



לתוכנות: המדריך לפתחים JSON

מדריך מוקף לעבודה עם JSON עבור מפתחים מתחילה ובינוניים. נלמד על המבנה, היתרונות והשימושים הנפוצים של JSON, וכייז לעבד אותו בפייתון ושפות תכנות אחרות.

Presented by Saleem Khoury



מהו JSON ולמה הוא חשוב?

הו פורטט טקסטואלי פשוט לייצוג ושמירת מידע (JavaScript Object Notation) בצורה מובנית. הוא נחשב לאחד הפורטטים הנפוצים ביותר להעברת נתונים בין מערכות, במיוחד באינטרנט:

- תקשורת בין שרת לאפליקציה
- העברת נתונים בין דפדפן ל-API
- שמירת הגדרות צורה
- תיעוד נתונים בפורטט קרייא



ה יתרונות המרכזיים של JSON

שפה בלתי תלואה

למרות שנוצר מתוכה JavaScript, Java, C/C++
כמעט כל שפת תכנות (Python, Java, PHP, #C ועוד) תומכת בקריאה וכתיבה
של JSON.



קל משקל

לא "כבד" כמו פורמטים אחרים (דוגמת XML). מעביר את המידע ביעילות
ובמהירות, מה שהוא חשוב במיוחד
באפליקציות רשת.



בסיס טקסט

קל לקרוא על ידי בני אדם ובזמן קצר
לפענוח על ידי מחשבים. המבנה
אינטואיטיבי ופשוט להבנה.



יתרונות אלו הפכו את JSON לסטנדרט התעשייתי העיקרי להעברת נתונים ברשת.

המבנה הבסיסי של JSON

בנייה משני סוגים מבנים עיקריים נספר:

- **אובייקטים** - מיוצגים ע"י {} (סוגרים מסולסלים) ומכילים זוגות של "מפתח: ערך"
- **מערכות** - מיוצגים ע"י [] (סוגרים מרובעים), רישומות של ערכים

הערכים בתוך נספן יכולים להיות:

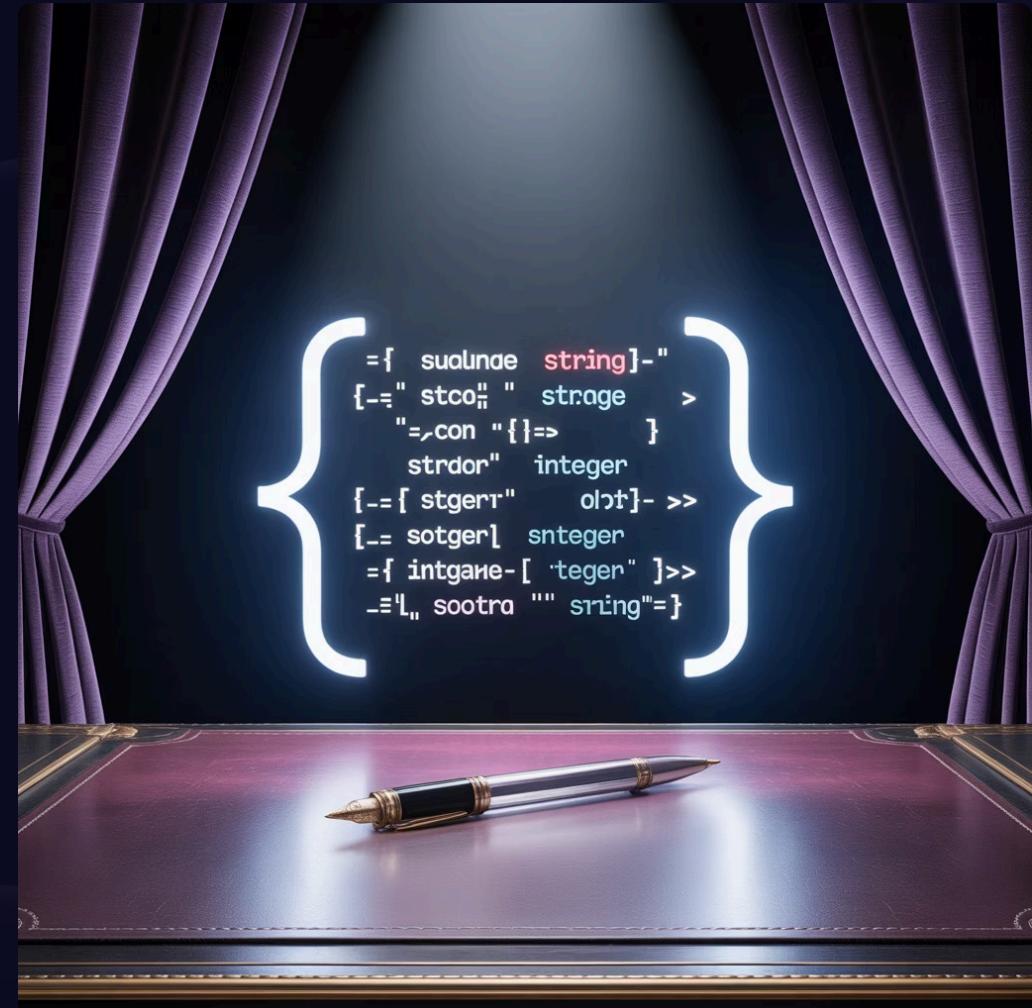
מספריים (לא גרשימי)

