교수의 파이썬

02_3 집합과 튜플 축약

창원대학교 정보통신공학과 교수 박동규
(feat 강영민교수님)

널널한 교수의 파이썬

02_3 집합과 튜플 축약

창원대학교 정보통신공학과 교수 박동규
(feat 강영민교수님)

널널한 교수의 고급 파이썬

02_3 집합과 튜플 축약

창원대학교 정보통신공학과 교수 박동규
(feat 강영민교수님)

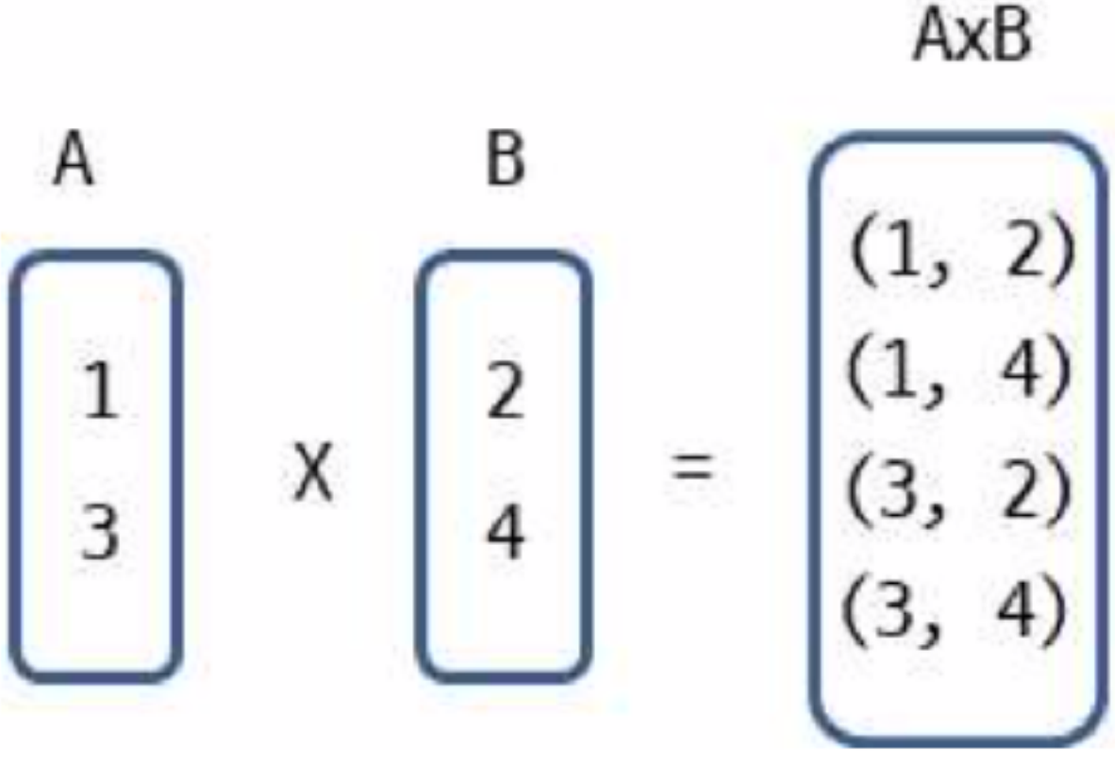
널널한 교수의 고급 파이썬

02_3 집합과 튜플 축약

창원대학교 정보통신공학과 교수 박동규
(feat 강영민교수님)

