JSON – текстовий формат обміну даними

Посилання на матеріали теми:

https://uk.wikipedia.org/wiki/JSON
https://www.JSON.org
https://www.w3schools.com
https://uk.soringpcrepair.com/how-to-open-json/
https://freexbcodes.com/rozvitok/19-instrumenti-json-dlja-rozboru-formatuvannja/

Загальна характеристика синтаксису JSON

JSON будується на двох структурах:
1) набір пар назва/значення. У різних мовах це реалізовано як об'єкт, запис, структура, словник, хеш-таблиця, список з ключем або асоціативним масивом;
2) впорядкований список значень. У багатьох мовах це реалізовано як масив, вектор, список, або послідовність.

Це універсальні структури даних. Теоретично всі сучасні мови програмування підтримують їх у тій чи іншій формі. Оскільки JSON використовується для обміну даними між різними мовами програмування, то є сенс будувати його на цих структурах.

У JSON використовуються такі їхні форми:

- **Об'єкт** — це послідовність пар назва/значення. Об'єкт починається з символу { і закінчується символом } . Кожне значення слідує за : і пари назва/значення відділяються комами.
- **Масив** — це послідовність значень. Масив починається символом [і закінчується символом]. Значення відділяються комами.
- **Значення** може бути рядком в подвійних лапках, або числом, або логічними true чи false, або null, або об'єктом, або масивом. Ці структури можуть бути вкладені одна в одну.
- **Рядок** — це послідовність з нуля або більше символів юнікода, обмежена подвійними лапками, з використанням escape-послідовностей, що починаються зі зворотної косої риски \. Символи представляються простим рядком.

Тип Рядок (String) дуже схожий на String в мовах C і Java. Число теж дуже схоже на C- або Java-число, за винятком того, що вісімкові та шістнадцяткові формати не використовуються. Пропуски можуть бути вставлені між будь-якими двома лексемами.

Наступний приклад показує JSON представлення об'єкта, що описує людину. У об'єкті є рядкові поля імені і прізвища, об'єкт, що описує адресу, і масив, що містить список телефонів.

```
{
  "firstName": "Іван",
  "lastName": "Коваленко",
  "address": {
      "streetAddress": "вул. Грушевського 14, кв.101",
      "city": "Київ",
      "postalCode": 21000
  },
  "phoneNumbers": [
      "044 123-1234",
      "050 123-4567"
  ]
}
```

Перегляд JSON-файлів

Раніше ми розглядали приклад файла "data.xml", в якому були зображені дані про окремі товари в магазині, а також інформація про сам магазин. Еквівалентне зображення цього файла в форматі JSON може виглядати так (файл "test.json"):

```
{
"magazine": {
    "title":"Магазин Ваші меблі",
    "address": "вул.Південна,28",
    "work": "Працюємо щоденно від 9 до 20 години без вихідних"
    },
"list": "Меблі для квартир та офісів",
"table": {
   "name": "Стіл",
   "rank": "Серія: меблі квартирні",
   "production": "Виробництво: Стрийська меблева фабрика",
   "price": [ "5670 грн.","8300 грн.","12500 грн.","7800 грн." ],
   "service": {
      "sale": "Оплата: перерахунок,готівка",
      "delivery": "Доставка: магазин,власна",
      "mounting": "монтаж: майстер,власний",
      "guarantee": "гарантія: 4 роки"
   }
  },
"chair": {
   "name": "Стілець",
   "rank": "Серія: меблі офісні",
   "production": "Виробництво: фабрика 'Закарпаття'",
   "price": [ "ціна 950 грн.","1350 грн.","640 грн.","800 грн." ],
   "service": {
      "sale": "Оплата: перерахунок,готівка",
      "delivery": "Доставка: власна"
   }
  },
"sofa": {
   "name": "Диван",
   "rank": "Серія: меблі квартирні",
   "production": "Виробництво: компанія 'Меблі-сервіс'",
   "price": [ "ціна 14600 грн.", "18000 грн.", "22900 грн." ],
   "service": {
      "sale": "Оплата: перерахунок",
      "delivery": "Доставка: магазин,власна",
      "mounting": "монтаж: майстер,власний",
      "guarantee": "гарантія: 3 роки"
   }
  }
}
```

Зауважимо, що перетворення з формату XML до формату JSON неоднозначне. Той самий текст XML можна зобразити різними варіантами тексту JSON. Проте, завжди можна зберегти однакову структуру даних.

