PY0101EN-4-1-ReadFile

November 17, 2020

```
<a href="https://cocl.us/PY0101EN_edx_add_top">
     <img src="https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass
</a>
```

Reading Files Python

Welcome! This notebook will teach you about reading the text file in the Python Programming Language. By the end of this lab, you'll know how to read text files.

Table of Contents

```
<ul>
    <li><a href="download">Download Data</a></li>
    <li><a href="read">Reading Text Files</a></li>
    <li><a href="better">A Better Way to Open a File</a></li>
</ul>
<p>
    Estimated time needed: <strong>40 min</strong>
</p>
```

Download Data

[1]:
```
# Download Example file

!wget -O /resources/data/Example1.txt https://s3-api.us-geo.objectstorage.
 ↪softlayer.net/cf-courses-data/CognitiveClass/PY0101EN/labs/example1.txt
```

```
--2020-06-04 01:34:09--  https://s3-api.us-geo.objectstorage.softlayer.net/cf-
courses-data/CognitiveClass/PY0101EN/labs/example1.txt
Resolving s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-
geo.objectstorage.softlayer.net)… 67.228.254.196
Connecting to s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-
geo.objectstorage.softlayer.net)|67.228.254.196|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 45 [text/plain]
Saving to: '/resources/data/Example1.txt'

/resources/data/Exa 100%[===================>]      45  --.-KB/s    in 0s

2020-06-04 01:34:09 (17.4 MB/s) - '/resources/data/Example1.txt' saved [45/45]
```

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 follow:

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 follow:

We read the file:

[2]:
```
# Read the Example1.txt

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

We can view the attributes of the file.

The name of the file:

[3]:
```
# Print the path of file

file1.name
```

[3]:
```
'/resources/data/Example1.txt'
```

The mode the file object is in:

[4]:
```
# Print the mode of file, either 'r' or 'w'

file1.mode
```

[4]:
```
'r'
```

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

[5]:
```
# Read the file

FileContent = file1.read()
FileContent
```

[5]:
```
'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:

[6]:
```
# 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:

[7]:
```
# Type of file content

type(FileContent)
```

[7]:
```
str
```

We must close the file object:

[8]:
```
# 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.

[9]:
```
# 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:

[10]:
```
# Verify if the file is closed

file1.closed
```

[10]:
```
True
```

We can see the info in the file:

[11]:
```
# 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()**:

[12]:
```
# 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():

[13]:
```
# 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:

[14]:
```
# 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():

[15]:
```
# Read one line

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

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

We can use a loop to iterate through each line:

[16]:
```
# 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
```

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

[17]:
```
# 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:

[18]:
```
# Print the first line

FileasList[0]
```

[18]:
```
'This is line 1 \n'
```

[19]:
```
# Print the second line

FileasList[1]
```

[19]:
```
'This is line 2\n'
```

[20]:
```
# Print the third line

```

```
FileasList[2]
```

[20]: '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.

Get IBM Watson Studio free of charge!

```
<p><a href="https://cocl.us/PY0101EN_edx_add_bbottom"><img src="https://s3-api.us-geo.objectst
```

About the Authors:

Joseph Santarcangelo is a Data Scientist at IBM, and holds a PhD in Electrical Engineering. His research focused on using Machine Learning, Signal Processing, and Computer Vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Other contributors: Mavis Zhou