내장함수

실전에서 유용한 표준 라이브러리

- **내장 함수**: 기본 입출력 함수부터 정렬 함수까지 기본적인 함수들을 제공합니다.
  - 파이썬 프로그램을 작성할 때 없어서는 안 되는 필수적인 기능을 포함하고 있습니다.
- **itertools**: 파이썬에서 반복되는 형태의 데이터를 처리하기 위한 유용한 기능들을 제공합니다.
  - 특히 순열과 조합 라이브러리는 코딩 테스트에서 자주 사용됩니다.
- **heapq**: 힙(Heap) 자료구조를 제공합니다.
  - 일반적으로 우선순위 큐 기능을 구현하기 위해 사용됩니다.
- **bisect**: 이진 탐색(Binary Search) 기능을 제공합니다.
- **collections**: 덱(deque), 카운터(Counter) 등의 유용한 자료구조를 포함합니다.
- **math**: 필수적인 수학적 기능을 제공합니다.
  - 팩토리얼, 제곱근, 최대공약수(GCD), 삼각함수 관련 함수부터 파이(pi)와 같은 상수를 포함합니다.

모든 경우의 수를 고려해야 할 때 어떤 라이브러리를 효과적으로 사용할 수 있을까요?

순열: 서로 다른 n개에서 서로 다른 r개를 선택하여 일렬로 나열하는 것

- {'A', 'B', 'C'}에서 세 개를 선택하여 나열하는 경우: 'ABC', 'ACB', 'BAC', 'BCA', 'CAB', 'CBA'

조합: 서로 다른 n개에서 순서에 상관 없이 서로 다른 r개를 선택하는 것

- {'A', 'B', 'C'}에서 순서를 고려하지 않고 두 개를 뽑는 경우: 'AB', 'AC', 'BC'

순열의 수: $$nPr = n * (n-1) * (n-2) * \cdots * (n-r+1)$$

조합의 수: $$nCr = \frac{n * (n-1) * (n-2) * \cdots * (n-r+1)}{r!}$$

순열: 서로 다른 n개에서 서로 다른 r개를 선택하여 일렬로 나열하는 것

- {'A', 'B', 'C'}에서 두 개를 선택하여 나열하는 경우: 'ABC', 'ACB', 'BAC', 'BCA', 'CAB', 'CBA'

```
from itertools import permutations

data = ['A', 'B', 'C'] # 데이터 준비

result = list(permutations(data, 3)) # 모든 순열 구하기
print(result)
```

실행 결과: `[('A','B','C'), ('A','C','B'), ('B','A','C'), ('B','C','A'), ('C','A','B'), ('C','B','A')]`

- **조합**: 서로 다른 n개에서 순서에 상관 없이 서로 다른 r개를 선택하는 것
  - {'A', 'B', 'C'}에서 순서를 고려하지 않고 두 개를 뽑는 경우: 'AB', 'AC', 'BC'

```
from itertools import combinations

data = ['A', 'B', 'C'] # 데이터 준비

result = list(combinations(data, 2)) # 2개를 뽑는 모든 조합 구하기
print(result)
```

실행 결과: `[('A','B'), ('A','C'), ('B','C')]`

중복 순열과 중복 조합

```
from itertools import product

data = ['A', 'B', 'C'] # 데이터 준비

result = list(product(data, repeat=2)) # 2개를 뽑는 모든 순열 구하기 (중복 허용)
print(result)
```

```
from itertools import combinations_with_replacement

data = ['A', 'B', 'C'] # 데이터 준비

result = list(combinations_with_replacement(data, 2)) # 2개를 뽑는 모든 조합 구하기 (중복 허용)
print(result)
```

Counter

Counter

- 파이썬 collections 라이브러리의 **Counter**는 등장 횟수를 세는 기능을 제공합니다.
- 리스트와 같은 반복 가능한(iterable) 객체가 주어졌을 때 내부의 원소가 몇 번씩 등장했는지를 알려줍니다.

```
from collections import Counter

counter = Counter(['red', 'blue', 'red', 'green', 'blue', 'blue'])

print(counter['blue']) # 'blue'가 등장한 횟수 출력
print(counter['green']) # 'green'이 등장한 횟수 출력
print(dict(counter)) # 사전 자료형으로 반환
```

실행 결과:

```
3
1
{'red': 2, 'blue': 3, 'green': 1}
```

수학적

최대 공약수와 최소 공배수

- 최대 공약수를 구해야 할 때는 math 라이브러리의 gcd() 함수를 이용할 수 있습니다.

```
import math

# 최소 공배수(LCM)를 구하는 함수
def lcm(a, b):
    return a * b // math.gcd(a, b)

a = 21
b = 14

print(math.gcd(21, 14)) # 최대 공약수(GCD) 계산
print(lcm(21, 14)) # 최소 공배수(LCM) 계산
```

실행 결과:

```
7
42
```