FBA | QUANTITATIVE FINANCE RESEARCH GROUP

Lesson5. File Input/Output

FAI Team2

Ⅰ. Path

Ⅱ. Read and write .txt file

Ⅲ. Read and write .json file

Ⅳ. Read and write .csv file

Ⅰ. Path

* Absolute Path(절대경로):
어떤 웹페이지나 파일이 가지고 있는 고유한 경로 (집의 주소와 비슷)

Ex) http://www.google.com, C:₩Users₩document₩fba.ipynb

- 절대경로를 사용하면 어느 곳에서 작업을 해도 해당파일을 찾을 수 있다
- 그렇지만 상위 폴더가 변화하는 등의 상황이 발생하면 해당 경로가 더 이상 유효하지 않을 수가 있다

Ⅰ. Path

- Relative Path(상대경로):
  현재 있는 곳을 기준으로 해서 파일의 위치를 나타낸 경로

상위 폴더 명이 바뀌는 등의 변화가 있어도 현재 있는 곳을 기준으로 경로를 나타내기 때문에 유동적으로 파일을 참조할 수 있다.

- 상대경로의 표시

/: 루트

./: 현재 위치

../: 현재 위치에서 상단 폴더

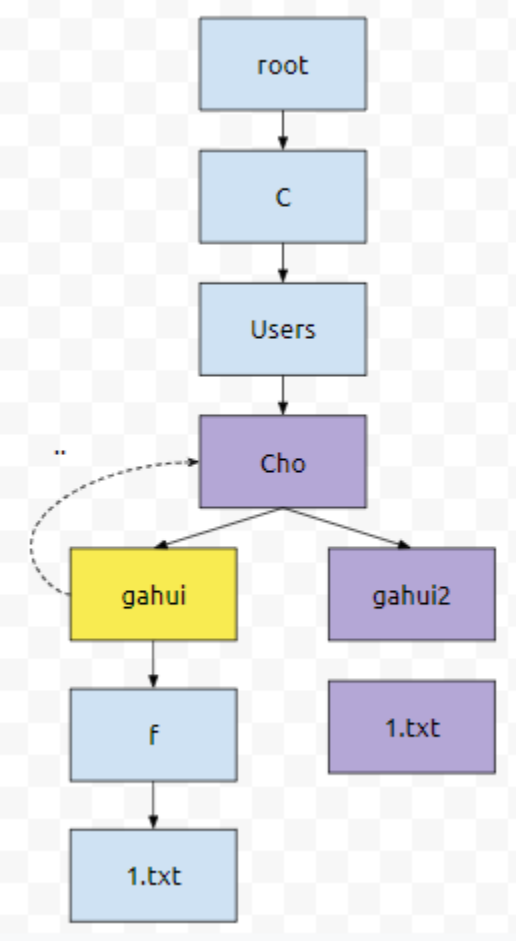
Ⅰ. Path

- Example
- gahui2에 있는 1.txt의 상대경로를 찾아보자!

* 현재 gahui2에 있을 때, 파일의 상대경로는?
답: ./1.txt

* 현재 Cho에 있을 때, 파일의 상대경로는?
답: ./gahui2/1.txt

* 현재 gahui에 있을 때, 파일의 상대경로는?
답: ../gahui2/1.txt



Ⅱ. Read and write .txt file

- Read File

* 파일을 읽을 때는 open의 mode를 ‘r’로 설정해준다

* .txt 파일을 읽는 방법에는 크게 세 가지가 있다

- readline(): 한 줄씩 읽어온다
- readlines(): 모든 줄들을 리스트의 형태로 읽어온다
- read(): 모든 데이터를 읽어온다

Ⅱ. Read and write .txt file

- Example
- readline() 사용

```python
with open('practice.txt', 'r') as infile:
    while True:
        line = infile.readline()
        if not line: break
        print(line)
```

