

JSON

JSON to dict

`json.loads` ??????; `json.load` ??????

```
import json

person = '{"name": "Bob", "languages": ["English", "French"]}'
person_dict = json.loads(person)

# Output: {'name': 'Bob', 'languages': ['English', 'French']}
print( person_dict)

# Output: ['English', 'French']
print(person_dict['languages'])
```

Dict to JSON

```
import json

person_dict = {'name': 'Bob',
'age': 12,
'children': None
}
person_json = json.dumps(person_dict)

# Output: {"name": "Bob", "age": 12, "children": null}
print(person_json)
```

Read JSON file

```
import json

with open('path_to_file/person.json', 'r') as f:
```

```
data = json.load(f)
```

```
# Output: {'name': 'Bob', 'languages': ['English', 'French']}  
print(data)
```

Write JSON file

```
json.dump ?????; json.dumps ???????
```

```
import json
```

```
person_dict = {"name": "Bob",  
               "languages": ["English", "French"],  
               "married": True,  
               "age": 32  
               }
```

```
with open('person.txt', 'w') as json_file:  
    json.dump(person_dict, json_file)
```

Print JSON

```
import json
```

```
person_string = '{"name": "Bob", "languages": "English", "numbers": [2, 1.6, null]}'
```

```
# Getting dictionary
```

```
person_dict = json.loads(person_string)
```

```
# Pretty Printing JSON string back
```

```
print(json.dumps(person_dict, indent = 4, sort_keys=True))
```

Access JSON

```
import json
```

```
json_data = '''
```

```
{  
    "students": [  
        ]
```

```

{
    "name": "David",
    "age": 19,
    "grades": {
        "math": 90,
        "english": 87
    }
},
{
    "name": "Harry",
    "age": 21,
    "grades": {
        "math": 85,
        "english": 95
    }
}
]
}
'''

```

Parse JSON Data

```
data = json.loads(json_data)
```

To access a large dataset we can use `for loop`

```
for student in data["students"]:
```

```
    name = student["name"]
```

```
    math_mark = student["grades"]["math"]
```

```
    english_mark = student["grades"]["english"]
```

```
    average_mark = (math_mark + english_mark) / 2
```

```
    print(f"{name}, Avarage Marks: {average_mark:.2f}")
```

Output:

```
# David, Average Marks: 88.50
```

```
# Harry, Average Marks: 90.00
```

```
import json
```

```
original_data_file="students_data.json"
```

```
updated_data_file="students_data_updated.json"
```

```

# reading `JSON file`
with open(original_data_file,"r") as file:
    students_result = json.load(file)

# Updating JSON Data
for student in students_result['students']:
    print(student['name'])

    if student['name'] == "Kabir":
        student['name'] = "John"

    grades = student['grades']
    avarage_mark= sum(grades.values()) / len(grades)
    student['avarage_mark'] = avarage_mark

# Saving updated data into a new file
with open(updated_data_file,"w") as file:
    json.dump(students_result,file,indent=4)

```

Get JSON from URL

```

import requests, json

# Response will be saved here
weather_data="weather_data.json"

# Request to `openweathermap` API
api_key = "6423af6e554f98cf1e6b8c6a7700986b" #REPLACE_WITH_YOUR_API_KEY
location = "Dhaka"
url = f"https://api.openweathermap.org/data/2.5/weather?q={location}&appid={api_key}&units=metric"

# Response
response = requests.get(url)

# Get `Place` and `Temperature` from the Response
if response.status_code == 200:
    json_data = response.json()
    print(f"Place: {json_data['name']}, Temperature: {json_data['main']['temp']} celsius")
else:

```

```
print(f"Request failed with status code {response.status_code}")

# Save the Response to a file
with open(weather_data,"w") as file:
    json.dump(json_data,file,indent=4)

# Output:
# Place: Dhaka, Temperature: 27.99 celsius
```

```
# Handling a JSONDecodeError in Python
from json import JSONDecodeError
import requests

resp = requests.get('https://reqres.in/api/users/page4')
try:
    resp_dict = resp.json()
except JSONDecodeError:
    print('Response could not be serialized')
```

Data Type

?? json.loads ?????????????????????? dict ?? array???????JSON ???????

JSON	Python
object	dict
array	list
string	str
number (integer)	int
number (real)	float
true	True
false	False
null	N

Library

[jsonpath-ng](#)

```
{
  "employees": [
    {
      "id": 1,
      "name": "Pankaj",
      "salary": "10000"
    },
    {
      "name": "David",
      "salary": "5000",
      "id": 2
    }
  ]
}
```

```
import json
from jsonpath_ng import jsonpath, parse

with open("db.json", 'r') as json_file:
    json_data = json.load(json_file)

print(json_data)

jsonpath_expression = parse('employees[*].id')

for match in jsonpath_expression.find(json_data):
    print(f'Employee id: {match.value}')
```

```
{'employees': [{ 'id': 1, 'name': 'Pankaj', 'salary': '10000'}, { 'name': 'David', 'salary': '5000', 'id': 2}]}
```

Employee id: 1

Employee id: 2

Revision #3

Created 7 June 2023 09:58:57 by Admin

Updated 7 June 2023 10:07:06 by Admin