METHODOLOGY FOR LEARNING JSON FILES USING THE PYTHON PROGRAMMING LANGUAGE

Nusratova Shakhzodakhon Bakhtiyorovna
Assistant , Department of Software and
Technical Support of Computer Systems,
Faculty of Digital Technologies and Artificial Intelligence,
Karshi State Technical University
nusratovashahzoda1010@gmail.com
0009-0000-0904-338X

Annotation: This article comprehensively covers the JSON (JavaScript Object Notation) file format, its structural design, core elements, and its significance in modern information technologies. JSON is a lightweight, simple, and universal format designed for data storage, transportation, and exchange. Since it is text-based, it is exceptionally easy for humans to read and edit. Concurrently, parsing and generating this format requires minimal computational resources from software systems and programming languages. The article explains the key-value system of JSON, its data structures in the form of objects and arrays, and complex nested structures through clear examples. In particular, it analyzes how perfectly the JSON architecture aligns with the built-in data types of the modern and popular Python programming language. The “Object” structure in JSON maps directly to the dictionary (dict) type in Python, while the “Array” maps straight to the list (list) type. This allows developers to process data structures effortlessly without the need for additional complex conversions.

Furthermore, the widespread adoption of JSON files in web development (RESTful API interfaces), mobile applications, client-server systems, and big data repositories (specifically in NoSQL and hybrid databases like MongoDB and PostgreSQL) is examined. Due to Python's dominance in artificial intelligence, Data

Science, and backend development (Django, FastAPI, Flask), it utilizes the JSON format as its primary medium for data exchange.

In addition, the advantages of JSON over the traditional XML format are comparatively analyzed, including its compactness due to the absence of redundant tags, network traffic savings, and high performance. While parsing XML documents demands substantial CPU time and memory, JSON accelerates this process multiple times over.

The article also demonstrates the practical capabilities of Python's standard json module. It enriches the study with concise and comprehensible code examples illustrating the processes of encoding data for transmission (serialization/json.dumps()) and converting received JSON text back into a Python object (deserialization/json.loads()). This practical section reinforces the theoretical and scientific conclusions of the article.

This article describes in detail the JSON (JavaScript Object Notation) file format, its structure, main elements, and its role in modern information technologies. JSON is a lightweight, simple, and universal format for storing and transferring data, which is easy for humans to read and fast for software systems to process. The article explains the key-value system of JSON, data structures in the form of objects and arrays, and nested structures using examples. It also examines the widespread use of JSON files in web programming, mobile applications, server-client systems, and databases (especially NoSQL systems). In addition, the advantages of JSON over the XML format, including simplicity, compactness, and fast performance, are analyzed. The article also provides a short programming example of working with JSON and shows its practical significance. In general, this work reveals the important role of JSON technology in modern programming.

Keywords: JSON, JavaScript Object Notation, data exchange, key-value, data structure, web programming, API, server-client, XML, NoSQL, mobile applications, programming languages, data formats.

Introduction: With the development of modern information technologies, the issue of fast, convenient and efficient data exchange is becoming increasingly important. Various formats are used to transfer data between web applications, mobile applications and server systems. Among them, JSON (JavaScript Object Notation) is one of the most common and convenient formats. JSON is distinguished by its simple structure, light weight and easy human readability. It mainly represents data based on key-value pairs and is supported by almost all modern programming languages. Therefore, JSON is widely used in web APIs, mobile applications and databases. This article provides detailed information about the structure, working principle, main elements and areas of its practical application of the JSON file format.

Research Methodology: In the process of preparing this article, theoretical and practical approaches to the JSON (JavaScript Object Notation) file format were used. The research is mainly based on the qualitative analysis method, and the structure, properties and areas of application of JSON were studied through existing scientific sources, official documents and Internet resources. As part of the methodology, a comparative analysis of the basic syntax of JSON, data structures (object, array, nested structures) and its comparison with other data formats such as XML was carried out. Also, practical examples of working with JSON in various programming languages were considered. During the study, the processes of reading and processing JSON data in programming languages such as Python and JavaScript were analyzed as a test. This helped to further understand the importance of JSON in real practice. In general, the methodology of this research is aimed at studying the theoretical foundations and practical applications of the JSON format based on a comprehensive approach.

Results and Discussion: JSON (JavaScript Object Notation) is a lightweight format for storing and exchanging data, widely used in modern programming. Although it was originally based on the JavaScript programming language, today it is supported by Python, Java, C#, PHP and many other programming languages. The JSON format has become widely popular due to its simple structure and ease of understanding. JSON data is mainly represented by key-value pairs. In this case, a certain value is associated with each key. In JSON files, objects are written using curly braces { }, and arrays are written using square brackets [ ].

