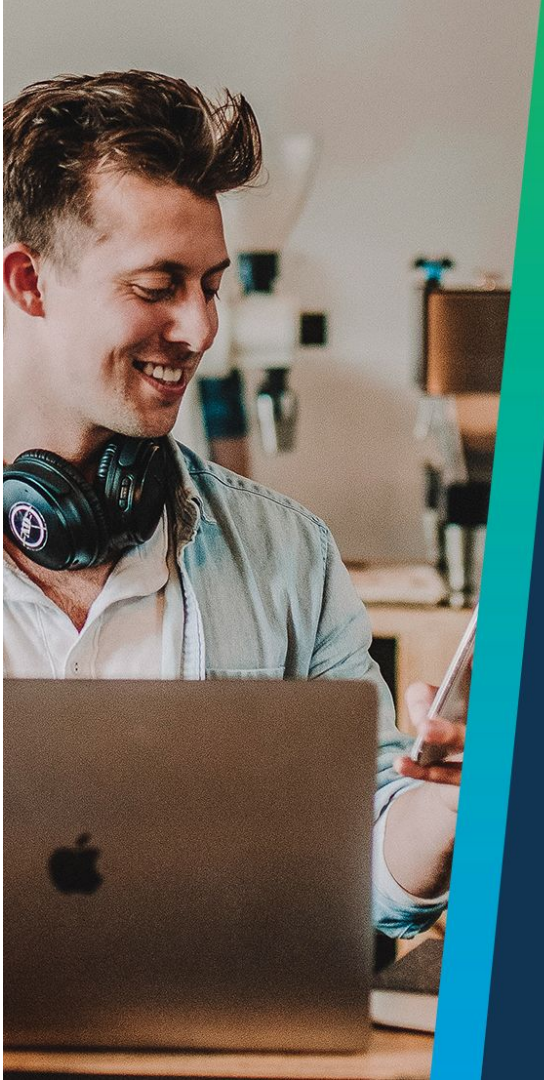


Hyperiondev

Working with External Data Sources - Output

Lecture – Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- No question is daft or silly - **ask them!**
- There are Q/A sessions midway and at the end of the session, should you wish to ask any follow-up questions.
- You can also submit questions here: http://hyperiondev.com/sbc4-se-questions
- For all non-academic questions, please submit a query: www.hyperiondev.com/support
- Report a safeguarding incident: http://hyperiondev.com/safeguardreporting
- We would love your feedback on lectures: https://hyperionde.wufoo.com/forms/zsgv4m40ui4i0g/

Objectives

1. Learn how to create external files with Python
    a. Create a text file if it does not already exists
    b. Specify the access mode eg. ‘w’, ‘a’ to write contents in a text file.
2. Learn how to write data to files
    a. Using the .write() function
    b. Writing user input to text files

Recap on File input - Reading from a text file

File I/O

- ★ File I/O stands for file input/output
- ★ It is a process that reads data from an external file on the computer or outputs to another file.
- ★ Python has a built-in file type, which is the complex data type.
- ★ This means that Python can create variables of type “file”.

Opening a File

- ★ To read from a file, we must first open it.
- ★ To open a file, we use Python's built-in open() function, which creates what is known as a file object.

- ★ To utilize the file object's data, we store the file object in a variable.

- ★ Once we are done, we then close the file.

Opening Files

- To use a file in our program, we store the file object in a variable as such :
  - **file = open(file_name , access_mode)**

- **Access mode** : what the user can do when the file has been opened, such as reading (r), writing (w), or reading and writing (r+).

Opening Example

```
# To make opening the file easier,
# best to keep the text file in the
# same location as your Python file
file_name = 'input.txt'

file = open(file_name,'r')

# File is now being read by Python
```

Reading Files

- ★ Files are opened in Python with the open() function. We know that open() will return a file object.
- ★ To then properly read the object, we will need to use the read method.
- ★ There are three methods : .read() , .readline(), .readlines()

Reading examples

```
# Use one of the read methods to read the contents
lines = file.read() # reads and stores all data as a string type
# Or
lines = file.readline() # reads and stores only the first line
# Or
lines = file.readlines() # reads and stores all data in a list

# Call the print function on the 'lines' variable to display contents
print(lines)
# Remember to close the file
file.close()
```

Writing to Files

- ★ Often, we will want to write data to a new file.
- ★ Usually after we have done a lot of computations or data processing and we would like to save the work and come back to it at another point.
- ★ Writing to a file has a simple multi-step process.