곱집합

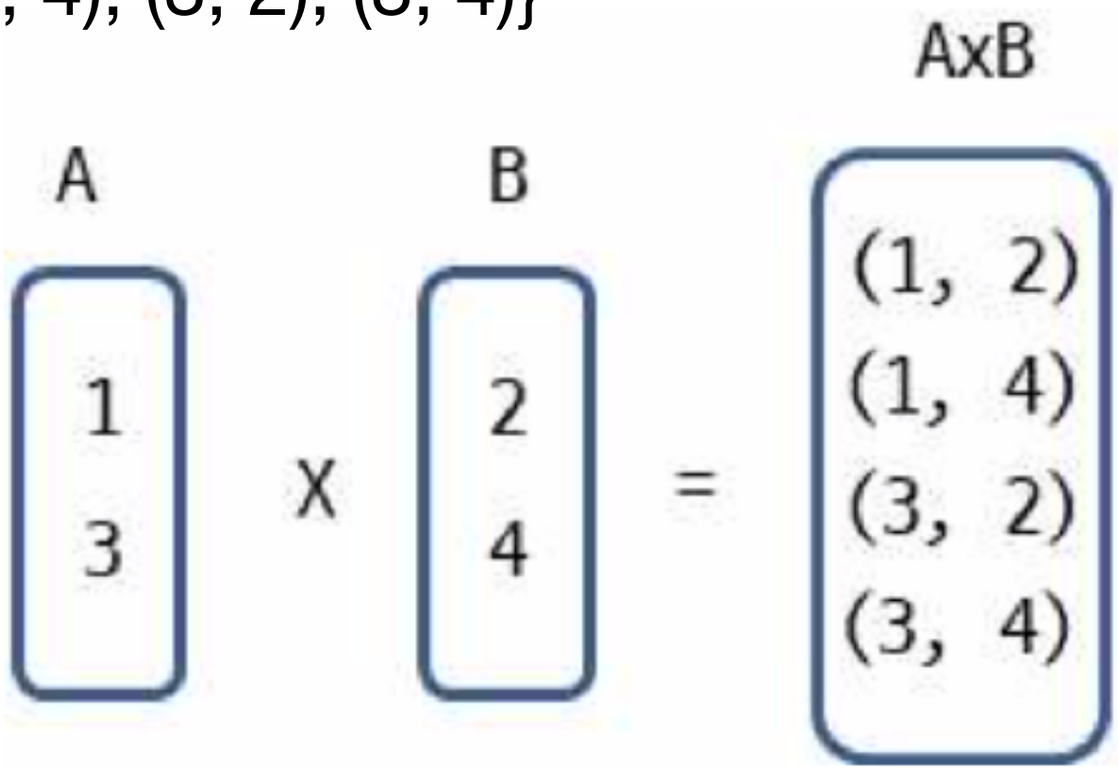
각 집합에서 임의의 원소를 가져와 만들 수 있는 조합을 하나의 튜플로 표현



