**@@@@@@@@@@@@여러 개의 변수 입력받기@@@@@@@@@@@@**

**#숫자일경우**

N, M, K = input().split(" ")

N = int(N)

M = int(M)

K = int(K)

**map사용하기**

map은 여러 개의 데이터를 한 번에 다른 형태로 변환하기 위해서 사용된다.

따라서 여러 개의 데이터를 담고 있는 list나 tuple을 대상으로 주로 사용한다.

a, b = map(int, input('숫자 두 개를 입력하세요: ').split())

map에 int와 input().split()을 넣으면 split의 결과를 모두 int로 변환해준다.

-여러 개의 입력값을 받아서 리스트로 만들기

s = [ x for x in map(int, input().split() ) ]

s = list( map( int, input().split() ) )

-0 인덱스의 특정값 추가한 상태로 입력값을 리스트로 만들기

p = [0] + list( map( int, input().split() ) )

-띄어쓰기 없는 입력값을 2차원 리스트로 만들기

arr = []

for i in range(0, 5):

arr.append( list( map( int, input() ) ) )

**-map()함수에서 lambda함수 사용하기**

x = [10, 20, 30]

xList = list( map( (lambda i: i+10), x ) )

print( xList ) # [20, 30, 40]

**-readline()으로 입력받기**import sys  
for i in range( int(input()) ):  
 a = sys.stdin.readline.rstrip() #뒤에 rstrip()은 개행문자를 제거해준다.

**-숫자 하나만 입력받을 때**n = int( \_\_import\_\_('sys').stdin.readline() )

**-문장 전체를 입력받기**

nList = sys.stdin.read().splitlines()

**@@@@@@@@@@@@ 프로그램 종료 @@@@@@@@@@@@**

quit()  
exit()  
sys.exit()

**@@@@@@@@@@@@ 수학 관련 @@@@@@@@@@@@**

**-나눗셈 소수점까지 구하기**

a = 10 / 3

print( a )

>> 3.3333

**-나눗셈 몫만 구하기**

a = 10 // 3

print( a )

>> 3

**-몫과 나머지를 튜플형태로 저장하기**

a = divmod( 10 , 3 )

print( a )

>> (3, 1) 반환

**-제곱구하기**  
print( 2 \*\* 10 ) # 2의 10제곱  
print( pow( 2, 10 ) ) #2의 10제곱

**-소수점 n번째 자리에서 반올림**



**@@@@@@@@@@@@ 문자 및 문자열 관련 @@@@@@@@@@@@**

**-문자열 거꾸로 뒤집기**  
v = “i am happy today”

print( v[ len(v)::-1 ], end=” “ )

>>i ma yppah yadot

**-문자를 아스키코드로 / 아스키코드를 문자로**

ord( 문자 ) : 문자의 아스키코드값 반환

chr( 숫자 ) : 아스키코드를 문자로 반환

**-숫자를 문자열로 변환**

a = 10  
str( a )

**-문자열에 특정 문자 포함 여부 확인**

s = “december Holiday Season”  
print( s.find( “Holiday” ) ) #9 특정문자열이 시작하는 인덱스값이 출력된다.

만일 특정 문자열이 포함되지 않을 경우 -1이 출력된다.

**-여러 개의 줄로 이루어진 문자열을 줄 별로 구분**

>>> y = 'haha

hoho

hihi'

>>> y

'haha, \nhoho, \nhihi'

>>> y.splitlines() # ['haha, ', 'hoho, ', 'hihi']

['haha, ', 'hoho, ', 'hihi']

이때 y.splitlines( True )를 해주면 개행문자가 제거되지 않는다.  
>>> y.splitlines( True ) # ['haha\n, ', 'hoho\n, ', 'hihi\n']

**@@@@@@@@@@@@반복문 관련@@@@@@@@@@@@**

**-리스트 내장( 한줄에 반복문과 조건문 같이 사용 )**

l = ["apple", "banana", "orange", "kiwi"]

print( [i for i in l if len(i) > 5] )  
# ['banana', 'orange']

**-filter() 함수 사용**

사용법: filter( 함수명 또는 None, 순회가능한 자료형 )

def GetBiggerThan20( i ):

return i > 20

l = [10, 25, 30]

iter = filter( GetBiggerThan20, l )

for i in iter:

print( i )

#25

30

람다함수로 간단하게 작성가능  
l = [10, 25, 30]

iter = filter( lambda i: i>20, l )

for i in iter:

print( i )

