

7장. Dictionary 와 Tuple

챕터의 포인트

- Dictionary
- Tuple
- List와 Tuple 메서드

Dictionary

1 ~ 6 까지 수가 각각 몇개 있는지 출력하는 문제

- 출력결과

1 : 0개

2 : 2개

3 : 2개

4 : 0개

5 : 0개

6 : 1개

| | | | | |
|---|---|---|---|---|
| 3 | 3 | 2 | 6 | 2 |
|---|---|---|---|---|

1 ~ 6 까지 수가 각각 몇개 있는지 출력

- 추가 배열 1 개와, 1 중 for문으로 구현할 수 있다.
- 빈 배열 만들기 : `bucket = [0] * 7`

• 출력결과

1 : 0개
2 : 2개
3 : 2개
4 : 0개
5 : 0개
6 : 1개

| | | | | |
|---|---|---|---|---|
| 3 | 3 | 2 | 6 | 2 |
|---|---|---|---|---|

값을 index로 활용

| | bucket |
|---|--------|
| 0 | |
| 1 | |
| 2 | 2 |
| 3 | 2 |
| 4 | |
| 5 | |
| 6 | 1 |

배열을 활용한 해결 방법

```
main.py x
1 a = [3, 3, 2, 6, 2]
2 bucket = [0] * 7
3
4 for i in a:
5     bucket[i] += 1
6
7 for i in range(7):
8     print(str(i) + " : " + str(bucket[i]) + "개")
```

```
0 : 0개
1 : 0개
2 : 2개
3 : 2개
4 : 0개
5 : 0개
6 : 1개
```

```
a = [3, 3, 2, 6, 2]
```

```
bucket = [0] * 7
```

```
a[3] += 1
```

```
a[3] += 1
```

```
a[2] += 1
```

```
a[6] += 1
```

```
a[2] += 1
```

리스트로 이 문제를 풀 수 있을까?

Confidential

ABC / MC / BTS의 개수를 구하는 문제

- 원하는 출력결과

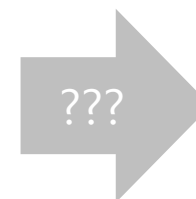
ABC : 0개

MC : 2개

BTS : 3개

| | | | | |
|----|-----|-----|----|-----|
| MC | BTS | BTS | MC | BTS |
|----|-----|-----|----|-----|

문자열을 index로 쓸 수 없다.
리스트로 풀 수 없다.



| | |
|---|--|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |

그래서 Dictionary를 사용한다.

- index로 문자열을 쓸 수 있다.

Dictionary 에서 index 역할 하는 값을
Key 값이라고 부른다.

| | | | | |
|----|-----|-----|----|-----|
| MC | BTS | BTS | MC | BTS |
|----|-----|-----|----|-----|

| | |
|-----|---|
| | |
| | |
| MC | 2 |
| BTS | 3 |
| | |
| | |
| | |
| | |

Dictionary 생성 두 가지 방법

```
a = dict()  
a['HI'] = 55  
a['BBQ'] = 'KFC'
```

또는

```
a = {'HI' : 55, 'BBQ' : 'KFC'}
```

```
main.py x  
1 a = dict()  
2 a['HAHA'] = 'KKK'  
3 a['OH'] = 345  
4 a[-4456] = 15367  
5  
6 b = {'HI' : 55, 'BBQ' : 'KFC'}  
7  
8 print(a[-4456])  
9 print(b["BBQ"])  
10
```

배열 index로 “문자열”을 사용 가능

- 다음과 같이 구현이 가능 (예시)

```
dt["MC"] += 1
dt["BTS"] += 1
dt["BTS"] += 1
dt["MC"] += 1
dt["BTS"] += 1
```

| | | | | |
|----|-----|-----|----|-----|
| MC | BTS | BTS | MC | BTS |
|----|-----|-----|----|-----|

1. 리스트 하드코딩
2. 가장 많이 등장하는 단어 출력하기

```
lst = ['MC', 'BTS', 'BTS', 'MC', 'BTS']

di = dict()
for i in lst:
    di[i] = 0

for i in lst:
    di[i] += 1
```

Key를 문자열로, Value를 숫자값으로 둔
Dictionary라고 한다.

Dictionary 는 다른 언어 에서도 사용되나요?

- C언어에는 없다.

지도가 아니라,
맵핑되어있다 라는 의미



- C++ : Unordered_map 이라는 이름으로 존재
- Java : HashMap 이라는 이름으로 존재
- C# : Dictionary 라는 이름으로 존재
- Python : Dictionary

많은 프로그래밍 언어에서
기본적으로 들어가 있는 필수 문법이니
꼭 알아두자!

다음 소스코드를 이해해보자.
왜 에러가 발생할까?

```
bbq.py x
1 dt = {'ABC':15, 'CCC':22}
2
3 dt['ABC'] += 1
4 dt['QQQQ'] += 1
5
```

초기화 부분
이렇게도 가능하다.

에러가 나는 코드

Key / Value 세트를 생성을 해야만,
그 이후로 사용할 수 있다.

- **값 대입을 하면**, 간단히 Key에 해당하는 공간을 생성할 수 있다.

```
bbq.py ×  
1 dt = {'ABC':15, 'CCC':22}  
2  
3 dt['ABC'] += 1  
4 dt['QQQQ'] += 1  
5
```

에러나는 코드

```
main.py ×  
1 dt = {'ABC':15, 'CCC':22}  
2  
3 dt['ABC'] += 1  
4  
5 dt['QQQQ'] = 0  
6 dt['QQQQ'] += 1  
7
```

정상적인 코드

한 문자열을 입력 받은 후,
그 문자열이 몇 개 존재하는지 출력한다.

- 리스트에 없는 문자열은 입력되지 않는다.
- Dictionary 를 이용하여 구현한다.

| | | | | |
|-----|----|-----|-----|-----|
| ABE | 53 | -99 | -99 | 124 |
|-----|----|-----|-----|-----|

str(i)를 Key 값으로 지정한다.

- 사용자가 53을 입력하면,
문자열 53인지
수 53인지 구분이 안가기에, 모두 str로 처리했다.

```
list = ['ABE', 53, -99, -99, 124]
d = dict()

for i in list: d[str(i)] = 0
for i in list:
    d[str(i)] += 1

a = input()
print(d[a])
```

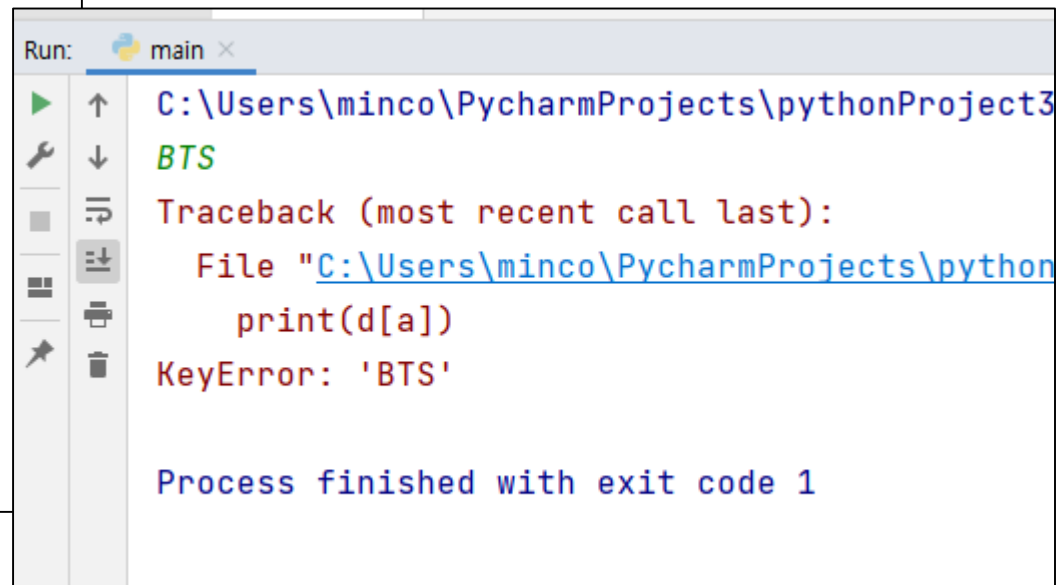
해당 소스코드를 입력값으로 “BTS”를 입력해보자.

- dictionary에 생성되지 않은 **Key값을 읽으려는 시도로 인해**, Key Error 가 발생했다.

```
list = ['ABE', 53, -99, -99, 124]
d = dict()

for i in list: d[str(i)] = 0
for i in list:
    d[str(i)] += 1

a = input()
print(d[a])
```



```
Run: main x
C:\Users\minco\PycharmProjects\pythonProject3
BTS
Traceback (most recent call last):
  File "C:\Users\minco\PycharmProjects\python
    print(d[a])
KeyError: 'BTS'

Process finished with exit code 1
```


Key 값이 존재하는지 확인하는 방법! **in**을 사용한다.

- 값을 수정하기 전, Key값이 있는지 검사를 해야 한다.

if ('abc' **in** dt) :

- abc값이 dt 안에 있는지 검사

if ('abc' **not in** dt) :

- abc값이 dt 안에 없는지 검사

```
main.py x
1  lst = ['MC', 'BTS', 'BTS', 'MC', 'BTS']
2  go = dict()
3
4  for i in lst:
5      if (i not in go) : go[i] = 0
6      go[i] += 1
7
8  print(go['MC'])
9  print(go['BTS'])
10
```

문자열 하나를 입력받고, 예약된 숫자 출력

- Dictionary를 이용하여 문제를 풀 것.

KFC 입력하면 → 10 출력

MC 입력하면 → 20 출력

MOMS 입력하면 → 30 출력

그 외의 값을 입력 받는 경우 → 1000 출력

| | |
|------|----|
| KFC | 10 |
| MC | 20 |
| MOMS | 30 |

키 값을 모두 출력한다.

- keys() 메서드 이용하기
- print(dt.keys()) 를 할 경우, 모든 키 값이 출력된다.

keys()와 for 문을 이용하여 모든 키값을 출력할 수 있다.

```
d = {'KFC':10, 'MC':20, 'MOMS':30}

print(d.keys())
```

main ×
C:\Users\minco\PycharmProjects\python
dict_keys(['KFC', 'MC', 'MOMS'])

```
dt = dict()
dt[15] = 1
dt[20] = 2
dt['ABC'] = 5

for i in dt.keys():
    print(i, dt[i])
```

해당 코드를 해석해보자.