Prepping the file

- ★ We already know how to open a file and store the file object in a variable.

- ★ Now the main difference between Input and Output is the access mode is different
  - Instead of reading from the file, we are now writing to the file using modes w , w+ , a
- ★ What comes next is then actually writing to the file, which we will take a look at now.

Writing Example 1

```
with open('output.txt','w') as file:

    file.write("Mankind knew, that they cannot change society.\n")
    file.write("So instead of reflecting on themselves. \n")
    file.write("They blamed the beast")

# The .write() function, will write any data we provide
# within the parentheses to our file
# and since we are using a with/as block
# we don't need to close the file with .close()
```

Writing Example 2

```
# An alternative method to write contents to a file
file_name = 'output.txt'

file = open(file_name,'w')

file.write("Mankind knew, that they cannot change society.\n")
file.write("So instead of reflecting on themselves. \n")
file.write("They blamed the beast")

file.close()
```

Things to Note

- ★ Remember that when the file is reopened and new data is written to the file, the previous data is then overwritten.
- ★ You can preserve the previous data by using the append (a) access mode. This will simply append the new data to the end of the file, instead of overwriting.
- ★ Always remember to close your file when you are done using it.

Writing Example 3

```
# Using the 'a' access mode will prevent data to be over written
# Open the file again
file_name = 'output.txt'  # This is the original text file

file = open(file_name,'a')

file.write("This is the new text")

file.close()
```

Writing Example 3 continued...

```
# Open and read the contents in the text file
file_name = 'output.txt'

file = open(file_name, 'r')

lines = file.read()

print(lines)

file.close()
```

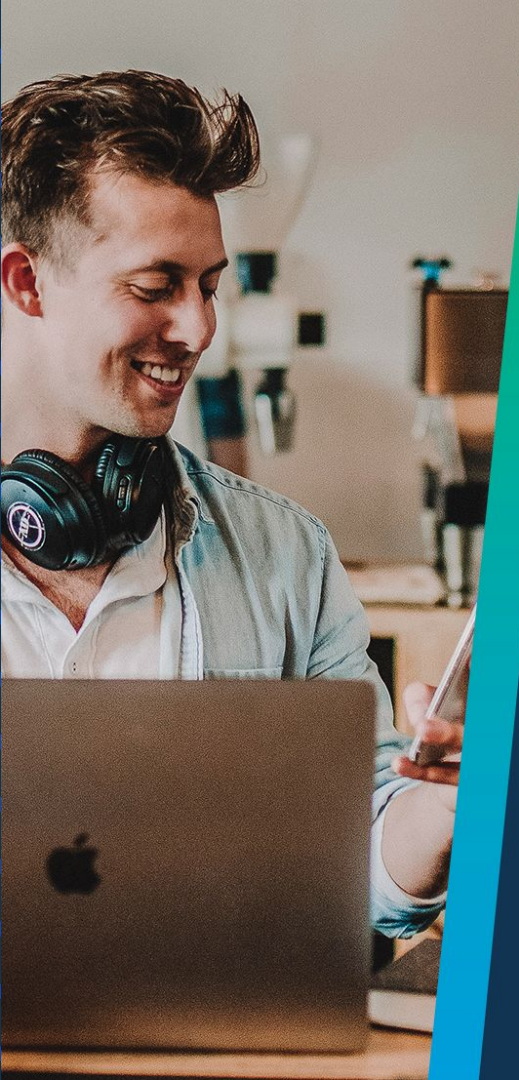
[output]

```
Mankind knew, that they cannot change society.
So instead of reflecting on themselves.
They blamed the beast
This is the new text
```

Hyperiondev

Q & A Section

Please use this time to ask any questions relating to the topic explained, should you have any



Hyperiondev

Thank you for joining us