a+"	Append and read from a file. If the file exists, new data is appended to it. If the file doesn't exist, it's created.

Files can be opened for reading, writing, or both.

Opening and Closing a Text File (continued)

```
text_file.close()
```

- `close()` file object method closes file
- Always close file when done reading or writing
- Closed file can't be read from or written to until opened again

Reading & Writing Files - Overview

- Opening and closing files
  - `the_file = open(filename, mode)`
  - `the_file.close()`
- Reading files
  - `string = the_file.read(number_of_characters)`
  - `string = the_file.readline(number_of_characters)`
  - `list_of_strings = the_file.readlines()`
- Writing files
  - `the_file.write(string)`
  - `the_file.writelines(list_of_strings)`

The Read It Program

- File read_it.txt contains

```
Line 1
This is line 2
That makes this line 3
```

Example

- The following code prompts the user for the name of the file, reads it substring by substring:
- filename = raw_input (*"Enter the file name: ")*
- text_file = open (filename, *'r')*
- for eachline in text_file:
- print eachline,
- text_file.close()

Reading Characters from a Text File

```
>>> print text_file.read(1)

L

>>> print text_file.read(5)

ine 1
```

- `read()` file object method
  - Allows reading a specified number of characters
  - Accepts number of characters to be read
  - Returns string
- Each `read()` begins where the last ended
- At end of file, `read()` returns empty string

Reading Characters from a Text File (continued)

```
>>> whole_thing = text_file.read()

>>> print whole_thing

Line 1

This is line 2

That makes this line 3
```

- `read()` returns entire text file as a single string if no argument passed

Exercise:

- Run the following code and explain:
- filename = raw_input ("Enter the file name: ")
- text_file = open (filename, 'r')
- print text_file.read(1)
- print text_file.read(5)
- print text_file.read(1)
- print text_file.read(1)
- text_file.close()

Reading Characters from a Line

```
>>> text_file = open("read_it.txt", "r")

>>> print text_file.readline(1)

L

>>> print text_file.readline(5)

ine 1
```

- `readline()` file object method
  - Reads from current line
  - Accepts number characters to read from current line
  - Returns characters as a string

Reading Characters from a Line (continued)

```
>>> text_file = open("read_it.txt", "r")

>>> print text_file.readline()

Line 1

>>> print text_file.readline()

This is line 2

>>> print text_file.readline()

That makes this line 3
```

- `readline()` file object method
  - Returns the entire line if no value passed
  - Once you read all of the characters of a line (including the newline), the next line becomes current line

Reading All Lines into a List

```
>>> text_file = open("read_it.txt", "r")

>>> lines = text_file.readlines()

>>> print lines

['Line 1\n', 'This is line 2\n', 'That makes this
   line 3\n']
```

- `readlines()` file object method
  - Reads text file into a list
  - Returns list of strings
  - Each line of file becomes a string element in list

Looping through a Text File

```
>>> text_file = open("read_it.txt", "r")
>>> for line in text_file:
        print line
Line 1
This is line 2
That makes this line 3
```

- Can iterate over open text file, one line at a time

Writing to a Text File

- Easy to write to text files
- Two basic ways to write

Writing Strings to a Text File

```
text_file = open("write_it.txt", "w")

text_file.write("Line 1\n")

text_file.write("This is line 2\n")

text_file.write("That makes this line 3\n")
```

- `write()` file object method writes new characters to file open for writing

Writing a List of Strings to a Text File

```
text_file = open("write_it.txt", "w")

lines = ["Line 1\n", "This is line 2\n", "That
   makes this line 3\n"]

text_file.writelines(lines)
```

- `writelines()` file object method
  - Works with a list of strings
  - Writes list of strings to a file

Selected Text File Methods

TABLE 7.2 SELECTED TEXT FILE METHODS

Method	Description
`close()`	Closes the file. A closed file cannot be read from or written to until opened again.
`read([size])`	Reads *size* characters from a text file and returns them as a string. If size is not specified, the method returns all of the characters from the current position to the end of the file.
`readline([size])`	Reads *size* characters from the current line in a text file and returns them as a string. If size is not specified, the method returns all of the characters from the current position to the end of the line.
`readlines()`	Reads all of the lines in a text file and returns them as elements in a list.
`write(output)`	Writes the string *output* to a text file.
`writelines(output)`	Writes the strings in the list *output* to a text file.

Table 7.2: Selected text file methods

Final Functions with files

- To know on position in bytes we are in a current file, we can use the function **tell().**
  - f=open (read_it.txt, 'r')
  - print f.read(1) — **L**
  - print f.read(5) — **ine 1**
  - print f.read (1) — **"newline"**
  - print f.read (1) — **T**
  - print f.tell() — **8**

Final Functions with files

- To be able to move the objects position in a file you use the function **seek()**
  - **object.seek(offset, fromwhat)**
- Example:
  - f.seek(0)
  - print f.read(1)

L