존재하는 단어, 각각 Counting

- 배열에 들어있는 각 단어들마다
몇 개가 존재하는지 출력하는 프로그램 작성
 - dictionary 자료구조를 사용

- 출력결과

MC : 2 개

BTS : 3 개

| | | | | |
|----|-----|-----|----|-----|
| MC | BTS | BTS | MC | BTS |
|----|-----|-----|----|-----|

Tuple 자료형

덩어리 값이다!

- $a = \text{"HI"}$
- $b = (3, 4, 6)$

"HI" 라는 값, 한 덩어리!

(3, 4, 6) 이라는 값, 한 덩어리!

함수 리턴하기

- 값 하나를 리턴할 수 있다.

여러 값을 리턴 하고 싶을 때, 튜플로 만들어서 리턴하면 된다.

```
def bts():  
    return 10  
  
a = bts()  
print(a)
```

객체 하나 리턴하기

```
def bts():  
    return (3, 4, 5)  
  
a = bts()  
print(a)
```

튜플이라는
객체 하나 리턴하기

괄호를 사용 안해도,
튜플로 인식한다.

- `type(a)` 도 출력해보자.

두 코드는 같다.

- `a = (3, 4, 5)`
- `a = 3, 4, 5`

```
main.py ×  
1 a = 3, 4, 5  
2  
3 print(a)  
4
```

(3, 4, 5) 한덩어리

for 사용 가능

- Python 스타일 for문
- C언어 스타일 for문

둘 다 사용 가능

```
main.py x
1 def abc():
2     return 1, 2, 3, 4, 5, 6
3
4
5 t = abc()
6 print(t)
```

```
a = 3, 4, 5

for i in a:
    print(i, end = ' ')

print()

for i in range(3):
    print(a[i], end = ' ')
```

튜플 안에 있는 내용들을
a, b, c 객체에 각각 넣었다.

```
main.py x
1 (a, b, c) = (1, 2, 3)
2
3 print(a)
4 print(b)
5 print(c)
```

```
main.py x
1 a, b, c = "ABC", 14, 555
2
3 print(a)
4 print(b)
5 print(c)
```

```
main.py x
t = (1, 2, 3)
a, b, c = t

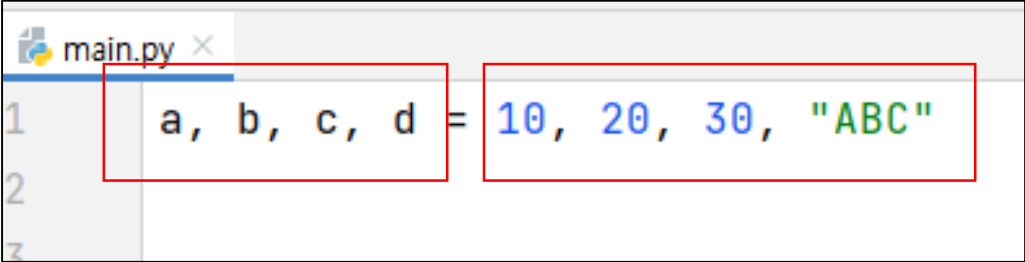
print(a)
```

이 코드는 튜플이 사용되었을까?

Confidential

a, b, c, d 각각에
값이 10, 20, 30, "ABC" 이다.

튜플이 눈에 보이는가?

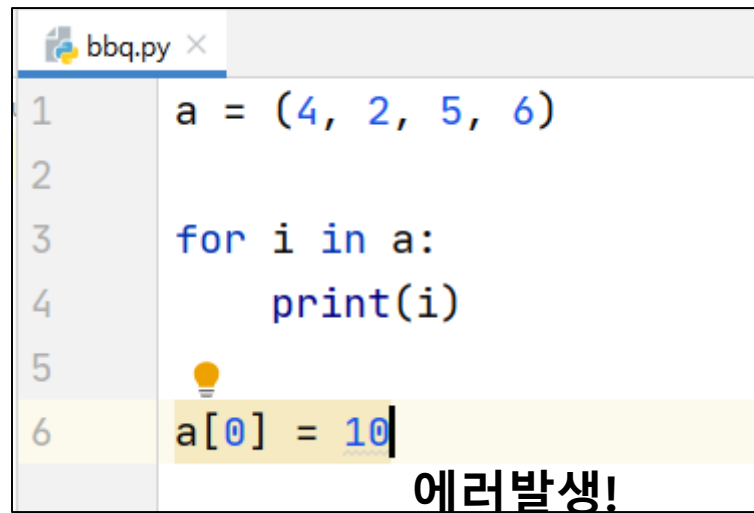


```
main.py x
1 a, b, c, d = 10, 20, 30, "ABC"
2
3
```

The screenshot shows a code editor window titled 'main.py'. The first line of code is 'a, b, c, d = 10, 20, 30, "ABC"'. The variables 'a, b, c, d' are enclosed in a red box, and the values '10, 20, 30, "ABC"' are also enclosed in a red box. The line numbers 1, 2, and 3 are visible on the left side of the editor.

Tuple은 요소의 값을 수정 **할 수 없다.**

- 만약 `a = (1, 2, 3)` 에서, `(1, 2, 3)`은 하나의 값 덩어리이다.
- `(1, 2, 3)` 에서 하나의 요소의 값을 제거하거나 수정할 수 없다.



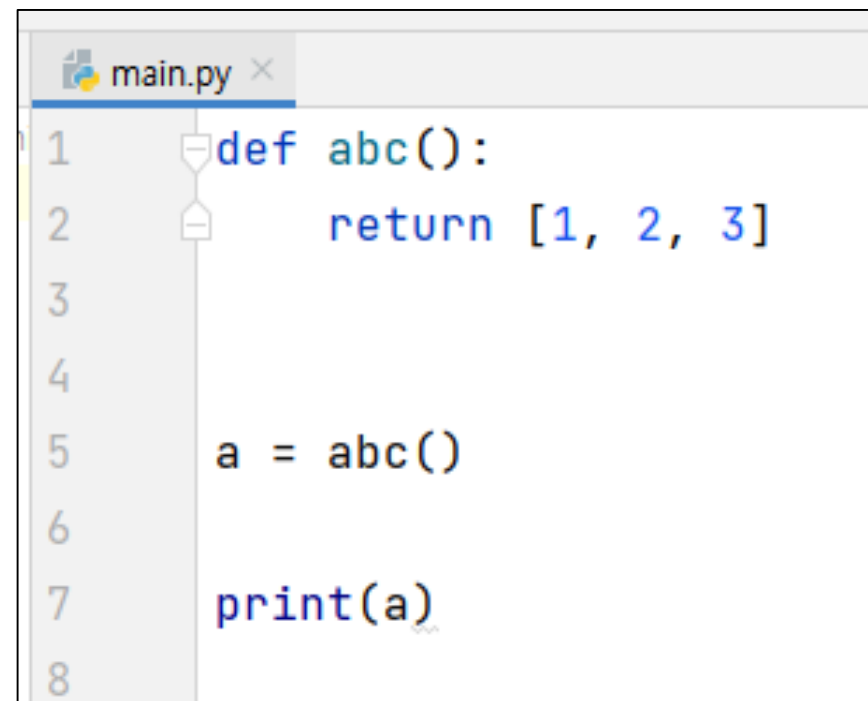
The screenshot shows a Python IDE window titled 'bbq.py'. The code is as follows:

```
1 a = (4, 2, 5, 6)
2
3 for i in a:
4     print(i)
5
6 a[0] = 10
```

Line 6 is highlighted in yellow, and a lightbulb icon is shown next to it, indicating an error. The text '에러발생!' (Error occurred!) is written below the code.

리스트 객체도,
튜플처럼 반환 된다.

리스트 객체도 사실
여러 값이 있는 **값 하나**로 취급된다.



```
main.py x
1  def abc():
2      return [1, 2, 3]
3
4
5  a = abc()
6
7  print(a)
8
```

그럼 뭐하러 튜플 쓰나요?

Confidential

리스트 쓰면 되는데
뭐하러 튜플 쓰나요?

상수값이기 때문

그 이상 이유는 미미하다.

- 메서드가 적어, 내부 메모리를 적게 사용된다.

튜플 쓰던지, 리스트 쓰던지 편하게 선택해서 쓰자.

단, 타인의 코드는 이해할수는 있어야 하기에
튜플을 학습한다.

다음 소스코드를 작성하고,
각자 동작을 분석해보자.

```
bbq.py x
1 def run(a):
2     for i in a:
3         print(i)
4
5 run((1, 2, 3, 4, 5))
6
```

여는 괄호가 2개 ((
닫기 괄호가 2개))

```
def abc():
    return 1, 2, 3, 4

print(abc())
```

```
bbq.py x
1 def run():
2     return 1, 2, 3
3
4 a, b, c = run()
5
6 print(a, b, c)
```


어떤 값이 출력될까?

```
main.py x
1 def abc( value ):
2     a = value[0]
3     print(a)
4
5 abc([1, 2, ('A', 'B'), [1, 2, [("KFC", "MOMS", "BHC")]]]);
```

어떤 값이 출력될까?

```
main.py x
1 def abc( value ):
2     a = value[2]
3     print(a)
4
5 abc([1, 2, ('A', 'B'), [1, 2, [("KFC", "MOMS", "BHC")]]]);
6
7
```

어떤 값이 출력될까?

```
main.py x
1 def abc( value ):
2     a = value[2][0]
3     print(a)
4
5 abc([1, 2, ('A', 'B'), [1, 2, [("KFC", "MOMS", "BHC")]]]);
6
```

치킨 브랜드의 이름을 숨겨두었다.

- for문을 돌려, 치킨 브랜드 이름만 출력하는 프로그램을 작성하자.

