yield

`yield` là một **từ khóa** trong Python, dùng để **tạm dừng một hàm và trả về một giá trị**, nhưng không **kết thúc hàm** như `return`.

Khi hàm sử dụng `yield`, nó trở thành một **generator function** – mỗi lần bạn gọi `next()`, hàm tiếp tục chạy từ chỗ đã dừng.

Ví dụ đơn giản dùng yield

```
def count_up_to(n):
    i = 1
    while i <= n:
        yield i
        i += 1

gen = count_up_to(3)

for num in gen:
    print(num)
```

Mỗi lần lặp, hàm `count_up_to` chạy đến `yield`, trả về `i`, rồi "ngủ đông" – đến vòng lặp tiếp theo thì tiếp tục chạy tiếp.

Kết quả

```
1
2
3
```

Ví dụ không dùng yeild, chúng ta phải dùng danh sach để lưu vì vậy nếu dữ liệu lớn thì cũng phải dùng nhiều RAM

```
def count_up_to(n):
    result = []
    i = 1
    while i <= n:
        result.append(i)
        i += 1
```

```
    return result

nums = count_up_to(3)

for num in nums:
    print(num)
```

`yield` khác gì với `return`?

`return`	`yield`
Trả về **một giá trị duy nhất**	Trả về **một chuỗi giá trị (generator)**
Kết thúc hàm ngay lập tức	Tạm dừng, giữ trạng thái và tiếp tục lần sau
Dùng để **hoàn thành** một tác vụ	Dùng để **tạo dòng dữ liệu** dần dần

Lợi ích của `yield`

- **Tiết kiệm bộ nhớ**: Không cần lưu toàn bộ danh sách trong RAM.
- **Hiệu suất cao**: Lười biếng – chỉ tính toán khi cần.
- **Rất hữu ích** khi làm việc với **file lớn, dữ liệu streaming, API phân trang**, v.v.

Ví dụ thực tế – đọc file lớn

```
def read_large_file(filename):
    with open(filename) as f:
        for line in f:
            yield line.strip()

for line in read_large_file("data.txt"):
    print(line)
```

Nếu `data.txt` chứa hàng triệu dòng, thì `yield` giúp đọc **từng dòng một** mà không làm đầy bộ nhớ.

Bạn có thể dùng `next()` để lấy từng giá trị từ generator:

```
gen = count_up_to(3)
print(next(gen))  # 1
print(next(gen))  # 2
```

Khi hết giá trị, nó sẽ raise `StopIteration`.

So sánh tổng quát:

Tiêu chí	Dùng `yield` (Generator)	Không dùng `yield` (Trả về list)
Bộ nhớ	Cực kỳ tiết kiệm (chỉ sinh 1 giá trị/lần)	Tốn bộ nhớ (lưu toàn bộ kết quả)
Hiệu suất	Cao khi xử lý dữ liệu lớn hoặc stream	Kém hơn với dữ liệu lớn
Tốc độ khởi tạo	Nhanh, không tính toán ngay	Tính toán toàn bộ trước
Dễ debug/logging	Hơi khó hơn một chút	Dễ quan sát dữ liệu
Tính liên tục (streaming)	Rất phù hợp (ví dụ đọc file, API nhiều trang)	Không phù hợp

Tác giả: Đỗ Ngọc Tú

Công Ty Phần Mềm VHTSoft

Phiên bản #2
Được tạo 18 tháng 4 2025 14:28:13 bởi Đỗ Ngọc Tú
Được cập nhật 18 tháng 4 2025 14:41:24 bởi Đỗ Ngọc Tú