🔷

기초 수학

소인수분해

약수 구하기

```
# 일반적인 방법
def get_divisor(n):
        divisor = []
        for i in range(1, n + 1):
                if n % i == 0:
                        divisor.append(i)
        return divisor

print(get_divisor(8)) # [1, 2, 4, 8]
```

```
# 범위를 좁혀서 구하기
import math

def get_divisor_square(n):
        divisor = set() # 제곱수의 경우 두 번 들어갈 수 있으므로 중복 제거!
        for i in range(1, int(math.sqrt(n) + 1))
                if n % 1 == 0:
                        divisor.update((i, n//i))
        return sorted(list(divisor))

print(get_divisor(8)) # [1, 2, 4, 8]
```

```
# list comprehension
def divisor(n):
        return [x for x in range(1, n + 1) if a % n == 0]
```

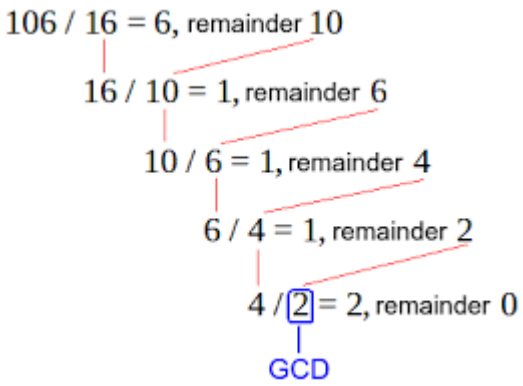
최대공약수 (Greatest Common Divisor)

```
for i in range(min(A,B), 0, -1):
    if A % i == 0 and B % i == 0:
        print(i)
        break
```

```
# 파이썬 라이브러리를 활용한 최대공약수 구하기
import math
math.gcd(A, B)
```

유클리드 호제법

- x와 y의 최대 공약수 == y와 r의 최대 공약수 (r = x%y)




```
# 파이썬으로 구현한 유클리드 호제법
def euclid_gcd(a, b): # a > b
    res = 0
    while b:
        res = a % b
        a = b
        b = res
    return a

print(uclid_gcd(106, 6)) # 2

# 재귀로 구현
def euclid_recursion(a, b): # a > b
        return a if b == 0 else euclid_recursion(b, a%b)

print(uclid_recursion(106, 6)) # 2
```

최소공배수 (Least Common Multiple)

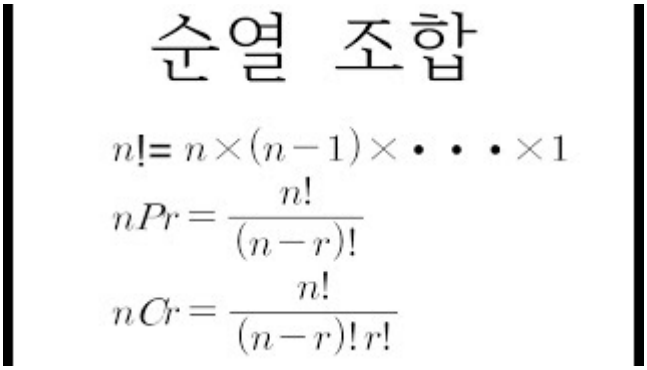
```
for i in range(max(A,B), (A*B) + 1):
    if i % A == 0 and i % B == 0:
        print(i)
        break
```

```
# 파이썬 라이브러리 활용

import math
math.lcm(A, B)
```

```
# 최대공약수 함수를 사용
def lcm(a, b):
    return (a * b) // euclid_gcd(a, b)
```

조합, 순열




```
# 파이썬 라이브러리 활용하기!

from itertools import permutations, combinations

num_list = [1, 2, 3, 4]

combi = list(combinations(num_list, 2))
print(len(combi)) # 6
print(combi) # [(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)]

perm =  list(permutations(num_list, 2))
print(len(perm)) # 12
print(perm) # [(1, 2), (1, 3), (1, 4), (2, 1), (2, 3), (2, 4), (3, 1), (3, 2), (3, 4),
(4, 1), (4, 2), (4, 3)]
```

소수

- 소수 판별하기

```
# 가장 기본적인 접근
def primenumber(x):
    for i in range(2, x): # 2부터 x-1까지의 모든 숫자
      if x % i == 0:    # 나눠떨어지는게 하나라도 있으면 False
          return False
    return True

print(primenumber(8191)) # True

# 범위를 좁혀서
import math
def primenumber_square(x):
    for i in range(2, int(math.sqrt(x) + 1)): # 2부터 x의 제곱근까지의 숫자
      if x % i == 0:    # 나눠떨어지는 숫자가 있으면 소수가 아님
          return False
    return True

print(primenumber_square(8191)) # True
```

- 소수 출력하기

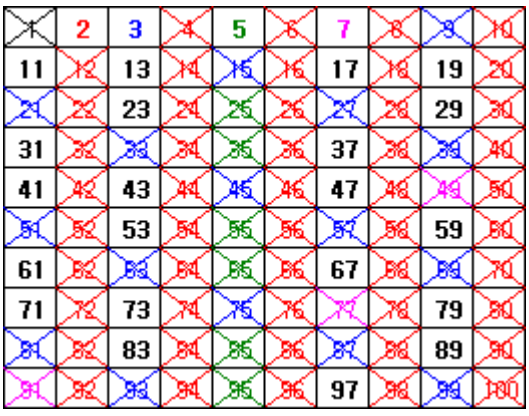
```
# m부터 n 사이의 소수 구하기 (백준 1929)

import math
m, n = map(int, input().split())

for i in range(m, n + 1):
    for j in range(2, int(math.sqrt(i)) + 1):
        if i % j == 0:
            break
    else:
        print(i)
```

에라토스테네스의 체

- 소수를 구하는 가장 기본적인 알고리즘



```
import math

def sieve(n):
        # 0, 1은 소수가 아니므로 False, 나머지는 True를 넣어줌
        prime = [0, 0] + ([1] * (n - 1))
        # 2부터 n의 제곱근까지 loop
        for i in range (2, int(math.sqrt(n) + 1)):
                # 해당 인덱스가 소수일 때(True)
                if prime[i]:
                        # 소수의 배수들을 모두 False로 변환
                        for j in range(i * 2, len(prime), i):
                                prime[j] = 0
        # enumerate를 통해 소수를 담은 리스트를 반환
        return [idx for idx, val in enumerate(prime) if val]

print(sieve(20)) [2, 3, 5, 7, 11, 13, 17, 19]
```

백준

9020 - 골드바흐 추측

2824 - 최대공약수

2981 - 검문

17087 - 숨바꼭질6

1182 - 부분수열의 합

4948 - 베르트랑 공준