- readlines() 사용

```python
with open('practice.txt', 'r') as infile:
    lines = infile.readlines()
    for line in lines:
        print(line)
```

```
row1

row2

row3

row4

row5

row6

row7

row8

row9

end
```

Ⅱ. Read and write .txt file

- Example
- read() 사용

```
with open('practice.txt', 'r') as infile:
    data = infile.read()
    print(data)
```

- 참고: 다음과 같은 방식도 가능하다.

```
with open('practice.txt', 'r') as infile:
    for line in infile:
        print(line)
```

Ⅱ. Read and write .txt file

- Write

* 새로운 파일을 작성할 때는 open의 mode를 ‘w’로 설정해준다
* 기존의 파일에 내용을 추가하고자 한다면 mode를 ‘a’로 설정해준다
* .txt 파일을 작성할 때는 write를 사용한다.

Ⅱ. Read and write .txt file

- Example
- 새로운 파일 생성

```
with open('new_practice.txt', 'w') as outfile:
  for i in range(1,11):
    data = f'line{i}\n'
    outfile.write(data)
```

```
line1
line2
line3
line4
line5
line6
line7
line8
line9
line10
```

- 기존의 파일에 내용 추가

```
with open('new_practice.txt', 'a') as addfile:
  for i in range(1,11):
    data = f'line{i}\n'
    addfile.write(data)
```

```
line1
line2
line3
line4
line5
line6
line7
line8
line9
line10
line11
line12
line13
line14
line15
line16
line17
line18
line19
```

Ⅱ. Read and write .txt file

- Exercise
- 가장 많이 나오는 단어 찾기

```python
import collections


def count_unique_words(filename='hamlet.txt'):
    # Use collections.Counter() to count frequency that each word appears
    words = collections.Counter()
    # import data(mode='r' is default)
    with open(filename) as f:
        for line in f:
            #Count frequency of the word line by line
            words.update(line.split())

    # Calculate the ten most common words
    for word, count in words.most_common(10):
        print(word, count)


if __name__ == '__main__':
    count_unique_words('hamlet.txt')
```

JSON

JSON stands for ‘Java Scrpit Object Notation’

- JSON is a file format can contain structured data.

Advantages using .JSON file

1) Easier to read for both human and machine.

2) Easy to parse data and faster to implement data.

3) Widely adopted; most of the programming languages have built-in library to deal with .json format data.

Disadvantages:

1) No date data type

2) Low signal to error

JSON file example

```
{
    "empid": "SJ011MS",
    "personal": {
        "name":"Smith Jones",
        "gender": "Male",
        "age": 28,
        "address":
        {
            "streetaddress": "7 24th Street",
            "city": "New York",
            "state": "NY",
            "postalcode": "10038"
        }
    },
    "profile": {
        "designation": "Deputy General",
        "department": "Finance"
    }
}
```

