

LESSON

pandas 모듈

```
animLength = toTime - fromTime

# Ask user for directory
filePath = c4d.storage.SaveDialog()
filePath, objName = os.path.split(filePath)
objName = objName + ". "
filePath = filePath + "\\ "
# Ask for confirmation
questionDialogText = "Obj Sequence will be saved as:\n\n" \
    "" + filePath + objName + "####.obj\n\n" \
    "From frame " + str(fromTime) + " to " + str(toTime) + " "
proceedBool = c4d.gui.QuestionDialog(questionDialogText)

if proceedBool == True:

    # Loop through animation and export frames
    for x in range(0,animLength):

        # change frame, redraw view
        moveTime = c4d.BaseTime(fromTime,docFps) + c4d.BaseTime(x,docFps)
        doc.SetTime(moveTime)
        c4d.EventAdd(c4d.EVENT_FORCEREDRAW)
        c4d.DrawViews(c4d.DRAWFLAGS_FORCEFULLREDRAW)

        # update status bar
        c4d.StatusSetText("Exporting " + str(x) + " of " + str(animLength))
        c4d.StatusSetBar(100.0*x/animLength)

        # get buffer pool
        bufferedNumber = str(doc.GetTime().GetFrame(docFps))
        if len(bufferedNumber) > 10:
            bufferedNumber = bufferedNumber[-10:]
```

pandas 모듈의 정의

- ❁ 파이썬의 대표적인 데이터 분석 라이브러리
 - 행과 열로 이루어진 데이터 객체를 만듦
- ❁ 특히 안정적으로 대용량 데이터를 처리에 매우 적합(빅데이터)
 - 프로그래밍 언어 R의 DataFrame과 유사
- ❁ 빅데이터의 생성, 선택, 연산, 병합, 그룹화, 변형, 그래프 등 엑셀과 유사한 기능을 손쉽게 구현할 수 있음

pandas 모듈의 설치

- pip install pandas로 설치
- Anaconda를 설치했다면 기본으로 함께 설치가 되어 있음
- `import pandas as pd`로 호출하여 사용

```
import pandas as pd  
print(pd.__version__)
```

```
0.22.0
```

pandas 모듈의 활용

♣ pandas 모듈이 제공하는 기본적인 자료구조



Series

DataFrame

pandas 모듈의 활용 : Series

🔗 인덱스를 가진 1차원 배열과 같은 자료형

```
s = pd.Series([1, 9, -3, 4])
print(s)
```

```
0    1
1    9
2   -3
3    4
dtype: int64
```

```
# 값 확인
print(s.values)
```

```
# 인덱스 확인
print(s.index)
```

```
# 자료형 확인
print(s.dtypes)
```

```
[ 1  9 -3  4]
RangeIndex(start=0, stop=4, step=1)
int64
```

pandas 모듈의 활용 : Series

- 인덱스를 가진 1차원 배열과 같은 자료형
- 기본 인덱스는 정수로 따로 지정할 수 있으며 파이썬 사전 자료형과 호환 가능

#인덱스 지정

```
s = pd.Series([4, 7, -5, 3]
               , index=['d', 'b', 'a', 'c'])
print(s)
```

```
d    4
b    7
a   -5
c    3
dtype: int64
```

#사전과 호환

```
dic = {'A': 3000, 'B': 6000, 'C': 2000, 'D': 4000}
s = pd.Series(dic)
print(s)
```

```
A    3000
B    6000
C    2000
D    4000
dtype: int64
```

pandas 모듈의 활용 : DataFrame

- ☛ 행과 열을 가진 자료구조로 직접 행과 열에 데이터를 입력하거나, 파이썬의 사전 자료형이나 numpy 모듈의 array로 정의함

```
df = pd.DataFrame({'A' : 1.,
                   'B' : '2020-01-01',
                   'C' : [1,2,3,4],
                   'D' : (5,6,7,8),
                   'E' : 0,
                   'F' : 'web' })

print(df)
```

