Write and Save Files in Python

Estimated time needed: **25** minutes

Objectives

After completing this lab you will be able to:

- Write to files using Python libraries

Table of Contents

Writing Files

We can open a file object using the method `write()` to save the text file to a list. To write the mode, argument must be set to write **w**. Let's write a file **Example2.txt** with the line: **"This is line A"**

```python
# Write line to file
exmp2 = 'Example2.txt'
with open(exmp2, 'w') as writefile:
    writefile.write("This is line A")
```

We can read the file to see if it worked:

```python
# Read file

with open(exmp2, 'r') as testwritefile:
    print(testwritefile.read())
```

```
This is line A
```

We can write multiple lines:

```python
# Write lines to file

with open(exmp2, 'w') as writefile:
    writefile.write("This is line A\n")
    writefile.write("This is line B\n")
```

The method `.write()` works similar to the method `.readline()`, except instead of reading a new line it writes a new line. The process is illustrated in the figure , the different colour coding of the grid represents a new line added to the file after each method call.

This is line A 1)writefile.write("This is line A\n")

2)writefile.write("This is line B\n")

You can check the file to see if your results are correct

```python
# Check whether write to file

with open(exmp2, 'r') as testwritefile:
    print(testwritefile.read())
```

```
This is line A
This is line B
```

We write a list to a **.txt** file as follows:

```python
# Sample list of text

Lines = ["This is line A\n", "This is line B\n", "This is line C\n"]
Lines
```

```
['This is line A\n', 'This is line B\n', 'This is line C\n']
```

```python
# Write the strings in the list to text file

with open('Example2.txt', 'w') as writefile:
    for line in Lines:
        print(line)
        writefile.write(line)
```

```
This is line A

This is line B

This is line C
```

We can verify the file is written by reading it and printing out the values:

```python
# Verify if writing to file is successfully executed

with open('Example2.txt', 'r') as testwritefile:
    print(testwritefile.read())
```

```
This is line A
This is line B
This is line C
```

However, note that setting the mode to **w** overwrites all the existing data in the file.

```python
with open('Example2.txt', 'w') as writefile:
    writefile.write("Overwrite\n")
with open('Example2.txt', 'r') as testwritefile:
    print(testwritefile.read())
```

```
Overwrite
```

Appending Files

We can write to files without losing any of the existing data as follows by setting the mode argument to append **a**. you can append a new line as follows:

```python
# Write a new line to text file

with open('Example2.txt', 'a') as testwritefile:
    testwritefile.write("This is line C\n")
    testwritefile.write("This is line D\n")
    testwritefile.write("This is line E\n")
```

You can verify the file has changed by running the following cell:

```python
# Verify if the new line is in the text file

with open('Example2.txt', 'r') as testwritefile:
    print(testwritefile.read())
```

```
Overwrite
This is line C
This is line D
This is line E
```

Additional modes

It's fairly ineffecient to open the file in **a** or **w** and then reopening it in **r** to read any lines. Luckily we can access the file in the following modes:

- **r+** : Reading and writing. Cannot truncate the file.
- **w+** : Writing and reading. Truncates the file.
- **a+** : Appending and Reading. Creates a new file, if none exists. You dont have to dwell on the specifics of each mode for this lab.

Let's try out the **a+** mode:

```python
with open('Example2.txt', 'a+') as testwritefile:
    testwritefile.write("This is line E\n")
    print(testwritefile.read())
```

There were no errors but `read()` also did not output anything. This is because of our location in the file.

Most of the file methods we've looked at work in a certain location in the file. `.write()` writes at a certain location in the file. `.read()` reads at a certain location in the file and so on. You can think of this as moving your pointer around in the notepad to make changes at specific location.

Opening the file in **w** is akin to opening the .txt file, moving your cursor to the beginning of the text file, writing new text and deleting everything that follows. Whereas opening the file in **a** is similiar to opening the .txt file, moving your cursor to the very end and then adding the new pieces of text.
It is often very useful to know where the 'cursor' is in a file and be able to control it. The following methods allow us to do precisely this -

- `.tell()` - returns the current position in bytes
- `.seek(offset,from)` - changes the position by 'offset' bytes with respect to 'from'. From can take the value of 0,1,2 corresponding to beginning, relative to current position and end

