```
lt1 = [(1.594251807546243e-06, 5.0024758456856947e-06),
 (1.3259880506666377e-06, 4.6269064841908407e-06),
 (1.2927745774504728e-06, 4.2947709516738538e-06),
 (1.4128545808489435e-06, 6.0091022896813227e-06),
 (1.4946109877200796e-06, 4.7418766553164456e-06),
 (1.8088624710799195e-06, 5.275848998280708e-06),
 (1.7066668442566879e-06, 5.7740522970561866e-06),
 (1.7449900406063535e-06, 4.9309386304230432e-06),
 (1.7884231056086719e-06, 5.5160086048999798e-06),
 (1.8727346832747572e-06, 6.0831944210804068e-06)]

# 1번 index
def extract_point1(lt,num):
  temp = [0 for i in range(len(lt))]
  for i in range(len(lt)):
    temp[i] = lt1[i][num]
  return temp



# 2번 index의 다른방법 -> enumerate
# enumerate : 인덱스 위치와 값을  가져온다. (2개의 값을 돌려준다)
def extract_point2(lt,num):
  temp = [0 for i in range(len(lt))]
  for i,k in enumerate(lt):
    temp[i] = k[num]
  return temp



# 3번 내장함수 append 이용
def extract_point3(lt,num):
  temp = []
  for i in range(len(lt)):
    temp.append(lt[i][num])
  return temp

import numpy as np
# # 4번 numpy의 append 이용
# numpy의 특징은 다음과 같습니다.
# 일반 List에 비해 빠르고, 메모리를 효율적으로 사용한다.
# 반복문 없이 데이터 배열에 대한 처리를 지원하여 빠르고 편리하다.
# 선형대수와 관련된 다양한 기능을 제공한다.
# C, C++, 포트란 등의 언어와 통합이 가능하다.

def extract_point4(lt,num):
  temp = []
  np_lt = np.array(lt)
```

```
  for i in range(len(np_lt)):
    temp = np.append(temp,np_lt[i][num])
  return temp



# 5번 List Conprehension
# 리스트 컴프리헨션은 쉽게 말해 '리스트를 쉽게, 짧게 한 줄로 만들 수 있는 파이썬의 문법'이다
# 장점
# 코드가 간결하다.
# 데이터베이스를 조회하여 리스트를 만들 때 많이 사용되므로, 새로운 배열을 만들때 특히 유용하다.
# 반복문을 활용하여 새로운 배열을 만드는것 보다 처리속도가 빠르다.
#단점 :복잡한 표현이 작성될 수록 가독성이 떨어진다.

def extract_point5(lt,num):
  temp = [i[num] for i in lt]
  return temp



# 6번 map함수 이용
###############map #########
# map은 리스트의 요소를 지정된 함수로 처리해주는 함수입니다(map은 원본 리스트를 변경하지 않고 새 리스트를 생성)
# 여러 개의 데이터를 한 번에 다른 형태로 변환하기 위해서 사용된다.
# 더 간단하고 메모리도 절약할 수 있는 iterator를 결과로 받을 수 있습니다.

############### Filter #########
# 함수의 조건에 일치하는 값만 반환하고 싶을 때 사용
# filter함수는 특정 조건으로 걸러서 걸러진 요소들로 iterator객체를 만들어서 리턴해줍니다
# map함수와 사용 방법은 동일하나, 함수의 결과가 참인지 거짓인지에 따라, 해당 요소를 포함할지를 결정합니다.
# 결과가 true 인 것만 여과한다.
# list(map(함수, 리스트))
# tuple(map(함수, 튜플))

def lt_map_x(num):
  return num[0]



def lt_map_y(num):
  return num[1]

def extract_point6(lt,num):
  if num ==0:
    temp = list(map(lt_map_x,lt))  # temp = list(map(lambda x: lt_map_x, li)) # 같은 결과임
    return temp
```

```
  else:
    temp = list(map(lt_map_y,lt))
    return temp


print('x 합 (extract_point1): ',sum(extract_point1(lt1,0)))
print('y 합 (extract_point1): ',sum(extract_point1(lt1,1)))

print('x 합 (extract_point2): ',sum(extract_point2(lt1,0)))
print('y 합 (extract_point2): ',sum(extract_point2(lt1,1)))

print('x 합 (extract_point3): ',sum(extract_point3(lt1,0)))
print('y 합 (extract_point3): ',sum(extract_point3(lt1,1)))

print('x 합 (extract_point4): ',sum(extract_point4(lt1,0)))
print('y 합 (extract_point4): ',sum(extract_point4(lt1,1)))

print('x 합 (extract_point5): ',sum(extract_point5(lt1,0)))
print('y 합 (extract_point5): ',sum(extract_point5(lt1,1)))

print('x 합 (extract_point6): ',sum(extract_point6(lt1,0)))
print('y 합 (extract_point6): ',sum(extract_point6(lt1,1)))
```

```
x 합 (extract_point1):  1.6042157149058764e-05
y 합 (extract_point1):  5.2255175178288485e-05
x 합 (extract_point2):  1.6042157149058764e-05
y 합 (extract_point2):  5.2255175178288485e-05
x 합 (extract_point3):  1.6042157149058764e-05
y 합 (extract_point3):  5.2255175178288485e-05
x 합 (extract_point4):  1.6042157149058764e-05
y 합 (extract_point4):  5.2255175178288485e-05
x 합 (extract_point5):  1.6042157149058764e-05
y 합 (extract_point5):  5.2255175178288485e-05
x 합 (extract_point6):  1.6042157149058764e-05
y 합 (extract_point6):  5.2255175178288485e-05
```

