ED5340 - Data Science: Theory and Practise

L12 - File Input and Output

Ramanathan Muthuganapathy (https://ed.iitm.ac.in/~raman)
Course web page: https://ed.iitm.ac.in/~raman/datascience.html
Moodle page: Available at https://courses.iitm.ac.in/

File I/O

Sequence of operations

- Open a file
- Read / Write data to it
- Close the file

File modes

L12_File1.py

- f = open(‘filename’, ‘mode’) e.g. f = open(‘messages.txt’, ‘w’)
  - ‘w’ - opens the file for writing in text mode
  - ‘r’ - opens the file for reading in text mode
  - ‘a’ - opens the file for appending in text mode
  - ‘wb’ - opens the file for writing in binary mode
  - ‘rb’ - opens the file for reading in binary mode
  - ‘ab’ - opens the file for appending in binary mode
  - Check for other available modes

Strings
L12_File1.py

- Note that both reading and writing are in the form of strings

```
msg1 = 'This is message 1 \n’

f = open('messages.txt', 'w') #Opens a file for writing

f.write(msg1)

f.close()
```

Strings

L12_File1.py

- Note that both reading and writing are in the form of strings

```
f = open('messages.txt', 'r')  #Opens a file for reading

data = f.read() #Reads ALL lines into data

print(data)

f.close()
```

‘with’ open

L12_File1.py

- Open but automatically closes the file.

```
with open('messages.txt', 'r') as f: #with closes the file automatically
    data = f.read()
```

f.seek - moving within a file

- f.seek(offset, reference)
- ‘reference’ can take 0 (beginning of a file), 1 (current position) and 2 (end of file)
- f.seek(12, 0) - moves to 12th position from bof.
- f.seek(-15, 2) - moves 15 positions to the left from eof.
- f.seek(0, 2)

File and Directory Operations

L12_FileDirs.py

- File operations - Creation, deletion, renaming etc.
- Directory operations - creation (recursive), deletion, listing etc.
- Path operations - absolute and relative paths, splitting / joining etc..
- . - current dir, .. - parent of current directory.

HW: Use file input / output for complex and matrix classes.

Comma separated values (csv)

L12_CSV.py

- Format is the most common import and export format for spreadsheets and databases

name1,name2, name3
ram1,ram2, ram3
cam1,cam2, cam3

Reading / Writing other datatypes

- Currently, all are strings!
- Tuple, dictionaries etc
- Using JSON module
    - JSON - JavaScript Object Notation

JSON Format

- Web development, configuration / settings
- JSON format
    - sequence of key-value pairs surrounded by curly brackets
    - Each key is mapped to a particular value using this format.
    - Key-value pairs are separated by a comma. Only the last pair is not followed by a comma
        - **Keys** must be strings.
        - **Values** can be either a string, a number, an array, a boolean value or a JSON object

```
{
    "name": "Raman",
    "languages": ["C", "C++"]
}
```

Some rules for JSON

- Always choose meaningful names.
- Array types should have plural key names. All other key names should be singular. For example: use `"orders"` instead of `"order"` if the corresponding value is an array.
- There should be no comments in JSON objects.

Relation to python

- JSON and Dictionaries might look very similar
- JSON is a file format used to represent and store data.
- Python Dictionary is a data structure (object) that is kept in memory.
- We can’t read the JSON files directly (as entire file is a single string, and individual key-value pairs cannot be accessed individually).
    - A dictionary can be created using Key-value pair
    - NOTE: JSON is the string rep and dictionaries are data structures in python

JSON module

- Python string in JSON format
    - per = '{"name": "Ram", "languages": ["Python", “C++"]}'
- To convert this to a python dictionary
    - per_dct = json.loads(per) #per_dct is a dictionary
- json.loads(per) — JSON string to a dictionary

Reverse - Python object to JSON format

e.g. dictionary to JSON format - L13_json1.py and L13_json2.py

- dct = {'name': 'Ram', 'languages': ['Python', 'C++']}
- str1 = json.dumps(dct) #This function returns a string
- print(str1)

Python object to JSON equivalent

Python	JSON Equivalent
`dict`	object
`list`, `tuple`	array
`str`	string
`int`, `float`, `int`	number
`TRUE`	true
`FALSE`	false
`None`	null

For User-defined datatypes

L12_Complex_JSON.py

- Complex object into JSON, encode_complex( ) in json.dump( )
- For load( ), decode_complex( ) through the object_hook parameter.