מחרוזות (בתוך גרשימים כפויים)

בוליאן (true/false)

null

אובייקט או מערך (מחזוניים)



לונגה בסיסית ל-JSON

```
{  
    "name": "Saleem",  
    "age": 30,  
    "isQA": true,  
    "skills": ["Python", "Selenium", "Testing"],  
    "address": {  
        "city": "Haifa",  
        "country": "Israel"  
    }  
}
```

במבנה זה אנו רואים:

- **מחרוזת**: "Saleem" - מוקפת תמיד בגרשיים כפויים
- **מספר**: 30 - לא גרשימים
- **בוליאן**: true - לא גרשימים (אך true או false בלבד)
- **מערך**: ["Python", "Selenium", "Testing"] - רשימה של ערכים
- **אובייקט מקומי**: address - אובייקט בתוך אובייקט

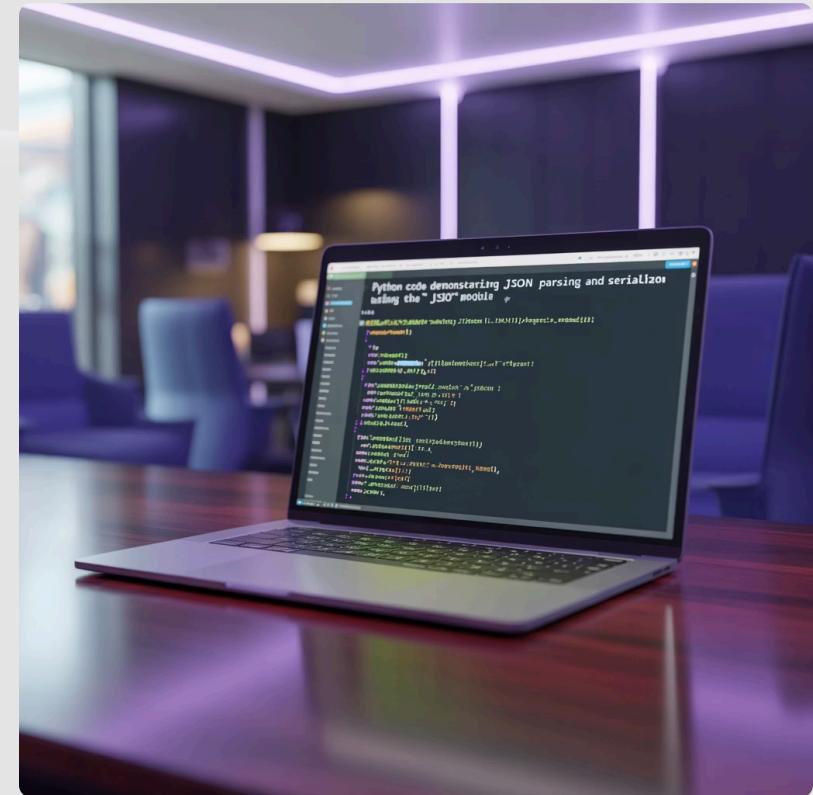
עובדת עם JSON בפייתון

פייתון מנגה עם מודול מובנה **json** שמאפשר לנו לעבוד בקלות עם JSON:

```
import json

# קרייה של JSON מחרוזת Parsing
data = '{"name": "Saleem", "age": 30}'
parsed = json.loads(data)
print(parsed["name"]) # Saleem

# כתיבה של JSON
person = {
    "name": "Saleem",
    "skills": ["Python", "Selenium"]
}
json_string = json.dumps(person, indent=4)
```



שיטות לב: True/False בפייתון יומרו ל-JSON true/false, ג-JSON false/true.
וכן להפוך.

קריאה וכתיבה לקובץ **JSON** בפייתן



כתיבה לקובץ

```
:with open("data.json", "w", encoding="utf-8") as file  
    json.dump(person, file, indent=4, ensure_ascii=False)
```



קריאה מקובץ

```
:with open("data.json", "r", encoding="utf-8") as file  
    data = json.load(file)
```

```
print(data["name"])
```

טיפ חשוב: שימוש לב לשימוש ב-`ensure_ascii=False` כאשר עובדים עם תוכן בעברית או שפות שאינן אנגלית, כדי לשמור על התווים המקוריים.

דוגמה מעשית: עבודה עם נתונים API

```
import json  