	A	B	C	D	E	F
0	1.0	2020-01-01	1	5	0	web
1	1.0	2020-01-01	2	6	0	web
2	1.0	2020-01-01	3	7	0	web
3	1.0	2020-01-01	4	8	0	web

```
#행
print(df.index)

#열
print(df.columns)
```

```
RangeIndex(start=0, stop=5, step=1)
Index(['name', 'points', 'year'], dtype='object')
```

```
data = {'name': ['A', 'B', 'C', 'D', 'E'],
        'year': [2017, 2018, 2019, 2020, 2021],
        'points': [3.5, 4.7, 2.6, 7.4, 1.9]}
df = pd.DataFrame(data)
print(df)
```

	name	points	year
0	A	3.5	2017
1	B	4.7	2018
2	C	2.6	2019
3	D	7.4	2020
4	E	1.9	2021

pandas 모듈의 활용 : DataFrame

🔗 **인덱싱(열)** : 열을 기준으로 데이터 추가 삭제 및 변경 가능

```
data = {'name': ['A', 'B', 'C', 'D', 'E'],
        'year': [2017, 2018, 2019, 2020, 2021],
        'points': [3.5, 4.7, 2.6, 7.4, 1.9]}
df = pd.DataFrame(data)
print(df)
```

	name	points	year
0	A	3.5	2017
1	B	4.7	2018
2	C	2.6	2019
3	D	7.4	2020
4	E	1.9	2021

```
#데이터 선택
print(df[['year']])
```

	year
0	2017
1	2018
2	2019
3	2020
4	2021

```
#여러 데이터 선택
print(df[['year', 'points']])
```

	year	points
0	2017	3.5
1	2018	4.7
2	2019	2.6
3	2020	7.4
4	2021	1.9

pandas 모듈의 활용 : DataFrame

🔗 **인덱싱(열)** : 열을 기준으로 데이터 추가 삭제 및 변경 가능

#새로운 열 추가

```
df['new'] = [1,2,3,4,5]
print(df)
```

	name	points	year	new
0	A	3.5	2017	1
1	B	4.7	2018	2
2	C	2.6	2019	3
3	D	7.4	2020	4
4	E	1.9	2021	5

#조건을 사용하여 추가

```
df['past'] = df['year'] < 2019
print(df)
```

	name	points	year	new	past
0	A	3.5	2017	1	True
1	B	4.7	2018	2	True
2	C	2.6	2019	3	False
3	D	7.4	2020	4	False
4	E	1.9	2021	5	False

#삭제

```
del df['past']
print(df)
```

	name	points	year	new
0	A	3.5	2017	1
1	B	4.7	2018	2
2	C	2.6	2019	3
3	D	7.4	2020	4
4	E	1.9	2021	5

pandas 모듈의 활용 : DataFrame

🔗 **인덱싱(행)** : 행을 기준으로 데이터 추가 삭제 및 변경 가능

```
data = {'name': ['A', 'B', 'C', 'D', 'E'],
        'year': [2017, 2018, 2019, 2020, 2021],
        'points': [3.5, 4.7, 2.6, 7.4, 1.9]}
df = pd.DataFrame(data)
print(df)
```

	name	points	year
0	A	3.5	2017
1	B	4.7	2018
2	C	2.6	2019
3	D	7.4	2020
4	E	1.9	2021

```
# 새로운 행 추가
df.loc[5,:] = ['F',4,'2022']
print(df)
```

	name	points	year
0	A	3.5	2017
1	B	4.7	2018
2	C	2.6	2019
3	D	7.4	2020
4	E	1.9	2021
5	F	4.0	2022

pandas 모듈의 활용 : DataFrame

🔗 **인덱싱(행)** : 행을 기준으로 데이터 추가 삭제 및 변경 가능

```
#범위 선택
print(df[0:3])
```

	name	points	year
0	A	3.5	2017
1	B	4.7	2018
2	C	2.6	2019

```
#범위 선택
print(df.loc[0:2])
```

	name	points	year
0	A	3.5	2017
1	B	4.7	2018
2	C	2.6	2019

```
# 범위 선택, 특정 열
print(df.loc[:, 'name'])
```

```
0    A
1    B
2    C
3    D
4    E
Name: name, dtype: object
```

pandas 모듈의 활용 : DataFrame

🔗 DataFrame 데이터 분석 : 데이터 분석 관련 다양한 함수 제공

```
data = {'name': ['A', 'B', 'C', 'D', 'E'],
        'year': [2017, 2018, 2019, 2020, 2021],
        'points': [3.5, 4.7, 2.6, 7.4, 1.9]}
df = pd.DataFrame(data)
print(df)
```

	name	points	year
0	A	3.5	2017
1	B	4.7	2018
2	C	2.6	2019
3	D	7.4	2020
4	E	1.9	2021

```
# 각 행의 합
print(df.sum(axis=1))
```

```
0    2020.5
1    2022.7
2    2021.6
3    2027.4
4    2022.9
dtype: float64
```

```
# 각 열의 합
print(df.sum(axis=0))
```

```
name      ABCDE
points    20.1
year      10095
dtype: object
```