**@@@@@@@@@@@@리스트 관련@@@@@@@@@@@@**

**-리스트 선언하기(타언어의 배열과 같음)**

ll = []

ll[0] = 0

ll[1] = 1

**-리스트 요소 추가 삭제**

a = [1, 2, 3, 4, 5]

a.append(6)

print( a ) #1, 2, 3, 4, 5, 6

a.insert( 인덱스값, 값 )

a.insert( 1, 10 )

print( a ) #1, 10, 2, 3, 4, 5

del a[ 인덱스값 ]

a = [1, 2, 3, 4, 5, 6, 7]

del a[1]

print( a ) #[1, 3, 4, 5, 6, 7]

list.remove( 특정값 )

a = [1, 2, 3, 4, 5, 6, 7]

a.remove(3)

a

[1, 2, 4, 5, 6, 7]

a = list[-1]

print( a ) #7

**-미리 크기를 정하고 0으로 초기화하기**

1차원 리스트의 경우: list = [ 0 for i in range(n) ]

2차원 리스트의 경우: list = [ [0] \* (n+1) for i in range(n+1) ]

**-두 개 리스트 합치기**

m = [2, 5, 7]

n = [3, 5, 9]

k = m + n

k

[2, 5, 7, 3, 5, 9]

k +=[11, 13]

k

[2, 5, 7, 3, 5, 9, 11, 13]

a = [1,2,3]

a.extend([4,5,6])

a

[1, 2, 3, 4, 5, 6]

a.append( [4, 5, 6] )

a

[1, 2, 3, [4, 5, 6] ]

**-리스트 거꾸로하기**

list = reversed( list )

**-리스트에서 최대 최소값 구하기 및 최대최소의 인덱스값 구하기**

nList = [3, 5, 1, 2, 7, 8, 5]

max(nList) #8

min(nList) #1

nList.index( max(nList) ) #5

nList.index( min(nList) ) #2

**-리스트 정렬하기(오름차순, 내림차순)**  
a = [1, 10, 5, 7, 6]

a.sort() #오름차순

a.sort( reverse=True ) #내림차순

**lambda식 사용**

lambda 매개변수 : 리턴값

lambda x : x+1 #매개변수에 1을 더해서 리턴해준다.

**첫 번째 인자 오름차순정렬하고 같을경우 두 번째 인자 오름차순 정렬**

e = [(1, 3), (0, 3), (1, 4), (1, 5), (0, 1), (2, 4)]

f = sorted(e, key = lambda x : (x[1], x[0]))

# f = [(0, 3), (0, 1), (1, 5), (1, 4), (1, 3), (2, 4)]

**길이순으로 정렬한 뒤 사전순으로 정렬하기**

e.sort( key= lambda x: ( len(x[0]), x[1] )

**단어수 정렬하고 같은 단어수면 사전순으로 정렬하기**

a.sort() #사전순 정렬  
a.sort( key=len ) #단어수 정렬

**-리스트의 값들 개수 구하기**

my\_list = ['Tick', 'Tock', 'Tock']  
from collections import Counter  
counter = Counter( my\_list )  
print( counter ) # ‘Tock’:2, ‘Tick’:1

**-리스트에서 가장 큰 값 찾기**

print( max(arr) )

**-리스트 요소들의 합 구하기**

import numpy as np

a = np.array( [1,2,3,4,5] )

np.sum( a ) #15

a.sum() #15

-array[ :: ] 사용하기

arr[ A:B:C ] 는 index A부터 index B까지 C의 간격으로 배열을 만들어라

A가 None이라면 처음부터라는 뜻이고

B가 None이라면 끝까지 이때 C가 양수라면 왼쪽에서 오른쪽으로 C가 음수라면 오른쪽에서 왼쪽으로

C가 None이면 그냥 1씩 증가

>> arr = range(10)

>> arr

[0,1,2,3,4,5,6,7,8,9]

>> arr[::2] # 처음부터 끝까지 두 칸 간격으로

[0,2,4,6,8]

>> arr[1::2] # index 1 부터 끝까지 두 칸 간격으로

[1,3,5,7,9]

>> arr[::-1] # 처음부터 끝까지 -1칸 간격으로 ( == 역순으로)

[9,8,7,6,5,4,3,2,1,0]

>> arr[::-2] # 처음부터 끝까지 -2칸 간격으로 ( == 역순, 두 칸 간격으로)

[9,7,5,3,1]

>> arr[3::-1] # index 3 부터 끝까지 -1칸 간격으로 ( == 역순으로)

[3,2,1,0]

>> arr[1:6:2] # index 1 부터 index 6 까지 두 칸 간격으로

[1,3,5]

**-zip() 함수 사용하기**

zip()함수는 순회가능한 이터레이터형 객체들을 결합해준다.

x = [10, 20, 30]

y = "abc"

z = (1.5, 2.5, 3.5)

zipList = list( zip(x, y, z) )

print( zipList )

# [(10, 'a', 1.5), (20, 'b', 2.5), (30, 'c', 3.5)]

zip()함수로 결합된 결과를 원래대로 분리할 수 있다.