Щоб переглянути дані формату JSON з сценарію python, можна використати дві можливості.

1) безпосередньо переглянути сам файл, запустивши на виконання дочірній процес:

```
destfile = "test.json"
import os
import subprocess
dirname = os.path.abspath('.')  # шлях від кореня до поточної папки
filename = os.path.join(dirname, destfile)
parprog = subprocess.Popen(["notepad.exe", filename])
```

2) перетворити json-файл у внутрішнє зображення і надрукувати у вікні виконання:

```
destfile = "test.json"
import json    # використаємо модуль json
from pprint import pprint  # "гарний" друк
# module provides a capability to "pretty-print" arbitrary Python
#  data structures in a form which can be used
#   as input to the interpreter
with open(destfile) as data_file:
  data = json.load(data_file)
pprint(data)
```

Зауважимо, що в цьому разі елементи даних друкують в алфавітному порядку.

Загальна схема сценарію роботи з JSON

1) Встановити повну веб-адресу ресурсу (API ендпоінт), наприклад:

```
remoteaddr =
   "https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange"
```

Ресурс за такою адресою має бути побудований в форматі JSON. Якщо формат ресурсу є інакшим, то, можливо, сервер дозволяє виконати перетворення формату – це треба визначити окремо. А також уточнити веб-адресу для варіанта JSON.

Часто для отримання даних в форматі JSON достатньо дописати в кінець url-адреси параметр `mode=json`. Якщо параметр єдиний, то попереду записуємо ?, а якщо вже є параметри – то об'єднуємо знаком &:

```
remoteaddr =
"https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange/?mode=json"
```

2) Виконати запит до сервера на отримання файла даних одним з методів, розглянутих раніше. Наприклад:

```
import urllib.request
# дані, отримані модулем urllib, завжди повертають як рядки байтів
remoteaddr =
"https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange/?mode=json"
destfile = "bank-exc.json"   # ім'я для отриманого файла
#   копіювати файл з перетворенням кодування ANSI в UTF-8
#    варіант 1: використати функцію urlopen()
remotefile = urllib.request.urlopen(remoteaddr)
#  remotefile - об'єкт файла для читання
with open(destfile,"w") as fsave :   # зберегти у текстовий файл
    fsave.write(remotefile.read().decode(encoding='utf-8'))
    # копіювання
remotefile.close()
```

Необхідно мати попередньо інформацію про стандарт кодування текстового файла на сервері. Для файлового ресурсу, зазначеного вище, кодування тексту є в стандарті ANSI. Подальші методи роботи з json-файлами вимагають кодування UTF-8. Тому необхідна функція перетворення кодування `decode(encoding='utf-8')`, яка декодує (розшифровує) байтовий потік ANSI і перетворює в коди UTF-8. Результатом перекодування є текстовий рядок, тому зберігати його в файл треба як текст: `open(destfile,"w")`.

Якщо ж перекодування не потрібне, тоді можна просто копіювати отриманий файл в інший файл як байтовий потік (розглядали раніше).

3) Отриманий і збережений файл можна переглянути для контролю способами, викладеними вище.

4) Перетворити json-документ в об'єкт python (десеріалізація). Десеріалізацію виконують за такою таблицею перетворень:

JSON	**Python**
object	dict
array	list
string	str
number (int)	int
number (real)	float
true	True
false	False
null	None

Як видно з таблиці, з числа структур даних python є лише дві: `dict` і `list`, а також текстові рядки `str`.