Now lets revisit **a+**

```python
with open('Example2.txt', 'a+') as testwritefile:
    print("Initial Location: {}".format(testwritefile.tell()))

    data = testwritefile.read()
    if (not data):  #empty strings return false in python
            print('Read nothing')
    else:
            print(testwritefile.read())

    testwritefile.seek(0,0) # move 0 bytes from beginning.

    print("\nNew Location : {}".format(testwritefile.tell()))
    data = testwritefile.read()
    if (not data):
            print('Read nothing')
    else:
            print(data)

    print("Location after read: {}".format(testwritefile.tell()) )
```

```
Initial Location: 75
Read nothing

New Location : 0
Overwrite
This is line C
This is line D
This is line E
This is line E

Location after read: 75
```

Finally, a note on the difference between **w+** and **r+**. Both of these modes allow access to read and write methods, However opening a file in **w+** overwrites it and deletes all existing data.
To work with a file on existing data, use **r+** and **a+**. While using **r+**, it can be useful to add a `.truncate()` method at the end of your data. This will reduce the file to your data and delete everything that follows.
In the following code block, Run the code as it is first and then run it with the `.truncate()`.

```python
with open('Example2.txt', 'r+') as testwritefile:
    data = testwritefile.readlines()
    testwritefile.seek(0,0) #write at beginning of file

    testwritefile.write("Line 1" + "\n")
    testwritefile.write("Line 2" + "\n")
    testwritefile.write("Line 3" + "\n")
    testwritefile.write("finished\n")
    #Uncomment the line below
    #testwritefile.truncate()
    testwritefile.seek(0,0)
    print(testwritefile.read())
```

```
Line 1
Line 2
Line 3
finished
 line D
This is line E
This is line E
```

Copy a File

Let's copy the file **Example2.txt** to the file **Example3.txt**:

```python
# Copy file to another

with open('Example2.txt','r') as readfile:
    with open('Example3.txt','w') as writefile:
          for line in readfile:
                writefile.write(line)
```

We can read the file to see if everything works:

```python
# Verify if the copy is successfully executed

with open('Example3.txt','r') as testwritefile:
    print(testwritefile.read())
```

```
Line 1
Line 2
Line 3
finished
 line D
This is line E
This is line E
```

After reading files, we can also write data into files and save them in different file formats like **.txt, .csv, .xls (for excel files) etc**. You will come across these in further examples

Now go to the directory to ensure the **.txt** file exists and contains the summary data that we wrote.

Exercise

Your local university's Raptors fan club maintains a register of its active members on a .txt document. Every month they update the file by removing the members who are not active. You have been tasked with automating this with your python skills.
Given the file currentMem, Remove each member with a 'no' in their inactive coloumn. Keep track of each of the removed members and append them to the exMem file. Make sure the format of the original files in preserved. (*Hint: Do this by reading/writing whole lines and ensuring the header remains*)
Run the code block below prior to starting the exercise. The skeleton code has been provided for you, Edit only the cleanFiles function.

```python
#Run this prior to starting the exercise
from random import randint as rnd

memReg = 'members.txt'
exReg = 'inactive.txt'
fee =('yes','no')

def genFiles(current,old):
    with open(current,'w+') as writefile:
        writefile.write('Membership No  Date Joined  Active  \n')
        data = "{:^13}  {:<11}  {:<6}\n"

        for rowno in range(20):
            date = str(rnd(2015,2020))+ '-' + str(rnd(1,12))+'-'+str(rnd(1,25))
            writefile.write(data.format(rnd(10000,99999),date,fee[rnd(0,1)]))


    with open(old,'w+') as writefile:
        writefile.write('Membership No  Date Joined  Active  \n')
        data = "{:^13}  {:<11}  {:<6}\n"
        for rowno in range(3):
            date = str(rnd(2015,2020))+ '-' + str(rnd(1,12))+'-'+str(rnd(1,25))
            writefile.write(data.format(rnd(10000,99999),date,fee[1]))


genFiles(memReg,exReg)
```

