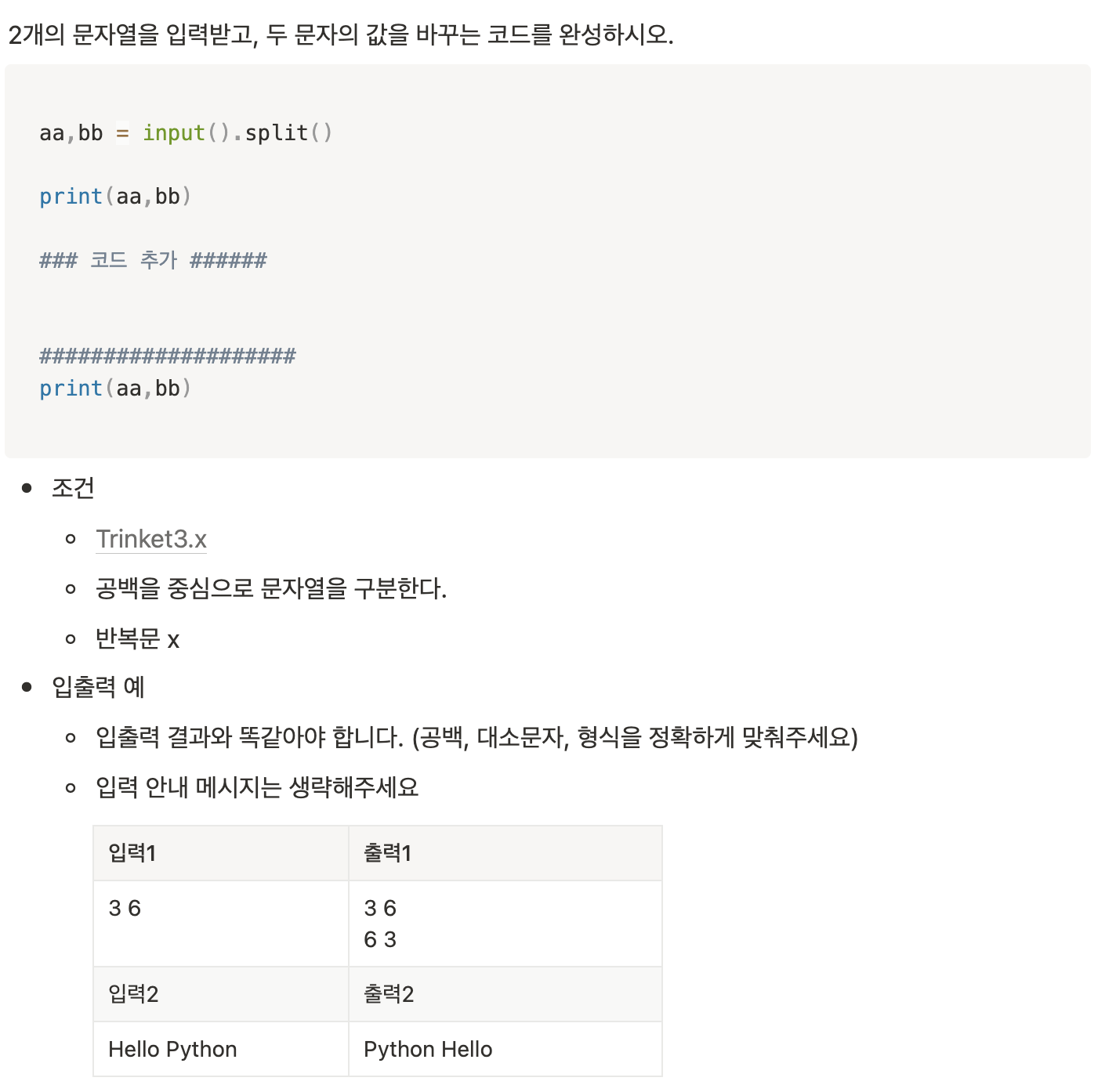
**REPORT**

**[ 과제 : 파이썬기초 문제지(9) 메모리맵,리스트(2) ]**



|  |  |
| --- | --- |
| 과 목 명 | 파이썬과학프로그래밍기초 |
| 교 수 명 | 김 병 정 |
| 학 번 | 20237107 |
| 작 성 자 | 하 태 영 |
| 제 출 일 | 2025.04.27 |