① sum : 행 또는 열의 합

② min, max : 최소, 최대 값

③ mean : 평균 값

④ median : 중간 값

⑤ std, var : 표준 편차, 분산

⑥ count : 값의 개수

⑦ sort_values : 정렬

⑧ corr, cov : 상관계수, 공분산

LESSON

Excel과 JSON 데이터 다루기

```
animLength = toTime - fromTime

# Ask user for directory
filePath = c4d.storage.SaveDialog()
filePath, objName = os.path.split(filePath)
objName = objName + ". "
filePath = filePath + "\\ "

# Ask for confirmation
questionDialogText = "Obj Sequence will be saved as:\n\n" \
    "" + filePath + objName + "####.obj\n\n" \
    "From frame " + str(fromTime) + " to " + str(toTime) + " "
proceedBool = c4d.gui.QuestionDialog(questionDialogText)

if proceedBool == True:

    # Loop through animation and export frames
    for x in range(0, animLength):

        # change frame, redraw view
        moveTime = c4d.BaseTime(fromTime, docFps) + c4d.BaseTime(x)
        doc.SetTime(moveTime)
        c4d.EventAdd(c4d.EVENT_FORCEREDRAW)
        c4d.DrawViews(c4d.DRAWFLAGS_FORCEFULLREDRAW)

        # update status bar
        c4d.StatusSetText("Exporting " + str(x) + " of " + str(animLength))
        c4d.StatusSetBar(100.0*x/animLength)

        # add buffer 0001
        bufferedNumber = str(doc.GetTime().GetFrame(docFps))
        if len(bufferedNumber) < 4:
```

pandas 모듈에서 외부 데이터 다루기

- ✦ pandas 모듈 내의 함수를 이용하여 Excel, JSON 등의 외부 데이터를 읽고 쓸 수 있음



Excel 데이터 다루기

- pandas 모듈 내 엑셀 함수를 활용하여 엑셀 파일을 읽을 수 있음
- `read_excel('파일명', '시트명')`: 불러온 엑셀 데이터는 DataFrame 객체

	A	B	C	D	E	F
1	N	year	A	B	start_page	end_page
2	1	2019	10	1.4	717	723
3	2	2020	11	2.5	724	736
4	3	2021	12	3.6	737	747
5	4	2022	13	4.7	748	764
6	5	2023	14	5.8	765	784
7	6	2024	15	6.9	785	796
8	7	2025	16	8	797	819
9	8	2026	17	9.1	820	832
10	9	2027	18	10.2	833	852
11	10	2028	19	11.3	853	864
12	11	2029	20	12.4	865	889
13	12	2030	21	13.5	890	903
14	13	2031	22	14.6	904	919
15	14	2032	23	15.7	920	930
16	15	2033	24	16.8	931	956
17	16	2034	25	17.9	957	966
18	17	2035	26	19	967	985
19	18	2036	27	20.1	986	1016
20	19	2037	28	21.2	1017	1028



```
excel = pd.read_excel('crawl.xls', 'Sheet1')
print(type(excel))
print(excel)
```

```
<class 'pandas.core.frame.DataFrame'>
   N  year  A      B  start_page  end_page
0  1  2019  10   1.4         717         723
1  2  2020  11   2.5         724         736
2  3  2021  12   3.6         737         747
3  4  2022  13   4.7         748         764
4  5  2023  14   5.8         765         784
5  6  2024  15   6.9         785         796
6  7  2025  16   8.0         797         819
7  8  2026  17   9.1         820         832
8  9  2027  18  10.2         833         852
9 10  2028  19  11.3         853         864
10 11  2029  20  12.4         865         889
11 12  2030  21  13.5         890         903
12 13  2031  22  14.6         904         919
13 14  2032  23  15.7         920         930
14 15  2033  24  16.8         931         956
15 16  2034  25  17.9         957         966
16 17  2035  26  19.0         967         985
17 18  2036  27  20.1         986        1016
18 19  2037  28  21.2        1017        1028
```