Start your solution below:

```python
def cleanFiles(currentMem,exMem):
    '''
    currentMem: File containing list of current members
    exMem: File containing list of old members

    Removes all rows from currentMem containing 'no' and appends them to exMem
    '''

    pass


# Code to help you see the files
# Leave as is
memReg = 'members.txt'
exReg = 'inactive.txt'
cleanFiles(memReg,exReg)


headers = "Membership No  Date Joined  Active  \n"
with open(memReg,'r') as readFile:
    print("Active Members: \n\n")
    print(readFile.read())

with open(exReg,'r') as readFile:
    print("Inactive Members: \n\n")
    print(readFile.read())
```

```
Active Members:


Membership No  Date Joined  Active
    42183      2017-3-19    yes
    85235      2017-7-22    no
    52777      2015-9-15    yes
    36244      2017-2-17    yes
    17359      2015-4-7     no
    70202      2017-7-20    yes
    59536      2016-5-18    yes
    81970      2020-8-11    yes
    12010      2019-8-22    yes
    88569      2016-6-20    yes
    18841      2020-1-14    yes
    45236      2018-4-14    no
    93283      2020-8-3     yes
    57413      2015-7-22    no
    28583      2020-4-20    no
    95687      2017-10-3    no
    92561      2020-8-12    yes
    27176      2016-5-15    yes
    67052      2015-11-24   yes
    13981      2020-8-20    no

Inactive Members:


Membership No  Date Joined  Active
    55856      2016-6-3     no
    38833      2020-5-4     no
    46503      2017-8-13    no
```

Run the following to verify your code:

```python
def testMsg(passed):
    if passed:
       return 'Test Passed'
    else :
       return 'Test Failed'

testWrite = "testWrite.txt"
testAppend = "testAppend.txt"
passed = True

genFiles(testWrite,testAppend)

with open(testWrite,'r') as file:
    ogWrite = file.readlines()

with open(testAppend,'r') as file:
    ogAppend = file.readlines()

try:
    cleanFiles(testWrite,testAppend)
except:
    print('Error')

with open(testWrite,'r') as file:
    clWrite = file.readlines()

with open(testAppend,'r') as file:
    clAppend = file.readlines()

# checking if total no of rows is same, including headers

if (len(ogWrite) + len(ogAppend) != len(clWrite) + len(clAppend)):
    print("The number of rows do not add up. Make sure your final files have the same header and format.")
    passed = False

for line in clWrite:
    if  'no' in line:
        passed = False
        print("Inactive members in file")
        break
    else:
        if line not in ogWrite:
            print("Data in file does not match original file")
            passed = False
print ("{}".format(testMsg(passed)))
```

```
Inactive members in file
Test Failed
```

```python
def cleanFiles(currentMem,exMem):
    with open(currentMem,'r+') as writeFile:
        with open(exMem,'a+') as appendFile:
            #get the data
            writeFile.seek(0)
            members = writeFile.readlines()
            #remove header
            header = members[0]
            members.pop(0)

            inactive = [member for member in members if ('no' in member)]
            '''
            The above is the same as

            for member in active:
            if 'no' in member:
                inactive.append(member)
            '''
            #go to the beginning of the write file
            writeFile.seek(0)
            writeFile.write(header)
            for member in members:
                if (member in inactive):
                    appendFile.write(member)
                else:
                    writeFile.write(member)
            writeFile.truncate()

memReg = 'members.txt'
exReg = 'inactive.txt'
cleanFiles(memReg,exReg)

# code to help you see the files

headers = "Membership No  Date Joined  Active  \n"

with open(memReg,'r') as readFile:
    print("Active Members: \n\n")
    print(readFile.read())

with open(exReg,'r') as readFile:
    print("Inactive Members: \n\n")
    print(readFile.read())
```

```
Active Members:


Membership No  Date Joined  Active
    42183      2017-3-19    yes
    52777      2015-9-15    yes
    36244      2017-2-17    yes
    70202      2017-7-20    yes
    59536      2016-5-18    yes
    81970      2020-8-11    yes
    12010      2019-8-22    yes
    88569      2016-6-20    yes
    18841      2020-1-14    yes
    93283      2020-8-3     yes
    92561      2020-8-12    yes
    27176      2016-5-15    yes
    67052      2015-11-24   yes

Inactive Members:


Membership No  Date Joined  Active
    55856      2016-6-3     no
    38833      2020-5-4     no
    46503      2017-8-13    no
    85235      2017-7-22    no
    17359      2015-4-7     no
    45236      2018-4-14    no
    57413      2015-7-22    no
    28583      2020-4-20    no
    95687      2017-10-3    no
    13981      2020-8-20    no
```

▸ Click here for the solution

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
2020-10-16	1.3	Arjun Swani	Added exercise
2020-10-16	1.2	Arjun Swani	Added section additional file modes
2020-10-16	1.1	Arjun Swani	Made append a different section
2020-08-28	0.2	Lavanya	Moved lab to course repo in GitLab

© IBM Corporation 2020. All rights reserved.