파이썬으로 배우는 데이터 구조

한동대학교
전산전자공학부
김영섭 교수

학습 목표

JSON 자료형 이해하고, 파이썬을 이용해 자유롭게
JSON 파일을 읽고 쓸 수 있다

Data Structures
Chapter 2 - 1

- **JavaScript Object Notation(JSON)**
- Software Design Principles
- Abstract Data Type(ADT)

그러므로 예수께서 자기를 믿은 유대인들에게 이르시되 너희가 내 말에 거하면 참으로 내 제자가 되고
진리를 알지니 진리가 너희를 자유롭게 하리라 (요8:31-32)

하나님은 모든 사람이 구원을 받으며 진리를 아는데 이르기를 원하시느니라 (딤전2:4)

내 아들들을 먼 곳에서 이끌며 내 딸들을 땅 끝에서 오게 하며 내 이름으로 불려지는 모든 자 곧 내가 내 영광을 위하여 창조한 자를 오게 하라 그를 내가 지었고 그를 내가 만들었노라 (사43:6-7)

너는 청년의 때에 너의 창조주를 기억하라 곧 곤고한 날이 이르기 전에, 나는 아무 낙이 없다고 할 해들이 가깝기 전에 (전12:1)

그런즉 너희가 먹든지 마시든지 무엇을 하든지 다 하나님의 영광을 위하여 하라 (고전10:31)

Quiz

- The items states about the execution of the following code. Select false statement(s):

```
def testing(x):
    try:
        print('Trying some code')
        2 / x
    except ZeroDivisionError:
        print('ZeroDivisionError raised here')
    except:
        print('Error raised here')
    else:
        print('Else clause')
    finally:
        print('Finally')
if __name__ == '__main__':
    for x in [1, 0, '1']:
        testing(x)
```

```
1. The case 0 produces 'ZeroDivisionError'.
2. The case 1 produces 'else' clauses'.
3. The case '1' produces 'Error raised here'.
4. All three cases go through 'else' block.
5. All three cases go through 'finally' block.
6. One case goes through 'else' block.
```

Quiz

- Which of the following statements is/are true?
  1. A try block is preceded by at least one finally block
  2. For each try block there must be at least one except block defined.
  3. A try block may be followed by any number of finally blocks
  4. If both except and finally blocks are defined, except block must precede the finally block

Learning outcomes

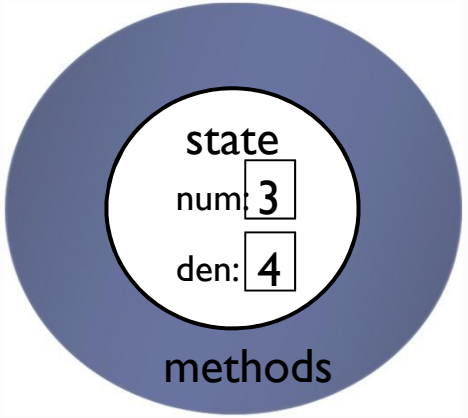
- At the end of this lecture, students should be able to:
  - Understand what JSON is used for.
  - Recognize information in JSON format.
  - Use the Python JSON library to read and write standard Python data types.
  - Extract useful information through the web page.

- Resources:
  - Tutorials Point: JSON with Python
    - JSON with Python (tutorialspoint.com)
  - Python Documentation
    - https://docs.python.org/3.9/library/json.html

Question?

- Given a particular set of data, how do you store it permanently?
  - What do you store on disk?
  - What format?
  - Can you easily transmit over the web?
  - Will it be readable by other languages?
  - Can humans read the data?

- Examples:
  - A square
  - A dictionary



JSON: JavaScript Object Notation

- Text-based data representation format for data interchange
  - Human readable (good for debugging / manual editing)
- Commonly Used for APIs and Configs
  - Makes the information more portable
- Lightweight and easy to read and write
  - Easy to transmit using web
- Integrates easily with most languages
  - Portable to different platforms

It is very widely used for many applications and its configuration. It is very hard to find an application which does not use JSON in some extent.

