



# לתכנות: המדריך למפתחים JSON

מדריך מקיף לעבודה עם JSON עבור מפתחים מתחילים ובינוניים. נלמד על המבנה, היתרונות והשימושים הנפוצים של JSON, וכיצד לעבוד איתו בפיתוח ושפות תכנות אחרות.

**Presented by Saleem Khoury**



**BRIGHTWAY**   
CAREER GROWTH STUDIO

# מהו JSON ולמה הוא חשוב?

הוא פורמט טקסטואלי פשוט לייצוג ושמירת מידע (JavaScript Object Notation) **JSON** בצורה מובנית. הוא נחשב לאחד הפורמטים הנפוצים ביותר להעברת נתונים בין מערכות, במיוחד באינטרנט:

- תקשורת בין שרת לאפליקציה
- העברת נתונים בין דפדפן ל-API
- שמירת הגדרות תצורה
- תיעוד נתונים בפורמט קריא



# היתרונות המרכזיים של JSON



## מבוסס טקסט

קל לקריאה על ידי בני אדם ובזמנית קל לפענוח על ידי מחשבים. המבנה אינטואיטיבי ופשוט להבנה.



## קל משקל

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



## שפה בלתי תלויה

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

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



# המבנה הבסיסי של JSON

בנוי משני סוגי מבנים עיקריים JSON:

- **אובייקטים** - מיוצגים ע"י { } (סוגריים מסולסלים) ומכילים זוגות של "מפתח: ערך"
- **מערכים** - מיוצגים ע"י [ ] (סוגריים מרובעים), רשימות של ערכים

## הערכים בתוך JSON יכולים להיות:

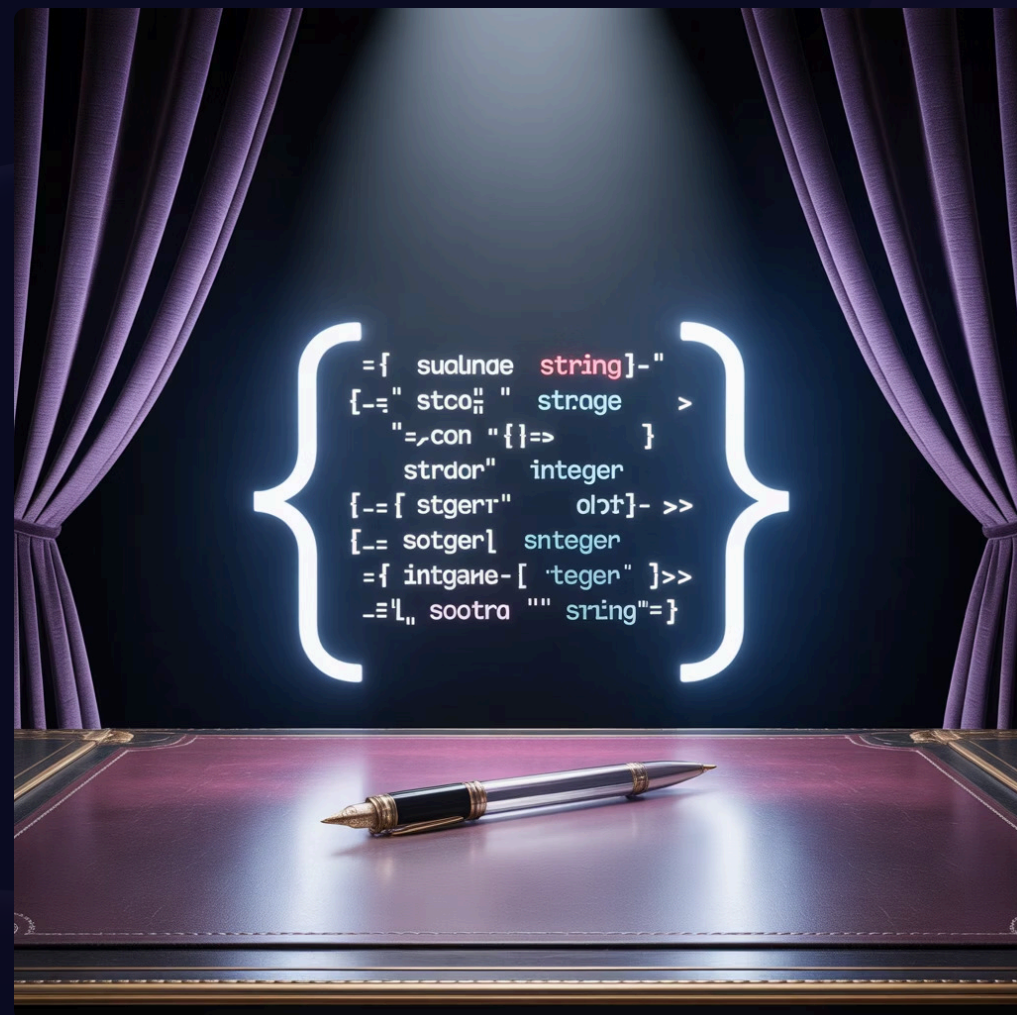
מספרים (ללא גרשיים)

