python #3

▼ 문제

제너레이터 기반 메모리 절약형 로직 작성

2. 아래 조건을 만족하는 제너레이터 함수 even_square_gen(n)을 작성

- 0 이상 n 미만의 정수 중 짝수만 제곱해서 하나씩 생성 (yield 활용)
- 해당 제너레이터를 이용해 0부터 1,000,000까지의 짝수의 제곱 총합을 계산
- 이 때의 메모리 사용량과 처리 속도를 비교 (time 모듈 활용)

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

```
import sys
import time

N = 1000000

# 1) 리스트 방식
start = time.time()
even_squares_list = [i * i for i in range(N) if i % 2 == 0]  # 모든 짝수 제곱을 리스트로 생성
even_sum_list = sum(even_squares_list) # 리스트의 모든 원소를 합산
end = time.time()

print("1) 리스트 방식")
print("합계:", even_sum_list) # 합계 출력
print("메모리 사용량:", sys.getsizeof(even_squares_list), "bytes")
print("실행 시간:", end - start, "초")

# 2) 제너레이터 방식
def even_square_gen(n): # 짝수 제곱 제너레이터
    for i in range(n): # 0부터 n-1까지 순회
        if i % 2 == 0: # 짝수인지 확인
            yield i * i # 짝수의 제곱을 하나씩 생성(yield)하여 반환

start = time.time()
even_sum_gen = sum(even_square_gen(N)) # 짝수 제곱을 하나씩 생성하며 합산
end = time.time()

print("2) 제너레이터 방식")
print("합계:", even_sum_gen) # 합계 출력
print("메모리 사용량:", sys.getsizeof(even_square_gen(N)), "bytes")
print("실행 시간:", end - start, "초")
print()
```

- even_squares_list = [i * i for i in range(N) if i % 2 == 0] → sum(even_squares_list)

    - 모든 짝수 제곱을 리스트로 생성 후 리스트의 모든 원소를 합산
- even_square_gen(n) → sum(even_square_gen(N))
    - 짝수 제곱을 생성하는 제너레이터 함수를 이용해 짝수 제곱을 하나씩 생성하며 합산

▼ 결과 해석

▼ 리스트 방식 vs 제너레이터 방식

```
1) 리스트 방식
합계: 166666166667000000
메모리 사용량: 4167352 bytes
실행 시간: 0.0649869441986084 초

2) 제너레이터 방식
합계: 166666166667000000
메모리 사용량: 208 bytes
실행 시간: 0.10016107559204102 초
```

- 리스트 방식의 메모리 사용량이 4167352 bytes로 제너레이터의 메모리 사용량 208 bytes보다 컸습니다.
- 리스트 방식의 sum 연산 실행 시간이 0.0649869441986084 초로 제너레이터 방식의 0.10016107559204102 초보다 빨랐습니다.
    - 두 방식의 속도 차이는 여러 번 수행 결과 리스트 방식이 빠른 경우도 있었고, 제너레이터 방식이 빠른 경우도 나왔습니다.

▼ N = 100000000에서의 비교

```
import sys
import time

N = 100000000
print(f"N={N}")

# 1) 리스트 방식
start = time.time()
even_squares_list = [i * i for i in range(N) if i % 2 == 0]  # 모든 짝수 제곱을 리스트로 생성
even_sum_list = sum(even_squares_list) # 리스트의 모든 원소를 합산
end = time.time()

print("1) 리스트 방식")
print("합계:", even_sum_list) # 합계 출력
print("메모리 사용량:", sys.getsizeof(even_squares_list), "bytes")
print("실행 시간:", end - start, "초")

# 2) 제너레이터 방식
def even_square_gen(n): # 짝수 제곱 제너레이터
    for i in range(n): # 0부터 n-1까지 순회
        if i % 2 == 0: # 짝수인지 확인
            yield i * i # 짝수의 제곱을 하나씩 생성(yield)하여 반환

start = time.time()
even_sum_gen = sum(even_square_gen(N)) # 제너레이터를 이용해 짝수 제곱을 하나씩 생성하며 합산
end = time.time()

print("2) 제너레이터 방식")
print("합계:", even_sum_gen) # 합계 출력
```

```
print("메모리 사용량:", sys.getsizeof(even_square_gen(N)), "bytes")
print("실행 시간:", end - start, "초")
print()
```

```
N=100000000
1) 리스트 방식
합계: 166666661666666700000000
메모리 사용량: 411943896 bytes
실행 시간: 8.67517375946045 초
2) 제너레이터 방식
합계: 166666661666666700000000
메모리 사용량: 208 bytes
실행 시간: 6.631064176559448 초
```

- N=100000000 (100배)로 수행 결과 리스트의 sum 연산 실행 시간이 8.67517375946045 초로 제너레이터 방식의 6.631064176559448 초보다 느리게 나왔습니다.