Reading Files Python

Estimated time needed: **40** minutes

Objectives

After completing this lab you will be able to:

- Read text files using Python libraries

Table of Contents

Download Data

```
In [1]: import urllib.request
        url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0101EN-SkillsNetwork/labs/Module%204/data/Example1.txt'
        filename = 'Example1.txt'
        urllib.request.urlretrieve(url, filename)
```

```
Out[1]: ('Example1.txt', <http.client.HTTPMessage at 0x1f1b9effca50>)
```

```
In [2]: # Download Example file


        !wget -O /resources/data/Example1.txt https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0101EN-SkillsNetwork/labs/Module%204/data/Example1.txt
```

```
'wget' is not recognized as an internal or external command,
operable program or batch file.
```

Reading Text Files

One way to read or write a file in Python is to use the built-in `open` function. The `open` function provides a **File object** that contains the methods and attributes you need in order to read, save, and manipulate the file. In this notebook, we will only cover **.txt** files. The first parameter you need is the file path and the file name. An example is shown as follows:

The mode argument is optional and the default value is **r**. In this notebook we only cover two modes:

- **r** Read mode for reading files
- **w** Write mode for writing files

For the next example, we will use the text file **Example1.txt**. The file is shown as follows:

This is line 1
This is line 2
This is line 3

We read the file:

```
In [3]: # Read the Example1.txt

        example1 = "Example1.txt"
        file1 = open(example1, "r")
```

We can view the attributes of the file.

The name of the file:

```
In [4]: # Print the path of file

        file1.name
```

```
Out[4]: 'Example1.txt'
```

The mode the file object is in:

```
In [5]: # Print the mode of file, either 'r' or 'w'

        file1.mode
```

```
Out[5]: 'r'
```

We can read the file and assign it to a variable :

```
In [6]: # Read the file

        FileContent = file1.read()
        FileContent
```

```
Out[6]: 'This is line 1 \nThis is line 2\nThis is line 3'
```

The **/n** means that there is a new line.

We can print the file:

```
In [7]: # Print the file with '\n' as a new line

        print(FileContent)
```

```
This is line 1
This is line 2
This is line 3
```

The file is of type string:

```
In [8]: # Type of file content

        type(FileContent)
```

```
Out[8]: str
```

It is very important that the file is closed in the end. This frees up resources and ensures consistency across different python versions.

```
In [9]: # Close file after finish

        file1.close()
```

A Better Way to Open a File

Using the `with` statement is better practice, it automatically closes the file even if the code encounters an exception. The code will run everything in the indent block then close the file object.

```
In [10]: # Open file using with

         with open(example1, "r") as file1:
             FileContent = file1.read()
             print(FileContent)
```

```
This is line 1
This is line 2
This is line 3
```

The file object is closed, you can verify it by running the following cell:

```
In [11]: # Verify if the file is closed

         file1.closed
```

```
Out[11]: True
```

We can see the info in the file:

```
In [12]: # See the content of file

         print(FileContent)
```

```
This is line 1
This is line 2
This is line 3
```

The syntax is a little confusing as the file object is after the `as` statement. We also don't explicitly close the file. Therefore we summarize the steps in a figure:

We don't have to read the entire file, for example, we can read the first 4 characters by entering three as a parameter to the method **.read()**:

```
In [13]: # Read first four characters

         with open(example1, "r") as file1:
             print(file1.read(4))
```

```
This
```

Once the method `.read(4)` is called the first 4 characters are called. If we call the method again, the next 4 characters are called. The output for the following cell will demonstrate the process for different inputs to the method `.read()`:

```
In [14]: # Read certain amount of characters

         with open(example1, "r") as file1:
             print(file1.read(4))
             print(file1.read(4))
             print(file1.read(7))
             print(file1.read(15))
```

```
This
 is
line 1

 This is line 2
```

The process is illustrated in the below figure, and each color represents the part of the file read after the method `.read()` is called:

Here is an example using the same file, but instead we read 16, 5, and then 9 characters at a time:

```
In [15]: # Read certain amount of characters

         with open(example1, "r") as file1:
             print(file1.read(16))
             print(file1.read(5))
             print(file1.read(9))
```

```
This is line 1

This
is
line 2
```

We can also read one line of the file at a time using the method `readline()`:

```
In [16]: # Read one line

         with open(example1, "r") as file1:
             print("first line: " + file1.readline())
```

```
first line: This is line 1
```

We can also pass an argument to `readline()` to specify the number of charecters we want to read. However, unlike `read()`, `readline()` can only read one line at most.

```
In [17]: with open(example1, "r") as file1:
             print(file1.readline(20)) # Does not read past the end of line
             print(file1.read(20)) # Returns the next 20 chars
```

```
This is line 1

This is line 2
This
```

We can use a loop to iterate through each line:

```
In [18]: # Iterate through the lines

         with open(example1,"r") as file1:
                 i = 0;
                 for line in file1:
                     print("Iteration", str(i), ": ", line)
                     i = i + 1
```

```
Iteration 0 :  This is line 1

Iteration 1 :  This is line 2

Iteration 2 :  This is line 3
```

```
In [19]: # Iterate through the lines

         with open(example1,"r") as file1:
                 i = 0;
                 for line in file1:
                     print("Iteration", str(i + 1), ": ", line)
                     i = i + 1
```

```
Iteration 1 :  This is line 1

Iteration 2 :  This is line 2

Iteration 3 :  This is line 3
```

We can use the method `readlines()` to save the text file to a list:

```
In [20]: # Read all lines and save as a list

         with open(example1, "r") as file1:
             FileasList = file1.readlines()
```

Each element of the list corresponds to a line of text:

```
In [21]: # Print the first line

         FileasList[0]
```

```
Out[21]: 'This is line 1 \n'
```

Print the second line

FileasList[1]

```
In [22]: # Print the third line

         FileasList[2]
```

```
Out[22]: 'This is line 3'
```

The last exercise!

Congratulations, you have completed your first lesson and hands-on lab in Python. However, there is one more thing you need to do. The Data Science community encourages sharing work. The best way to share and showcase your work is to share it on GitHub. By sharing your notebook on GitHub you are not only building your reputation with fellow data scientists, but you can also show it off when applying for a job. Even though this was your first piece of work, it is never too early to start building good habits. So, please read and follow this article to learn how to share your work.

Author

Joseph Santarcangelo

Other contributors

Mavis Zhou

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-09-30	1.3	Malika	Deleted exercise "Weather Data"
2020-09-30	1.2	Malika Singla	Weather Data dataset link added
2020-09-30	1.1	Arjun Swani	Added exercise "Weather Data"
2020-09-30	1.0	Arjun Swani	Added blurbs about closing files and read() vs readline()
2020-08-26	0.2	Lavanya	Moved lab to course repo in GitLab

© IBM Corporation 2020. All rights reserved.