JSON in Python – Using dumps() and loads()

🧠 What is JSON?

JSON (JavaScript Object Notation) is a lightweight data format used to store and transfer data between applications.
It is language-independent but easily understood by both humans and machines.
JSON is often used in web APIs, databases, and file storage.

Example of JSON:

```
{
  "name": "John",
  "age": 25,
  "city": "Hyderabad"
}
```

◇ Why do we use JSON in Python?

Python data types like dictionary, list, tuple, etc., can be easily converted into JSON format.
JSON is useful when we want to:
- Store Python data in a text file.
- Send data between a Python program and a web server.
- Share data between different programming languages (e.g., Python → JavaScript).

◇ 1. json.dumps() – Convert Python object to JSON string

Syntax: json.dumps(python_object)

Meaning: Converts Python data (like dict, list, etc.) into a JSON string (not file).

Example:

```
import json

data = {'name': 'John', 'age': 25, 'city': 'Hyderabad'}
json_string = json.dumps(data)

print(json_string)
print(type(json_string))
```

Output:

```
{"name": "John", "age": 25, "city": "Hyderabad"}
<class 'str'>
```

⚙ Optional Parameters in dumps()

- indent: Adds indentation for readability
- sort_keys: Sorts keys alphabetically
- separators: Custom separators for JSON

Example:
```
print(json.dumps(data, indent=4))
```

Output:
```
{
    "name": "John",
    "age": 25,
    "city": "Hyderabad"
}
```

🧩 Example 2: Convert List to JSON String

```
data = ['apple', 'banana', 'grapes']
json_str = json.dumps(data)

print(json_str)
print(type(json_str))
```

Output:
```
["apple", "banana", "grapes"]
<class 'str'>
```

◇ 2. json.loads() – Convert JSON string to Python object

Syntax: json.loads(json_string)

Meaning: Converts a JSON string back into a Python object (list, dict, etc.)

Example 1:
```
import json

json_string = '{"name": "John", "age": 25, "city": "Hyderabad"}'
python_obj = json.loads(json_string)

print(python_obj)
print(type(python_obj))
```

Output:
```
{'name': 'John', 'age': 25, 'city': 'Hyderabad'}
<class 'dict'>
```

Example 2: Convert JSON string (list) → Python List

```
json_str = '["apple", "banana", "grapes"]'
python_list = json.loads(json_str)

print(python_list)
print(type(python_list))

# Perform list operation
python_list.append('mango')
print(python_list)
```

Output:

```
['apple', 'banana', 'grapes', 'mango']
```

Common Mistakes

- Using single quotes ' instead of double quotes " in JSON string
- Passing Python objects directly to loads() instead of string
- Forgetting to import json module

Summary

Purpose	Function	Input	Output
Convert Python → JSON string	json.dumps()	Python dict/list	JSON string
Convert JSON string → Python	json.loads()	JSON string	Python object

Real-World Example

Suppose we receive student details in JSON format from a web API:

```
import json

json_data = '{"students": [{"name": "Harish", "age": 22}, {"name": "Kiran", "age": 23}]}'

# Convert to Python
data = json.loads(json_data)

print(data['students'][0]['name'])   # Harish

# Modify and convert back to JSON
data['students'].append({'name':  'Abdul', 'age': 24})
updated_json = json.dumps(data, indent=4)

print(updated_json)
```

Output:
```
{
  "students": [
    {"name": "Harish", "age": 22},
    {"name": "Kiran", "age": 23},
    {"name": "Abdul", "age": 24}
  ]
}
```

✨ Key Points to Remember

- dumps() → Convert Python object → JSON string
- loads() → Convert JSON string → Python object
- JSON strings must always use double quotes ("")
- Ideal for data storage, APIs, file handling, and data exchange
- JSON works with:
  - Dictionary → Object
  - List → Array
  - String → String
  - Number → Number
  - Boolean → true/false
  - None → null