

# Working with JSON in Python

INF 551

Wensheng Wu

# Python primer

- String: 'abc' or "abc"
- List: `x = ['abc', 25]`
  - `x[0] = ?`
  - `x.append(True)` // True is boolean
  - `x.append(None)` // None is `NoneType` = null
- Tuple: `y = ('abc', 25)`
  - `y[0] = ?`
  - What about `y.append(True)`?

# Python primer

- Dictionary: `z = {'name': 'john', 25: 'age'}`
  - Note key in Python can also be integer or tuple
  - `z['name'] = ?`
  - `z[25]=?`
  - What about `z['age']` or `z['25']`?
- `z['gender'] = 'male'`
  - `z = {25: 'age', 'name': 'john', 'gender': 'male'}`

# Working with JSON in Python

- `import json`
  - Loading json library module
- `json.dumps()`
  - JSON encoder
  - Python object => JSON document
- `json.loads()`
  - JSON decoder
  - JSON document => Python object

# Python object => JSON document

- Python list => JSON array
  - `json.dumps([1, 2]) => '[1, 2]'`
  - `json.dumps([3, 'abc', True, None]) => '[3, "abc", true, null]'`
- Python tuple => JSON array
  - `json.dumps((1, 'abc')) => '[1, "abc"]'`

# Python object => JSON document

- Python dictionary => JSON object
  - `json.dumps({'name': 'john', 25: 'age'}) => '{"25": "age", "name": "john"}'`
- Notes
  - `None` => `null`
  - `True` => `true`
  - `'abc'` => `"abc"`

# Python object => JSON document

- `json.dumps(['foo', {'bar': ('baz', None, 1.0, 2)}])`
  - `'["foo", {"bar": ["baz", null, 1.0, 2]}]'`
- `json.dumps({(1,2): 5})`
  - Error (key is a tuple, Ok in Python)
  - `dumps()` doesn't take tuple as key (but see below)
- `json.dumps({(2): 5}) => '{"2": 5}'`

# JSON document => Python object

- JSON object => Python dictionary
  - `json.loads('{"name": "john", "age": 5}')` => `{u'age': 5, u'name': u'john'}`
  - Note: 'u' means "unicode"
- JSON array => Python list
  - `json.loads('[25, "abc"]')` => `[25, u'abc']`



# JSON document => Python object

- `json.loads('\"abc\"')` => `u'abc'`
- `json.loads('25.2')` => `25.2`
- `json.loads('true')` => `True`
- `json.loads('null')` => `None`
- `json.loads('{\"name\": \"john\", \"age\": 25, \"phone\": [123, 456]})'`  
=> `{u'phone': [123, 456], u'age': 25, u'name': u'john'}`

# Conversion summary

JSON	Python
Object	Dictionary
Array	List
Array	Tuple (from Python)
null	None
true	True
false	False

Python dictionary => JSON object

- Keys in Python can be number, string, or tuple.
- Number is also converted to string.
- But tuple (with two or more components) is not acceptable by dumps()/dump().

# Working with files

- `f = open('lax.json')`
- `lax= json.load(f)`
- `out_file = open('output.json', 'w')`
- `json.dump(lax, out_file)`

# LAX passenger traffic data

- Download data at:
  - <https://data.lacity.org/A-Prosperous-City/Los-Angeles-International-Airport-Passenger-Traffi/g3qu-7q2u>
- A copy is available on Blackboard too
  - In the Resources folder

# Data spreadsheet

	ReportPeriod ⓘ ⋮	Terminal	Arrival_Departure ⓘ ⋮	Domestic_International	Passenger_Count ⓘ ⋮
4650 ⓘ	07/01/2016 12:00:00 AM	Terminal 3	Arrival	Domestic	402,452
4651 ⓘ	07/01/2016 12:00:00 AM	Terminal 3	Departure	Domestic	390,418
4652 ⓘ	07/01/2016 12:00:00 AM	Terminal 3	Departure	International	4,365
4653 ⓘ	07/01/2016 12:00:00 AM	Terminal 4	Arrival	Domestic	422,257
4654 ⓘ	07/01/2016 12:00:00 AM	Terminal 4	Arrival	International	35,800
4655 ⓘ	07/01/2016 12:00:00 AM	Terminal 4	Departure	Domestic	384,861
4656 ⓘ	07/01/2016 12:00:00 AM	Terminal 4	Departure	International	64,590
4657 ⓘ	07/01/2016 12:00:00 AM	Terminal 5	Arrival	Domestic	456,188
4658 ⓘ	07/01/2016 12:00:00 AM	Terminal 5	Arrival	International	78,781
4659 ⓘ	07/01/2016 12:00:00 AM	Terminal 5	Departure	Domestic	478,348
4660 ⓘ	07/01/2016 12:00:00 AM	Terminal 5	Departure	International	71,021
4661 ⓘ	07/01/2016 12:00:00 AM	Terminal 6	Arrival	Domestic	374,394

# JSON file (lax.json)

- Ignore "meta" info (in the beginning of file)
- Records are in the value of "data"

```
  },  
  "data" : [ [ 1, "31CAC749-9F88-4EA3-8EC0-0DFC3EE6DA81", 1, 1401275468, "883844",  
    "1401275468", "883844", "{\n}", "2014-05-01T00:00:00", "2006-01-01T00:00:00", "Imperial Terminal", "Arrival", "Domestic", "490" ],  
    [ 2, "085E4C26-9CE8-41F1-AD72-7A8E42DCB3B9", 2, 1401275468, "883844", 1401275468, "883844", "{\n}", "2014-05-01T00:00:00", "2006-01-01T00:00:00", "Imperial Terminal", "Departure", "Domestic", "498" ],  
    [ 3, "5EA27B8A-859C-4FA9-B697-38B7292E3689", 3, 1401275468, "883844", 1401275468, "883844", "{\n}", "2014-05-01T00:00:00", "2006-01-01T00:00:00", "Misc. Terminal", "Arrival", "Domestic", "753" ],  
    [ 4, "7E96DA71-1DCF-4DF6-98E6-3BD3AF165821", 4, 1401275468, "883844", 1401275468, "883844", "{\n}", "2014-05-01T00:00:00", "2006-01-01T00:00:00", "Misc. Terminal", "Departure", "Domestic", "688" ],  
    [ 5, "0FEAB567-5777-4FC4-9D0F-8C1B67CA44D7", 5, 1401275468, "883844", 1401275468, "883844", "{\n}", "2014-05-01T00:00:00", "2006-01-01T00:00:00", "Terminal 1", "Arrival", "Domestic", "401535" ] ] ]
```

# Querying it in Python

ReportPeriod



```
>>> lax["data"][0]
[1, u'31CAC749-9F88-4EA3-8EC0-0DFC3EE6DA81', 1, 1401275468, u'883844', 1
401275468, u'883844', u'{\n}', u'2014-05-01T00:00:00', u'2006-01-01T00:0
0:00', u'Imperial Terminal', u'Arrival', u'Domestic', u'490']
>>> lax["data"][0][9]
u'2006-01-01T00:00:00'
>>> lax["data"][0][10]
u'Imperial Terminal'
>>> lax["data"][0][11]
u'Arrival'
>>> lax["data"][0][12]
u'Domestic'
>>> lax["data"][0][13]
u'490'
>>> |
```

# Resources

- JSON
  - <https://en.wikipedia.org/wiki/JSON>
  - Syntax: <http://www.json.org/>
- JSON encoder and decoder
  - <https://docs.python.org/2/library/json.html>