Excel 데이터 다루기 예시1

DataFrame의 함수를 활용해 엑셀 데이터를 다룰 수 있음

#각 열 평균

```
print(excel.mean())
```

```
N          10.000000
year       2028.000000
A          19.000000
B          11.300000
start_page 853.473684
end_page   868.894737
dtype: float64
```

#요약 통계

```
print(excel.describe())
```

	N	year	A	B	start_page	end_page
count	19.000000	19.000000	19.000000	19.000000	19.000000	19.000000
mean	10.000000	2028.000000	19.000000	11.300000	853.473684	868.894737
std	5.627314	5.627314	5.627314	6.190046	94.189743	96.249615
min	1.000000	2019.000000	10.000000	1.400000	717.000000	723.000000
25%	5.500000	2023.500000	14.500000	6.350000	775.000000	790.000000
50%	10.000000	2028.000000	19.000000	11.300000	853.000000	864.000000
75%	14.500000	2032.500000	23.500000	16.250000	925.500000	943.000000
max	19.000000	2037.000000	28.000000	21.200000	1017.000000	1028.000000

#상위 x개 출력

```
print(excel.head(5))
```

	N	year	A	B	start_page	end_page
0	1	2019	10	1.4	717	723
1	2	2020	11	2.5	724	736
2	3	2021	12	3.6	737	747
3	4	2022	13	4.7	748	764
4	5	2023	14	5.8	765	784

#열 기준 정렬

```
print(excel.sort_values(by='year', ascending=False))
```

	N	year	A	B	start_page	end_page
18	19	2037	28	21.2	1017	1028
17	18	2036	27	20.1	986	1016
16	17	2035	26	19.0	967	985
15	16	2034	25	17.9	957	966
14	15	2033	24	16.8	931	956
13	14	2032	23	15.7	920	930
12	13	2031	22	14.6	904	919
11	12	2030	21	13.5	890	903
10	11	2029	20	12.4	865	889
9	10	2028	19	11.3	853	864
8	9	2027	18	10.2	833	852
7	8	2026	17	9.1	820	832
6	7	2025	16	8.0	797	819
5	6	2024	15	6.9	785	796
4	5	2023	14	5.8	765	784
3	4	2022	13	4.7	748	764
2	3	2021	12	3.6	737	747
1	2	2020	11	2.5	724	736
0	1	2019	10	1.4	717	723

Excel 데이터 다루기 예시2 엑셀의 특정 데이터만 골라서 가져 올 수 있음

```
excel = pd.read_excel('crawl.xls', 'Sheet1')
excel.head(10)
```

	N	year	A	B	start_page	end_page
0	1	2019	10	1.4	717	723
1	2	2020	11	2.5	724	736
2	3	2021	12	3.6	737	747
3	4	2022	13	4.7	748	764
4	5	2023	14	5.8	765	784
5	6	2024	15	6.9	785	796
6	7	2025	16	8.0	797	819
7	8	2026	17	9.1	820	832
8	9	2027	18	10.2	833	852
9	10	2028	19	11.3	853	864



```
excel = pd.read_excel('crawl.xls', 'Sheet1', usecols='B, E, F')
excel.head(10)
```

	year	start_page	end_page
0	2019	717	723
1	2020	724	736
2	2021	737	747
3	2022	748	764
4	2023	765	784
5	2024	785	796
6	2025	797	819
7	2026	820	832
8	2027	833	852
9	2028	853	864

Excel 데이터 다루기 예시3 반복문과 함께 활용

	N	year	A	B	start_page	end_page
0	1	2019	10	1.4	717	723
1	2	2020	11	2.5	724	736
2	3	2021	12	3.6	737	747
3	4	2022	13	4.7	748	764
4	5	2023	14	5.8	765	784
5	6	2024	15	6.9	785	796
6	7	2025	16	8.0	797	819
7	8	2026	17	9.1	820	832
8	9	2027	18	10.2	833	852
9	10	2028	19	11.3	853	864

DataFrame 객체를
그대로 반복문에 넣었을 경우

```
for i in excel:
    print(i)
```

```
N
year
A
B
start_page
end_page
```

