22일 (화)

1. 프로그래머스 1단계 문제 풀이

1) 문자열 다루기 기본 (못 푼 문제)

문자열 다루기 기본

문제 설명

문자열 s의 길이가 4 혹은 6이고, 숫자로만 구성돼있는지 확인해주는 함수, solution을 완성하세요. 예를 들어 s가 "a234"이면 False를 리턴하고 "1234"라면 True를 리턴하면 됩니다.

제한 사항

- `s` 는 길이 1 이상, 길이 8 이하인 문자열입니다.

입출력 예

s	return
"a234"	false
"1234"	true

```
def alpha_string46(s):
    return s.isdigit() and len(s) in (4, 6)
```

```
def solution(s):
# 문자열이 4 또는 6 이면서 문자열 s이 모두 숫자로 이루어져 있는지 확인
    if (len(s) == 4 or len(s) == 6) and s.isdigit():
```

```
        return True  # 조건에 맞으면 True 출력
    else:
        return False # 조건에 맞지않으면 False 출력
```

isdigit( ) 메소드 - 문자열 안이 숫자로 되어 있는지 확인

```
num = "13431"


print(num.isdigit())


==> True


string = "sfad2312"

print(string.isdigit())

==> False
```

2) 문자열 내 p와 y의 개수

문자열 내 p와 y의 개수

문제 설명

대문자와 소문자가 섞여있는 문자열 s가 주어집니다. s에 'p'의 개수와 'y'의 개수를 비교해 같으면 True, 다르면 False를 return 하는 solution를 완성하세요. 'p', 'y' 모두 하나도 없는 경우는 항상 True를 리턴합니다. 단, 개수를 비교할 때 대문자와 소문자는 구별하지 않습니다.

예를 들어 s가 "pPoooyY"면 true를 return하고 "Pyy"라면 false를 return합니다.

제한사항

- 문자열 s의 길이 : 50 이하의 자연수
- 문자열 s는 알파벳으로만 이루어져 있습니다.

입출력 예

s	answer
"pPoooyY"	true
"Pyy"	false

입출력 예 설명

입출력 예 #1
'p'의 개수 2개, 'y'의 개수 2개로 같으므로 true를 return 합니다.

입출력 예 #2
'p'의 개수 1개, 'y'의 개수 2개로 다르므로 false를 return 합니다.

```
def solution(s):
    answer = True
    s = s.lower()

    p_count = s.count('p')   #TIL 리스트 내 문자열 개수 count() 함수 사용
    y_count = s.count('y')

    if p_count != y_count:
```

```
        answer = False


    return answer
```

```
def numPY(s):

    return s.lower().count('p') == s.lower().count('y')

# 아래는 테스트로 출력해 보기 위한 코드입니다.
print( numPY("pPoooyY") )
print( numPY("Pyy") )
```

3) 문자열 내림차순으로 배치하기

문자열 내림차순으로 배치하기

문제 설명

문자열 s에 나타나는 문자를 큰것부터 작은 순으로 정렬해 새로운 문자열을 리턴하는 함수, solution을 완성해주세요.
s는 영문 대소문자로만 구성되어 있으며, 대문자는 소문자보다 작은 것으로 간주합니다.

제한 사항

- str은 길이 1 이상인 문자열입니다.

입출력 예

s	return
"Zbcdefg"	"gfedcbZ"

```
def solution(s):
    answer = ''
    s_l = sorted(s, reverse=True)

    for i in s_l:
        answer += i

    return answer
```

```
def solution(s):
    return ''.join(sorted(s, reverse=True))
```

4) 서울에서 김서방 찾기

서울에서 김서방 찾기

문제 설명

String형 배열 seoul의 element중 "Kim"의 위치 x를 찾아, "김서방은 x에 있다"는 String을 반환하는 함수, solution을 완성하세요. seoul에 "Kim"은 오직 한 번만 나타나며 잘못된 값이 입력되는 경우는 없습니다.

제한 사항

- seoul은 길이 1 이상, 1000 이하인 배열입니다.
- seoul의 원소는 길이 1 이상, 20 이하인 문자열입니다.
- "Kim"은 반드시 seoul 안에 포함되어 있습니다.