곱집합

각 집합에서 임의의 원소를 가져와 만들 수 있는 조합을 하나의 튜플로 표현

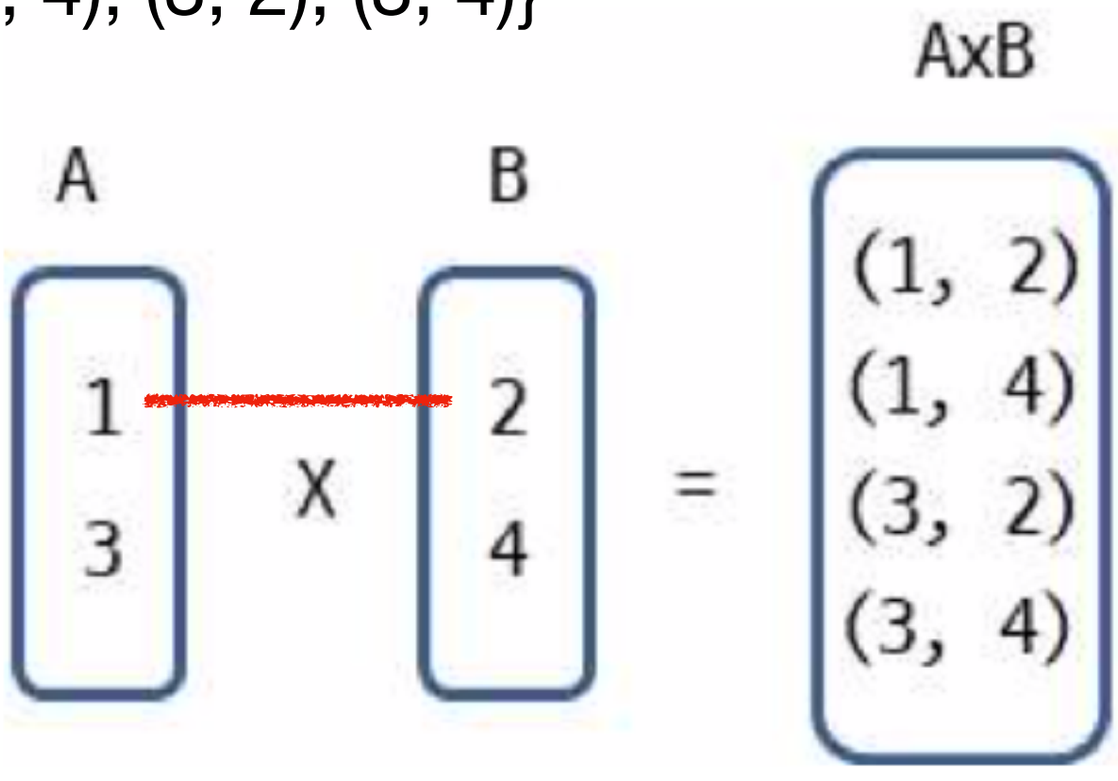
A = {1, 3}
B = {2, 4}
A x B = {(1, 2), (1, 4), (3, 2), (3, 4)}



곱집합

각 집합에서 임의의 원소를 가져와 만들 수 있는 조합을 하나의 튜플로 표현

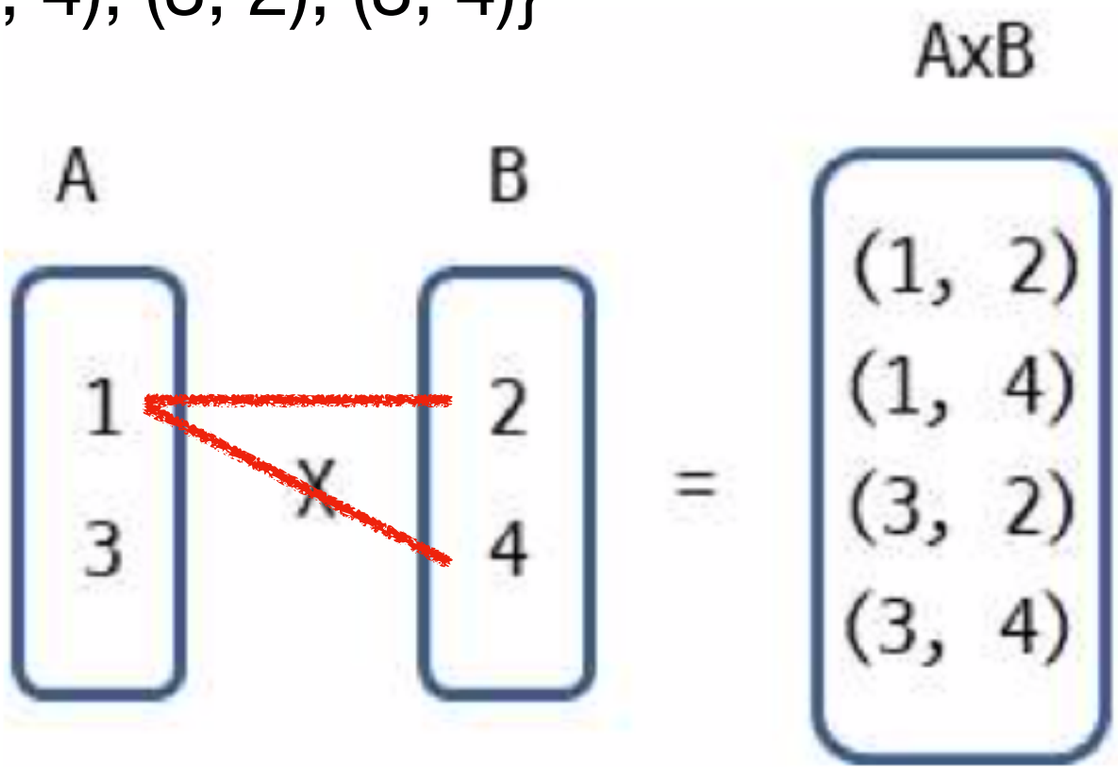
A = {1, 3}
B = {2, 4}
A x B = {(1, 2), (1, 4), (3, 2), (3, 4)}