교수님이 가르켜주신 코드

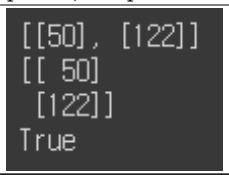
```
# import numpy as np

# def isNumber(v):
#     if type(v) == int or type(v) == float:
#         return True
#     elif type(v) == list or type(v) == tuple: #형태가 리스트나 튜플이면 하나씩 꺼내서 type을 확인하는
#         li = [isNumber(i) for i in v]
#         return True if all(li) else False
#     else:
#         return False #문자열일 경우


# def ltall2(lt):
#      if len(lt) % 2 != 0:
#         lt.append(lt[0])
#     lt2 = lt.copy()
#     lt2.reverse()
#     return lt == lt2


# def lt_shape(lt):
#     row = len(lt)
#     col_in_row = [len(lt[i]) for i in range(row)]#각 행별 길이가 저장됨
#     if ltall2(col_in_row): #결과가 true 이면 실행
#         col = col_in_row[0]
#     return row,col


# def lt_trans(lt):

#     row,col = lt_shape(lt)
#     lt_a = []
#     for i in range(col):
#         temp = []
#         for j in range(row):
#             temp.append(lt[j][i])
#         lt_a.append(temp)
#     return lt_a


# def lt_mul(lta,ltb): # 두 리스트를 곱해주는 함수

#     lt3 = [lta[i] * ltb[i] for i in range(len(lta))]
#     return lt3


# def lt_dot(lta,ltb):
```

	# if not isNumber(lta) or not isNumber(ltb): # print("잘못된 리스트 형태입니다. 리스트를 다시 입력하세요 ") # return False # if lt_shape(lta)[1] != lt_shape(ltb)[0]: #리스트 주고 행,열을 리스트 형태로 반환받아, 앞에는 열만 뒤 # print("두 리스트의 열과 행이 다릅니다. 동일하게 설정하세요. ") # return False # ltc = [] # for i in lta: # temp = [] # for j in lt_trans(ltb): # temp.append(sum(lt_mul(i,j))) # 곱한값을 더해줌 # ltc.append(temp) # return ltc # def lt_np_arr_equal(lt,np): # if lt_shape(lt) == lt_shape(np): # temp = [] # for i in range(len(lt)): # if lt[i] == np[i]: # temp.append('true') # else: # temp.append('false') # if all(temp): # return True # else: # return False # else: # return False # print(lt[0],np[0]) # lt1 = [[1,2,3],[4,5,6]] # 2행 3열 # lt2 = [[7],[8],[9]] # 3행 1열 # lt_in = lt_dot(lt1,lt2) # np_in = np.dot(np.array(lt1),np.array(lt2)) # print(lt_np_arr_equal(lt_in,np_in)) # 같으면 True 다르면 False

동영상 코드	# 문제 2번 -> 두 리스트의 내적 구하기
	
	def lt_dot(lta,ltb):# 3*2 2*3
	  if not isNumber(lta) or not isNumber(lta):
	  print("잘못된 리스트 ")
	  if lt_shape(lta)[1] != lt_shape(ltb)[0]:
	  print('두 리스트는 열과 행이 다릅니다. ')
	  ltc =[[sum(lt_mup(i,j)) for j in lt_trans(ltb)] for i in lta]

```
  return ltc

def isNumber(lt):
  if type(lt) == int or type(lt) == float:
    return True
  elif type(lt) == list or type(lt) == tuple:
    li = list(map(isNumber,lt))
    return True if ltall(li) else False
  else:
    return False

def ltall(lt):# 1,2,1,1
  max = 0
  for i in range(len(lt)):
    temp = 0
    for j in range(len(lt)-1):
      if lt[j+1] > lt[i]:
        max = j+1
    temp = lt[i]
    lt[i] = lt[max]
    lt[max] = temp
  if lt[0] != lt[-1]:
    return False
  else:
    return True

def lt_shape(lt):
  row = len(lt)
  li_col = [len(lt[i]) for i in range(row)]

  if ltall2(li_col):
    col = li_col[0]
    return row,col
  else:
    '잘못입력'

def ltall2(lt):
  lt_a = lt.copy()
  for i in range(len(lt_a)):
    if lt_a[0] != lt_a[i]:
      return False
  return True

def lt_trans(lt):# 2*3
  row, col = lt_shape(lt)
  lt_a = [[lt[j][i] for j in range(row)] for i in range(col)]
  return lt_a
```