입출력 예

seoul	return
["Jane", "Kim"]	"김서방은 1에 있다"

```
def solution(seoul):

    count = 0

    for i in seoul:
        if i == "Kim":
```

```
            break
        count += 1

    answer = '김서방은 {0}에 있다'.format(count)

    return answer
```

```
def findKim(seoul):
    return "김서방은 {}에 있다".format(seoul.index('Kim'))
```

5) 소수 찾기 (못 푼 문제)

소수 찾기

문제 설명

1부터 입력받은 숫자 n 사이에 있는 소수의 개수를 반환하는 함수, solution을 만들어 보세요.

소수는 1과 자기 자신으로만 나누어지는 수를 의미합니다.
(1은 소수가 아닙니다.)

제한 조건

- n은 2이상 1000000이하의 자연수입니다.

입출력 예

n	result
10	4
5	3

입출력 예 설명

입출력 예 #1
1부터 10 사이의 소수는 [2,3,5,7] 4개가 존재하므로 4를 반환

입출력 예 #2
1부터 5 사이의 소수는 [2,3,5] 3개가 존재하므로 3를 반환

```
def solution(n):
    #에라토스테네스의 체 구현

    # 소수여부 리스트 초기화
    a = [False, False] + [True] * (n-1)
```

```
    #소수 리스트 생성
    prime = []

    #소수 찾기
    for i in range(2, n+1):
        if a[i]:
            prime.append(i)

            for j in range(i+i, n+1, i):
                a[j] = False
    return len(prime)
```

```
def solution(n):
    num=set(range(2,n+1))

    for i in range(2,n+1):
        if i in num:
            num-=set(range(2*i,n+1,i))
    return len(num)
```

6) 수박수박수박수박수?

수박수박수박수박수박수?

문제 설명

길이가 n이고, "수박수박수박수...."와 같은 패턴을 유지하는 문자열을 리턴하는 함수, solution을 완성하세요. 예를들어 n이 4이면 "수박수박"을 리턴하고 3이라면 "수박수"를 리턴하면 됩니다.

제한 조건

- n은 길이 10,000이하인 자연수입니다.

입출력 예

n	return
3	"수박수"
4	"수박수박"

```
def solution(n):
    answer = ''

    for i in range(n):
        if i % 2 == 0:
            answer += '수'
```

```
        else:
            answer += '박'
    return answer
```

```
def water_melon(n):
    s = "수박" * n
    return s[:n]
```

7) 문자열을 정수로 바꾸기

문자열을 정수로 바꾸기

문제 설명

문자열 s를 숫자로 변환한 결과를 반환하는 함수, solution을 완성하세요.

제한 조건

- s의 길이는 1 이상 5이하입니다.
- s의 맨앞에는 부호(+, -)가 올 수 있습니다.
- s는 부호와 숫자로만 이루어져있습니다.
- s는 "0"으로 시작하지 않습니다.

입출력 예

예를들어 str이 "1234"이면 1234를 반환하고, "-1234"이면 -1234를 반환하면 됩니다.
str은 부호(+,-)와 숫자로만 구성되어 있고, 잘못된 값이 입력되는 경우는 없습니다.

```
def solution(s):
    answer = 0

    if int(s) < 0 :
        answer = int(s[1:])
```

```
    answer = int(s)
    return answer
```

```
def strToInt(str):
    #함수를 완성하세요
    a = int(str)
    return a
```

8) 시저 암호 (못푼 문제)

시저 암호

문제 설명

어떤 문장의 각 알파벳을 일정한 거리만큼 밀어서 다른 알파벳으로 바꾸는 암호화 방식을 시저 암호라고 합니다. 예를 들어 "AB"는 1만큼 밀면 "BC"가 되고, 3만큼 밀면 "DE"가 됩니다. "z"는 1만큼 밀면 "a"가 됩니다. 문자열 s와 거리 n을 입력받아 s를 n만큼 민 암호문을 만드는 함수, solution을 완성해 보세요.

제한 조건

- 공백은 아무리 밀어도 공백입니다.
- s는 알파벳 소문자, 대문자, 공백으로만 이루어져 있습니다.
- s의 길이는 8000이하입니다.
- n은 1 이상, 25이하인 자연수입니다.

