python #2

▼ 문제

1. 100만 개의 숫자 리스트를 처리하는 프로그램
- 일반 리스트 방식은 메모리 과부하 발생 가능
- 이를 해결하기 위해 제너레이터 기반 반복 구조를 직접 구현

1) 0부터 999,999까지의 정수를 담는 리스트를 생성하고 총합 구하기.
2) 같은 결과를 제너레이터 함수로 구현.
3) 두 방법의 메모리 사용 차이를 sys.getsizeof()로 확인.

▼ 코드

▼ 전체 코드

```
import sys

# 1) 리스트 방식
numbers = list(range(1000000)) # 0부터 999,999 리스트 생성
list_sum = sum(numbers) # 합계 구하기
list_mem = sys.getsizeof(numbers) # 메모리 사용량 확인 (리스트 객체 크기)

print(f"리스트 합: {list_sum:,}")
print(f"리스트 메모리 사용량: {list_mem} bytes")

# 2) 제너레이터 방식
def number_gen(): # 제너레이터 함수 정의
    for i in range(1_000_000):
        yield i

gen = number_gen() # 제너레이터 객체 생성
gen_sum = sum(gen) # 합계 구하기
gen_mem = sys.getsizeof(gen) # 메모리 사용량 확인 (제너레이터 객체 크기)

print(f"제너레이터 합: {gen_sum:,}")
print(f"제너레이터 메모리 사용량: {gen_mem} bytes")

# 3) 결과 비교
print(f"메모리 사용량 비교: 리스트 {list_mem} bytes vs 제너레이터 {gen_mem} bytes")
```

▼ 1) 리스트 방식

```
import sys

# 1) 리스트 방식
numbers = list(range(1000000)) # 0부터 999,999 리스트 생성
list_sum = sum(numbers) # 합계 구하기
list_mem = sys.getsizeof(numbers) # 메모리 사용량 확인 (리스트 객체 크기)

print(f"리스트 합: {list_sum:,}")
print(f"리스트 메모리 사용량: {list_mem} bytes")
```

```
리스트 합: 499,999,500,000
리스트 메모리 사용량: 8000056 bytes
```

- numbers=list(range(1000000)) → sum(numbers)
  - 0~999,999를 리스트(numbers)로 만들어 합계를 구함
- sys.getsizeof(numbers)
  - 리스트 객체의 크기를 바이트 단위로 반환

▼ 2) 제너레이터 방식

```
# 2) 제너레이터 방식
def number_gen(): # 제너레이터 함수 정의
    for i in range(1_000_000):
        yield i

gen = number_gen() # 제너레이터 객체 생성
gen_sum = sum(gen) # 합계 구하기
gen_mem = sys.getsizeof(gen) # 메모리 사용량 확인 (제너레이터 객체 크기)

print(f"제너레이터 합: {gen_sum:,}")
print(f"제너레이터 메모리 사용량: {gen_mem} bytes")
```

```
제너레이터 합: 499,999,500,000
제너레이터 메모리 사용량: 200 bytes
```

- gen = number_gen() → sum(gen)
  - 제너레이터 객체 생성, 내부적으로 하나씩 값을 생성해 합산
- sys.getsizeof(gen)
  - 제너레이터 객체의 크기를 바이트 단위로 반환

▼ 3) 결과 비교

```
# 3) 결과 비교
print(f"메모리 사용량 비교: 리스트 {list_mem} bytes vs 제너레이터 {gen_mem} bytes")
```

```
메모리 사용량 비교: 리스트 8000056 bytes vs 제너레이터 200 bytes
```

- list(range(1000000))
  - list()로 감싸면 메모리에 100만 개 원소의 배열이 만들어지므로 크기가 크다(O(N)).
- gen = number_gen()
  - 제너레이터 객체는 “다음에 뭘 생산할지에 대한 상태”만 저장하고 실제 값(0, 1, 2, ...)을 미리 메모리에 올리지 않아서 크기가 작다(O(1)).