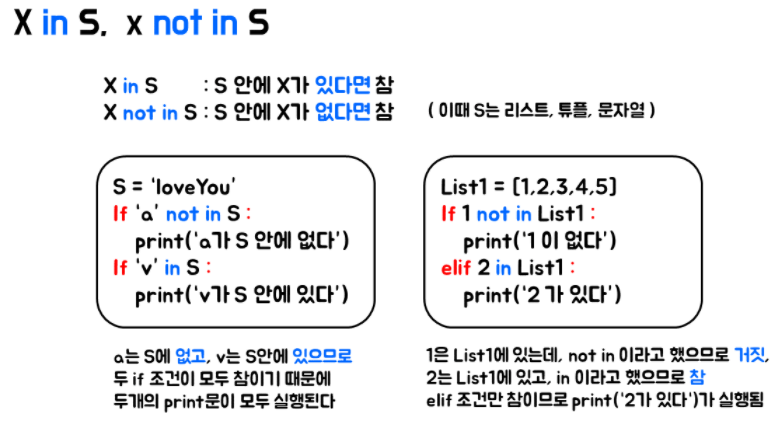
a, b, c = zip( \*zipList )

print( a, b, c )

# (10, 20, 30) ('a', 'b', 'c') (1.5, 2.5, 3.5)

튜플형태로 반환된다.

**@@@@@@@@@@@@조건문 관련@@@@@@@@@@@@**



**@@@@@@@@@@@@반복문@@@@@@@@@@@@**

**-for문**

for i in range(0, 10): #0부터 9까지 반복

print( i )

for i in range(10, 0, -3):

print( i )

#10, 7, 4, 1

**-for문 한 줄로 사용**

v = list( range(10 )

[ i for i in v ]

#[ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ]

print( “ “.join( str( i ) for i in v ) )

#0 1 2 3 4 5 6 7 8 9

**-인덱스와 요소를 함께 출력하기**

a = [ 38, 21, 53, 62, 19 ]

for index, value in enumerate( a ):

print( index, value )

#결과

0 38

1 21

2 53

3 62

4 19

**-while**

while True: #무한루프

num = 1

while num <= 100:

print( num )

num = num + 1

**@@@@@@@@@@@@정규표현식@@@@@@@@@@@@**

**-자주쓰이는 정규식**

[a-zA-Z] : 알파벳 모두

[0-9] : 숫자

\d - 숫자와 매치, [0-9]와 동일한 표현식이다.

\D - 숫자가 아닌 것과 매치, [^0-9]와 동일한 표현식이다.

\s - whitespace 문자와 매치, [ \t\n\r\f\v]와 동일한 표현식이다. 맨 앞의 빈 칸은 공백문자(space)를 의미한다.

\S - whitespace 문자가 아닌 것과 매치, [^ \t\n\r\f\v]와 동일한 표현식이다.

\w - 문자+숫자(alphanumeric)와 매치, [a-zA-Z0-9\_]와 동일한 표현식이다.

\W - 문자+숫자(alphanumeric)가 아닌 문자와 매치, [^a-zA-Z0-9\_]와 동일한 표현식이다.

**-Dot(.)**

정규 표현식의 Dot(.) 메타 문자는 줄바꿈 문자인 \n을 제외한 모든 문자와 매치됨을 의미한다.

정규식을 작성할 때 re.DOTALL 옵션을 주면 \n 문자와도 매치된다.

정규식 a[.]b는 "a.b" 문자열과 매치되고, "a0b" 문자열과는 매치되지 않는다.

※ 만약 앞에서 살펴본 문자 클래스([]) 내에 Dot(.) 메타 문자가 사용된다면

이것은 "모든 문자"라는 의미가 아닌 문자 . 그대로를 의미한다. 혼동하지 않도록 주의하자.

**-반복 (\*)**

ca\*t

이 정규식에는 반복을 의미하는 \* 메타 문자가 사용되었다.

여기에서 사용한 \*은 \* 바로 앞에 있는 문자 a가 0부터 무한대로 반복될 수 있다는 의미이다.

