1. N개의 최소공배수

def solution(arr):

    arrmax = 1

    allpass = 0

    answer = 0

    for maxnum in arr:

        arrmax \*= maxnum

    for i in range(max(arr), arrmax+1):

        for j in range(0,len(arr)):

            if i % arr[j] == 0:

                allpass += 1

        if allpass == len(arr):

            answer = i

            break

        allpass = 0

    return answer

arr = [1,2,3]

print(solution(arr))

<https://programmers.co.kr/learn/courses/30/lessons/12953>

1. 튜플

def solution(s):

    answer = []

    result = []

    result1 = []

    num = ""

    for i in range(len(s)-1):

        if s[i].isdigit() and s[i+1].isdigit():

            num += s[i]

        elif s[i].isdigit() and not s[i+1].isdigit():

            num += s[i]

            result1.append(num)

            num = ""

        if s[i] == "}":

            result.append(result1)

            result1 = []

    result.sort(key=lambda x:len(x))

    for i in range(len(result)):

        for j in range(len(result[i])):

            if int(result[i][j]) not in answer:

                answer.append(int(result[i][j]))

    return answer

s = "{{4,2,3},{3},{2,3,4,1},{2,3}}"

print(solution(s))

<https://programmers.co.kr/learn/courses/30/lessons/64065>

1. 행렬의 곱셈

def solution(arr1, arr2):

    x = len(arr2[0])

    y = len(arr1)

    answer = []

    for i in range(y):

        x\_list = []

        for j in range(x):

            number = 0

            for k in range(len(arr1[0])):

                nf1 = arr1[i][k]

                nf2 = arr2[k][j]

                number += nf1 \* nf2

            x\_list.append(number)

        answer.append(x\_list)

    return answer

arr1 = [[1, 4], [3, 2], [4, 1]]

arr2 = [[3, 3], [3, 3]]

print(solution(arr1,arr2))

<https://programmers.co.kr/learn/courses/30/lessons/12949>

1. H-Index

def solution(citations):

    citations.sort()

    print(citations)

    for i in range(0,len(citations)):

        if citations[i] >= len(citations)-i:

            return len(citations)-i

    return 0

citations = [3, 0, 6, 1, 5]

print(solution(citations))

H-index가 무엇인지 이해하면 쉽다 문제는 그게 이해가 잘 안되니 검색이 중요하다

<https://programmers.co.kr/learn/courses/30/lessons/42747>

1. 가장 큰 수

def solution(numbers):

    #0. key point

    numbers\_str = [str(num) for num in numbers]                 #1. 사전 값으로 정렬하기

    numbers\_str.sort(key=lambda num: num\*3, reverse=True)       #2. number는 1000이하의 숫자이므로 x3(반복)한 값으로 비교

    return str(int(''.join(numbers\_str)))

    # 만약 numbers=[0,0,0,0] 이라면 0 이 나와야 한다.

    # join한 값을 int로 만들어 준 후 원하는 return값이 str이기 때문에 다시 str로 변환한다.

print(solution([6, 10, 2]))             # result : 6210

print(solution([3, 30, 34, 5, 9]))      # result : 9534330

print(solution([0,0,0,0]))

sorted와 람다를 이용해 정렬한다는 것에는 잘 접근 했지만 num\*3을 찾지 못했다.

<https://programmers.co.kr/learn/courses/30/lessons/42746>

1. 기능개발

def solution(progresses, speeds):

    answer = []

    time = 0

    count = 0

    while len(progresses)> 0:

        if (progresses[0] + time\*speeds[0]) >= 100:

            progresses.pop(0)

            speeds.pop(0)

            count += 1

        else:

            if count > 0:

                answer.append(count)

                count = 0

            time += 1

    answer.append(count)

    return answer

progresses = [93, 30, 55]

speeds = [1, 30, 5]

print(solution(progresses,speeds))

<https://programmers.co.kr/learn/courses/30/lessons/42586>

1. 다음 큰 숫자

def numberOneSearch(binNum):

    oneCount = 0

    for i in range(2,len(binNum)):

        if binNum[i] == "1":

            oneCount += 1

    return oneCount

def solution(n):

    answer = 0

    binN = numberOneSearch(str(bin(n)))

    while answer != binN:

        n += 1

        answer = numberOneSearch(str(bin(n)))

    return n

n = 15

print(solution(n))

https://programmers.co.kr/learn/courses/30/lessons/12911