JSON Types and Python Types

- Strings - `"Hello World", "James"`
- Numbers - `10, 1.5,  -30, 1.2e10`
- Booleans - `true, false`
- null - `null`
- Arrays - `[1, 2, 3], ["Hello", "World"]`
- Objects - `{"name": "Joe", "age": 22}`

Python	JSON
dict	object
list, tuple	array
str	string
int, float, int- & float-derived Enums	number
True	true
False	false
None	null

JSON File: user.json

```
{
  "name": "joe",
  "id": 3,
  "coder": true,
  "hobbies": ["Climbing", "Bowling"],
  "friends": [ {
     "name": "kim",
     "id": 100,
     "coder": false,
     "friends": [ . . . ]
   } ]
}
```

JSON File: companies.json

```
[
  {
    "name": "Big Corpration",
    "numberOfEmployee": 10000,
    "ceo": "Jun",
    "rating": 3.9
  },
  {
    "name": "Small Startup",
    "numberOfEmployee": 5,
    "ceo": null,
    "rating": 4.5
  }
]
```

Example Using JSON File: c_cpp_properties.json

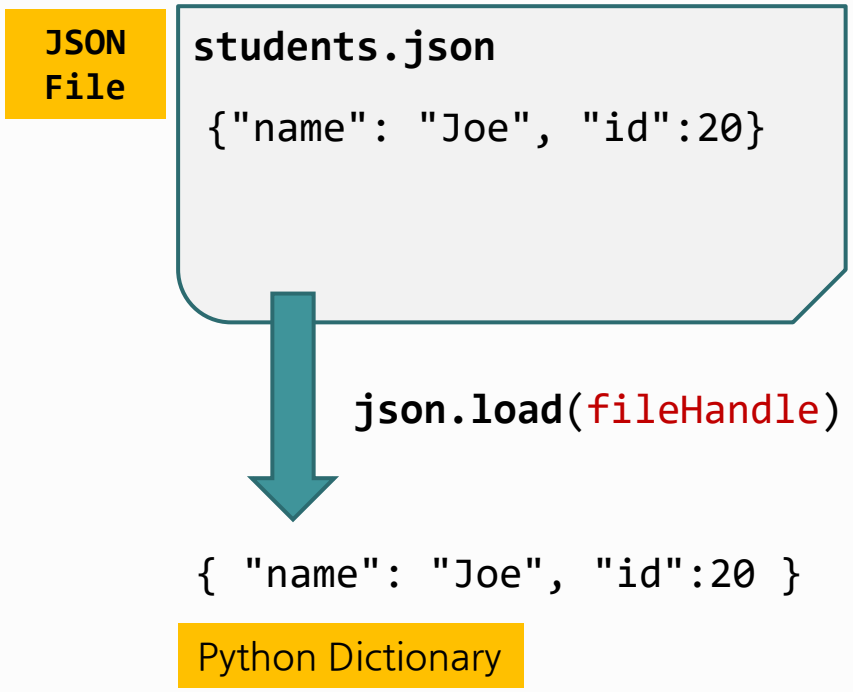
```
{
    // https://code.visualstudio.com/docs/cpp/c-cpp-properties-schema-reference
    "configurations": [
        {
            "name": "Win32",
            "includePath": [ "${workspaceFolder}/**" ],
            "defines": [ "_DEBUG", "UNICODE", "_UNICODE", "DEBUG" ],
            "windowsSdkVersion": "10.0.18362.0",
            "compilerPath": "C:/msys64/mingw64/bin/g++.exe",
            "cStandard": "c17",
            "cppStandard": "c++17",
            "intelliSenseMode": "gcc-x64"
        }
    ],
    "version": 4
}
```