**정규식 문자열 Match 여부 설명**

ca\*t ct Yes "a"가 0번 반복되어 매치

ca\*t cat Yes "a"가 0번 이상 반복되어 매치 (1번 반복)

ca\*t caaat Yes "a"가 0번 이상 반복되어 매치 (3번 반복)

**-반복 (+)**

반복을 나타내는 또 다른 메타 문자로 +가 있다. +는 최소 1번 이상 반복될 때 사용한다.

즉 \*가 반복 횟수 0부터라면 +는 반복 횟수 1부터인 것이다.

ca+t

위 정규식의 의미는 다음과 같다.

"c + a(1번 이상 반복) + t"

위 정규식에 대한 매치여부는 다음 표와 같다.

정규식 문자열 Match 여부 설명

ca+t ct No "a"가 0번 반복되어 매치되지 않음

ca+t cat Yes "a"가 1번 이상 반복되어 매치 (1번 반복)

ca+t caaat Yes "a"가 1번 이상 반복되어 매치 (3번 반복)

**-반복 ({m,n}, ?)**

여기에서 잠깐 생각해 볼 게 있다.

반복 횟수를 3회만 또는 1회부터 3회까지만으로 제한하고 싶을 수도 있지 않을까?

{ } 메타 문자를 사용하면 반복 횟수를 고정할 수 있다.

{m, n} 정규식을 사용하면 반복 횟수가 m부터 n까지 매치할 수 있다.

또한 m 또는 n을 생략할 수도 있다.

만약 {3,}처럼 사용하면 반복 횟수가 3 이상인 경우이고

{,3}처럼 사용하면 반복 횟수가 3 이하를 의미한다.

생략된 m은 0과 동일하며, 생략된 n은 무한대(2억 개 미만)의 의미를 갖는다.

※ {1,}은 +와 동일하고, {0,}은 \*와 동일하다.

{ }을 사용한 몇 가지 정규식을 살펴보자.

1. {m}

ca{2}t

위 정규식의 의미는 다음과 같다.

"c + a(반드시 2번 반복) + t"

위 정규식에 대한 매치여부는 다음 표와 같다.

정규식 문자열 Match 여부 설명

ca{2}t cat No "a"가 1번만 반복되어 매치되지 않음

ca{2}t caat Yes "a"가 2번 반복되어 매치

2. {m, n}

ca{2,5}t

위 정규식의 의미는 다음과 같다:

"c + a(2~5회 반복) + t"

위 정규식에 대한 매치여부는 다음 표와 같다.

정규식 문자열 Match 여부 설명

ca{2,5}t cat No "a"가 1번만 반복되어 매치되지 않음

ca{2,5}t caat Yes "a"가 2번 반복되어 매치

ca{2,5}t caaaaat Yes "a"가 5번 반복되어 매치

3. ?

반복은 아니지만 이와 비슷한 개념으로 ? 이 있다.

? 메타문자가 의미하는 것은 {0, 1} 이다.

ab?c

위 정규식의 의미는 다음과 같다:

"a + b(있어도 되고 없어도 된다) + c"

위 정규식에 대한 매치여부는 다음 표와 같다.

정규식 문자열 Match 여부 설명

ab?c abc Yes "b"가 1번 사용되어 매치

ab?c ac Yes "b"가 0번 사용되어 매치

**@@@@@@@@@@@@Counter클래스@@@@@@@@@@@@**

from collections import Counter

Counter('hello world') # Counter({'l': 3, 'o': 2, 'h': 1, 'e': 1, ' ': 1, 'w': 1, 'r': 1, 'd': 1})

**@@@@@@@@@@@@ deque @@@@@@@@@@@@**

앞뒤 양쪽 방향에서 요소를 추가하거나 제거할 수 있음. deque는 스택처럼 사용할 수 있고 큐처럼 사용할 수 있다.

from collections import deque

que = deque()

que.appendleft( 10 ) #요소를 왼쪽에 추가

que.append( 0 ) #요소를 오른쪽끝에 추가

que.pop() #오른쪽 끝 요소를 가져오는 동시에 삭제

que.popleft() #왼쪽 끝 요소를 가져오는 동시에 삭제

que.extend( array ) #주어진 배열을 순회하면서 배열값을 오른쪽에 추가

que.extendleft( array )

que.remove( item ) #특정 item을 찾아서 삭제

que.rotate( 1 ) #큐를 시계방향으로 180도 회전해서 오른쪽값이 왼쪽끝으로 가게한다. 좌우반전

que.rotate( -1 ) #큐를 반시계방향으로 180도 회전해서 오른쪽값이 왼쪽끝으로 가게한다. 좌우반전

while que: #큐 안에 요소있으면 반복, 없으면 반복문 종료

**@@@@@@@@@@@@ 리스트 조합구하기 @@@@@@@@@@@@**

리스트에 있는 값들의 모든 조합을 구하기

from itertools import product

from itertools import permutations

from itertools import combinations

**-하나의 리스트에서 모든 조합 구하기**

items = ['1', '2', '3', '4', '5']

from itertools import permutations

list(permutations(items, 2))