출력결과

KFC
MOMS
BHC

```
main.py x
1 def abc( value ):
2     ???
3
4
5 abc([1, 2, ('A', 'B'), [1, 2, [("KFC", "MOMS", "BHC")]]]);
6
```

최종 정답코드

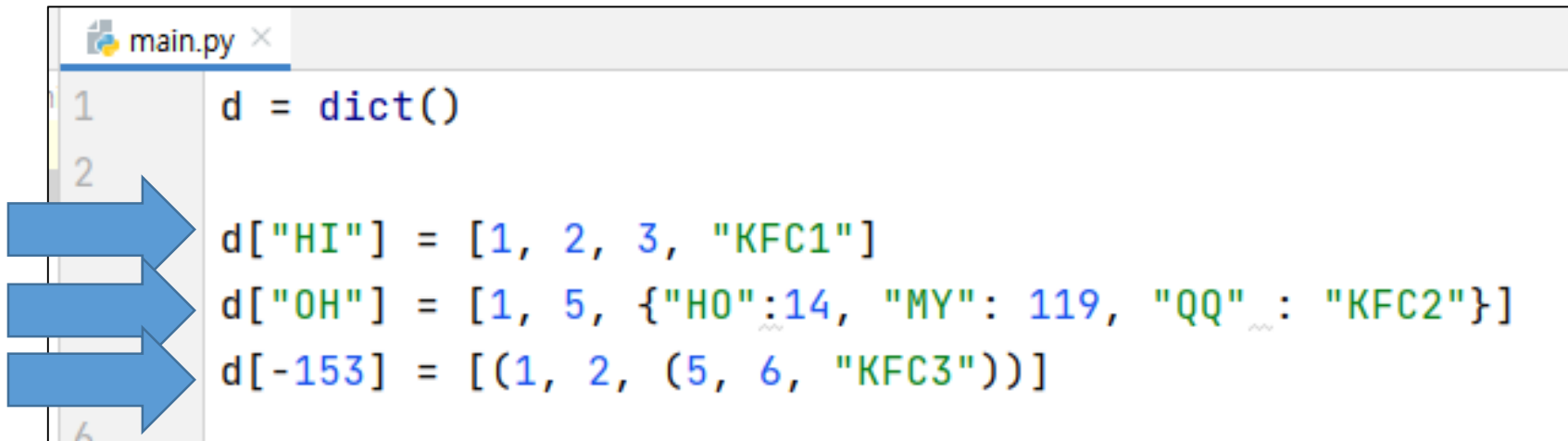
```
main.py x
1 def abc( value ):
2     a = value[3][2][0]
3     print(a)
4
5 abc([1, 2, ('A', 'B'), [1, 2, [("KFC", "MOMS", "BHC")]]]);
6
7
```



```
main.py x
1 def abc( value ):
2     a = value[3][2][0]
3     for i in a:
4         print(i)
5
6 abc([1, 2, ('A', 'B'), [1, 2, [("KFC", "MOMS", "BHC")]]]);
7
8
```

KFC1, 2, 3 형제를 구출하자.

- d["HI"], d["OH"], d[-153] 에서 각각 KFC 값을 꺼내서, print 명령어로 출력한다.



```
main.py x
1 d = dict()
2
3 d["HI"] = [1, 2, 3, "KFC1"]
4 d["OH"] = [1, 5, {"HO": 14, "MY": 119, "QQ": "KFC2"}]
5 d[-153] = [(1, 2, (5, 6, "KFC3"))]
6
```

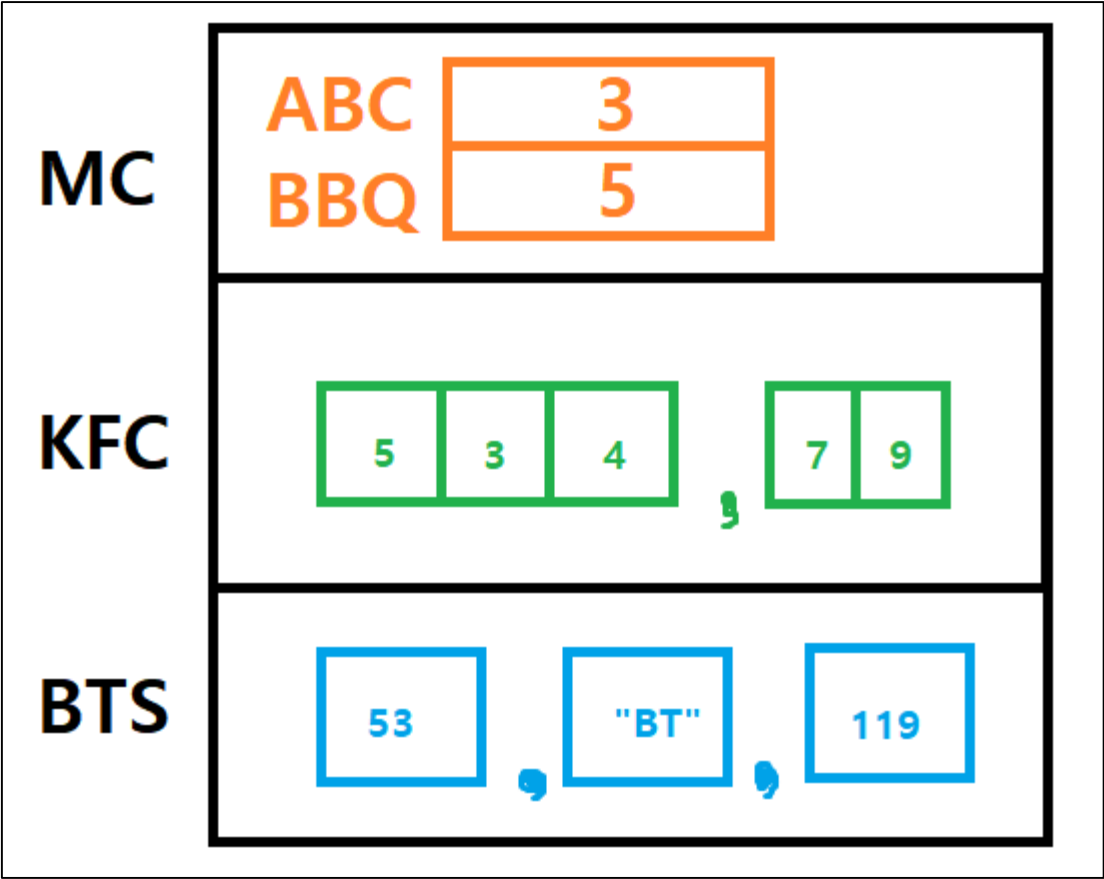
[도전] 다음 그림대로 구현해보자.

Confidential

이곳에 튜플은 없다.

Dictionary와 List로 구현

다음 그림처럼
구현해보자.



```
main.py x
1 d = dict()
2
3 d["HI"] = [1, 2, 3, "KFC1"]
4 d["OH"] = [1, 5, {"HO":14, "MY": 119, "QQ": "KFC2"}]
5 d[-153] = [(1, 2, (5, 6, "KFC3"))]
6
```

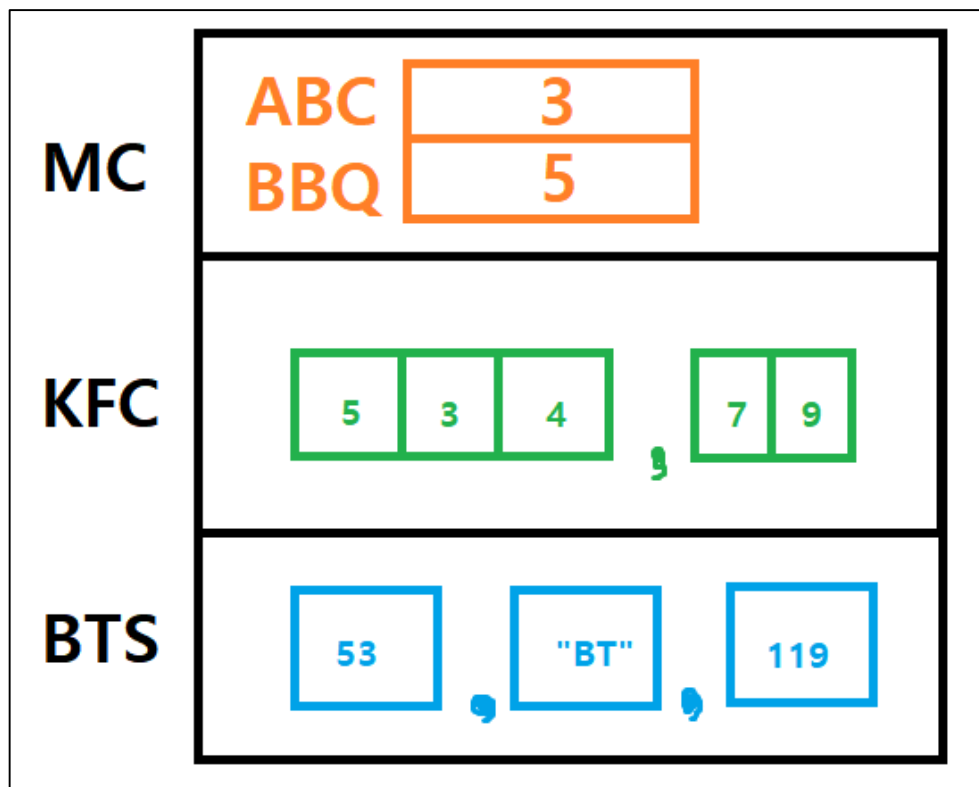
다음 방식으로 구현도 가능하다.

Confidential

엔터를 적절히 넣어,
구조를 이해하기 쉽게 표현해도 된다.

- 본인의 정답을, 엔터를 넣어 표현해보자.

```
main.py x
1 value = {
2     "MC" : {
3         "ABC":3,
4         "BBQ":5
5     },
6     "KFC" : [
7         [5,3,4],
8         [7, 9]
9     ],
10    "BTS" : [53, "BT", 119]
11 }
12
13
```



8장. SWEA 문제풀기 HARD

SWEA 풀기 전, 사용할 메서드 소개

특정 값을 찾아서, index 값을 리턴하는 방법

- 값이 없다면 error
- in 을 이용하여 내부에 값이 있는지 먼저 확인하자.

```
bbq.py x
1 a = tuple([1, 2, 377])
2
3 print(a.index(2))
/
```

```
bbq.py x
1 a = [1, 2, 55, 4, 11]
2
3 print(a.index(55))
/
```

List에 들어있는 값을 쉽게 Counting 가능

- `count(요소)`

```
bbq.py x
1 a = [4, 3, 4, 3, 4, 3, 4]
2
3 print(a.count(4))
```

```
bbq.py x
1 b = (4, 3, 4, 3, 4, 3, 4)
2
3 print(b.count(3))
```

max값과 min값을 쉽게 구할 수 있음

- max([]) 또는 max(()) 사용
- Tuple / List 모두 사용 가능

```
bbq.py x
1 b = (4, 3, 4, 3, 4, 3, 4)
2
3 print(max(b))
```

List / Tuple 특정 범위의 객체 새롭게 생성

- 슬라이싱 형태
 - a[시작부터 : 전까지 : Step]
- print(a[1 : 5 : 1])
 - a[1], a[2], a[3], a[4] 값을 출력한다.
- print(b[0 : 6 : 3])
 - b[0], b[3] 값을 출력한다.

```
bbq.py x
1 a = [1, 5, 4, 5, 3, 2]
2
3 b = a[0:6:2]
4 print(b)
```

[1, 4, 3]

특정 문자만 추출할 수 있다.