입출력 예

s	n	result
"AB"	1	"BC"
"z"	1	"a"
"a B z"	4	"e F d"

```
#다른 사람 풀이 1위

def caesar(s, n):
    s = list(s)
    for i in range(len(s)):
        if s[i].isupper():
```

```
            s[i]=chr((ord(s[i])-ord('A')+ n)%26+ord('A'))
        elif s[i].islower():
            s[i]=chr((ord(s[i])-ord('a')+ n)%26+ord('a'))

    return "".join(s)

#issupper() - 파라미터 없이 문자열 내부에 있는 모든 문자가 대문자이면 True 를 리턴
#islower() - 파라미터 없이 문자열 내부에 있는 모든 문자가 소문자이면 True 를 리턴
#chr() -> 아스키 ( 숫자를 문자로 바꿈 ) , ord( ) -> 문자를 숫자로 바꿈
```

```
#다른 사람 풀이 2위

def caesar(s, n):
    lower_list = "abcdefghijklmnopqrstuvwxyz"
    upper_list = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

    result = []

    for i in s:
        if i is " ":
            result.append(" ")
        elif i.islower() is True:
            new_ = lower_list.find(i) + n
            result.append(lower_list[new_ % 26])
        else:
            new_ = upper_list.find(i) + n
            result.append(upper_list[new_ % 26])
    return "".join(result)
```

9) 약수의 합

약수의 합

문제 설명

정수 n을 입력받아 n의 약수를 모두 더한 값을 리턴하는 함수, solution을 완성해주세요.

제한 사항

- `n` 은 0 이상 3000이하인 정수입니다.

입출력 예

n	return
12	28
5	6

입출력 예 설명

입출력 예 #1
12의 약수는 1, 2, 3, 4, 6, 12입니다. 이를 모두 더하면 28입니다.

입출력 예 #2
5의 약수는 1, 5입니다. 이를 모두 더하면 6입니다.

```
def solution(n):
    answer = []

    for i in range(1, n+1):
        if n % i == 0:
```

```
            answer.append(i)
    return sum(answer)
```

```
def sumDivisor(num):
    # num / 2 의 수들만 검사하면 성능 약 2배 향상잼
    return num + sum([i for i in range(1, (num // 2) + 1) if num % i == 0])
```

10) 이상한 문자 만들기 (못 푼 문제)

이상한 문자 만들기

문제 설명

문자열 s는 한 개 이상의 단어로 구성되어 있습니다. 각 단어는 하나 이상의 공백문자로 구분되어 있습니다. 각 단어의 짝수번째 알파벳은 대문자로, 홀수번째 알파벳은 소문자로 바꾼 문자열을 리턴하는 함수, solution을 완성하세요.

제한 사항

- 문자열 전체의 짝/홀수 인덱스가 아니라, 단어(공백을 기준)별로 짝/홀수 인덱스를 판단해야합니다.
- 첫 번째 글자는 0번째 인덱스로 보아 짝수번째 알파벳으로 처리해야 합니다.

입출력 예

s	return
"try hello world"	"TrY HeLlO WoRlD"

입출력 예 설명

"try hello world"는 세 단어 "try", "hello", "world"로 구성되어 있습니다. 각 단어의 짝수번째 문자를 대문자로, 홀수번째 문자를 소문자로 바꾸면 "TrY", "HeLlO", "WoRlD"입니다. 따라서 "TrY HeLlO WoRlD" 를 리턴합니다.

```
#내가 푼 것 틀림

def solution(s):
    s_ls = s.split()
    new_ls = []
    for arr in s_ls:
        for i in range(len(arr)):
            if i % 2 == 0:
                arr[i].upper()
```

```
            else:
                arr[i]

    answer = ' '.join(new_ls)


    return answer
```

```
def solution(s):
    words_list = s.split(" ")  #" " 의 사용유무가 정답을 가린
    new_list = []
    for word in words_list:
        new_words = ""
        for i in range(len(word)):
            new_words += word[i].upper() if i%2 == 0 else word[i].lower()
        new_list.append(new_words)
    return " ".join(new_list)
```

11) 자릿 수 더하기

자릿수 더하기

문제 설명

자연수 N이 주어지면, N의 각 자릿수의 합을 구해서 return 하는 solution 함수를 만들어 주세요.
예를들어 N = 123이면 1 + 2 + 3 = 6을 return 하면 됩니다.