**한림대학교**

**문제 PY91-0001**

aa,bb = input().split() # aa, bb를 입력값 공백 기준으로 나누어 저장

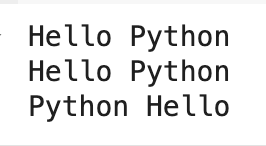
print(aa,bb) # aa, bb를 출력

### 코드 추가 ######

aa, bb = bb, aa # aa를 bb에 저장, bb를 aa에 저장

####################

print(aa,bb) # aa, bb를 출력

****

**문제 PY91-0002**

****

**문제 PY91-0002**

* myswap1()

def myswap1(a, b): # myswap1 사용자 정의 함수, 매개변수 a, b를 받는다

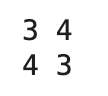
return b, a # a와 b의 값을 바꾸어 반환

a, b = 3, 4 # a에는 3을, b에는 4를 저장

print(a,b) # 바꾸기 전의 a, b를 출력

a,b = myswap1(a,b) # myswap1 함수를 호출하여 a, b 값을 교환

print(a,b) # 바꾼 후의 a, b를 출력

****

* myswap2()

def myswap2(a, b): # myswap2 사용자 정의 함수, 매개변수 a, b를 받는다

a, b = b, a # a를 b에 저장, b를 a에 저장

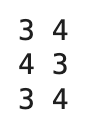
print(a, b) # 바뀐 a, b를 출력

a, b = 3, 4 # a에는 3을, b에는 4를 저장

print(a,b) # 바꾸기 전의 a, b를 출력

myswap2(a,b) # myswap2 함수를 호출하여 a, b 값을 교환

print(a,b) # 바꾼 후의 a, b를 출력

****

* myswap3()

def myswap3(): # myswap3 사용자 정의 함수

global a, b # 전역변수 a, b 선언

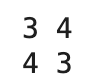
a, b = b, a # a에 b를 저장, b에 a를 저장

a, b = 3, 4 # a에 3을 저장, b에 4를 저장

print(a, b) # a, b를 출력

myswap3() # myswap3 함수를 호출

print(a, b) # 바꾼 후의 a, b를 출력

****

**문제 PY91-0003**

**텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.**

def myswap4(): # myswap3 사용자 정의 함수

# lt 0번째를 lt 1번째에 저장, lt 1번째를 lt 0번째를 저장

lt[0], lt[1] = lt[1], lt[0]

lt= [3,4] # lt 리스트 선언

print(lt) # lt 리스트 출력

myswap4() # myswap4 함수를 호출

print(lt) # 바꾼 후의 lt 리스트 출력

**텍스트, 폰트, 번호, 시계이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.**

**문제 PY91-0004**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.**

lt1 = ['AAA','BBB','AAA','CCC'] # lt1 lt1 리스트 선언 및 초기화

def delete\_List(): # delete\_List 사용자 정의 함수

global lt1 # lt1을 전역 변수로 선언하여 함수 내부에서 수정 가능하게 함

print(lt1) # 기존 lt1 리스트 출력

lt1 = ['DDD','BBB','FFF','CCC'] # lt1 값을 새로운 리스트로 갱신

print(lt1) # 갱신된 lt1 출력

delete\_List() # delete\_List() 함수 호출

**텍스트, 폰트, 화이트, 타이포그래피이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.**

**문제 PY91-0005**

**텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.**

**문제 PY91-0005**

* 방법1) deepcopy 사용 (모듈추가 코드 포함)

import copy # copy 모듈을 사용하기 위한 import (deepcopy 함수 사용 목적)

lt1 = [[3,4],5,6]

lt2 = []

print(lt1,lt2) #[[3, 4], 5, 6] []

