2일차(파이썬 문법 - 조건문/반복문/함수/입출력/라이브러리)

💡 조건문

```
score = 85

if score >= 90:
  print("90")
elif score >= 80:
  print("80")
else:
  print("zz")
```

💡 반복문

```
# while
i = 1
result = 0

while i <= 9:
  result += i
  i++

result # 45



# for
result = 0

for i in range(1, 10):
  result += i

result # 45

for i in range(2, 10):
```

```
  for j in range(1, 10):
    print(f'{i} X {j} = {i*j}`)
```

💡 함수

```
# 일반적인 함수
def add(a, b):
  return a + b


# global 사용
a = 0

def func():
  global a
  a++

for i in range(10):
  func()

a # 10


# 람다식
print(lambda(a, b: a + b)(3, 8)) # 11
```

💡 입출력

```
n = int(input())
data = list(map(int, input().split()))
# input()으로 받은 모든 데이터를 split()으로 공백으로 나눈다
# map()을 이용해 해당 리스트의 모든 원소에 int() 적용
# 그 결과를 list()로 바꾸면서 문자열을 띄어쓰기로 구분하여
# 각각 숫자 자료형으로 저장


n, m, k = map(int, input().split())
```

```
# 입력을 최대한 빨리 받아야 하는 경우에는 readline 사용
# 예를 들어 1000만개를 한번에 받아야 되면...
import sys

data = sys.stdin.readline().rstrip()
print(data)

# readline으로 입력하면 입력 후 엔터가 줄 바꿈 기호로 입력되는데,
# 이 공백문자를 제거하기 위해 rstrip() 함수 사용
```

출력

- 기본적으로 콤마(,)를 이용해 출력하며, 타입이 다른 내용 출력 불가
- 콤마 사용시 띄어쓰기 적용된다.

```
a = 1
b = 1
add = a + b

print("a+b" + add + "입니다") # Error
print("a+b" + str(add) + "입니다") # a+b2입니다
print("a+b", str(add), "입니다") # a+b 2 입니다
```

- f-string 코드도 사용 가능(파이썬 3.6 이상에서)

```
a = 1
b = 1
add = a + b

print(f"a와 b를 더한값은 {add}입니다.")
# js의 백틱과 같은 기능
# 다른 타입도 써넣을 수 있음
```

💡 내장함수

- import 없이 바로 사용 가능하다.

eval()

- 해당 수식을 계산한 결과 반환

```
result = eval('(3+5) * 7') # 56
```

sorted()

- iterable 객체에 대해 정렬 반환
  - iterable 객체 : List, Tuple, Dictionary

```
d_list = [3, 1, 2, 5]
d_tuple = (3, 1, 2, 5)
d_dict = [('a', 26), ('b', 23), ('c', 11), ('d', 13)]

print(sorted(d_list)) # [1, 2, 3, 5]
print(sorted(d_list, reserve=True)) # [5, 3, 2, 1]
print(sorted(d_tuple)) # [1, 2, 3, 5]
print(sorted(d_tuple, reserve=True)) # [5, 3, 2, 1]
print(sorted(d_dict))
# [('a', 26), ('b', 23), ('c', 13), ('d', 11)]
print(sorted(d_dict, lambda a: a[1]))
# [('a', 26), ('b', 23), ('d', 13), ('c', 11)]

# sort(only list)
d_list = [3, 1, 2, 5]
d_list.sort()
print(d_list) # [1, 2, 3, 5]
```

💡 itertools

- permutation() : list iterable 객체에서 r개의 데이터를 뽑아 일렬로 나열하는 모든 경우(순열)
- combination() : list iterable 객체에서 r개의 데이터를 뽑아 순서를 고려하지 않고 나열하는 모든 경우(조합)

```
from itertools import permutations
from itertools import product
from itertools import combinations_with_replacement

data = ['a', 'b', 'c']

# 순열
d_permutation = list(permutations(data, 2))
# [('a', 'b'), ('b', 'a')]

# 순열(중복 허용)
d_product = list(product(data, repeat = 2))
[('a', 'a'), ('a', 'b'), ('b', 'a'), ('b', 'b')]

# 조합(중복 허용)
d_combination = list(combinations_with_replacement(data, 2))
# [('a', 'a'), ('a', 'b'), ('b', 'b')]
```

💡 heapq

- 최단 경로 알고리즘을 포함해 다양한 알고리즘에서 우선순위 큐 기능을 구현하는 함수

```
import heapq

def heapsort(iterable):
  h = []
  result = []

  # heap에 데이터 다 넣고
```

```
  for value in iterable:
    heapq.heappush(h, value)

  # 끝에서부터 빼서 result에 저장하면 오름차순 정렬됨
  for i in range(leng(h)):
    result.append(heapq.heappop(h))

  return result

result = heapsort([1, 3, 5, 7, 9, 2, 4, 6, 8, 0])
print(result) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

bisect

- 정렬된 배열에서 특정한 원소를 주로 찾을 때 사용하고 인덱스를 반환한다.
- bisect_left : 정렬된 순서를 유지하면서 리스트 a에 데이터 x를 삽입할 가장 왼쪽 인덱스 반환
- bisect_right : 정렬된 순서를 유지하면서 리스트 a에 데이터 x를 삽입할 가장 오른쪽 인덱스 반환

- [1, 2, 4, 4, 8]이 있을 때, 4를 삽입한다고 가정하면
- bisect_left(a, 4) = 2, bisect_right(a, 4) = 4

```
from bisect import bisect_left, bisect_right

# 특정한 원소의 배열 위치
a = [1, 2, 4, 4, 8]

print(bisect_left(a, 2)) # 2
print(bisect_right(a, 4)) # 4

# 값이 특정 범위에 속하는 원소의 개수
def count_by_range[a, lVal, rVal]:
    rIdx = bisect_right(a, rVal)
    lIdx = bisect_left(a, lVal)

    return rIdx - lIdx

a = [ 1, 2, 3, 3, 3, 3, 4, 4, 8, 9]
print(count_by_range(a, 4, 4)) # 2
print(count_by_range(a, 3, 3)) # 4
```

```
# 1 ~ 9 범위의 데이터 개수
print(count_by_range(a, 1, 9)) # 10
```

💡 Collections

- deque를 이용해 큐(FIFO)를 구현해야 한다.

```
from collection import deque

data = deque([2, 3, 4])

data.appendleft(1)
data.append(5)

print(data)
# deque([1, 2, 3, 4, 5])

print(list(data))
# [1, 2, 3, 4, 5]
```

💡 Counter

- 등장 횟수를 센다
- list iterable 객체에만 가능

```
from collections import Counter
data_counter = ['red', 'blue', 'blue', 'orange', 'green', 'blue', 'green']

print(Counter(data_counter))
# Counter({'blue': 3, 'green': 2, 'red': 1, 'orange': 1})

print(Counter(dict(data_counter)))
# {'red': 1, 'blue': 3, 'orange': 1, 'green': 2}
```

```
print(Counter(data_counter, 'blue'))
# 3
```

Math

- 수학 계산

```
import math

# factorial(x)
print(math.factorial(5)) # 120

# sqrt(x): x의 제곱근
print(int(math.sqrt(49))) # 7

# gcd(x): 최대 공약수
print(math.gcd(21, 14)) # 7

# 파이(pi) 출력
print(math.pi) # 3.141592653589793

# 자연상수(e) 출력
print(math.e) # 2.718281828459045
```

- 백준 입출력 + 조건문 17문제