INFO 450 Spring 2021

Week 6

Agenda

- files
- json
- csv
- hackerrank

Files

Files are ways of storing data permanently, typically on a persistent media like an internal hard drive, network drive or USB storage.

Files, in most operating systems, are in a 'folder' or 'path' and have a filename + extension. Exentions can be used to associate programs for opening files of a certain type.

- e.g. 'pdf' -> Adobe PDF Reader
- e.g. 'xls' -> Microsoft Excel

Files are either text/ascii or Binary

Python Operations for files

```
open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)¶
```

- open: returns a file object
- file: path-like object giving the pathname, typically a string, relative or absolute
- mode:
  - r - read
  - w - write
  - x - exclusive creation
  - a - append
  - b - binary
  - t - text
  - + - reading and writing

Always close your files!

```
f = open('fname.txt', 'r')
contents = f.read()
f.close()
```

- If an exception, that close function may NOT get called.. so we:

```
with open('fname.txt', 'r') as f:
    contents = f.read()
print("File will automatically be closed.")
```

- Computer operating systems have a limit of how many 'file handles' can be opened. Bad things can happen if you cross that limit.

Writing Data to files

For the majority of this course, we'll deal with 'text' files. This may be slower and more disk space than binary packing data, but, it is simpler to verify.

When we write data to files, we write it as plain text. An integer becomes a String that looks like an integer. Strings are strings. Parsing them is the hard part.

```
cars = []
cars.append({"id": 1, "color":"blue", "make": "ford", "model":"flex"})
cars.append({"id": 2, "color":"orange", "make": "hyundai", "model":"santa fe sport"})
cars.append({"id": 3, "color":"green", "make": "honda", "model":"accord"})
cars.append({"id": 4, "color":"white" "make": "kia", "model":"sorento"})
cars.append({"id": 5, "color":"black", "make": "jeep", "model":"wrangler"})
cars.append({"id": 6, "color":"red", "make": "jaguar", "model":"f-type"})

with open('data.txt', 'w') as f:
    for car in cars:
        f.write(f"{car['id']},{car['color']},{car['make']},{car['model']")
```

What's it look like?

```
$ cat data.txt
1,blue,ford,flex
2,orange,hyundai,santa fe sport
3,green,honda,accord
4,white,kia,sorento
5,black,jeep,wrangler
6,red,jaguar,f-type
```

Read it back

```
cars = []
with open("data.txt", "r") as f:
    for line in f.readlines():
        line = line.strip()
        parts = line.split(",")
        cars.append({ "id": parts[0],
                      "color": parts[1],
                      "make": parts[2],
                      "model": parts[3]
                      })
print("My resulting cars: ", cars)
```

Read - output

```
$ python read_cars.py
My resulting cars:  [{'id': '1', 'color': 'blue', 'make': 'ford', 'model': 'flex'}, {'id': '2', 'color': '
```

- Ok , that's really ugly, let's make it JSON to read better:

JSON

- Ok , that's really ugly, let's make it JSON to read better:

```
import json
cars = []
with open("data.txt", "r") as f:
for line in f.readlines():
    line = line.strip()
    parts = line.split(",")
    cars.append({ "id": parts[0],
                  "color": parts[1],
                  "make": parts[2],
                  "model": parts[3]
                  })
print(cars)
print(json.dumps(cars))
```

```
$ python read_cars.py
[{'id': '1', 'color': 'blue', 'make': 'ford', 'model': 'flex'}, {'id': '2', 'color': 'orange', 'make': 'hy
[{"id": "1", "color": "blue", "make": "ford", "model": "flex"}, {"id": "2", "color": "orange", "make": "hy
```

- Well that didn't change.