곱집합

각 집합에서 임의의 원소를 가져와 만들 수 있는 조합을 하나의 튜플로 표현

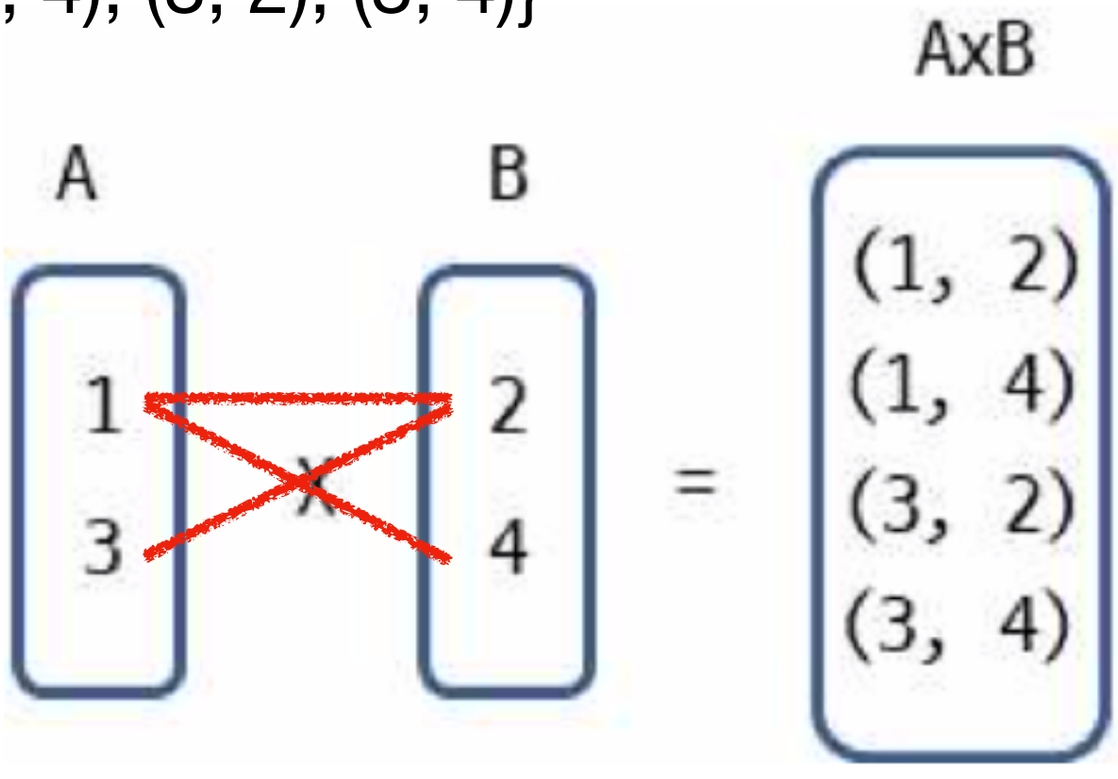
A = {1, 3}
B = {2, 4}
A x B = {(1, 2), (1, 4), (3, 2), (3, 4)}