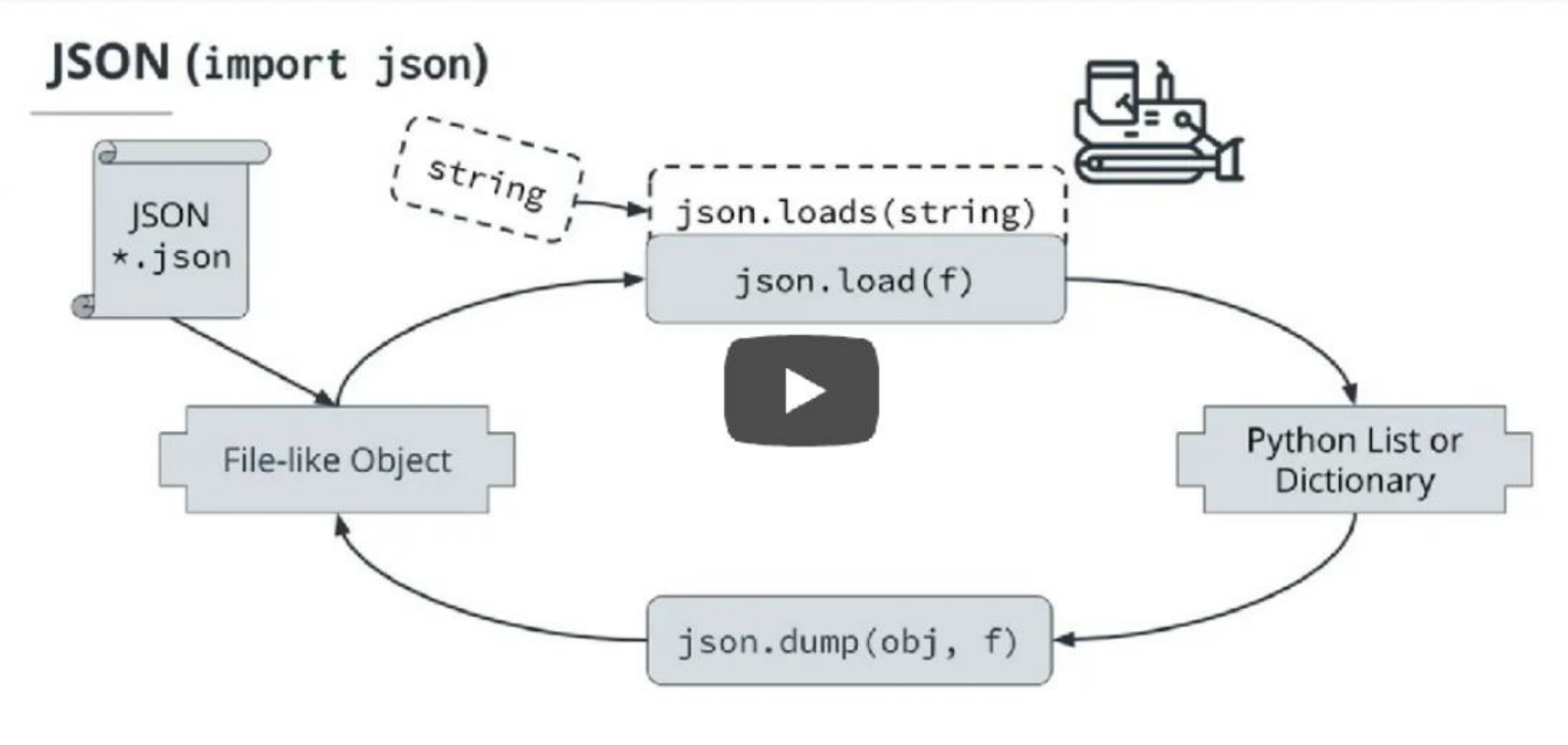

JSON file
- highly structured data
  with array or object

- it can be filled with
string, number, true/false and nulls.

Python vs JSON

Python	JSON
dict	object
list, tuple	array
str	string
int, long, float	number
True	true
False	false
None	null