Example Using JSON File: launch.json

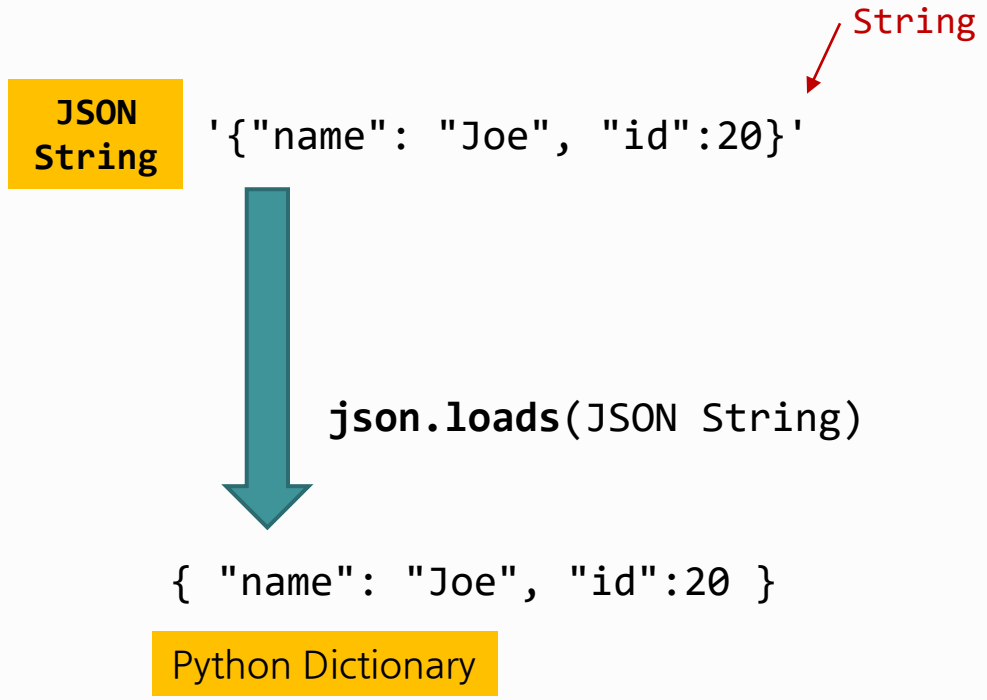
```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "g++.exe - 활성 파일 빌드 및 디버그",
            "type": "cppdbg",
            "request": "launch",
            "program": "${fileDirname}\\${fileBasenameNoExtension}.exe",
            // "args": [ "john", "Dr. Lee", "Handong Global University", "peter" ],
            "args": [],
            "stopAtEntry": false,
            "cwd": "C:/msys64/mingw64/bin",
            "environment": [],
            "externalConsole": false,
            "MIMode": "gdb",
            "miDebuggerPath": "C:\\msys64\\mingw64\\bin\\gdb.exe",
            "setupCommands": [
                {
                    "description": "gdb에 자동 서식 지정 사용",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                }
            ],
            "preLaunchTask": "build ${fileBasenameNoExtension}"
        }
    ]
}
```

Python JSON Parsing: Decode JSON using load() & loads()

json.load()

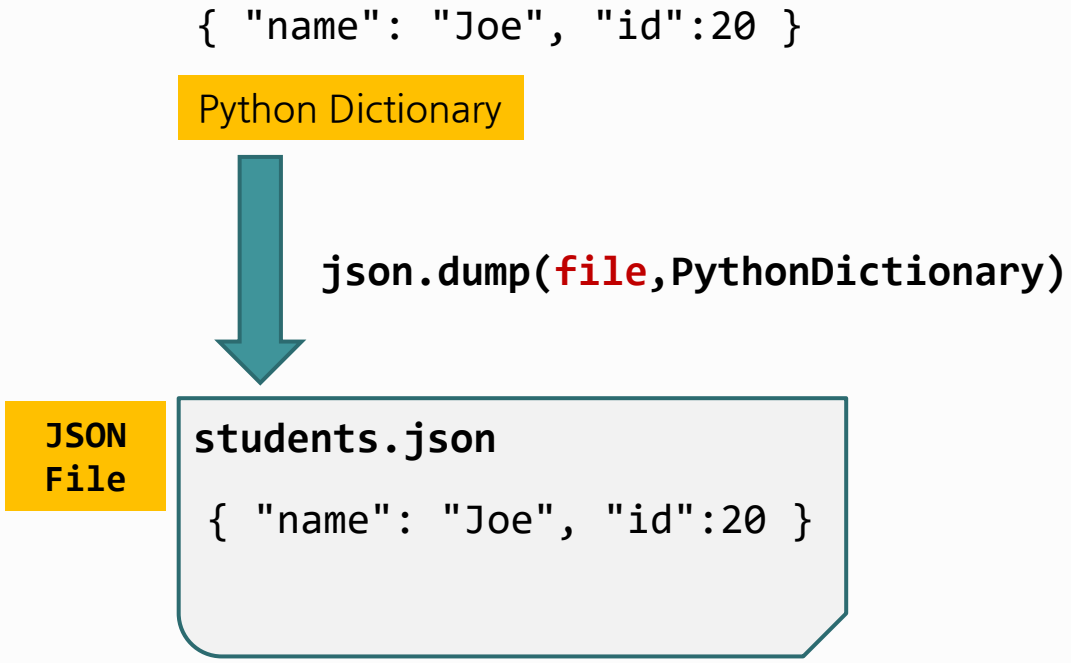


json.loads()