- 3부터 5 전까지 글자 추출
- 5부터 3 전까지 글자 추출

```
bbq.py x
1 a = "ABCDEFGF"
2
3 print(a[3:5])
4 print(a[5:3:-1])
```

DE

FE

SWEA 문제 풀기

목표

- DAY 2 문제 풀기
- 제한 시간 내, 최대한 많이 풀기!

다 풀고 코딩테스트 기출 **유사** 문제 풀 예정

9장. JSON

챕터의 포인트

- JSON Format
- Encoding / Decoding
- JSON 파일 파싱하기

JSON Format

데이터들을 교환할 때 정해진 규격

- Dictionary 자료구조와 비슷하다.
- 읽기 쉽다.

```
{  
    "name": "sanghai",  
    "price": 4,900,  
    "brand": "mcdonald"  
}
```

```
kfc.py x  
1  a = {  
2      'name': 'sanghi',  
3      'price': 4900,  
4      'brand': 'mcdonald'  
5  }  
6  
7  print(a)
```

표준파일교환 Format

- WEB에서 AJAX 기술에 사용된다.
- API 사용시에 사용된다.
- Database (NoSQL)에 사용된다.
- 환경 설정 값 저장용

JSON 문법 연습

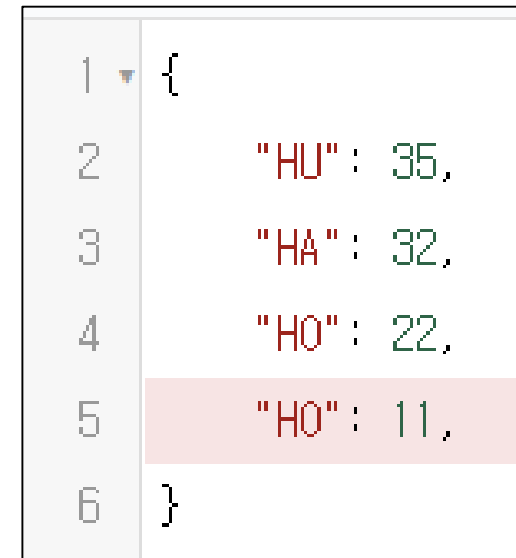
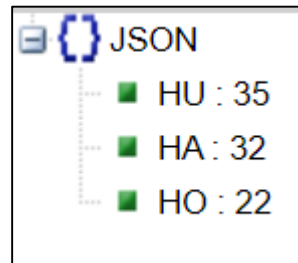
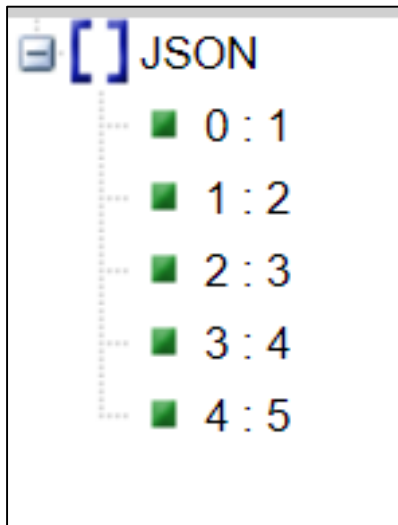
- <https://jsonlint.com/>에서 Valid Check
- <http://jsonviewer.stack.hu/> 으로 Visualization

JSON Sample

- <https://json.org/example.html>

JSON은 array / object 로 시작한다.

- [] 괄호로 묶여있는 것이 배열이다.
- { } 괄호로 묶여있는 것이 객체이다. (Class와 관련 없다. Dictionary와 비슷)
- 객체는 중복 Key 값을 허용하지 않는다.



객체의 기본 규칙

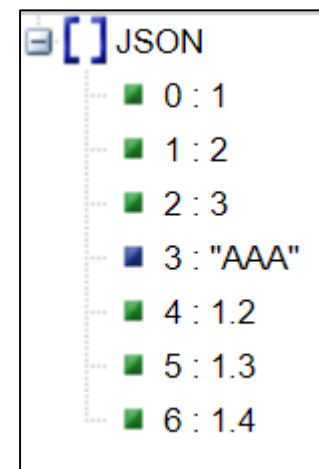
- 모든 키값은 **쌍 따옴표로 반드시 묶여야 한다.**
- 숫자와 소숫점은 쌍따옴표가 없어도 되지만, 문자열은 쌍따옴표 필수
- 콤마는 정확히 맞추어야 한다.

| Name ▲ | Value |
|--------|------------|
| brand | "mcdonald" |
| name | "sanghai" |
| price | 4900 |

| Name ▲ | Value |
|--------|--------|
| age | 20 |
| height | 160.7 |
| name | "alex" |
| weight | 42.1 |

배열의 기본 규칙

- 중복된 값이 있어도 된다.
- 숫자, 문자열, 소수점 등 기입이 가능하다.

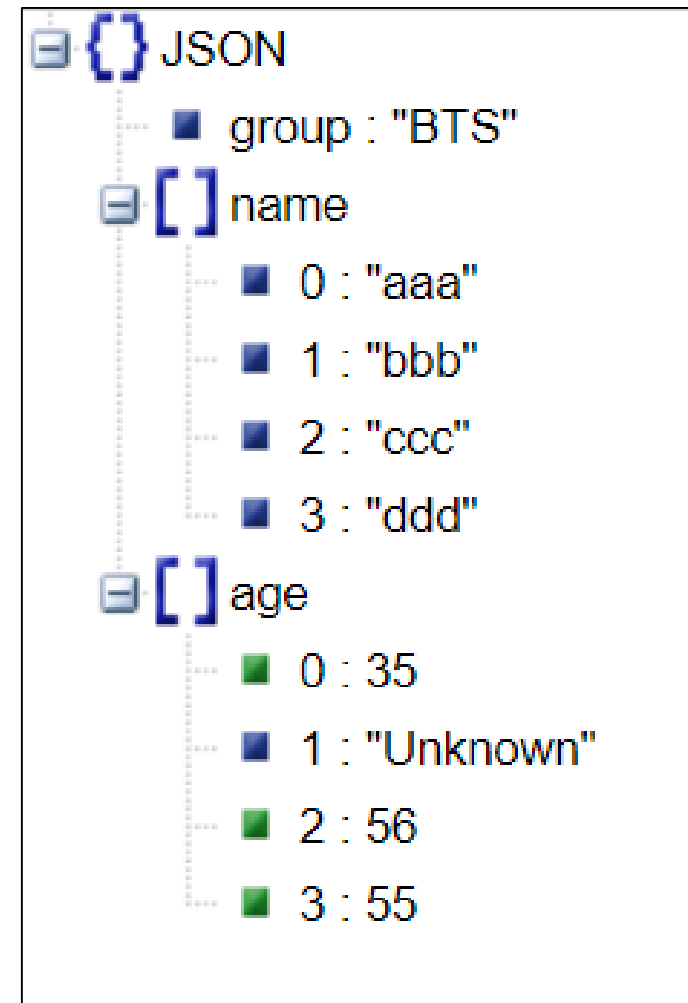


객체 내 배열 형태로도 저장이 가능

- “name” = [“aaa”, “bbb”, ...],
- 타입은 숫자와 문자열이 섞여도 된다.

오른쪽 결과를 실제 json 만들기

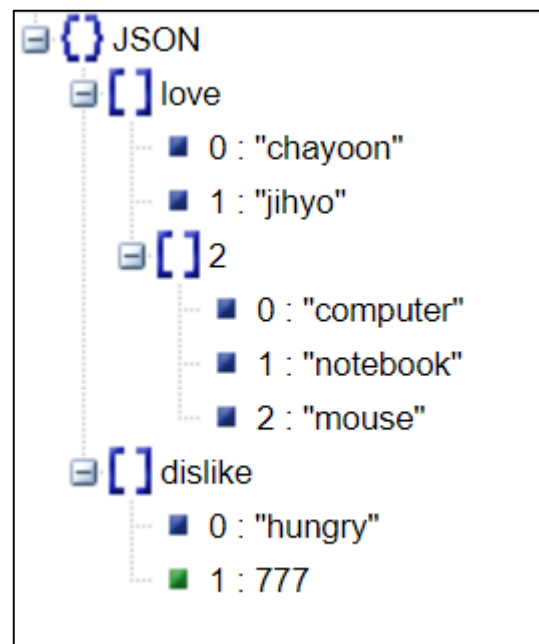
- lint 검사 해준 후,
Visulizer로 똑같이 나오는지 확인하기



