Data Scientist
Codecademy
Python Fundamentals
Files

Reading a File

Computers use file systems to store and retrieve data. Each file is an individual container of related information.

Reading Text files from Python:

```
with open('text_document.txt') as doc:
  content = doc.read()
print(content)
```

Iterating Through Lines

.readLines() function helps to read a text file line by line instead of having the whole file as a single string.

```
with open('file.txt') as doc:
  for line in doc.readLines():
      print(line)
```

Reading a Line

.readline() will read only a single line at a time. If whole document is read line by line in this way subsequent calls to .readline() will not throw an error but will start running an empty string "".

```
with open('file.txt') as doc:
  line1 = doc.readline()
  line2 = doc.readline()
```

Writing a File

With python we can create files of our own. open() function that we use to open a file to read when given another argument ‘w’ along with file name it opens a file to write to.

```
with open(‘file.txt’,’w’) as doc:
  doc.write(“message for file.txt”)
```

‘w’ - indicates to open the file in write-mode

‘r’ - indicates to open the file in read-mode it is default mode

If a file already exists by the name file.txt then its content will be completely overwritten once this script is run.

Appending to a File

Since opening a file in write mode overwrites the content of the existing file, we can use:

‘a’ - indicates to open the file in append-mode

In this mode new content is added to file without affecting the existing content.

```
with open(‘file.txt’,’a’) as doc:
  doc.write(“message to be appended”)
```

What’s with “with”?

with keyword invokes something called a *context manager* for the file that we’re calling open() on. This context manager takes care of opening the file when we call open() and then closing the file after we leave the indented block.

All variables we create in a python script are python objects and python knows how to clean them up when it’s done with them, but files exist outside python script, so we need to tell python when we’re done with them so that it can close the connection to that file. Leaving a file connection open unnecessarily can affect performance or impact other programs on your computer that might be trying to access that file.

The with syntax replaces older ways to access files where we need to call .close() on the file object manually. We can still open up a file and append to it with old syntax, as long as we remember to close the file connection afterwards.

What is a CSV file?

Text files aren't the only thing that Python can read, but they're the only thing that we don't need any additional parsing library to understand. CSV files are an example of a text file that impose a structure to their data. CSV stands for *Comma-Separated Values* and CSV files are usually the way that data from spreadsheet softwares is exported into a portable format.

Reading a CSV file

CSV files are in text format with different cells in a row separated by commas.

In python we can convert that data into a dictionary using the csv library's DictReader object.

```
import csv

list_of_emails = []

with open('users.csv',newline='') as users_csv:

  user_reader = csv.DictReader(users_csv)

  for row in user_reader:

      list_of_emails.append(row['Email'])
```

csv.DictReader(users_csv) converts the lines of CSV file to dictionaries. The keys of the dictionary are, by default the entries in the first line of CSV file. users.csv file calls the third field in CSV "Email", we can use that as the key in each row of our DictReader. By accessing the 'Email' key of each row we can grab the value corresponding to email field and append it to our list.

The newline='' keyword argument is passed to open() function to account for possibility of \n characters in data.

Reading Different Types of CSV Files

As other separators grew in popularity everyone realized that creating a new .[a-z]sv file format for every value- separating character used is not sustainable.

So we call all files with a list of different values a CSV file and then use different delimiters(comma, tab, etc) to indicate where the different values start and stop.

```
with open('addresses.csv',newline='') as addresses_csv:
  address_reader = csv.DictReader(addresses_csv,
delimiter=';')
  for row in address_reader:
    print(row['Address'])
```

Writing a CSV File

```
big_list = [{..},{..},...]


import csv
with open('output.csv','w') as output_csv:
  fields = ['name','user_id','is_admin']
  output_writer = csv.DictWriter(output_csv,fieldnames=fields)
  output_writer.writeheader()
  for item in big_list:
    output_writer.writerow(item)
```

Reading a JSON File

JSON or JavaScript Object Notation is a file format inspired by the programming language JS. JSON's format is endearingly similar to python dictionary syntax, and so JSON files might be easy to read from a python developer standpoint.

```
import json
with open('file.json') as f_json:
```

```
  f_data = json.load(f_json)
  print(f_data['key_name'])
```

Writing a JSON File

```
dict = {....}
import json
with open('output.json','w') as json_file:
  json.dump(dict,json_file)
```