Pretty JSON

```
import json
cars = []
with open("data.txt", "r") as f:
    for line in f.readlines():
        line = line.strip()
        parts = line.split(",")
        cars.append({ "id": parts[0],
                      "color": parts[1],
                      "make": parts[2],
                      "model": parts[3]
                      })
# print(cars)
# print(json.dumps(cars))
print(json.dumps(cars, sort_keys=True, indent=4))
```

```
$ python read_cars.py
[
    {
        "color": "blue",
        "id": "1",
        "make": "ford",
        "model": "flex"
    },
    {
        "color": "orange",
        "id": "2",
        "make": "hyundai",
        "model": "santa fe sport"
    },
    {
        "color": "green",
        "id": "3",
        "make": "honda",
        "model": "accord"
    },
    {
```

CSV

Python has built in capabilities for dealing with CSV files.
[CSV Python]

```
import csv
cars = []
cars.append({"id": 1, "color":"blue", "make": "ford", "model":"flex"})
cars.append({"id": 2, "color":"orange", "make": "hyundai", "model":"santa fe sport"})
cars.append({"id": 3, "color":"green", "make": "honda", "model":"accord"})
cars.append({"id": 4, "color":"white", "make": "kia", "model":"sorento"})
cars.append({"id": 5, "color":"black", "make": "jeep", "model":"wrangler"})
cars.append({"id": 6, "color":"red", "make": "jaguar", "model":"f-type"})

with open('data.csv', 'w', newline='') as csvfile:
    fieldnames = ['id', 'color','make','model']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writeheader() # Up to you
    # for car in cars:
        # writer.writerow(car)
    writer.writerows(cars)
```

```
$ cat data.csv
id,color,make,model
1,blue,ford,flex
2,orange,hyundai,santa fe sport
3,green,honda,accord
4,white,kia,sorento
5,black,jeep,wrangler
6,red,jaguar,f-type
```

Dealing with Multiple Files

This can really be used for reading and writing multiple files at the same time.

This simple example will be to 'put the big words' in one file and small words in another.

Homer's Iliad - Project Gutenberg

Steps:

- Open iliad.txt to read
- Open bigwords.txt to write
- Open smallwords.txt to write
- Read iliad.txt, process to remove characters we don't care about
- Loop through words
  - If long (more than 5 characters) write to bigwords.txt
  - else write to smallwords.txt

```
with open("iliad.txt", 'r') as iliad, open("bigwords.txt", "w") as big_words, open("smallwords.txt", "w")
    contents = iliad.read()
    for char in '?!-=+/\'"-.,\n':
        contents = contents.replace(char,' ')
    contents = contents.lower()
    word_list = contents.split()
    print(word_list[0:100])
    for word in word_list:
        if len(word) > 5:
            big_words.write(f"{word}\n")
        else:
            small_words.write(f"{word}\n")
```

Output

```
$ head -n 10 bigwords.txt
invoking
minerva
repressing
achilles
departure
briseis
achilles
thetis
calling
briareus

$ head -n 10 smallwords.txt
the
iliad
of
homer
homer
the
muse
mars
the
fury
```

Word count

Let's count the most frequent words:

Break it all up into functions for a 'real' Python program

0) Read File - return contents

1) Count words - for fun

2) Perform frequency analysis of words

```
import os
import sys


def frequency_analysis(word_list):
    def mapper(word):
        return 1

    mapped = map(mapper, word_list)
    mapped = zip(word_list, mapped)
    output_dict = dict()
    for x in mapped:
        if x not in output_dict:
            output_dict[x] = 0
        output_dict[x] += 1
    return output_dict

def read_file(filename):
    with open(filename, 'r') as f:
        contents = f.read()
        return contents

def count_words(word_list):
    return len(word_list)

if __name__ == "__main__":

    filename = "iliad.txt"
    contents = read_file(filename)
    print(f"Content length: {len(contents)}")
    for char in '?!-=+/\'"-.,\n':  # Super inefficient
        contents = contents.replace(char,' ')
    contents = contents.lower()
    word_list = contents.split()
    word_count = count_words(word_list)
    print(f"Word count: {word_count}")
    reduced = frequency_analysis(word_list)
    sorted_reduced = {k: v for k, v in sorted(reduced.items(), key=lambda item: item[1], reverse=True)}
    for x in sorted_reduced:
        print(f"{x}: {sorted_reduced[x]}")
```

Homework

No homework!

Exam Next Week

Cumulative

Lots of code

Lots of fill in the blank

Lots of multi choice

Expected 60-90 minutes

Will have 2 hours alloted

One attempt, open at approximately 5PM and close by midnight.