One of the main advantages of JSON is its lightness and compactness. Compared to XML, JSON requires fewer characters and is easier to read. Therefore, JSON is more commonly used for data transfer between server and client.JSON quyidagi sohalarda keng qo‘llaniladi:

- In web programming, when exchanging information via API;
- In mobile applications, when communicating with the server;
- In NoSQL databases;
- In configuration files;
- In cloud technologies.

For example, many web services return user data in JSON format. This allows programmers to process data quickly. When comparing JSON and XML formats, JSON has the advantage of simplicity, speed, and small size. XML may be more convenient for complex documents and special structures.

The results show that JSON is one of the most popular data exchange formats today, occupying an important place in the field of programming and information technology. Its universality and convenience ensure its widespread use in the future. JSON (JavaScript Object Notation) is today the most popular format used for storing

and transmitting data over the Internet. Originally created for the JavaScript language, this format is now used by almost all programming languages. In addition, the contents of JSON format files can be viewed and edited using a simple text editor. The Python programming language has JSON elements and a module for working with it, for this it is enough to call it as follows.

import json

To **convert JSON** data to a Python data type, we use the json.loads() or json.load() functions. As above**, the json.loads()** function works directly with **JSON text**, while **the json.load()** function is used to read JSON files.**json.loads()**

This function takes a JSON string as a parameter and converts it to a Python data type. If it is a JSON string, you can parse it using the json.loads() function.

```
import json
x =  '{ "name":"Farrux", "age":26,
"city":"Tashkent"}'
y = json.loads(x)
print(y["age"])
```

the json.dumps()

figure 1**. Python JSON Parsing**

Example

Convert JSON to JSON string. If you have a Python object, you can convert it to a JSON string using the json.dumps() function. We use the json.dumps() function to convert the data to JSON format:

```
import json
x = 10
x_json = json.dumps(x)
ism = "anvar"
ism_json = json.dumps(ism)
```

```
sonlar = [12, 45, 23, 67]
sonlar_json = json.dumps(sonlar)
```

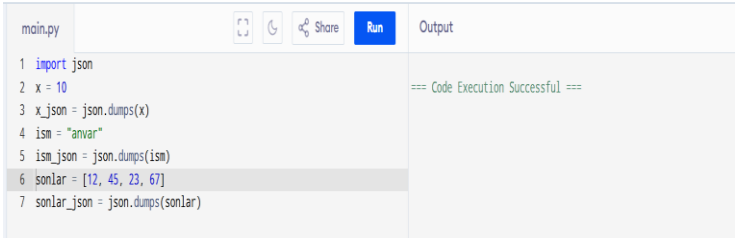

figure 2. Python JSON Serialization Example Using json.dumps()

the json.dump()

To convert the data to JSON format and write it to a file, we call the json.dump() function. We specify the variable and file names as function parameters. Of course, we must first open the file for writing:

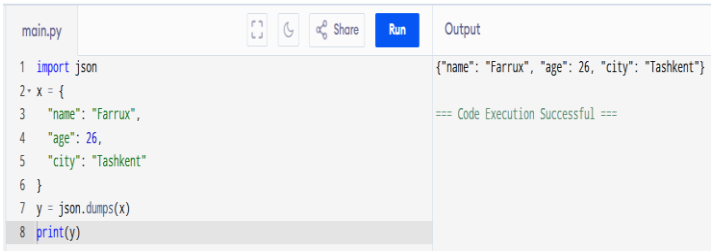

figure 3. Python Dictionary to JSON String Conversion Using json.dumps()

```
import json
x = {
  "name": "Farrux",
  "age": 26,
  "city": "Tashkent"
}
y = json.dump(x)
print(y)
```

the json.load()

This function is used to load the contents of JSON files into Python.

```
filename = 'bemor.json'
```

```
with open(filename) as f:
 bemor = json.load(f)
print(type(bemor))
```

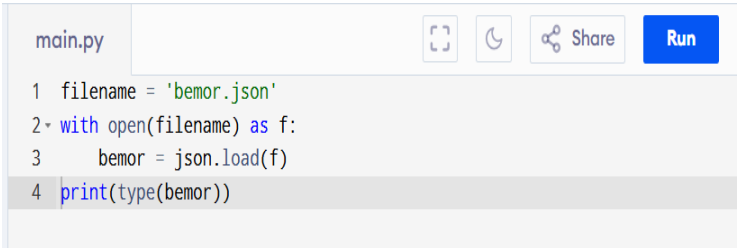

figure 4. Reading JSON File and Loading Data into Python Using json.load()

We can convert the following types of Python objects to JSON strings:

- dict
- list
- tuple
- string
- int
- float
- True
- False
- None

WORKING WITH JSON

Often when we receive JSON files over the Internet, the data is in the form of a multi-layered dictionary. To extract the information we need from the JSON text, we may need to parse the dictionary a bit, finding its keys and values. This is especially true for very long JSON files. Therefore, we can also learn to work with dictionaries to work effectively with JSON.