Перетворення json-документа було показано вище:

```
import json
destfile = "test.json"
with open(destfile) as data_file:
  data = json.load(data_file)
```

Отримали посилання `data` на структуру python. Тепер вся робота аналізу json-документа перекладається на `data`. Отже, будуть потрібні методи і функції для роботи з словниками, списками і текстовими рядками.

5) Виконати програмне дослідження python-структури документа. Приклад окремих досліджень:

```
import json

destfile = "test.json"

from pprint import pprint

with open(destfile) as data_file:
  data = json.load(data_file)
```

```
# ------------ вивчення структури документа -----------
print("Тип цілого документа: ", type(data).__name__)
print("Документ має ", len(data), "елементів")
print("Тип кожного елемента як цілого:",
      *[type(ob).__name__ for ob in data])
print("\nТипи значень окремих елементів документа за ключами:",
      *[type(data[ob]).__name__ for ob in data.keys()])
print("\nТип елемента 'chair':", type(data["chair"]).__name__)
print("\nЗначення цілого елемента 'chair':\n", data["chair"])
print("\nПоелементний перелік частин 'chair':")
print(*[ subvalue for subvalue in data["chair"] ], sep="\n")
print(*[ subvalue for subvalue in data["chair"].items() ],
       sep="\n")
print("\nСписок значень окремих елементів документа:",
       *[str(ob) + " : " + str(data[ob]) for ob in data.keys()],
       sep="\n\n")
```

Зауважимо, що показані функції друкування будуть дійсними лише в тому випадку, коли тип і значення кожного елемента в документі json відповідає способу звертання до нього. Це не є універсальна схема дослідження будь-якого документа json.

Деякі задачі аналізу json-документа

Будова розв'язків задач залежить від структури конкретного json-документа. Зокрема, структура тестового прикладу `test.json` є частково регулярною. Тому в загальному випадку треба: 1) перевіряти типи елементів документа; 2) перевіряти присутність потрібних елементів.

Задача 1. Надрукувати список назв товарів і ціни, які є в документі.

```
import json
destfile = "test.json"

with open(destfile) as data_file:
  data = json.load(data_file)

from pprint import pprint
#pprint(data)

# --- надрукувати список назв товарів і ціни ------
key1 = data.keys()  # отримати всі ключі першого рангу
#print(list(key1))   # контрошль списку ключів першого рангу
for k in key1:  # перевірка за кожним ключем
  if isinstance(data[k], dict):  # чи значення є словником
    key2 = data[k].keys()   # ключі внутрішнього словника
    #print(list(key2))  # контроль списків внутрішніх ключів
    if 'name' in key2 and 'price' in key2:  # чи є потрібні дані
      #print('+')
      print("{0:10}{1}".format(data[k]['name'], data[k]['price']))
```

В текст програми додані функції pprint, print, за допомогою яких можна спостерігати кроки виконання програми. Для цього треба їх розкоментувати, але для самого розв'язку вони не потрібні.

Приклад отриманих результатів:

```
Стіл      ['ціна 5670 грн.', '8300 грн.', '12500 грн.', '7800 грн.']
Стілець   ['ціна 950 грн.', '1350 грн.', '640 грн.', '800 грн.']
Диван     ['ціна 14600 грн.', '18000 грн.', '22900 грн.']
```

Задача 2. Показати список постачальників продукції в магазин.

```
import json
destfile = "test.json"

# --- показати список постачальників продукції в магазин -----
key1 = data.keys()  # ключі першого рангу
for k in key1:  # перевірка за кожним ключем
  if isinstance(data[k], dict):  # чи значення є словником
    key2 = data[k].keys()   # ключі внутрішнього словника
    if 'name' in key2 and 'production' in key2 :
      # якщо є дані про постачальника
      print("{0:10}{1}".format(data[k]['name'],
                              data[k]['production']))
```

Приклад отриманих результатів:

```
Стіл      Виробництво: Стрийська меблева фабрика
Стілець   Виробництво: фабрика 'Закарпаття'
Диван     Виробництво: компанія 'Меблі-сервіс'
```

.