

# פרויקט: מנתח לוגים של תעבורת רשת

שלב 4 - Generators (yield)

## מטרת השלב

עד עכשיו טענו את כל הקובץ לזיכרון. זה עובד עם 10,000 שורות, אבל מה קורה עם קובץ של 10 מיליון שורות?

בשלב זה נלמד לעבד קבצים ענקיים בצורה יעילה באמצעות **Generators**.

### הבעיה

הקוד הבא טוען את כל הקובץ לזיכרון:

```
def get_all_suspicious(filepath):  
    suspicious_lines = []  
    with open(filepath, 'r') as file:  
        for line in file:  
            if is_suspicious(line):  
                suspicious_lines.append(line)  
    return suspicious_lines # מחזיר רשימה ענקית!
```

אם הקובץ ענק - התוכנית תקרוס או תהיה איטית מאוד.

**הפתרון:** במקום לאסוף הכל לרשימה ולהחזיר בסוף, נשתמש ב- `yield` כדי להחזיר תוצאות אחת אחת תוך כדי עיבוד.

```
def get_all_suspicious(filepath):
    with open(filepath, 'r') as file:
        for line in file:
            if is_suspicious(line):
                yield line # מחזיר שורה אחת ומחכה
```

## דרישות שלב 4

בשלב זה נמיר חלק מהפונקציות שכתבנו לשימוש ב-yield.

### 1 קריאת קובץ עם yield

כתבו פונקציה שקוראת את קובץ הלוג ומחזירה (yield) כל שורה כרשימה של שדות - בלי לטעון את כל הקובץ לזיכרון.

### 2 סינון שורות חשודות עם yield

כתבו פונקציה שמקבלת generator של שורות ומחזירה (yield) רק שורות שיש בהן לפחות חשד אחד.

הפונקציה לא טוענת את כל השורות - היא מעבדת שורה, בודקת, ואם חשודה - מחזירה אותה.

### 3 החזרת חשדות עם פרטי שורה

כתבו פונקציה שמקבלת generator של שורות ומחזירה (yield) עבור כל שורה חשודה tuple של (שורה, רשימת\_חשדות).

דוגמה לפלט:

```
(
    ["2024-01-15 03:23:45", "45.33.32.156", "10.0.0.5", "22", "SSH", "6000"],
```

```
[ "EXTERNAL_IP", "SENSITIVE_PORT", "LARGE_PACKET", "NIGHT_ACTIVITY" ]  
)
```

#### 4 ספירה בלי טעינה לזיכרון

כתבו פונקציה שמקבלת generator ומחזירה את מספר השורות החשודות - בלי לשמור אותן בזיכרון.

רמז: אפשר לספור עם לולאה פשוטה או עם `sum(1 for ...)`

#### 5 שרשרת generators

חברו את הפונקציות יחד: קריאה → סינון → הוספת פרטים. הריצו על קובץ הלוג וספרו כמה שורות חשודות יש.

##### דוגמה לשימוש:

```
lines = read_log("network_traffic.log")      # generator  
suspicious = filter_suspicious(lines)        # generator  
detailed = add_suspicion_details(suspicious) # generator  
  
count = count_items(detailed)  
print(f"Total suspicious: {count}")
```

**למה זה חשוב?** במודיעין עובדים עם קבצי לוג ענקיים - ג'יגהבייטים ואפילו טרהבייטים. אי אפשר לטעון הכל לזיכרון. generators מאפשרים לעבד כמויות אינסופיות של מידע בצורה יעילה.