

מהו generator בשפת פייתון ?

זהו כלי שיכול לחסוך לנו זיכרון.

בפונקצית פייתון לעיתים ישנו משפט return שמחזיר ערך או לא מחזיר כלום ומסיים את הפונקציה.

פונקציה מסוג generator היא פונקציה שגם יודעת להחזיר ערך בעזרת משפט מיוחד שנקרא: **yield** - וגם יודעת להשהות את עצמה, כך שאם נקרא לה שוב, היא תמשיך מהפקודה הבאה לאחר פקודת ה- yield האחרונה שהתבצעה.

במילים אחרות, אפשר לחזור אליה והיא לא תתחיל מהתחלה, אלא תמשיך בביצוע. השימושיות של דבר כזה באה לידי ביטוי במצב שרוצים לחסוך בזיכרון על ידי כך שלא מייצרים מיידית את כל הערכים שנרצה, אלא כל פעם ערך אחד בסדרה.

דוגמה למצב שבו אפשר לחסוך בזיכרון. נניח שנרצה לייצר רשימה של כל הריבועים של מספרים טבעיים מאפס ועד 100 אלף, אבל רשימה כזו תצרוך המון זיכרון. אפשר לייצר מה שנקרא: generator comprehension

מילת המפתח **next** מאפשרת לקרוא שוב ל- generator אחרי יצירתו.

לסיכום:

In summary, generators and iterators in Python offer a more memory-efficient and flexible approach to iteration, especially when dealing with large datasets or infinite sequences. They provide a cleaner and more readable code structure compared to traditional for loops.

```
import sys

def main():

    squares_lst = [n**2 for n in range(100000)]
    # List comprehenssion

    squares_gen = (n**2 for n in range(100000))
    # generator comprehension

    # print few squares

    print (next(squares_gen))
    print(next(squares_gen))
    print(next(squares_gen))

    # check memory allocation

    size_lst = sys.getsizeof(squares_lst)
    size_gen = sys.getsizeof(squares_gen)

    print ('list size is:', size_lst, ' gen
size is:', size_gen)

main()
```