곱집합

각 집합에서 임의의 원소를 가져와 만들 수 있는 조합을 하나의 튜플로 표현

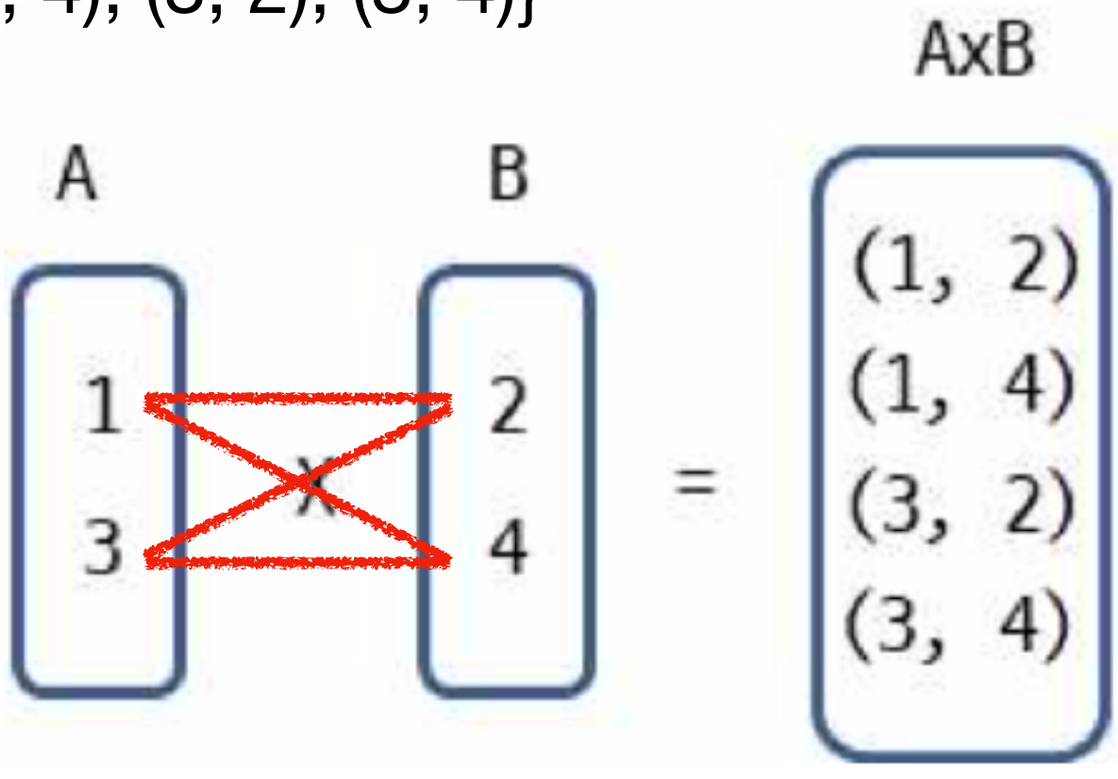
A = {1, 3}
B = {2, 4}
A x B = {(1, 2), (1, 4), (3, 2), (3, 4)}



곱집합

각 집합에서 임의의 원소를 가져와 만들 수 있는 조합을 하나의 튜플로 표현

A = {1, 3}
B = {2, 4}
A x B = {(1, 2), (1, 4), (3, 2), (3, 4)}



곱집합

```
def product_set(set1, set2) :
    res = set()
    for i in set1:
        for j in set2:
            res = res | {(i,j)}  # 이중 for 루프를 이용한 곱집합
    return res

A = {1, 3}     # 집합 A의 원소
B = {2, 4}     # 집합 B의 원소
AxB = product_set(A, B)      # A와 B의 곱집합 AxB(A x B가 아님)
print('A =', A)
print('B =', B)
print('AxB =', AxB)          # A와 B의 곱집합을 출력함
```

실행 결과

```
A = {1, 3}
B = {2, 4}
AxB = {(1, 2), (3, 2), (3, 4), (1, 4)}
```

축약표현으로 구한 곱집합

```
def product_set(set1, set2) :
     return {(i, j) for i in set1 for j in set2}

A = {1, 3}
B = {2, 4}
AxB = product_set(A,B)
print('A =', A)
print('B =', B)
print('AxB =', AxB)
```