배열 안, 배열 사용 가능

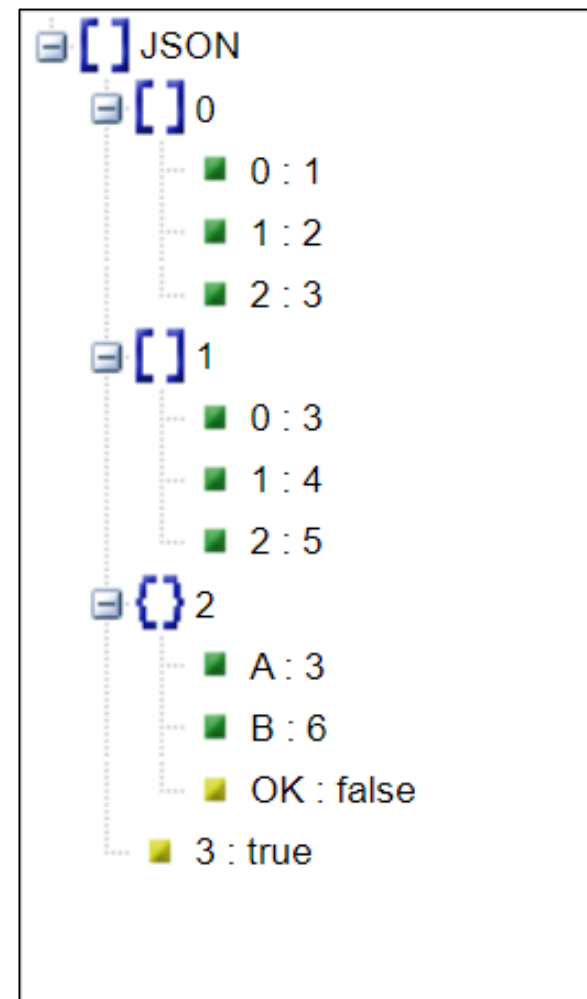
- [] 내부에 [] 사용 가능
- , 컴마를 정확히 지켜주어야 함

객체 내, 객체 사용 가능



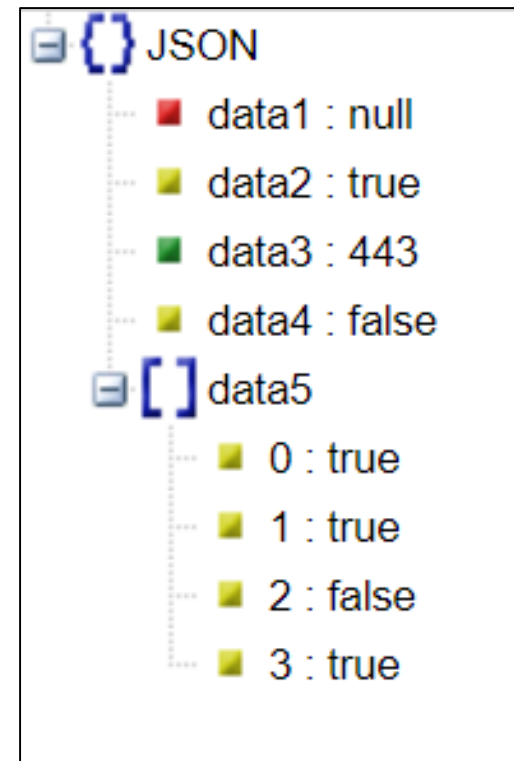
true / false도 집어넣을 수 있음

- 사용 가능한 타입
 - 배열 / 객체
 - string
 - number
 - 소수점
 - Boolean 타입



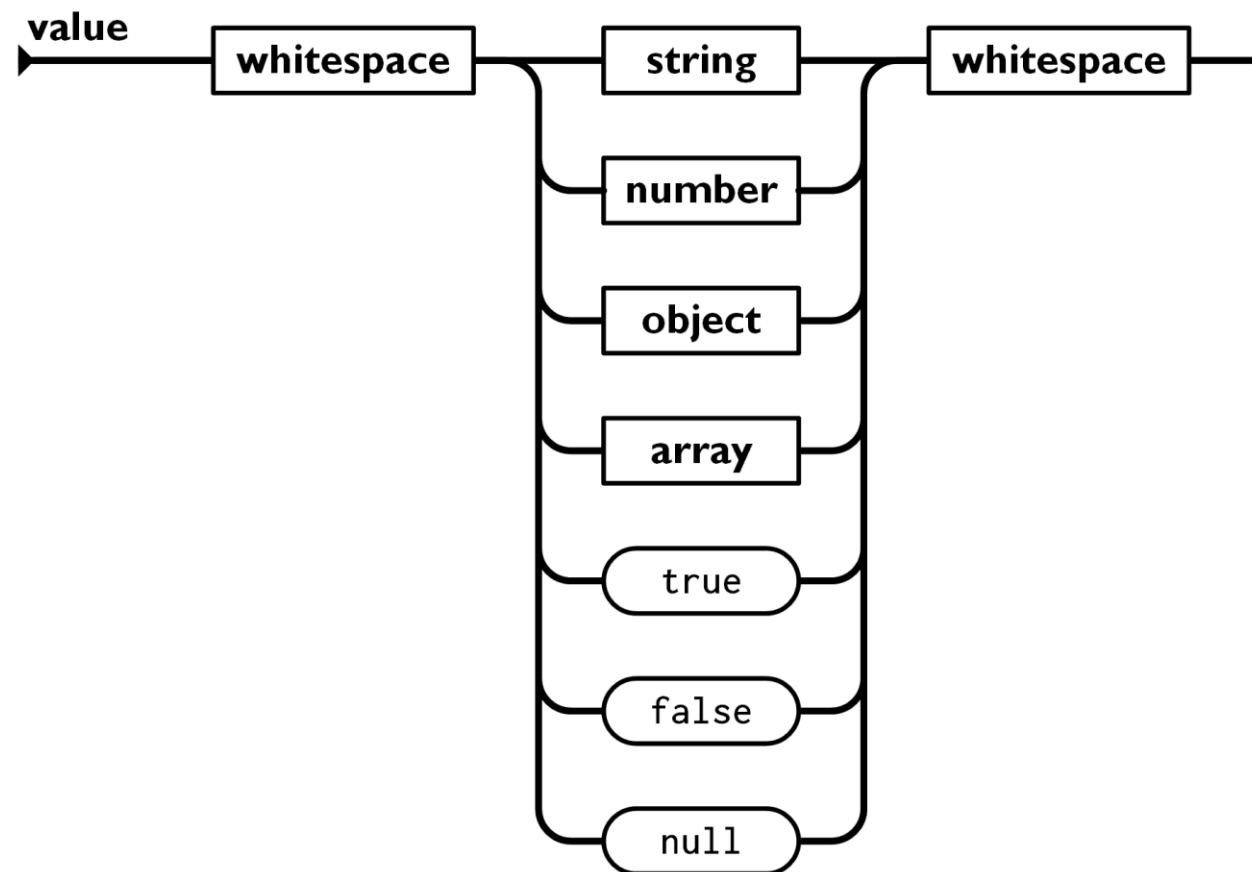
Boolean과 null 사용

- true / false 사용 가능
- null 사용 가능, 비어 있음을 나타내는 '값'



JavaScript Objection Notation

- JavaScript에서 파생 되었고,
언어 독립적으로 쓸 수 있다.



Wikipedia 예제

```
1 {  
2   "이름": "홍길동",  
3   "나이": 25,  
4   "성별": "여",  
5   "주소": "서울특별시 양천구 목동",  
6   "특기": ["농구", "도술"],  
7   "가족관계": {"#": 2, "아버지": "홍판서", "어머니": "춘섬"},  
8   "회사": "경기 수원시 팔달구 우만동"  
9 }
```


Encoding / Decoding

Python 기준에서, Encoding

- Python 객체를 JSON Data로 쓰는 동작

Python 기준에서, Decoding

- 읽은 JSON Data를 Python 객체로 저장한다.

| 파이썬 | → | JSON |
|----------------------------------|---|------------------|
| dict | | 오브젝트 (object) |
| list, tuple | | 배열(array) |
| str | | 문자열(string) |
| int, float, int와 float에서 파생된 열거형 | | 숫자(number) |
| True | | true |
| False | | false |
| None | | null |

Encoding

| JSON | → | 파이썬 |
|--------------|---|-------|
| 오브젝트(object) | | dict |
| 배열(array) | | list |
| 문자열(string) | | str |
| 숫자 (정수) | | int |
| 숫자 (실수) | | float |
| true | | True |
| false | | False |
| null | | None |

Decoding

JSON 내장 Package

- `json.dumps()` : Encoding 메서드
Dictionary 객체를 손쉽게 Encoding 가능
`str type` 객체를 생성한다.

```
kfc.py x
1 import json
2
3 a = dict()
4 a['name'] = 'sanghi'
5 a['price'] = 4900
6 a['brand'] = 'mcdonald'
7
8 b = json.dumps(a, indent=4)
9 print(b)
```

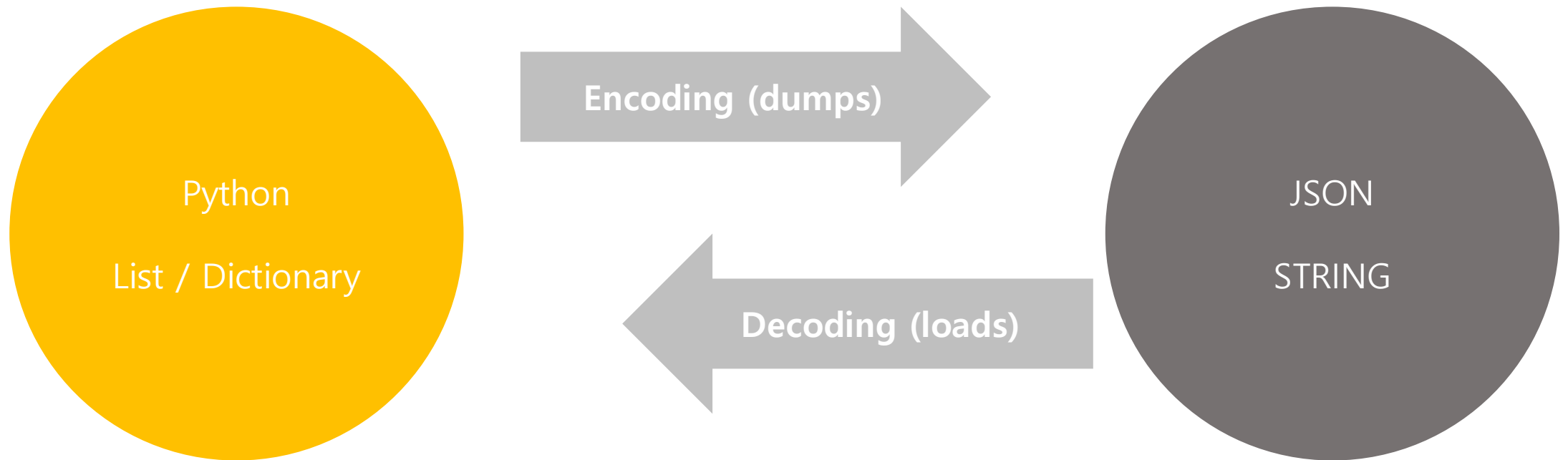
```
{
    "name": "sanghi",
    "price": 4900,
    "brand": "mcdonald"
}
```

Decoding 메서드

- `a = json.loads(str객체)`
- 유의사항 : load**s** 이다. (load라고 쓰지 말것)
string 을 파싱하여 python 객체로 만든다.