제한사항

- N의 범위 : 100,000,000 이하의 자연수

입출력 예

N	answer
123	6
987	24

입출력 예 설명

입출력 예 #1
문제의 예시와 같습니다.

입출력 예 #2
9 + 8 + 7 = 24이므로 24를 return 하면 됩니다.

```
def solution(n):
    answer = 0

    for i in str(n):
        answer += int(i)
    return answer
```

```
#재귀함수 이용
def sum_digit(number):
    if number < 10:
        return number;
    return (number % 10) + sum_digit(number // 10)

# 아래는 테스트로 출력해 보기 위한 코드입니다.
print("결과 : {}".format(sum_digit(123)));



# 리스트 컴프레이션 이용

def sum_digit(number):
    return sum([int(i) for i in str(number)])
# 아래는 테스트로 출력해 보기 위한 코드입니다.
print("결과 : {}".format(sum_digit(123)));
```

12) 자연수 뒤집어 배열로 만들기 (못 푼 문제)

자연수 뒤집어 배열로 만들기

문제 설명

자연수 n을 뒤집어 각 자리 숫자를 원소로 가지는 배열 형태로 리턴해주세요. 예를들어 n이 12345이면 [5,4,3,2,1]을 리턴합니다.

제한 조건

- n은 10,000,000,000이하인 자연수입니다.

입출력 예

n	return
12345	[5,4,3,2,1]

13) 정수 내림차순으로 배치하기

정수 내림차순으로 배치하기

문제 설명

함수 solution은 정수 n을 매개변수로 입력받습니다. n의 각 자릿수를 큰것부터 작은 순으로 정렬한 새로운 정수를 리턴해주세요. 예를들어 n이 118372면 873211을 리턴하면 됩니다.

제한 조건

- `n` 은 1이상 8000000000 이하인 자연수입니다.

입출력 예

n	return
118372	873211

```
def solution(n):
    answer = ''

    li = sorted(str(n), reverse = True)

    for i in li:
```

```
        answer += str(i)

    return int(answer)
```

```
def solution(n):
    ls = list(str(n))
    ls.sort(reverse = True)
    return int("".join(ls))
```

14) 정수 제곱근 판별

정수 제곱근 판별

문제 설명

임의의 양의 정수 n에 대해, n이 어떤 양의 정수 x의 제곱인지 아닌지 판단하려 합니다.
n이 양의 정수 x의 제곱이라면 x+1의 제곱을 리턴하고, n이 양의 정수 x의 제곱이 아니라면 -1을 리턴하는 함수를 완성하세요.

제한 사항

- n은 1이상, 50000000000000 이하인 양의 정수입니다.

입출력 예

n	return
121	144
3	-1

입출력 예 설명

입출력 예#1
121은 양의 정수 11의 제곱이므로, (11+1)를 제곱한 144를 리턴합니다.

입출력 예#2
3은 양의 정수의 제곱이 아니므로, -1을 리턴합니다.

```
def solution(n):
    answer = 0
    x = (n**0.5)

    if x.is_integer():
        answer =+ (x+1)**2

    else:
        answer =+ -1
```

```
    return answer
```

```
def nextSqure(n):
    sqrt = n ** (1/2)

    if sqrt % 1 == 0:
        return (sqrt + 1) ** 2
    return 'no'
```

15) 제일 작은 수 제거하기

제일 작은 수 제거하기

문제 설명

정수를 저장한 배열, arr 에서 가장 작은 수를 제거한 배열을 리턴하는 함수, solution을 완성해주세요. 단, 리턴하려는 배열이 빈 배열인 경우엔 배열에 -1을 채워 리턴하세요. 예를들어 arr이 [4,3,2,1]인 경우는 [4,3,2]를 리턴 하고, [10]면 [-1]을 리턴 합니다.

제한 조건

- arr은 길이 1 이상인 배열입니다.
- 인덱스 i, j에 대해 i ≠ j이면 arr[i] ≠ arr[j] 입니다.