단순히 열의
이름을 출력

Excel 데이터 다루기 예시3 반복문과 함께 활용

	N	year	A	B	start_page	end_page
0	1	2019	10	1.4	717	723
1	2	2020	11	2.5	724	736
2	3	2021	12	3.6	737	747
3	4	2022	13	4.7	748	764
4	5	2023	14	5.8	765	784
5	6	2024	15	6.9	785	796
6	7	2025	16	8.0	797	819
7	8	2026	17	9.1	820	832
8	9	2027	18	10.2	833	852
9	10	2028	19	11.3	853	864

각 행마다의 값을 사용하고
싶을 때 : items() 사용

```
for i,j in excel.items():
    print(j[0])
```

i는 인덱스,
j는 각 행의 값

```
1
2019
10
1.4
717
723
```

j[i]의 값 출력

Excel 데이터 다루기 예시4 반복문과 함께 활용

	N	year	A	B	start_page	end_page
0	1	2019	10	1.4	717	723
1	2	2020	11	2.5	724	736
2	3	2021	12	3.6	737	747
3	4	2022	13	4.7	748	764
4	5	2023	14	5.8	765	784
5	6	2024	15	6.9	785	796
6	7	2025	16	8.0	797	819
7	8	2026	17	9.1	820	832
8	9	2027	18	10.2	833	852
9	10	2028	19	11.3	853	864

각 열마다의 값을 사용하고
싶을 때 : iterrows() 사용

```
for i,j in excel.iterrows():
    print(j[3])
```

1.4
2.5
3.6
4.7
5.8
6.9
8.0
9.1
10.2
11.3
12.4
13.5
14.6
15.7
16.8
17.9
19.0
20.1
21.2

Excel 데이터 다루기 예시5

확장자가 '.csv'인 파일 불러오기

	crawl.csv
1	,year,A,B,start_page,end_page
2	1,2019,10,1.4,717,723
3	2,2020,11,2.5,724,736
4	3,2021,12,3.6,737,747
5	4,2022,13,4.7,748,764
6	5,2023,14,5.8,765,784
7	6,2024,15,6.9,785,796
8	7,2025,16,8,797,819
9	8,2026,17,9.1,820,832
10	9,2027,18,10.2,833,852
11	10,2028,19,11.3,853,864
12	11,2029,20,12.4,865,889
13	12,2030,21,13.5,890,903
14	13,2031,22,14.6,904,919
15	14,2032,23,15.7,920,930
16	15,2033,24,16.8,931,956
17	16,2034,25,17.9,957,966
18	17,2035,26,19,967,985
19	18,2036,27,20.1,986,1016
20	19,2037,28,21.2,1017,1028

```
excel = pd.read_csv('crawl.csv', 'Sheet1')
excel.head(10)
```

read_csv() 함수 사용

	,year,A,B,start_page,end_page
0	1,2019,10,1.4,717,723
1	2,2020,11,2.5,724,736
2	3,2021,12,3.6,737,747
3	4,2022,13,4.7,748,764
4	5,2023,14,5.8,765,784
5	6,2024,15,6.9,785,796
6	7,2025,16,8,797,819
7	8,2026,17,9.1,820,832
8	9,2027,18,10.2,833,852
9	10,2028,19,11.3,853,864

csv(comma-separated values) 파일

JSON 데이터 다루기

- ❖ 파이썬의 표준 모듈 json을 활용(별도의 설치 없이 사용 가능)
- ❖ **import json**으로 호출

```
import json  
print(json.__file__)
```

/anaconda3/lib/python3.6/json/__init__.py



크게 파이썬 데이터를 JSON 데이터로 변환하는 인코딩과
JSON 데이터를 파이썬 데이터로 변환하는 디코딩, 두 가지 기능을 제공함

JSON 데이터 다루기

❖ 인코딩(파이썬 → JSON) : 파이썬 객체(문자열, 숫자, 리스트, 튜플 등)를 JSON 문자열로 변경

- dumps 함수 활용

```
import json
student = {
    'id': 20200101,
    'name': 'Hong',
    'history': [
        {'subject': 'math', 'grade': 3.0},
        {'subject': 'english', 'grade': 4.5},
    ]
}
```

```
Encoding_json = json.dumps(student)
print(Encoding_json)
print(type(Encoding_json))
```

```
{"id": 20200101, "name": "Hong", "history": [{"subject": "math", "grade": 3.0}, {"subject": "english", "grade": 4.5}]}
<class 'str'>
```