Working with JSON



Working with JSON

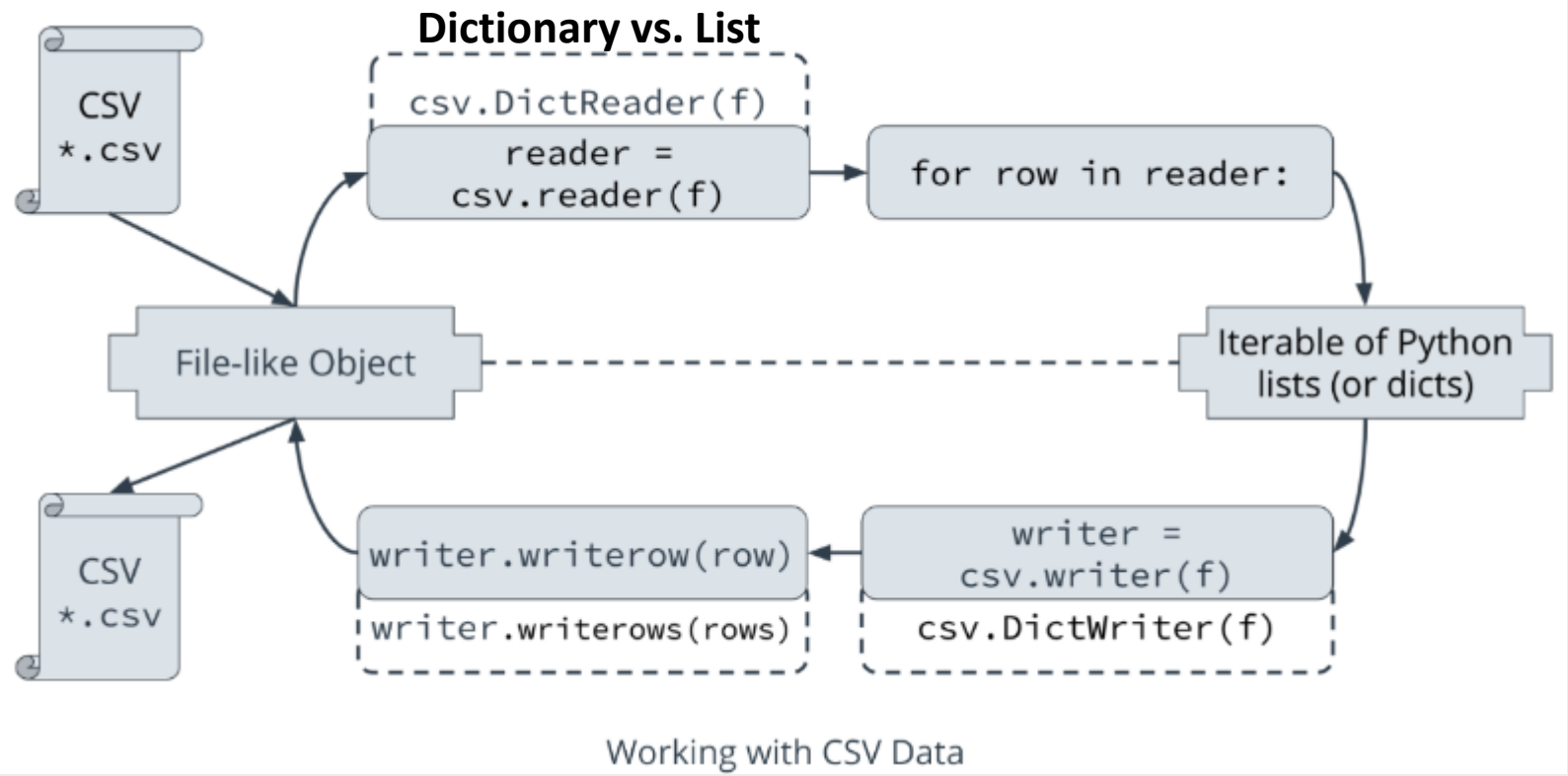
```python
import json

#Extract .json file into python using json.loads
with open ('listings.json', 'r') as infile:
  contents = json.loads(infile)



# Write data to .json format output file using json.dump
with open ('output-listings.json', 'w') as outfile:
  json.dump(pythondata, outfile)
```