이전 페이지의 수행결과와 동일하지만 더 축약된 코드임

실행 결과

```
A = {1, 3}
B = {2, 4}
AxB = {(1, 2), (3, 2), (3, 4), (1, 4)}
```

Lab

문제

```
A = {1, 2}
B = {'A', 'B', 'C'}
```

1) AxB =

2) BxA =

3) AxA =

4) BxB =

문제

A = {1, 3} 일 때

A x A x A =

해설

문제

- 주사위를 두 번 던져서 나올 수 있는 모든 경우

```
def product_set(set1, set2) :
    return {(i, j) for i in set1 for j in set2}

cases = { 1, 2, 3, 4, 5, 6 }
cases_2times = product_set(cases, cases)
print(cases_2times)
```

실행 결과

```
{(1, 1),
 (1, 2),
 (1, 3),
 (1, 4),
 ...
 (6, 6)}
```

문제

- 주사위를 두 번 던져 나오는 모든 경우에 대하여 숫자의 합을 구하기

```
def product_set(set1, set2) :
    return {(i, j) for i in set1 for j in set2}

cases = { 1, 2, 3, 4, 5, 6 }
cases_2times = product_set(cases, cases)

sum_set = { sum(tup) for tup in cases_2times }
print('sum_set =', sum_set)
sum_list = [ sum(tup) for tup in cases_2times ]
print('sum_list =', sum_list)
```

실행 결과

```
sum_set = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
sum_list = [4, 12, 11, 3, 8, 7, 6, 7, 3, 6, 10,
```

문제

- 주사위를 두 번 던져 나오는 모든 경우에 대하여 숫자의 합을 구하기

```
def product_set(set1, set2) :
    return {(i, j) for i in set1 for j in set2}

cases = { 1, 2, 3, 4, 5, 6 }
cases_2times = product_set(cases, cases)

sum_set = { sum(tup) for tup in cases_2times }
print('sum_set =', sum_set)
sum_list = [ sum(tup) for tup in cases_2times ]
print('sum_list =', sum_list)
```

실행 결과

```
sum_set = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
sum_list = [4, 12, 11, 3, 8, 7, 6, 7, 3, 6, 10,
```

문제

- 주사위를 두 번 던져 나오는 모든 경우에 대하여 숫자의 합을 구하기

```
def product_set(set1, set2) :
    return {(i, j) for i in set1 for j in set2}

cases = { 1, 2, 3, 4, 5, 6 }
cases_2times = product_set(cases, cases)

sum_set = { sum(tup) for tup in cases_2times }
print('sum_set =', sum_set)
sum_list = [ sum(tup) for tup in cases_2times ]
print('sum_list =', sum_list)
```

실행 결과

```
sum_set = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
sum_list = [4, 12, 11, 3, 8, 7, 6, 7, 3, 6, 10,
```

주사위를 세 번 던지는 경우

대화창 실습 : 주사위를 세 번 던져 얻는 모든 경우

```
>>> cases_3times = product_set(cases, cases_2times)
>>> cases_3times
{(3, (5, 4)), (1, (1, 3)), (1, (4, 3)), (2, (5, 2)), (3, (5, 1)), (3, (4, 3)), (2,
(3, 5)), (5, (6, 1)), (3, (6, 5)), (2, (2, 6)), (1, (4, 1)), (5, (1, 2)), (4, (3,
2)), (3, (2, 1)), (6, (2, 1)), (2, (6, 2)), (6, (4, 5)), (4, (3, 3)), (6, (3, 4)),
 << 중간 생략 >>
(4, 1)), (6, (3, 1)), (3, (2, 3)), (1, (1, 5)), (5, (2, 1)), (5, (4, 2)), (6, (4,
2)), (2, (3, 6)), (4, (3, 6)), (1, (6, 6)), (1, (1, 1)), (5, (5, 2)), (4, (6, 4)),
(4, (3, 1)), (4, (1, 2))}
```