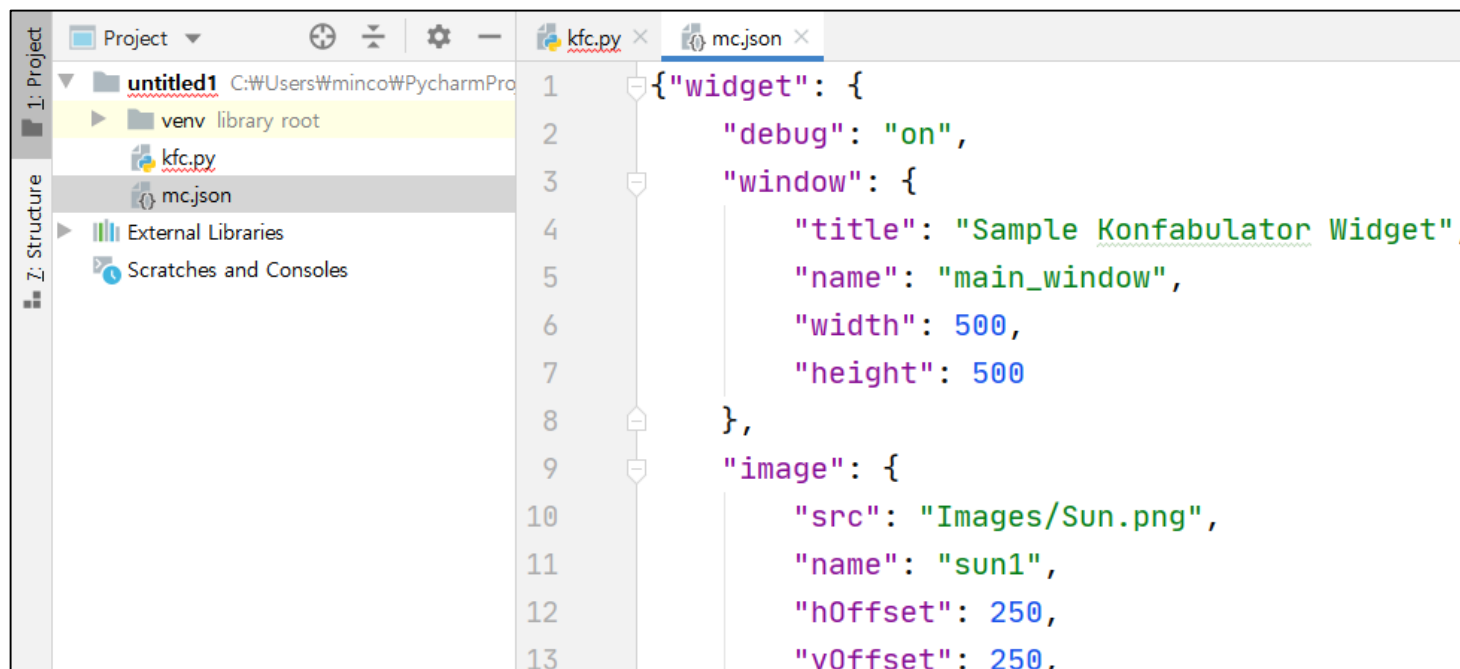
```
kfc.py x
1  import json
2
3  a = dict()
4  a['name'] = 'sanghi'
5  a['price'] = 4900
6  a['brand'] = 'mcdonald'
7
8  b = json.dumps(a, indent=4)
9
10 c = json.loads(b)
11 print(c)
```

```
{'name': 'sanghi', 'price': 4900, 'brand': 'mcdonald'}
```



.json 파일을 생성한다.

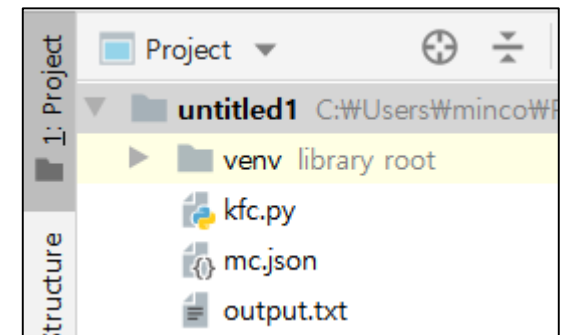
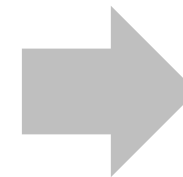
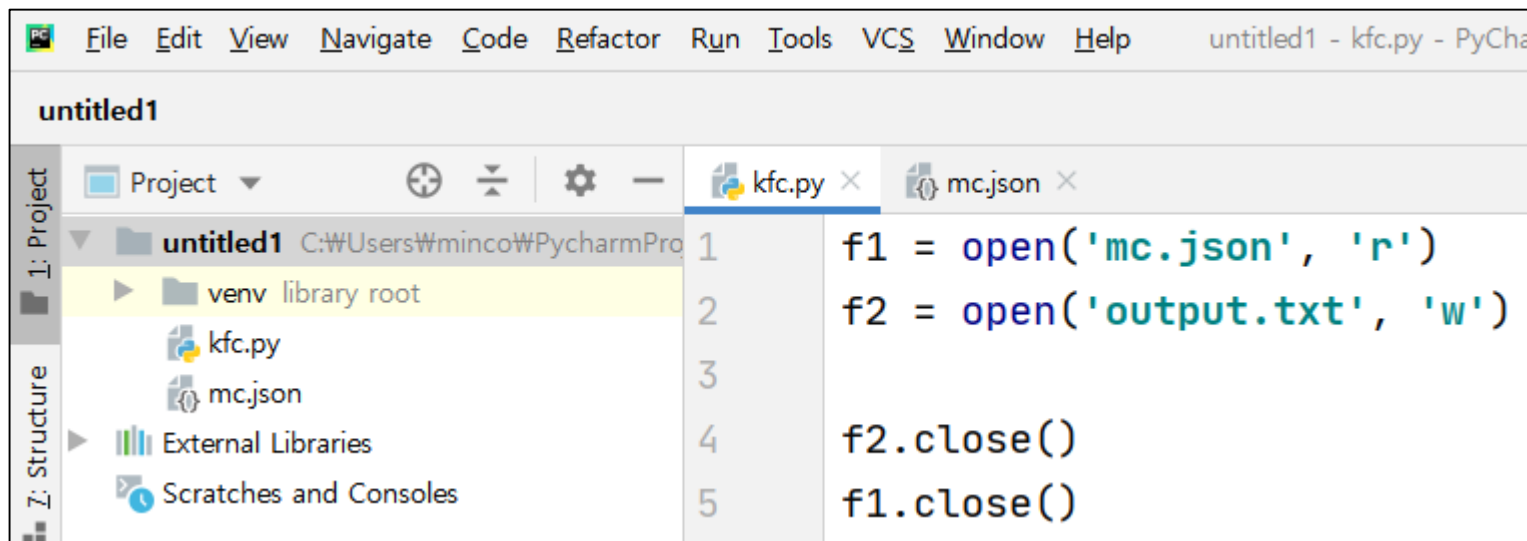
- <https://json.org/example.html> 에서 다음 예제를 복사
- 프로젝트 내 .json 파일을 생성하여 저장한다.



```
1 {"widget": {
2     "debug": "on",
3     "window": {
4         "title": "Sample Konfabulator Widget",
5         "name": "main_window",
6         "width": 500,
7         "height": 500
8     },
9     "image": {
10        "src": "Images/Sun.png",
11        "name": "sun1",
12        "hOffset": 250,
13        "vOffset": 250,
```

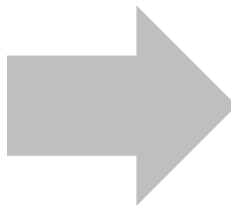
read / write 모드로 file open을 수행

- write open 시 자동으로 프로젝트 내 파일이 추가 됨
- 'a' 모드로 읽으면 마지막 내용을 추가하는 방식



파일을 전체 읽어서 출력해보자

```
kfc.py x mc.json x
1 f1 = open('mc.json', 'r')
2 f2 = open('output.txt', 'w')
3
4 txt = f1.read()
5 print(txt)
6
7 f2.close()
8 f1.close()
```



```
{"widget": {
  "debug": "on",
  "window": {
    "title": "Sample Konfabulator Widget",
    "name": "main_window",
    "width": 500,
    "height": 500
  },
  "image": {
    "src": "Images/Sun.png",
    "name": "sun1",
    "hoffset": 250,
```


output.txt에 모두 저장해보자

- with 문을 사용하면 자동으로 close 처리가 된다.

```
kfc.py x output.txt x mc.json x
1 f1 = open('mc.json', 'r')
2 f2 = open('output.txt', 'w')
3
4 txt = f1.read()
5 f2.write(txt)
6 print(txt)
7
8 f2.close()
9 f1.close()
10
```



```
kfc.py x output.txt x
1 txt = ""
2
3 with open('mc.json', 'r') as f1:
4     txt = f1.read()
5
6 with open('output.txt', 'w') as f2:
7     f2.write(txt)
8
```

