MapReduce in Python (emulation only)

INF 553

Wensheng Wu

Python

- Use 2.7 for assignments
- Tutorial:
  - http://learnpythonthehardway.org/book/

Python

- Object-oriented
  - Module, class, object

- Data structures
  - Tuple: t = (1, 2, 3), t[0] = ?
  - List: l = [4, 5, 6], l[0] = ?
  - Dictionary: d = {'name': 'john', 'age': 25}, d['name'] = ?

Tuple vs list

- Tuples are fixed size
  – Can not append, remove, …

- Lists are dynamic
  – Can grow: l.append(4) adds 4 to the end of list
  – Can shrink: l.pop(0) removes 1st element

Loop

-
  ```
  for x in t:
          print x
  ```

-
  ```
  for i in range(0, len(t)):
          print t[i]
  ```

-
  ```
  for key in d:
   print key, d[key]
  ```

Loop

-
  ```
  for k, v in d.iteritems():
          print k, v
  ```

Object-oriented

- class myclass:

  ```
      items = []
      def add(self, item):
          self.items.append(item)
  ```

- myobj = myclass()
- myobj.add(2)
- myobj.items

JSON (Javascript Object Notation)

- Light-weight data exchange format
  - Much simpler than XML
  - Language-independent
  - Inspired by Javascript object literal notation

- Some differences from Javascript objects, e.g.,
  - String in JSON must be double-quoted
  - Ok to single-quote in Javascript

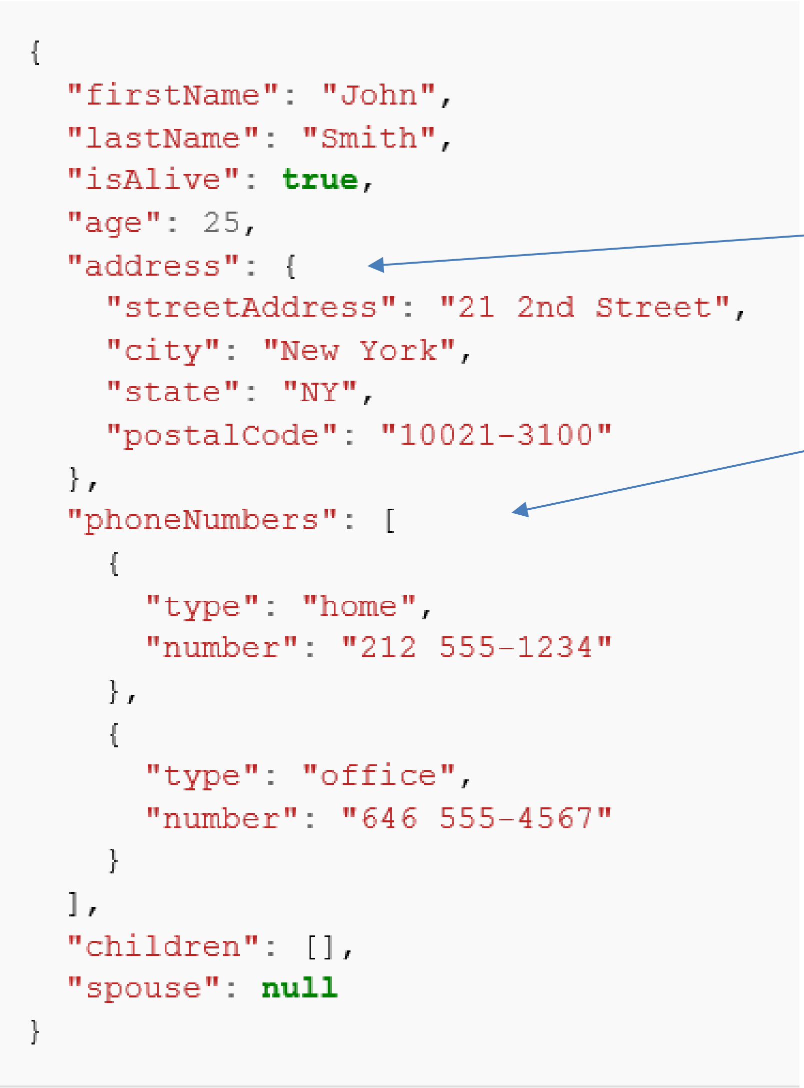
Syntax of JSON

- object = {} | { members }
  - – members = pair | pair, members
  - – pair = string : value