JSON actually has two main structures:

- **Dictionary (object)** — consists of keys and values
- **List (array)** — a sequence of several values

In practice, JSON files are made up of a combination of these two structures.

1. Receiving data

JSON is often obtained from the Internet (via API) or from a file.

2. Understanding the structure

It is necessary to consider how the data is located in the JSON. It is important to determine which key contains which value.

3. Extracting the necessary information

Only the necessary part is obtained from the JSON. Because JSON is usually very large and not all the information is needed.

4. Parsing

Working on the obtained information — checking it, sorting it, or using it elsewhere.

• **Internal structure**

JSON can contain more JSON. This makes finding information a little more complicated.

• **Keys (names)**

Each piece of information is obtained by key. It is very important to correctly identify the keys.

• **Data types**

JSON can contain text, numbers, lists, and even empty values.

• **Errors**

Sometimes the required key may not exist. Therefore, it is always recommended to check and work.

• JSON is widely used in the following areas:

• Web programming (exchange of information between sites and applications)

• Working with APIs

• Mobile applications

• Configuration files

• Data storage and transfer

Practical tips

• First try to understand the JSON file in general

• When working with large JSON, study it in parts

• It is very useful to know how to work with dictionaries and lists

• Always extract only the necessary information

Working with JSON is more about understanding and correctly parsing the data than reading it. If you understand the JSON structure well, you can easily work with large and complex data. Since JSON is used in almost all modern applications today, learning it is very important for every programmer.

Conclusion: JSON (JavaScript Object Notation) files are one of the most important formats in programming and information exchange today. They allow you to store data in a structured, easy-to-understand and compact form. The main advantage of JSON is its universality, that is, it is supported by almost all programming languages (Python, JavaScript, Java, C #, etc.). Therefore, it is widely used as a standard format for transferring information over the Internet (APIs, web services).

When working with JSON, it is important to load data into Python objects (json.load() and json.loads()), and vice versa (json.dump() and json.dumps()). These functions make it very easy to work with files, save and retrieve data. The JSON structure is very similar to Python data types such as dictionary and list, so it is easy to understand and use.

Also, JSON files often have a complex, multi-level (nested) structure. In such cases, skills in working with dictionaries and lists are required to extract the necessary information. Effective work with JSON requires a programmer to have a good understanding of data structures and their internal relationships.

In practice, JSON is widely used in the following areas:

• In web programming, when exchanging information between the server and the client

• In working with APIs

• In storing configuration (settings) files

• In storing data temporarily or permanently

In short, mastering working with JSON files is one of the necessary skills for every programmer.

Because most modern programs exchange information via JSON. Correct and effective use of JSON increases the efficiency of the program, simplifies the code, and facilitates integration between different systems.

References

1. Bray, T. (2017). *The JavaScript Object Notation (JSON) data interchange format (RFC 8259).* IETF. https://www.rfc-editor.org/rfc/rfc8259
2. ECMA International. (2017). *The JSON data interchange syntax (ECMA-404).* https://www.ecma-international.org/publications-and-standards/standards/ecma-404/
3. Lutz, M. (2013). *Learning Python* (5th ed.). O'Reilly Media.
4. McKinney, W. (2017). *Python for data analysis* (2nd ed.). O'Reilly Media.
5. Downey, A. B. (2015). *Think Python: How to think like a computer scientist* (2nd ed.). Green Tea Press. https://greenteapress.com/wp/think-python/
6. Zed Shaw. (2014). *Learn Python the hard way* (3rd ed.). Addison-Wesley.
7. Python Software Foundation. (2024). *json — JSON encoder and decoder.* Python Documentation. https://docs.python.org/3/library/json.html
8. Severance, C. (2016). *Python for everybody: Exploring data in Python 3.* CreateSpace Independent Publishing Platform.
9. Tanenbaum, A. S., & Van Steen, M. (2017). *Distributed systems: Principles and paradigms* (2nd ed.). Pearson.