JSON 데이터 다루기

인코딩(파이썬 → JSON)

- indent : 들여쓰기(가독성 향상)

```
Encoding_json = json.dumps(student, indent=4)
print(Encoding_json)
```

```
{
  "id": 20200101,
  "name": "Hong",
  "history": [
    {
      "subject": "math",
      "grade": 3.0
    },
    {
      "subject": "english",
      "grade": 4.5
    }
  ]
}
```

```
Encoding_json = json.dumps(student, indent=8)
print(Encoding_json)
```

```
{
    "id": 20200101,
    "name": "Hong",
    "history": [
        {
            "subject": "math",
            "grade": 3.0
        },
        {
            "subject": "english",
            "grade": 4.5
        }
    ]
}
```


JSON 데이터 다루기

🔗 디코딩(JSON → 파이썬) : JSON을 파이썬 객체로 변경

- loads 함수

```
import json
json_data = """{"id": 20200101, "name": "Hong",
"history": [{"subject": "math", "grade": 3.0}, {"subject": "english", "grade": 4.5}]}"""

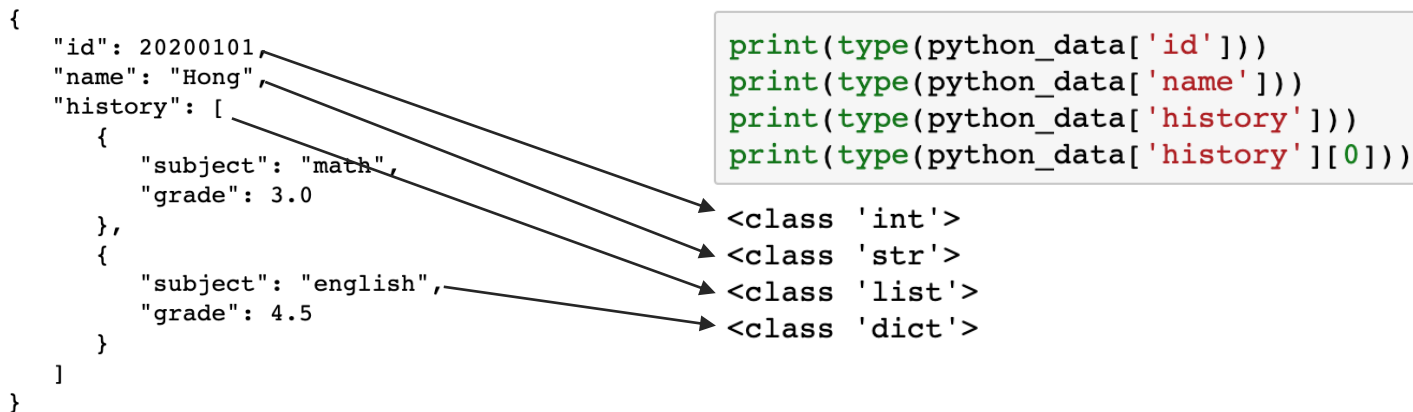
Decoding_data = json.loads(json_data)
print(Decoding_data)
print(type(Decoding_data))

{'id': 20200101, 'name': 'Hong', 'history': [{'subject': 'math', 'grade': 3.0}, {'subject': 'english', 'grade': 4.5}]}
<class 'dict'>
```

JSON 데이터 다루기

❖ 디코딩(JSON → 파이썬) : JSON을 파이썬 객체로 변경

- 각각 파이썬 자료형으로 변환



```
{
  "id": 20200101,
  "name": "Hong",
  "history": [
    {
      "subject": "math",
      "grade": 3.0
    },
    {
      "subject": "english",
      "grade": 4.5
    }
  ]
}
```

```
print(type(python_data['id']))
print(type(python_data['name']))
print(type(python_data['history']))
print(type(python_data['history'][0]))
```

```
<class 'int'>
<class 'str'>
<class 'list'>
<class 'dict'>
```

JSON 데이터 다루기

인코딩: 파이썬 → JSON

파이썬	JSON
dict	오브젝트(object)
list, tuple	배열(array)
str	문자열(string)
int, float	숫자(number)
True	true
False	false
None	null

디코딩: JSON → 파이썬

JSON	파이썬
오브젝트(object)	dict
배열(array)	list
문자열(string)	str
숫자(정수)	int
숫자(실수)	float
true	True
false	False