מחרוזות (בתוך גרשיים כפולים)

בוליאן (true/false)

null

אובייקט או מערך (מקוננים)



# דוגמה 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 כפולים
- **מספר:** 30 - ללא גרשיים
- **בוליאן:** true - ללא גרשיים (true או false בלבד)
- **מערך:** ["Python", "Selenium", "Testing"] - רשימה של ערכים
- **אובייקט מקונן:** address - אובייקט בתוך אובייקט

# עבודה עם JSON בפייתון

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

```
import json
```

```
# ממחרוזת JSON של (Parsing) קריאה
```

```
data = '{"name": "Saleem", "age": 30}'
```

```
parsed = json.loads(data)
```

```
print(parsed["name"]) # Saleem
```

```
# JSON של (Serialization) כתיבה
```

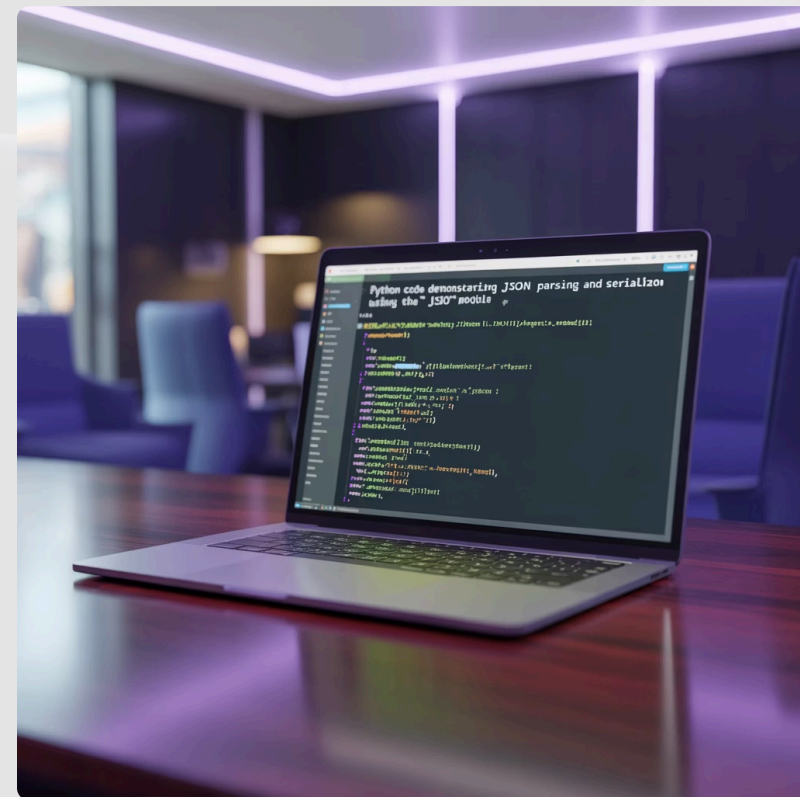
```
person = {
```

```
    "name": "Saleem",
```

```
    "skills": ["Python", "Selenium"]
```

```
}
```

```
json_string = json.dumps(person, indent=4)
```



**שימו לב:** True/False בפייתון יומרו ל-true/false ב-JSON,

וכן להפך.

# קריאה וכתיבת קבצי JSON בפייתון



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

# דוגמה מעשית: עבודה עם נתוני API

```
import json
import requests

# קבלת נתונים מ-API
response = requests.get(
    "https://api.example.com/data")

# JSON-המרת התגובה ל
data = response.json()

# שימוש בנתונים
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 ופענוח ה-JSON
3. עיבוד הנתונים באפליקציה
4. שמירת הנתונים לשימוש עתידי



## טיפים מתקדמים ומלכודות נפוצות

### עבודה עם תאריכים

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

```
from datetime import datetime

# המרה לפני המרה ל-JSON
date_str =
(datetime.now()).isoformat
```

### טיפול בשגיאות

תמיד עטפו פענוח JSON ב-try/except:

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

### פורמט תקין

מחמיר מאוד בתחביר JSON:

- מפתחות חייבים להיות במרכאות כפולות (" ")
- לא ניתן להשתמש בתגובות (//)
- לא ניתן להשתמש בפסיקים מיותרים בסוף רשימה

**זכרו:** תמיד בדקו את תקינות ה-JSON לפני שליחה או אחרי קבלה, במיוחד בסביבות ייצור.

# סיכום: JSON לשימוש יומיומי

למדנו היום:

- מהו JSON ומדוע הוא כה נפוץ
- המבנה הבסיסי של JSON: אובייקטים ומערכים
- סוגי הערכים השונים בJSON
- עבודה עם JSON בפיייתון: קריאה וכתיבה
- טיפול בקבצי JSON
- שימושים מעשיים עם API
- מלכודות נפוצות וכיצד להימנע מהן

**הצעד הבא:** התחילו להשתמש ב-JSON בפרויקטים שלכם לשמירת הגדרות, תקשורת עם שירותים מרוחקים, או כפורמט לאחסון נתונים.