- array = [] | [elements]
  - – elements = value | value elements

- value = string|number|object|array|true|false|null

Example JSON



```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

Value is an object

Value is an array

JSON validator

- http://jsonlint.com/

JSON in Python: load and dump

- json_string = '{"first_name": "John", "last_name":"Smith"}'

- parsed_json = json.loads(json_string)
  - Parsed_json is a dictionary
  - parsed_json["first_name"]

- json.dumps(parsed_json)
  - Convert python object back to JSON string

Example

- js = '[1, 2, 3]'
- js1 = json.loads(js)
- What is in js1?

Example

- value = (1, 2, 3)
- str = json.dumps(value)
- What is in str?
- What if value = [1,2,3]?

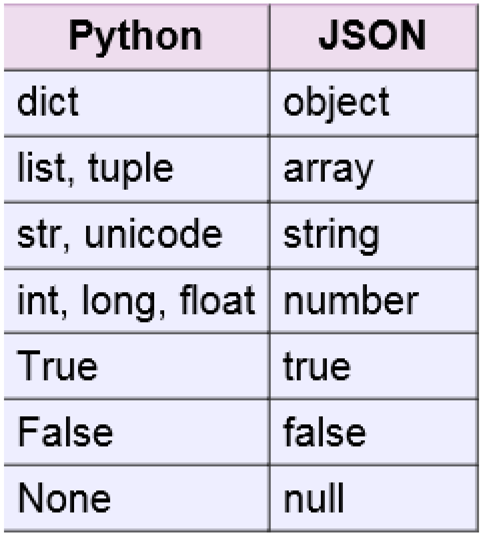
Load and dump