Working with CSV



Final Project: Read CSV

```csv
2B,410,AER,2965,KZN,2990,,0,CR2
2B,410,ASF,2966,KZN,2990,,0,CR2
2B,410,ASF,2966,MRV,2962,,0,CR2
2B,410,CEK,2968,KZN,2990,,0,CR2
2B,410,CEK,2968,OVB,4078,,0,CR2
2B,410,DME,4029,KZN,2990,,0,CR2
2B,410,DME,4029,NBC,6969,,0,CR2
2B,410,DME,4029,TGK,₩N,,0,CR2
2B,410,DME,4029,UUA,6160,,0,CR2
2B,410,EGO,6156,KGD,2952,,0,CR2
2B,410,EGO,6156,KZN,2990,,0,CR2
2B,410,GYD,2922,NBC,6969,,0,CR2
2B,410,KGD,2952,EGO,6156,,0,CR2
2B,410,KZN,2990,AER,2965,,0,CR2
2B,410,KZN,2990,ASF,2966,,0,CR2
```

```python
def read_airports(filename='airports.dat'):
    airports = {}
    with open(filename, encoding='utf-8') as f:
        reader = csv.reader(f)
        for line in reader:
            airports[line[4]] = line[1]
    return airports


def read_routes(filename='routes.dat'):
    routes = {}
    with open(filename, encoding='utf-8') as f:
        reader = csv.reader(f)
        for line in reader:
            if line[2] not in routes:
                routes[line[2]] = []
            routes[line[2]].append(line[4])
    return routes
```

Final Project: Data Processing

```
{('SFO', 'ABQ', 'HOU', 'BOS'),
 ('SFO', 'AMS', 'AUA', 'BOS'),
 ('SFO', 'AMS', 'BOS'),
 ('SFO', 'AMS', 'FCO', 'BOS'),
 ('SFO', 'AMS', 'KEF', 'BOS'),
 ('SFO', 'AMS', 'LIS', 'BOS'),
 ('SFO', 'AMS', 'MAD', 'BOS'),
 ('SFO', 'AMS', 'PDL', 'BOS'),
 ('SFO', 'AMS', 'PTY', 'BOS'),
 ('SFO', 'AMS', 'PUJ', 'BOS'),
 ('SFO', 'AMS', 'SXM', 'BOS'),
 ('SFO', 'ATL', 'ALB', 'BOS'),
 ('SFO', 'ATL', 'AUA', 'BOS'),
 ('SFO', 'ATL', 'BDA', 'BOS'),
 ('SFO', 'ATL', 'BNA', 'BOS'),
 ('SFO', 'ATL', 'BOS'),
```

```python
def find_paths(routes, source, dest, max_segments):
    frontier = {source}
    seen = {source: {(source, )}}
    for segs in range(max_segments):
        next_frontier = set()
        for airport in frontier:
            for target in routes.get(airport, ()):
                if target not in seen:
                    next_frontier.add(target)
                    seen[target] = set()
                for path in seen[airport]:
                    if len(path) != segs + 1:
                        continue
                    seen[target].add(path + (target, ))
        frontier = next_frontier
    return seen[dest]
```

Final Project: Write JSON

```
{
  "1": [
    [
      "Syracuse Hancock International Airport",
      "General Edward Lawrence Logan International Airport"
    ]
  ],
  "2": [
    [
      "Syracuse Hancock International Airport",
      "Chicago O'Hare International Airport",
      "General Edward Lawrence Logan International Airport"
    ],
    [
      "Syracuse Hancock International Airport",
      "La Guardia Airport",
      "General Edward Lawrence Logan International Airport"
    ],
    [
      "Syracuse Hancock International Airport",
      "Charlotte Douglas International Airport",
      "General Edward Lawrence Logan International Airport"
    ],
```

```python
def main(source, dest, max_segments):
    airlines = read_airlines()
    airports = read_airports()
    routes = read_routes()

    paths = find_paths(routes, source, dest, max_segments)
    output = {}

    for path in paths:
        segments = len(path) - 1
        if segments not in output:
            output[segments] = []
        output[segments].append(rename_path(path, airports))

    with open(f"{source}_{dest}_(max_{max_segments})_.json", 'w') as f:
        json.dump(output, f, indent=2, sort_keys=True)
```

FBA | QUANTITATIVE FINANCE RESEARCH GROUP

Contact fbaquant@gmail.com Website fbaquant.com