입출력 예

arr	return
[4,3,2,1]	[4,3,2]
[10]	[-1]

```
def solution(arr):

    arr.remove(min(arr))

    if len(arr) == 0:
        arr.append(-1)

    return arr
```

```
def rm_small(mylist):
    return [i for i in mylist if i > min(mylist)]

# 아래는 테스트로 출력해 보기 위한 코드입니다.
my_list = [4,3,2,1]
print("결과 {} ".format(rm_small(my_list)))
```

16) 짝수와 홀수

짝수와 홀수

문제 설명

정수 num이 짝수일 경우 "Even"을 반환하고 홀수인 경우 "Odd"를 반환하는 함수, solution을 완성해 주세요.

제한 조건

- num은 int 범위의 정수입니다.
- 0은 짝수입니다.

입출력 예

num	return
3	"Odd"
4	"Even"

```
def solution(num):
    if num % 2 == 0:
        answer = "Even"
    else:
        answer = "Odd"
    return answer

#TIL - if , elif, else 문 안에 return 입력가능
```

```
def evenOrOdd(num):
    if (num%2):
        return "Odd"
    else:
        return "Even"
```

17) 최대공약수와 최소공배수

최대공약수와 최소공배수

문제 설명

두 수를 입력받아 두 수의 최대공약수와 최소공배수를 반환하는 함수, solution을 완성해 보세요. 배열의 맨 앞에 최대공약수, 그다음 최소공배수를 넣어 반환하면 됩니다. 예를 들어 두 수 3, 12의 최대공약수는 3, 최소공배수는 12이므로 solution(3, 12)는 [3, 12]를 반환해야 합니다.

제한 사항

- 두 수는 1이상 1000000이하의 자연수입니다.

입출력 예

n	m	return
3	12	[3, 12]
2	5	[1, 10]

입출력 예 설명

입출력 예 #1
위의 설명과 같습니다.

입출력 예 #2
자연수 2와 5의 최대공약수는 1, 최소공배수는 10이므로 [1, 10]을 리턴해야 합니다.

```
def solution(n, m):
    answer = []
    n_ls = []
    m_ls = []
    k_ls = []

    # n 약수 구하기
    for i in range(1, n+1):
        if n % i == 0:
```

```
            n_ls.append(i)
    # m 약수 구하기
    for j in range(1, m+1):
        if m % j == 0:
            m_ls.append(j)

    # n, m 공약수 구하기
    for k in n_ls:
        if k in m_ls:
            k_ls.append(k)

    # 최대공약수 answer 내에 삽입
    answer.append(max(k_ls))

    # 최소공배수 answer 내에 삽입
    answer.append(max(k_ls) * ((n/max(k_ls)) * (m/max(k_ls))))


    return answer
```

```
# 유클리드의 호제법 사용

def gcdlcm(a, b):
    c, d = max(a, b), min(a, b)
    t = 1
    while t > 0:
        t = c % d
        c, d = d, t
    answer = [c, int(a*b/c)]

    return answer

# 아래는 테스트로 출력해 보기 위한 코드입니다.
print(gcdlcm(3,12))
```

18) 콜라츠 추측

콜라츠 추측

문제 설명

1937년 Collatz란 사람에 의해 제기된 이 추측은, 주어진 수가 1이 될때까지 다음 작업을 반복하면, 모든 수를 1로 만들 수 있다는 추측입니다. 작업은 다음과 같습니다.

```
1-1. 입력된 수가 짝수라면 2로 나눕니다.
1-2. 입력된 수가 홀수라면 3을 곱하고 1을 더합니다.
2. 결과로 나온 수에 같은 작업을 1이 될 때까지 반복합니다.
```

예를 들어, 입력된 수가 6이라면 6→3→10→5→16→8→4→2→1 이 되어 총 8번 만에 1이 됩니다. 위 작업을 몇 번이나 반복해야하는지 반환하는 함수, solution을 완성해 주세요. 단, 작업을 500번을 반복해도 1이 되지 않는다면 –1을 반환해 주세요.

제한 사항

- 입력된 수, `num` 은 1 이상 8000000 미만인 정수입니다.

입출력 예

n	result
6	8
16	4
626331	-1

입출력 예 설명

입출력 예 #1
문제의 설명과 같습니다.

입출력 예 #2
16 -> 8 -> 4 -> 2 -> 1 이되어 총 4번만에 1이 됩니다.