- Use default json.JSONEncoder() and json.JSONDecoder()

- Can define customized encoder and decoder

Default encoding

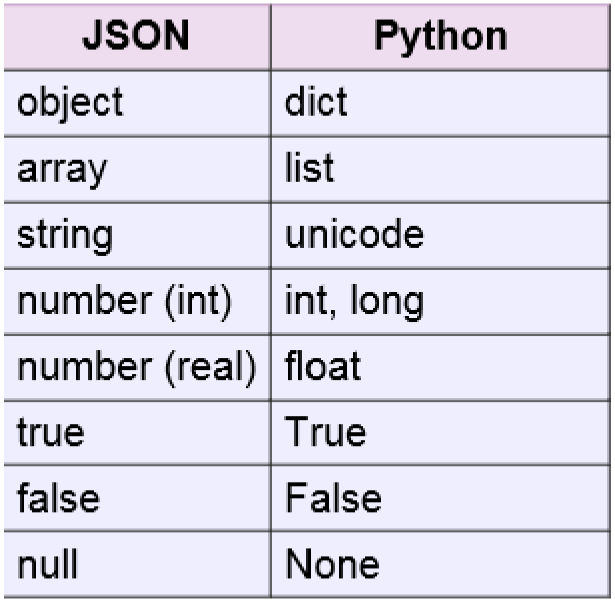
- json.JSONEncoder().encode({"foo": ["bar", "baz"]})
  - '{"foo": ["bar", "baz"]}'



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

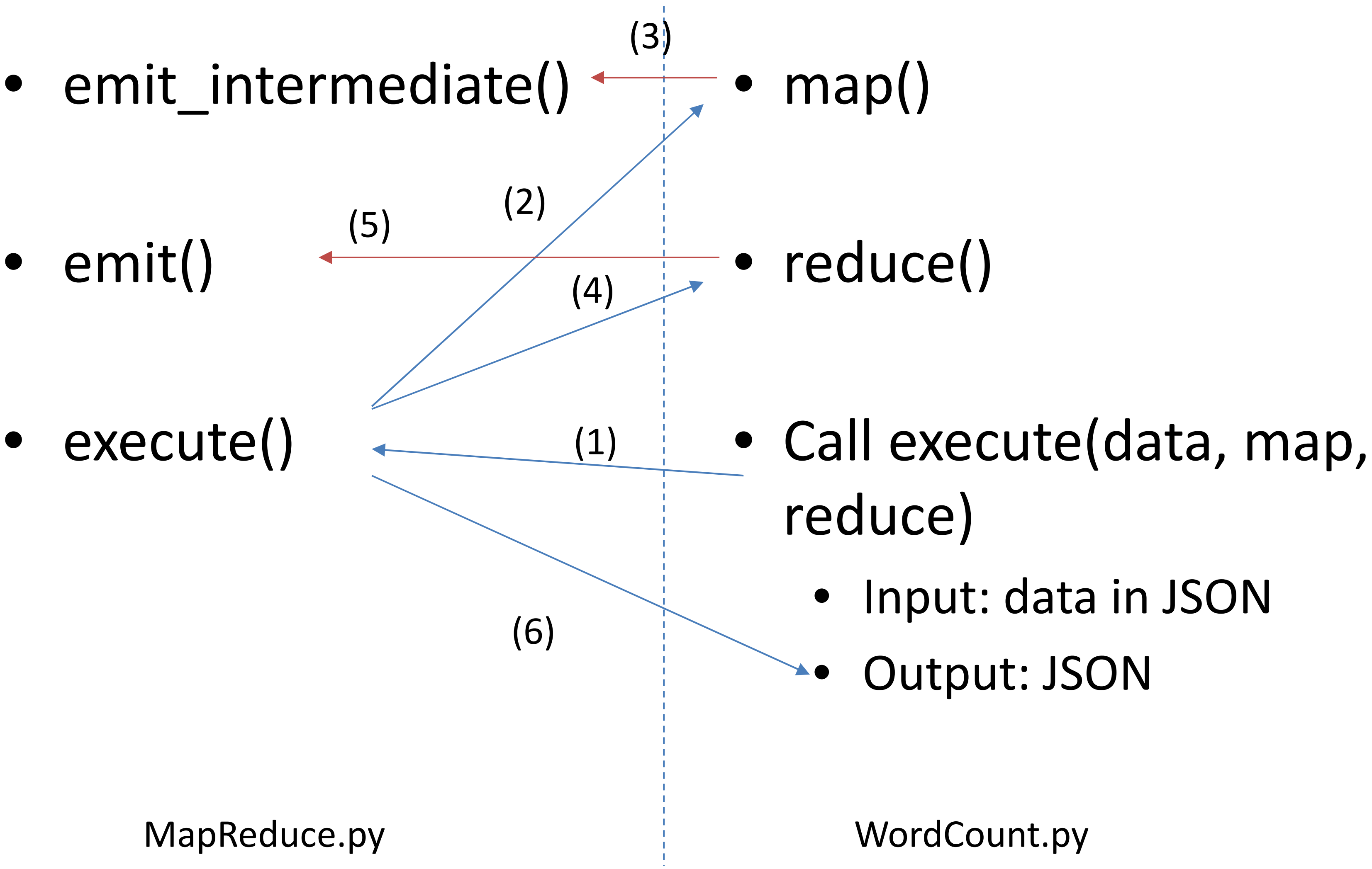
Decoding

- json.JSONDecoder().decode('{"foo": ["bar", "baz"]}')



JSON	Python
object	dict
array	list
string	unicode
number (int)	int, long
number (real)	float
true	True
false	False
null	None

MapReduce in Python: WordCount



- emit_intermediate()
- emit()
- execute()

MapReduce.py

- map()
- reduce()
- Call execute(data, map, reduce)
  - Input: data in JSON
  - Output: JSON

WordCount.py

Squaring a matrix

- Compute C = A^2
- Map task: ?
- Reduce task: ?