	def lt_mup(lta,ltb):
	  ltc = [0 for i in range(len(lta))]
	  for i in range(len(lta)):
	  ltc[i] = lta[i] * ltb[i]
	  return ltc
	
	def lt_np_arr_equal(lt,np):
	  if lt_shape(lt) == lt_shape(np):
	  temp = []
	  for i in range(len(lt)):
	  if lt[i] == np[i]:
	  temp.append('True')
	  else:
	  temp.append('False')
	  if all(temp) :
	  return True
	  else:
	  return False
	
	lt1 = [[1,2,3],[4,5,6]] # 2행 3열
	lt2 = [[7],[8],[9]] # 3행 1열
	
	import numpy as np
	lt_in = lt_dot(lt1,lt2)
	print(lt_in)
	np_in = np.dot(np.array(lt1),np.array(lt2))
	print(np_in)
	
	print(lt_np_arr_equal(lt_in,np_in)) # 같으면 True 다르면 False

동영상 코드결과	

	# python list는 다양한 자료형이 저장가능 하지만 numpy array는 같은 자료형만 넣을 수 있다. # 따라서 python list는 값을 추가하고 제거하는 일에 쓰고, # numpy array는 수치 계산이 많고 복잡할 때나 행렬같은 다차원 배열을 쓸 때 사용하면 된다. # Array는 정적 할당에 해당하고, List의 경우는 동적 할당에 해당함 # 동적으로 계속 크기가 변할 수 있는 Python List와 달리 NumPy Array는 고정된 크기를 갖는다. import numpy as np lt1 = [(0,1),(1,1),(2,1),(1,10),(10,10),(11,10)] #입력값 # np_1 = np.array(lt1) print(lt1) # print(np_1) # print(11/10) # print(11//10) # print(11%10) # print(2//3) # print(4//3) # print(4/3)

초기 구성	# 1보다 작으면 1.
	# 1보다 크면
	
	def lt_result(lt):
	
	  for i in lt:
	  a = i[0] / i[1]
	  if a <= 1:# 작을때
	  if a == 0:
	  print("(m,n)=",i , "--> ",0)
	  else:
	  print("(m,n)=",i , "--> ",1)
	  else:# 클때
	  b = i[0] // i[1]
	  # print('빼기',b-a)
	  if a-b > 0:
	  print("(m,n)=",i , "--> ",b+1)
	  else:
	  print("(m,n)=",i , "--> ", b)
	
	lt_result(lt1)

초기 결과

```
[(0, 1), (1, 1), (2, 1), (1, 10), (10, 10), (11, 10)]
(m,n)= (0, 1) -->  0
(m,n)= (1, 1) -->  1
(m,n)= (2, 1) -->  2
(m,n)= (1, 10) -->  1
(m,n)= (10, 10) -->  1
(m,n)= (11, 10) -->  2
```

동영상 코드

```
# 문제 3번

lt1 = [(0,1),(1,1),(2,1),(1,10),(10,10),(11,10)]  #입력값

def lt_result(lt):
  x = lt[0]
  y = lt[1]

  a = x / y
  b = x // y

  if a <= 1:
    if a ==0:# 나눌값이 없을경우( 게시물 개수가 없경우)
      return 0
    else:
      return 1
  else:
    if a-b >0:
      return b+1
    else:
      return b




# 1번 numpy 사용할때
import numpy as np

np_lt = np.array(lt1)
temp = np.array([])
cnt = 0



for i in np_lt:
  temp = np.append(temp,lt_result(i))
  print("(m,n)=",i , "--> ",int(temp[cnt]))

  cnt = cnt+1
```

	print()
	# 2번 numpy 사용 안할때
	temp = []
	cnt = 0
	for i in lt1:
	  temp.append(lt_result(i))
	  print("(m,n)=",i , "--> ",temp[cnt])
	  cnt = cnt +1
	
	print()
	# 3번 List Comprehension 사용할때
	temp = [lt_result(i) for i in lt1]
	
	for i in range(len(lt1)):
	  print("(m,n)=",lt1[i] , "--> ",temp[i])
	
	# print("(m,n)=",i , "--> ",)

동영상 결과	(m,n)= [0 1] --> 0 (m,n)= [1 1] --> 1 (m,n)= [2 1] --> 2 (m,n)= [1 10] --> 1 (m,n)= [10 10] --> 1 (m,n)= [11 10] --> 2 (m,n)= (0, 1) --> 0 (m,n)= (1, 1) --> 1 (m,n)= (2, 1) --> 2 (m,n)= (1, 10) --> 1 (m,n)= (10, 10) --> 1 (m,n)= (11, 10) --> 2 (m,n)= (0, 1) --> 0 (m,n)= (1, 1) --> 1 (m,n)= (2, 1) --> 2 (m,n)= (1, 10) --> 1 (m,n)= (10, 10) --> 1 (m,n)= (11, 10) --> 2