입출력 예 #3
626331은 500번을 시도해도 1이 되지 못하므로 -1을 리턴해야합니다.

```
def solution(num):

    count = 0

    for i in range(500):
        if num != 1:
            if num % 2 == 0:
                num = num / 2
            else:
```

```
                num = (num * 3) + 1

            count += 1

            if count >= 500:
                count = -1
                break


    return count
```

```
def collatz(num):
    for i in range(500):
        num = num / 2 if num % 2 == 0 else num*3 + 1
        if num == 1:
            return i + 1
    return -1
# if문안에 리턴 하나 외부에 하나 따로 입력할 수 있다.
# 아래는 테스트로 출력해 보기 위한 코드입니다.
print(collatz(6))
```

19) 평균 구하기

평균 구하기

문제 설명

정수를 담고 있는 배열 arr의 평균값을 return하는 함수, solution을 완성해보세요.

제한사항

- arr은 길이 1 이상, 100 이하인 배열입니다.
- arr의 원소는 -10,000 이상 10,000 이하인 정수입니다.

입출력 예

arr	return
[1,2,3,4]	2.5
[5,5]	5

```
def solution(arr):

    answer = (sum(arr)/len(arr))

    return answer
```

```
def average(list):
    return (sum(list) / len(list))
```

20) 하샤드 수

하샤드 수

문제 설명

양의 정수 x가 하샤드 수이려면 x의 자릿수의 합으로 x가 나누어져야 합니다. 예를 들어 18의 자릿수 합은 1+8=9이고, 18은 9로 나누어 떨어지므로 18은 하샤드 수입니다. 자연수 x를 입력받아 x가 하샤드 수인지 아닌지 검사하는 함수, solution을 완성해주세요.

제한 조건

- `x` 는 1 이상, 10000 이하인 정수입니다.

입출력 예

arr	return
10	true
12	true
11	false
13	false

입출력 예 설명

입출력 예 #1
10의 모든 자릿수의 합은 1입니다. 10은 1로 나누어 떨어지므로 10은 하샤드 수입니다.

입출력 예 #2
12의 모든 자릿수의 합은 3입니다. 12는 3으로 나누어 떨어지므로 12는 하샤드 수입니다.

입출력 예 #3
11의 모든 자릿수의 합은 2입니다. 11은 2로 나누어 떨어지지 않으므로 11는 하샤드 수가 아닙니다.

입출력 예 #4
13의 모든 자릿수의 합은 4입니다. 13은 4로 나누어 떨어지지 않으므로 13은 하샤드 수가 아닙니다.

```
def solution(x):
    answer = True

    # x 자릿수의 합 구하기
    ls = []

    for i in str(x):
        ls.append(int(i))

    # x 를 자릿수 합으로 나누기
    if x % sum(ls) == 0:
        answer = True

    else:
        answer = False

    return answer
```

```
def Harshad(n):
    # n은 하샤드 수 인가요?
    return n % sum([int(c) for c in str(n)]) == 0

# 아래는 테스트로 출력해 보기 위한 코드입니다.
print(Harshad(18))
```

21) 핸드폰 번호 가리기

핸드폰 번호 가리기

문제 설명

프로그래머스 모바일은 개인정보 보호를 위해 고지서를 보낼 때 고객들의 전화번호의 일부를 가립니다. 전화번호가 문자열 phone_number로 주어졌을 때, 전화번호의 뒷 4자리를 제외한 나머지 숫자를 전부 `*` 으로 가린 문자열을 리턴하는 함수, solution을 완성해주세요.

제한 조건

- s는 길이 4 이상, 20이하인 문자열입니다.

입출력 예

phone_number	return
"01033334444"	"*******4444"
"027778888"	"*****8888"

```
def solution(phone_number):
    answer = ''
    answer += ('*' * (len(phone_number) - 4)) + phone_number[-4:]
    return answer
```

```
def hide_numbers(s):
    return "*"*(len(s)-4) + s[-4:]
```

22) 행렬의 덧셈 (못 푼 문제)

행렬의 덧셈

문제 설명

행렬의 덧셈은 행과 열의 크기가 같은 두 행렬의 같은 행, 같은 열의 값을 서로 더한 결과가 됩니다. 2개의 행렬 arr1과 arr2를 입력받아, 행렬 덧셈의 결과를 반환하는 함수, solution을 완성해주세요.

