Python Assignment – JSON

Name: PRIYESHWAR

Mail: priyesh2664@gmail.com

What is JSON?

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write and easy for machines to parse and generate. JSON is language-independent but uses conventions familiar to programmers of the C family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others.

Key Features of JSON

- **Lightweight and Text-Based:** Data is stored as plain text.
- **Language-Independent:** Can be used with virtually all modern programming languages.
- **Structured Data:** Represents structured data using key-value pairs and ordered lists.
- **Human-Readable:** Easy to read and understand compared to binary data formats.
- **Widely Used:** Commonly used for APIs, configuration files, and data exchange between client and server.

JSON Data Types

Type	**Example**	**Notes**
string	"hello"	Always double-quoted, supports Unicode escape \uXXXX.
number	42, 3.14, -1e-9	No NaN, Infinity, or hex.
boolean	true, false	Lowercase only.
null	null	Represents absence of value.
object	{ "k": "v" }	Unordered key/value pairs.
array	[1, 2, 3]	Ordered, zero-based indexing conceptually.

Example of JSON

```
{
  "name": "Alice",
  "age": 25,
  "isStudent": false,
  "skills": ["Python", "JavaScript", "SQL"],
  "address": {
    "city": "New York",
    "zip": "10001"
  }
}
```

Converting JSON to Python

Python provides a built-in module called json for working with JSON data. Here's how you can convert between JSON and Python objects.

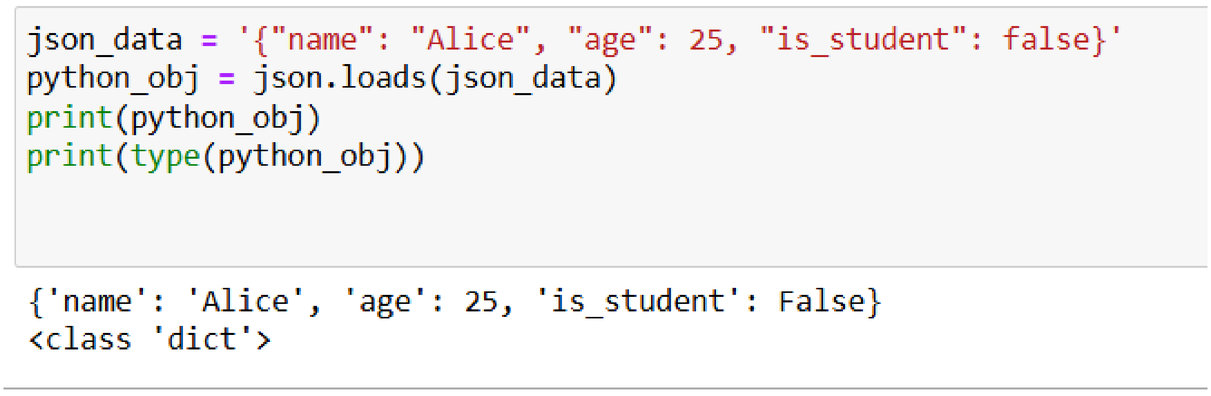
Importing the JSON Module

```
import json
```

JSON to Python

You can parse a JSON string and convert it into a Python object using json.loads():

```
import json


json_data = '{"name": "Alice", "age": 25, "is_student": false}'
python_obj = json.loads(json_data)
print(python_obj)
print(type(python_obj))
```



```
json_data = '{"name": "Alice", "age": 25, "is_student": false}'
python_obj = json.loads(json_data)
print(python_obj)
print(type(python_obj))
```

```
{'name': 'Alice', 'age': 25, 'is_student': False}
<class 'dict'>
```

Python to JSON

To convert a Python object into a JSON-formatted string, use json.dumps():

import json

python_obj = {

  "name": "Bob",

  "age": 30,

  "is_student": True,

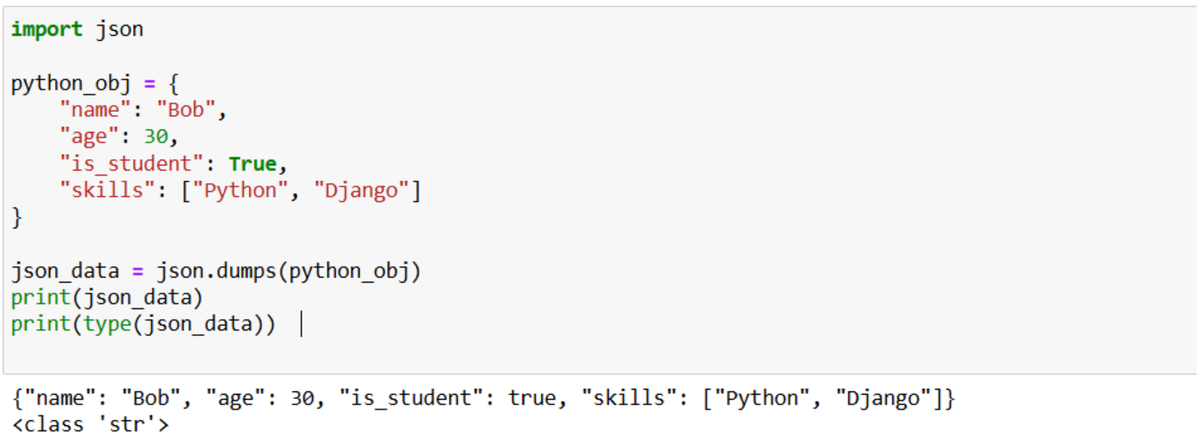
  "skills": ["Python", "Django"]

}

json_data = json.dumps(python_obj)

print(json_data)

print(type(json_data))



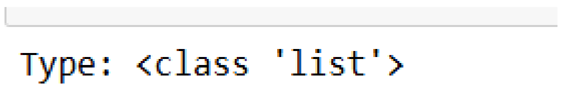
```
import json

python_obj = {
    "name": "Bob",
    "age": 30,
    "is_student": True,
    "skills": ["Python", "Django"]
}

json_data = json.dumps(python_obj)
print(json_data)
print(type(json_data))
```

```
{"name": "Bob", "age": 30, "is_student": true, "skills": ["Python", "Django"]}
<class 'str'>
```

Reading from and Writing to Files

Read JSON from a File

```
with open('data.json', 'r') as f:
    data = json.load(f)
    print("Type:" ,type(data))
```



```
Type: <class 'list'>
```

Write JSON to a File

```
with open('data.json', 'w') as f:
    json.dump(python_obj, f, indent=4)
```

Use json.load() and json.dump() for file operations (not loads() / dumps() which are for strings).