대화창 실습 : 중첩 튜플의 원소에 sum을 사용할 경우에 발생하는 오류

```
>>> sums = [sum(tup) for tup in cases_3times]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 1, in <listcomp>
TypeError: unsupported operand type(s) for +: 'int' and 'tuple'
```

튜플내에 튜플이 포함된 경우 재귀적으로 원소의 합을 구한다

```
def tuple_sum(tup) :                  # tup내의 모든 항목의 합을 구하는 함수
    if isinstance(tup, int) :  # tup가 int 형이면 tup를 반환
        return tup
    else:
        accum = 0
        for element in tup :    # tup내의 모든 항목을 조회함
            accum += tuple_sum(element)  # 이 항목의 합을 구하기 위한 재귀적
return accum
```

isinstance(a, b) 함수 : 인자 a가 인자 b의 인스턴스이면 참(True)을 반환함
그렇지 않을 경우 거짓(False)를 반환함

```
def tuple_sum(tup) :                  # tup내의 모든 항목의 합을 구하는 함수
    if isinstance(tup, int) :  # tup가 int 형이면 tup를 반환
        return tup
    else:
        accum = 0
        for element in tup :    # tup내의 모든 항목을 조회함
            accum += tuple_sum(element)  # 이 항목의 합을 구하기 위한 재귀적 호출
    return accum

def product_set(set1, set2) :
    res = set()
    for i in set1:
        for j in set2:
            res = res | {(i,j)}  # 이중 for 루프를 이용한 곱집합
    return res

cases = { 1, 2, 3, 4, 5, 6 }
cases_2times = product_set(cases, cases)
cases_3times = product_set(cases, cases_2times)

sums = [tuple_sum(tup) for tup in cases_3times]
print(sums)   # tup내의 모든 항목의 합을 리스트로 출력
```

실행 결과

```
[4, 12, 12, 15, 12, 11, 11, 10, 10, 13, 9, 8, 10, 9, 8, 8, 12, 13, 10, 9, 8, 10, 8, 7
, 11, 13, 12, 11, 14, 7, 10, 9, 5, 10, 9, 14, 11, 9, 10, 12, 8, 13, 10, 16, 13, 8, 13
, 15, 8, 8, 16, 11, 6, 8, 10, 13, 8, 15, 10, 11, 12, 11, 10, 12, 6, 11, 14, 9, 10, 13
, 16, 5, 13, 17, 7, 10, 16, 18, 9, 11, 8, 15, 8, 10, 13, 13, 14, 6, 8, 7, 17, 12, 12,
9, 12, 6, 4, 8, 14, 7, 8, 15, 10, 9, 14, 15, 12, 14, 9, 9, 5, 12, 11, 12, 7, 14, 13,
13, 12, 12, 7, 10, 10, 11, 12, 7, 11, 6, 11, 17, 6, 13, 6, 14, 12, 15, 13, 9, 8, 9, 1
1, 7, 10, 8, 11, 10, 12, 14, 11, 5, 12, 7, 13, 16, 6, 13, 11, 9, 10, 9, 9, 9, 7, 3, 9
, 8, 7, 9, 11, 7, 8, 14, 15, 14, 11, 10, 4, 10, 10, 11, 6, 12, 9, 13, 12, 14, 11, 8,
13, 11, 12, 6, 12, 7, 10, 16, 7, 15, 14, 5, 9, 9, 11, 14, 11, 11, 10, 10, 9, 13, 13,
12, 5, 9, 11, 15]
>>>
```

주사위를 세 번 던져나오는 모든 눈의 합

Lab

응용

- 주사위를 세 번 던져 얻을 수 있는 경우의 수를 다음처럼 출력하는 코드를 작성하라.

```
주사위를 세 번 던져 발생할 수 있는 사건은 216 가지 경우가 존재합니다.
```

- tuple_sum 함수로 눈의 합이 10 이상인 경우의 수를 계산해 다음처럼 출력하라.

```
주사위를 세 번 던져 나온 눈의 합이 10 이상인 경우는 135 가지입니다.
```

감사합니다.