import requests  
  
#-קלות נתונים מ-API  
response = requests.get(  
    "https://api.example.com/data")  
  
#JSON-המרת התגובה ל-JSON  
data = response.json()  
  
#שימוש נתונים  
print(f"שם: {data['name']}")  
print(f"גיל: {data['age']}")  
  
#שמירת הנתונים לקובץ  
with open("api_data.json", "w",  
          encoding="utf-8") as f:  
    json.dump(data, f, indent=4,  
              ensure_ascii=False)
```



דוגמה זו מראה תהליך נפוץ מאוד בפיתוח מודרני:

1. שליחת בקשה ל-API
2. קבלת תגובה ופונCTIONALITY-unixה ה-JSON
3. שימוש הנתונים באפליקציה
4. שמירת הנתונים לשימוש עתידי

טיפים מתקדמים ומלכודות נפוצות

עבודה עם תאריכים

אינו תומך בתאריכים באופן מובנה - JSON
צריך להמיר תאריכים למחוזות ולהחזיר:

```
from datetime import datetime

# המרת לפניו המרת ל-JSON
date_str =
(datetime.now()).isoformat
```

טיפול בשגיאות

תמיד עטפו פונCTION ב-try/except

```
:try
data = json.loads(json_string)
except json.JSONDecodeError as
:e
:f"שגיאה בפונCTION: {e}"print
```

פורמט תקין

מחמיר מאד בתחריר JSON

- מפתחות חייבים להיות במרקאות כפולות (" ")
- לא ניתן להשתמש בתגבות (//)
- לא ניתן להשתמש בפסיקים מיוחדים בסוף רשימה

לכון: תמיד בדקו את תקינות ה-JSON לפני שליחה או אחרי קבלת, במיוחד בסביבות ייצור.

סיכום: NOSER לשימוש יומיומי

למזרנו היום:

- מהו NOSER ומדוע הוא כה נפוץ
- המבנה הבסיסי של NOSER: אובייקטים ומערכות
- סוגים הערכים השונים בנוסר
- עובודה עם NOSER בPython: קריאה וכתיבה
- טיפול בקבצי NOSER
- שימושים מעשיים עם API
- מלכודות נפוצות וכיוצא להימנע ממן

הצעד הבא: התחלו להשתמש בנוסר בפרויקטים שלכם לשמרות הנדרות, תקשורת עם שירותים מרוחקים, או כפורמת לאחסון נתונים.