제한 조건

- 행렬 arr1, arr2의 행과 열의 길이는 500을 넘지 않습니다.

입출력 예

arr1	arr2	return
[[1,2],[2,3]]	[[3,4],[5,6]]	[[4,6],[7,9]]
[[1],[2]]	[[3],[4]]	[[4],[6]]

```
def solution(arr1, arr2):
    answer = [[] for x in range(len(arr2))]  # 참고한 부분 원래 [[] * len(arr1)] 이라고 적음

    for i in range(0, len(arr1)):
        for j in range(0, len(arr1[i])): # 인터넷 보고 고친 부분 len(arr1[i])
            answer[i].append(arr1[i][j] + arr2[i][j])   # 참고한 부분

    return answer
```

```
def sumMatrix(A,B):
    answer = [[c + d for c, d in zip(a, b)] for a, b in zip(A,B)]
    return answer

# 아래는 테스트로 출력해 보기 위한 코드입니다.
print(sumMatrix([[1,2], [2,3]], [[3,4],[5,6]]))
```

행렬의 덧셈이나 뺄셈은 zip( ) 메소드를 이용하자!!!

23) x 만큼 간격이 있는 n개의 숫자

x만큼 간격이 있는 n개의 숫자

문제 설명

함수 solution은 정수 x와 자연수 n을 입력 받아, x부터 시작해 x씩 증가하는 숫자를 n개 지니는 리스트를 리턴해야 합니다. 다음 제한 조건을 보고, 조건을 만족하는 함수, solution을 완성해주세요.

제한 조건

- x는 -10000000 이상, 10000000 이하인 정수입니다.
- n은 1000 이하인 자연수입니다.

입출력 예

x	n	answer
2	5	[2,4,6,8,10]
4	3	[4,8,12]
-4	2	[-4, -8]

```
def solution(x, n):
    answer = []
    for i in range(1, n+1):
        answer.append(i * x)
    return answer
```

```
def number_generator(x, n):
    # 함수를 완성하세요
    return [i * x + x for i in range(n)]
print(number_generator(2, 5))
```

24) 직사각형 별찍기

직사각형 별찍기

문제 설명

이 문제에는 표준 입력으로 두 개의 정수 n과 m이 주어집니다.
별(*) 문자를 이용해 가로의 길이가 n, 세로의 길이가 m인 직사각형 형태를 출력해보세요.

제한 조건

- n과 m은 각각 1000 이하인 자연수입니다.

예시

입력

```
5 3
```

출력

```
*****
*****
*****
```

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

for i in range(m):
    for j in range(n):
        print('*', end='')
    print()
```

```
a, b = map(int, input().strip().split(' '))
answer = ('*'*a +'\n')*b
print(answer)
```

2. 오늘 배운 메서드들

```
#if 조건문

if len(s) == 4 or len(s) == 6

==> if len(s) == 4 or 6 로 간단하게 작성 가능하다 .
```

```
# index( ) 메소드 - 찾고자 하는 문자열 이나 숫자의 인덱스를 출력한다.

list = [2, 4, 6]

list.index(2)
> [0]
```

```
# 문자열 곱하기

string = "수박" * 5

print(string)

> "수박수박수박수박수박"
```

```
# list( ) 메소드 - 문자열을 리스트로 변환해준다.

num = "123242"

list(num)
> ['1', '2', .... , '2']

단, int 는 사용불가
```

```
# .join() 메소드 - 숫자 배열 --> 숫자로 변경할 때 유용
```

```
# is_integer( ) 메소드 --> 변수가 정수인지 아닌지 확인

num = 1.0

print(num.is_integer())

> True
```

```
# 리스트 내 원소 삭제 - del( ) or remove( ) 메소드
```

```
# return 내에 사용 가능한 것들

return if, else, and, or 등 모두 사용 가능하다.
```

```
# 최대공약수 구하는 것은 유클리드 호제법 이용하여 구해보자
```

```
# str( ) 메소드

숫자 1204 를 하나씩 리스트에 담기 위해서는 숫자를 str( ) 메소드로 문자
열로 변환 후 list( ) 함수를 사용하여 배열로 만든다.
```

```
# 이차원 배열 len( ) 메소드 사용가능

ls = [[1, 2], [3, 5]]
```

```
len(ls)
> 2
```