File IO

9.23.24

Learning Objectives

- How to restructure directories using Python
- Reading/Writing to files
- Data collection in semi-structured databases

File Jockeying with os

- Builtin os library

Major:

Make Directories	**os.makedirs()**
Folder exists?	**os.path.exists()**
Change Working Directory	**os.chdir()**
Contents	**os.listdir()**
Remove file	**os.remove()**
Remove empty directory	**os.rmdir()**
File size (bytes)	**os.path.getsize()**

os continued...

os.system('command_as_string')

- Directly pass commands to CMD (windows) or terminal (UNIX).
- If you already know CMD/BASH, this is all you need...

os.path.join('p1', 'p2', 'p3'...)

- Connect file-paths using \ if on Windows, and / if on UNIX
- Good for platform-indep. file-wrangling

Practice Together: Cleaning a Directory

- I want to get rid of empty files/folders in my directory
- The directory contains subfolders with txt files

1. Delete empty files (those with size 0) & folders
2. Make a log of what was deleted

Make Directories	**os.makedirs()**
Folder exists?	**os.path.exists()**
Change Working Directory	**os.chdir()**
Contents	**os.listdir()**
Remove file	**os.remove()**
Remove empty directory	**os.rmdir()**
File size (bytes)	**os.path.getsize()**

Copying Files with shutil

- shutil.copy('source', 'destination’): copies data & permissions
- shutil.copy2(--,--): also tries to keep metadata
- shutil.move(--,--): moves file

- shutil.copytree(--,--): copy full directory tree
- shutil.rmtree(--): delete full directory tree

Practice: Condensing a Directory

- Now let’s combine all those .txt files into the base folder.

use:

shutil.copy('source', 'destination’)

shutil.rmtree('source')

Opening files

with open(‘filename’, **‘access_mode’**) **as** file:

.....

....

- Use the with structure to ensure files are auto-closed

- Read (r): requires file to exist
- Write (w): will create new file if none exists
- Append (a): only add new lines
- Write & read: w+
- For text: optionally add ‘t’: ‘rt’, ‘wt’
- For binary: add ‘b’: ‘rb’, ‘wb’, ‘w+b’

Reading from files

file.**read**(n_bytes)

n bytes-at-a-time

In most cases # bytes = # characters (UTF-8)

file.**readline**()

One line at a time

file.**readlines**()

All lines as a list

Reading Position

- Files are read like a story—from start to finish
- Read methods go through the file in order (bytes or lines)
- You can loop across lines:

```
with open(fName,'r') as file:
    for lines in file:
        print(lines)
```

- Current reading position in bytes: file.**tell**()
- Change reading position: file.**seek**(n_bytes, offset)
  - Offset=0: from start, 1: from current pos

Writing to a file

- Like reading, writing only accepts strings

- file.**write**(): single string

  Add '\n' to terminate lines

- file.**writelines**(foo): same as file.write(''.join(foo))
  - Repeated calls to file.write for an iterator: not separate lines
  - Use file.writelines([ii+'\n' for ii in foo]) for separate lines

Saving Variables: Pickle vs. JSON

Pickle

- Can store almost anything, including custom classes
- “Unsecure”--files contain code which is executed
- Byte storage: ‘wb’ and ‘rb’

- import pickle
- pickle.dump(data,file_obj)
- pickle.load()

JSON

- Dict, list, str, float, int
- No sets, tuples, or complex

- import json
- json.dump(data,file_obj)
- json.load()

Pickle vs. Json

Pickle is stored in bytes: use wb/rb

```
import os as os
import pickle
import json

foo={'a':[1,2,3],'b':'cd'};

pkName=os.getcwd()+r'\foo.pkl';
jsName=os.getcwd()+r'\foo.json';
with open(pkName,'wb') as file:
    pickle.dump(foo,file)
with open(jsName,'w') as file:
    json.dump(foo,file);
```

os.getcwd=Current directory

Both load as the original type (dictionary)

```
with open(pkName,'rb') as file:
    pkFoo=pickle.load(file);
with open(jsName,'r') as file:
    jsFoo=json.load(file)
```

Practice: Combining Files

- The directory Cal_Data has files separated by day with separate folders for each month.
- Files contain either alpha or numeric values
- Create one for each to collect all of the alpha data and for the numeric data with dates.

Lines should look like: Aug-6: 12345
Aug-12: 1532, etc.

And Aug-15: abcde
Sept-2: vwxyz, etc.

- Create another file for each logging the original file's address on each line.

fin