해당하는 요소만 뽑아내기

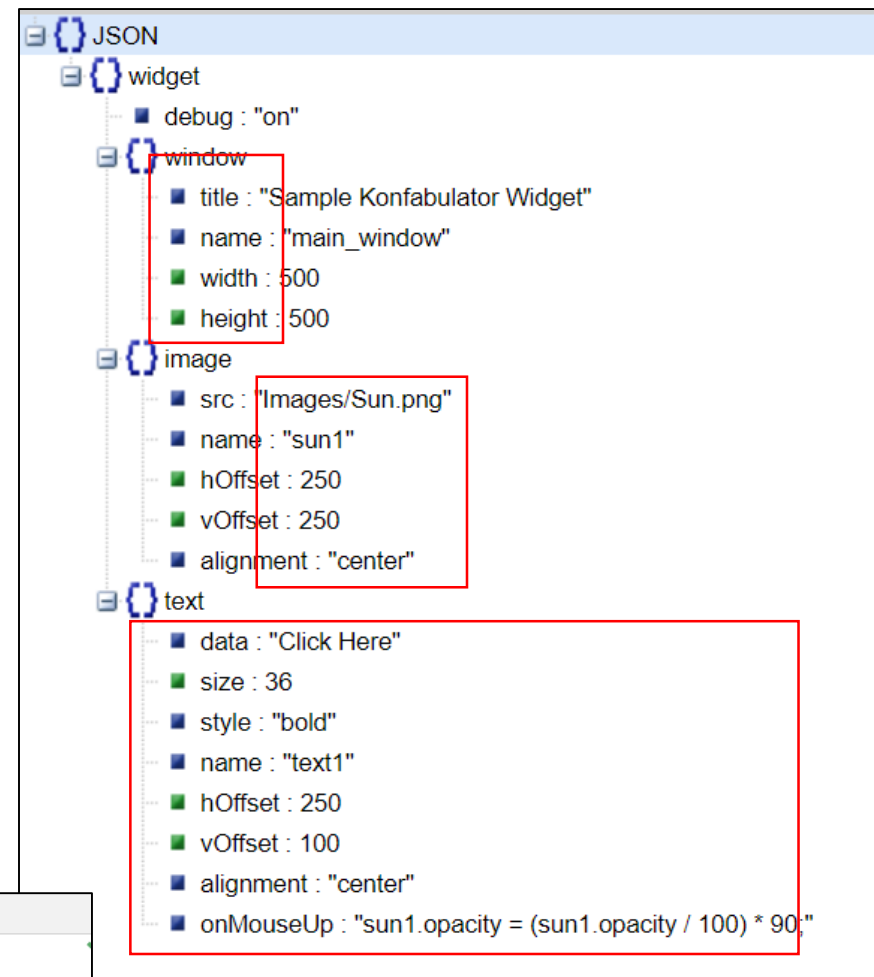
파일 출력을 한다 → output.txt

- window 객체에서 key만, 모두 파일 출력
- image 객체에서 value만, 모두 파일 출력
- text 객체에서 items 메서드를 사용해서, key, value 세트 모두 출력하기

[참고] Dictionary 메서드

- keys()
- values()
- items()

```
kfc.py x output.txt x
1 title name width height
2 Images/Sun.png sun1 250 250 center
3 ('data', 'Click Here') ('size', 36) ('style', 'bold') ('name', 'text
```



JSON 파일 파싱하기

Awesome JSON Datasets

- JSON 파일을 공유하는 사이트
<https://github.com/jdorman/awesome-json-datasets>
- JSON 구조를 쉽게 파악할 수 있는 사이트
<http://jsonviewer.stack.hu/>
- **TV Shows 에 Netflix 의 Narcos 클릭**
 - URL 을 복사하자

TV Shows

- Mr. Robot (USA)
- Better Call Saul (AMC)
- Homeland (Showtime)
- Silicon Valley (HBO)
- The Walking Dead (AMC)
- South Park (Comedy Central)
- Game of Thrones (HBO)
- House of Cards (Netflix)
- The Big Bang Theory (CBS)
- Narcos (Netflix)
- Black Mirror (Netflix)
- Stranger Things (Netflix)
- Rick and Morty (Adult Swim)
- Westworld (HBO)

JSON 파일을 가져오기

- requests Package 설치
- json 파일 읽어오기

```
kfc.py x output.txt x
1 import requests
2
3 r = requests.get('주소')
4 print(r.text)
```

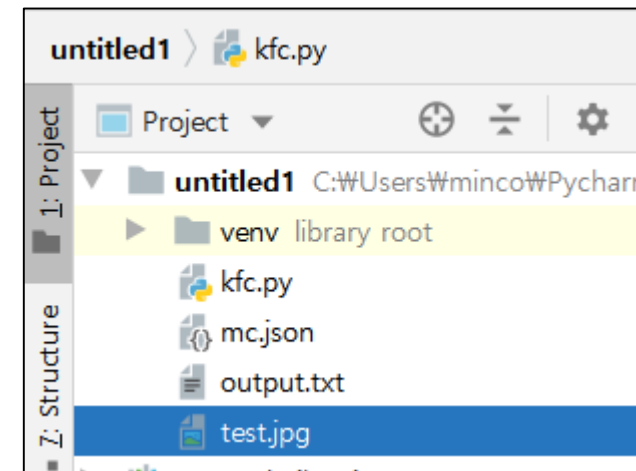
```
import json
import requests

url = 'http://api.tvmaze.com/singlesearch/shows?q=westworld&embed=episodes'
r = requests.get(url)
print(r.text)
```

urllib package

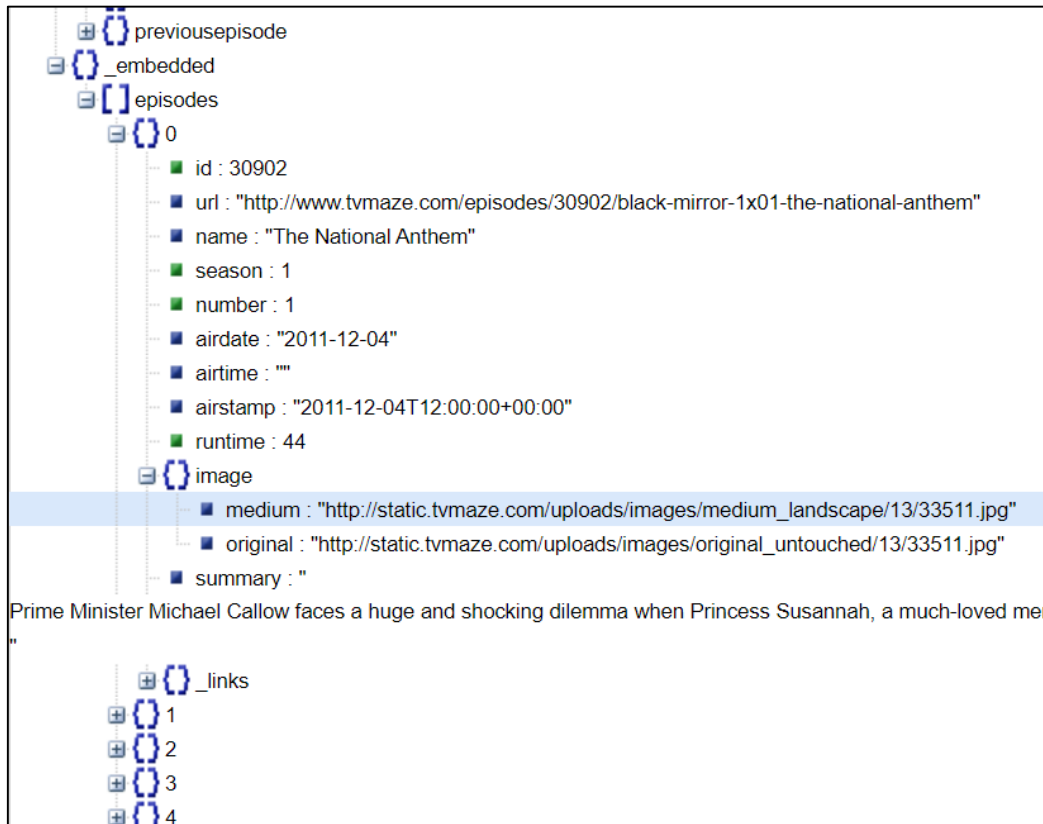
- 파이썬 기본 내장 Package
- URL 을 통해 접속 가능, requests package의 소규모 버전
- urllib의 urlretrieve 메서드로 쉽게 이미지 다운로드 가능

```
kfc.py x output.txt x
1 import urllib.request
2 url = 'http://static.tvmaze.com/uploads/images/or
3 urllib.request.urlretrieve(url, "test.jpg")
```



모든 에피소드의 Image를 다운로드 받는 프로그램 제작하기

- medium (작은사이즈의 이미지) 만 다운로드 받을 것.



10장. 공공데이터 Open API 사용하기

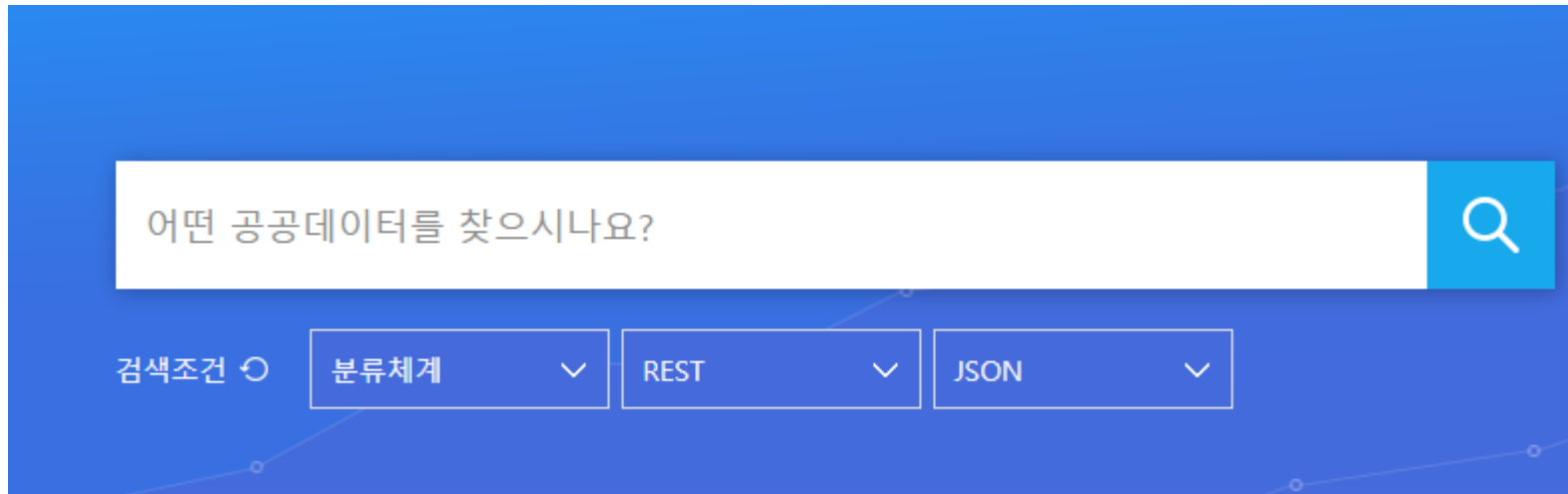
Application Programming Interface

- App에서 사용할 수 있게끔 만든 Interface
- App과 서비스 제공자의 연결고리 = API
 - ex. Windows API
 - ex. 증권사 API

공공데이터포털

- <https://www.data.go.kr/>
- 공공기관이 만들고, 관리하는 데이터의 종합 창구
- 국민들에게 JSON / XML 등 형태로, 공공 데이터를 제공

회원가입 후 REST / JSON 선택하여 검색



어떤 공공데이터를 찾으시나요?