# [('1', '2'), ('1', '3'), ('1', '4'), ('1', '5'), ('2', '1'), ('2', '3'), ('2', '4'), ('2', '5'), ('3', '1'), ('3', '2'), ('3', '4'), ('3', '5'), ('4', '1'), ('4', '2'), ('4', '3'), ('4', '5'), ('5', '1'), ('5', '2'), ('5', '3'), ('5', '4')]

from itertools import combinations

list(combinations(items, 2))

# [('1', '2'), ('1', '3'), ('1', '4'), ('1', '5'), ('2', '3'), ('2', '4'), ('2', '5'), ('3', '4'), ('3', '5'), ('4', '5')]

**-두 개 이상의 리스트의 모든 조합 구하기**

from itertools import product

items = [['a', 'b', 'c,'], ['1', '2', '3', '4'], ['!', '@', '#']]

list(product(\*items))

# [('a', '1', '!'), ('a', '1', '@'), ('a', '1', '#'), ('a', '2', '!'), ('a', '2', '@'), ('a', '2', '#'), ('a', '3', '!'), ('a', '3', '@'), ('a', '3', '#'), ('a', '4', '!'), ('a', '4', '@'), ('a', '4', '#'), ('b', '1', '!'), ('b', '1', '@'), ('b', '1', '#'), ('b', '2', '!'), ('b', '2', '@'), ('b', '2', '#'), ('b', '3', '!'), ('b', '3', '@'), ('b', '3', '#'), ('b', '4', '!'), ('b', '4', '@'), ('b', '4', '#'), ('c,', '1', '!'), ('c,', '1', '@'), ('c,', '1', '#'), ('c,', '2', '!'), ('c,', '2', '@'), ('c,', '2', '#'), ('c,', '3', '!'), ('c,', '3', '@'), ('c,', '3', '#'), ('c,', '4', '!'), ('c,', '4', '@'), ('c,', '4', '#')]

**@@@@@@@@@@@@ set 자료형 @@@@@@@@@@@@**

set은 집합과 같다.

**-선언하기**

s = set()

s = set( [ 1, 3, 5, 7 ] )

중복된 값은 중복이 제거 된다.

집합자료형은 순서가 없어서 print해보면 아무 원소가 랜덤으로 출력된다.

**-순서대로 나오게 하기**

for i in s:

print( i ) # 1 3 5 7

**-원소 추가**

s.add( 50 )

**-여러 데이터 한 번에 추가( 중복은 저절로 제거된다. )**

s.update( [3, 4, 5] )

s # 1, 3, 4, 5, 7

-원소 제거

s.remove( 3 )

s #1, 4, 5, 7

**@@@@@@@@@@@@ 우선순위큐 @@@@@@@@@@@@**

#최소힙  
import heapq  
heap = []  
heapq.heappush( heap, 4 )  
heapq.heappush( heap, 1 )  
heapq.heappush( heap, 7 )  
heapq.heappush( heap, 3 )  
print( heap ) #1 3 7 4  
print( heapq ) # 1 3 4 7  
  
#최대힙  
import heapq

nums = [4, 1, 7, 3, 8, 5]

heap = []

for num in nums:

heapq.heappush(heap, (-num, num)) # (우선 순위, 값)

while heap:

print(heapq.heappop(heap)[1]) # 8 7 5 4 3 1  
  
  
#기존 리스트를 힙으로 변환  
heap = [4, 1, 7, 3, 8, 5]

heapq.heapify(heap)

print(heap) # [1, 3, 5, 4, 8, 7]

**@@@@@@@@@@@@ accumulate 사용 @@@@@@@@@@@@**

from itertools import accumulate

a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

b = list(accumulate(a))

print(a) # [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(b) # [1, 3, 6, 10, 15, 21, 28, 36, 45, 55]

**@@@@@@@@@@@@ enumerate() 사용 @@@@@@@@@@@@**

enumerate()는 순회가능한 객체에서 인덱스 값과 요소의 값을 둘 다 반환해준다.  
for i, season in enumerate( ['Spring', 'Summer', 'Fall', 'Winter'] ):

print( i, season )  
>>  
0 Spring

1 Summer

2 Fall

3 Winter

**@@@@@@@@@@@@ 리스트내장 사용 @@@@@@@@@@@@**