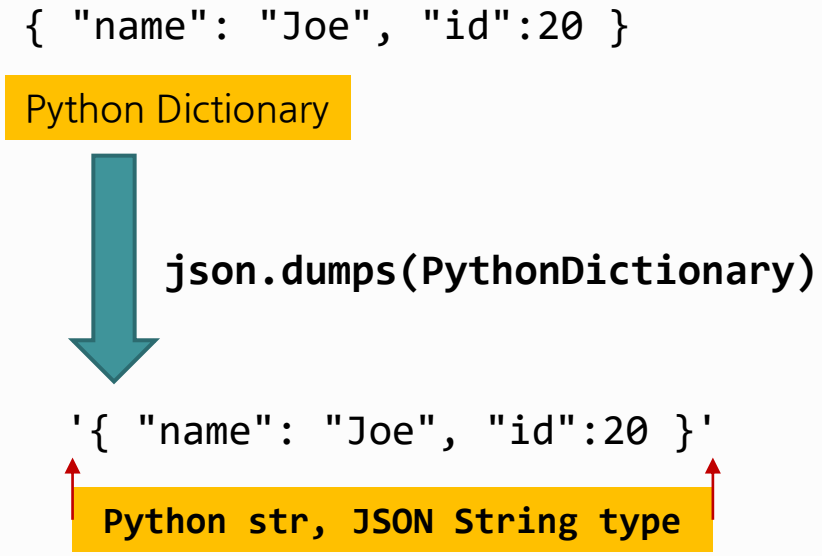


Python JSON Encoding: Encode JSON using dump() & dumps()

json.dump()



json.dumps()



(ex) print(json.dumps(data, indent=2))

Writing or Printing a dictionary in Python

- Create a dictionary

```
my_dict = {'Angela':866, 'Adrian':871, 'Ann': 123}
file_name = 'test_dict.txt'
write(my_dict, file_name)
```

```
print(read(file_name))
```

```
{'Angela': 866, 'Adrian': 871, 'Ann': 123}
```

Writing JSON using pretty printing

```
data = { 'b':['HELLO', 'WORLD'], 'a':['hello', 'world'] }
```

- `json.dumps( data )`

double quotes

str type

```
'{"b": ["HELLO", "WORLD"], "a": ["hello", "world"]}'
```

- `json.dumps( data, indent=2, sort_keys=True )`

str type

```
'{\n  "a": [\n    "hello",\n    "world"\n  ],\n  "b": [\n    "HELLO",\n    "WORLD"\n  ]\n}'
```

- `print(json.dumps( data, indent=2, sort_keys=True ))`

```
{
    "a": [
        "hello",
        "world"
    ],
    "b": [
        "HELLO",
        "WORLD"
    ]
}
```

What about user-defined classes?

- Alternatively, you may use **`__dict__`** property which is already available in the Python objects.

```
print(p.__dict__)
```

```
{'x': 2, 'y': 3}
```

- We can take an advantage on the fact that every object stores all of its members in a dictionary called **`self.__dict__`**. Making a string representation of the object is just a matter of return a string representation of **`__dict__`**.

```
def __str__(self):
   return str(self.__dict__)
```

```
{'x': 2, 'y': 3}
```

```
def __str__(self):
    return json.dumps(self.__dict__, indent=2, separators=(',', ': '))
```

```
{
  "x": 2,
  "y": 3
}
```

Summary

- JSON is a standard way to exchange data
  - Easily parsed by machines
  - Human readable form
- Symbols used in JSON are the same as Python
  - **Double quotes** used for strings in JSON
- Python module **`json`** provides decoding and encoding functions.
  - **`load()`** and **`loads()`**, **`dump()`** and **`dumps()`**

학습 정리

1) JSON 자료형과 파이썬은 서로 호환되며 JSON 모듈을 import하여 쉽게 작업할 수 있다

2) load()함수는 JSON 파일을 읽어 파이썬 딕셔너리 객체를 반환하고, loads()함수는 JSON str 자료형 객체를 딕셔너리 객체로 변환한다

3) dump()함수는 파이썬 딕셔너리를 JSON 파일로 생성하고, dumps()함수는 딕셔너리를 JSON str 객체로 변환한다

파이썬으로 배우는 데이터 구조

수고했습니다
곧 다음 시간에
다시 뵙겠습니다