검색조건 ↻ 분류체계 REST JSON

The image shows a search interface on a blue background. At the top is a white search bar with the placeholder text '어떤 공공데이터를 찾으시나요?' (Which public data are you looking for?). To the right of the search bar is a blue button with a white magnifying glass icon. Below the search bar, there are three filter buttons: '검색조건' (Search Conditions) with a circular arrow icon, '분류체계' (Classification System) with a dropdown arrow, 'REST' with a dropdown arrow, and 'JSON' with a dropdown arrow.

검색

- 환경부 국립환경과학원 미세먼지

오픈 API (336건)

더보기 >

환경기상

국가행정기관

미리보기

XML

JSON

환경부 국립환경과학원 미세먼지(금속성분) 실시간 정보

국립환경과학원 대기연구분야 데이터 입니다.(대기중 중금속 성분 2시간, 24시간 평균 측정 자료)

제공기관 환경부 국립환경과학원

수정일 2021-08-31

조회수 10645

활용신청 445

키워드 대기질,예보통보,미세먼지,중금속 실시간 공개,대기환경연구소

활용신청

서비스키를 발급 받아서, 나의 서비스키를 입력하면, 결과를 받을 수 있음

- 서비스 신청 후 발급받은 “디코딩 키” 값을 입력하면 됨

샘플코드

Java

Javascript

C#

PHP

Curl

Objective-C

Python

Nodejs

R

Python3 샘플 코드

import requests

url = 'http://apis.data.go.kr/1480523/MetalMeasuringResultService/MetalService'

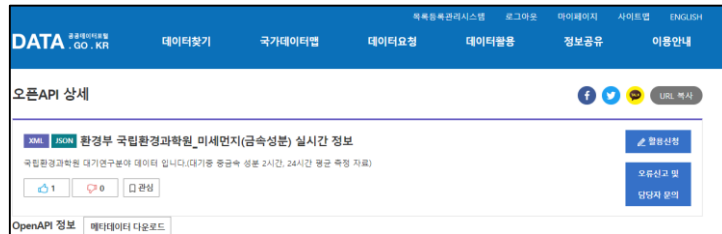
params = {'serviceKey': '서비스키', 'pageNo': '1', 'numOfRows': '10', 'resultType': 'XML', 'date': '20171208', 'stationcode': '1', 'itemcode': '90303', 'timecode': 'RH02' }

response = requests.get(url, params=params)

print(response.content)

목록

활용신청 > 기본정보 입력 후, 활용신청 > 즉시 승인처리 됨



활용목적 선택

*활용목적

☐ 웹 사이트 개발
 ☐ 앱개발 (코바일,올루션등)
 ☒ 기타
 ☐ 광고자료
 ☐ 연구(논문 등)

학습용

3/250

정부과목

파일 선택

Drag & Drop으로 파일을 선택 가능합니다.

시스템유형

시스템 유형

☒ 일반

상세기능정보 선택

| | 상세기능 | 설명 | 일일 트래픽 |
|-------------------------------------|--------------|--|--------|
| <input checked="" type="checkbox"/> | 중금속 성분 측정 결과 | 조폐본부, 알코프드, 측정소코드, 시간구분용 기준으로 중금속 성분 측정 결과를 2시간 평균, 24시간 평균 측정 수치 가트르 제공하는 서비스 | 10000 |

라이선스 표시

*이용허락범위

이용허락범위 제한 없음

☒ 동의합니다.

취소

활용신청

마이페이지

오픈API

개발계정

운영계정

인증키 발급현황

개발계정

신청 0건 >

신청중인 단계

보류

0건

반려

0건

활용 5건 >

승인되어 활용중인 단계

변경신청

0건

중지0건 >

중지신청하여 운영이 중지된 단계

DATA

나의 문의

나의 관심

나의 제공신청

나의 분쟁조정

회원정보 수정

상세검색 열기

총5건

환경가성

환경부 국립환경과학원

활용신청

[승인] 환경부 국립환경과학원_미세먼지(금속성분) 실시간 정보

신청일

2022-01-10

만료예정일

2024-01-10

Decoding 키
복사해두기

개발계정 상세보기

기본정보

| | | | |
|-------|---|------|------|
| 데이터명 | 환경부 국립환경과학원_미세먼지(금속성분) 실시간 정보 상세설명 | | |
| 서비스유형 | REST | 심의여부 | 자동승인 |
| 신청유형 | 개발계정 활용신청 | 처리상태 | 승인 |
| 활용기간 | 2022-01-10 ~ 2024-01-10 | | |

서비스정보

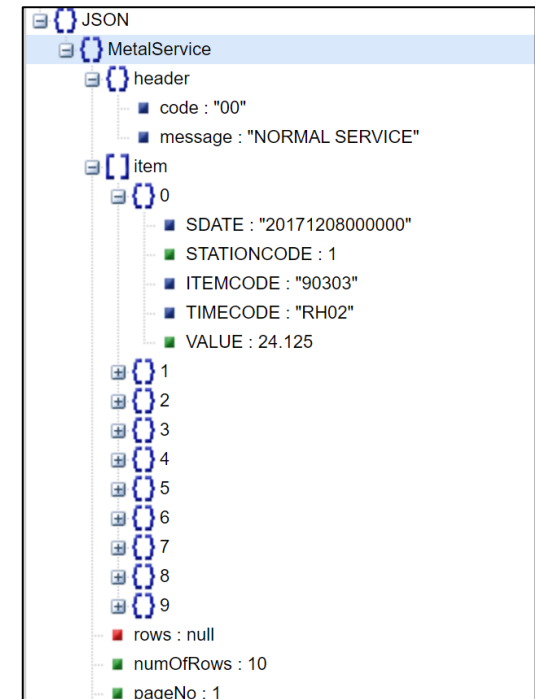
| | |
|---|--|
| 참고문서 | 13.OpenAPI활용가이드 국립환경과학원 대기환경연구소 대기오염측정자료.pdf |
| 데이터포맷 | JSON+XML |
| End Point | http://apis.data.go.kr/1480523/MetalMeasuringResultService |
| API 환경 또는 API 호출 조건에 따라 인증키가 적용되는 방식이 다를 수 있습니다. 포털에서 제공되는 Encoding/Decoding 된 인증키를 적용하면서 구동되는 키를 사용하시기 바랍니다. * 향후 포털에서 더 명확한 정보를 제공하기 위해 노력하겠습니다. | |
| 일반 인증키 (Encoding) | OwCTrPdQSAZpz5hBZ0dYxTazhVw5ziOvZvRpBLp9RoAqQHsIIDX%2F3n5IgvbQ%2BS3KbcgSoN2O1Xjs%2FdOmaDZ11A%3D%3D |
| 일반 인증키 (Decoding) | OwCTrPdQSAZpz5hBZ0dYxTazhVw5ziOvZvRpBLp9RoAqQHsIIDX/3n5IgvbQ+S3KbcgSoN2O1Xjs/dOmaDZ11A== |

json Type 으로 가져온 데이터

```
main x
C:\Users\inho\PycharmProjects\pythonProject4\venv\Scripts\python.exe C:/Users/inh
b'{"MetalService":{"header":{"code":"00","message":"NORMAL SERVICE"},"item":[{"SD
Process finished with exit code 0
```


```
import requests

url = 'http://apis.data.go.kr/1480523/MetalMeasuringResultService/MetalService'
key = "0wCTrPdQSAZpz5hBZ0dYxTazhVw5zi0vZvRpBLp9RoAqQHsLIDX/3n5IgvbQ+S3KbcgSoN201X
params = {'serviceKey' : key, 'pageNo' : '1', 'numOfRows' : '10', 'resultType' : '
response = requests.get(url, params=params)
print(response.content)
```



PDF 명세를 확인하여, 보기 좋게 출력하기

- 날짜
- 대기환경연구소명
- 측정항목 종류 (납 / 칼슘)
- 측정수치



국립환경과학원

공공데이터 개방 · 공유 · 활용 · 체계 개발 OpenAPI 활용가이드

2.1.3.2 응답 메시지 명세

| 항목명(영문) | 항목명(국문) | 항목크기 | 항목구분 | 샘플데이터 | 항목설명 |
|-------------|-------------|------|------|----------------|-----------------|
| resultCode | 결과코드 | 2 | 1 | 00 | 결과코드 |
| resultMsg | 결과메세지 | 36 | 1 | success | 결과메세지 |
| item | 아이템 목록 | - | 0..n | - | 목록 |
| sdate | 날짜 | 14 | 1 | 20171208040000 | 날짜 |
| stationcode | 대기환경연구소 코드 | 1 | 1 | 1 | 대기환경연구소 코드 |
| | | | | | 코드 |
| | | | | | 대기환경연구소명 |
| | | | | | 1 수도권 |
| | | | | | 2 백령도 |
| | | | | | 3 호남권 |
| | | | | | 4 중부권 |
| | | | | | 5 제주권 |
| | | | | | 6 영남권 |
| | | | | | 7 경기권 |
| | | | | | 8 충청권 |
| | | | | | 9 전라권 |
| | | | | | 10 강원권 |
| itemcode | 측정항목 코드 | 5 | 1 | 90303 | 측정항목 코드 |
| | | | | | 코드 |
| | | | | | 대기환경연구소명 |
| timecode | 측정데이터 시간 구분 | 4 | 1 | RH02 | 구분 |
| | | | | | 설명 |
| | | | | | RH02 2시간 이동평균 |
| | | | | | RH24 24시간 이동평균 |
| value | 측정수치 | 8 | 1 | 7.5 | 측정수치 (단위:ng/m3) |
| numOfRows | 한 페이지 결과 수 | 4 | 1 | 10 | 한 페이지 결과 수 |
| pageNo | 페이지 번호 | 4 | 1 | 1 | 페이지 번호 |
| totalCount | 전체 결과 수 | 4 | 1 | 1234 | 전체 결과 수 |

※ 항목구분: 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n)

내일 방송에서 만나요!

삼성 청년 SW 아카데미