## 코드 추가

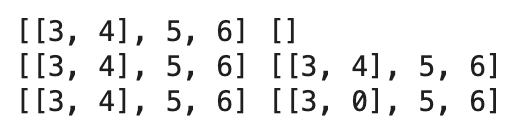
lt2 = copy.deepcopy(lt1) # lt1의 깊은 복사본을 lt2에 저장 (내부 리스트까지 복사됨)

####################

print(lt1,lt2) #[[3, 4], 5, 6] [[3, 4], 5, 6]

lt2[0][1] = 0 # lt2 내부 리스트의 값 변경 (lt1에는 영향 없음)

print(lt1,lt2) #[[3, 4], 5, 6] [[3, 0], 5, 6]

****

**문제 PY91-0005**

* 방법2) LC, 슬라이스연산자 사용

lt1 = [[3,4],5,6]

lt2 = []

print(lt1,lt2) #[[3, 4], 5, 6] []

## 코드 추가

# 1. lt1의 각 요소를 kk로 하나씩 꺼낸다.

# 2. kk가 리스트 타입이면 (type(kk) == list) -> kk[:]로 복사해서 사용한다.

# 3. list 타입이 아니면 (예: 숫자 같은 거) -> 그냥 kk를 그대로 사용한다.

# 4. 이렇게 만들어진 값들을 모아서 lt2를 만든다.

lt2 = [kk[:] if type(kk) == list else kk for kk in lt1]

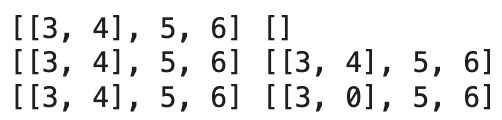
# lt2 = lt1[:] => 리스트 껍데기만 복사

####################

print(lt1,lt2) #[[3, 4], 5, 6] [[3, 4], 5, 6]

lt2[0][1] = 0

print(lt1,lt2) #[[3, 4], 5, 6] [[3, 0], 5, 6]

****

**문제 PY91-0010**

****

**문제 PY91-0010**

* 방법1) collections.Counter를 사용 x

import ast # ast 모듈을 가져온다

# ast.literal\_eval : 입력된 문자열을 안전하게 리스트로 변환

lt1 = ast.literal\_eval(input("lt1 = ")) # lt1 입력

lt2 = ast.literal\_eval(input("lt2 = ")) # lt2 입력

# lt1에서 원소 개수 세기

c1 = {} # c1 딕셔너리 선언

for num in lt1: # lt1의 각 문자를 num에 대입하여 반복

if num in c1: # c1 딕셔너리에 num 키가 존재하면

c1[num] += 1 # c1[num] 값을 1 증가시킨다

else: # c1 딕셔너리에 num 키가 없으면

c1[num] = 1 # c1[num]에 1을 저장한다 (처음 등장)

# lt2에서 원소 개수 세기

c2 = {} # c2 딕셔너리 선언

for num in lt2: # lt2의 각 문자를 num에 대입하여 반복

if num in c2: # c2 딕셔너리에 num 키가 존재하면

c2[num] += 1 # c2[num] 값을 1 증가시킨다

else: # c2 딕셔너리에 num 키가 없으면

c2[num] = 1 # c1[num]에 1을 저장한다 (처음 등장)

# 비교

if c1 == c2: # c1과 c2가 같다면 (원소 종류와 개수가 모두 같음)

print(f"{lt1} 와 {lt2} 의 원소의 종류와 갯수가 같습니다.")

else: # c1과 c2가 다르면

print(f"{lt1} 와 {lt2} 의 원소의 종류와 갯수가 다릅니다.")

# lt1이 더 많은 원소 찾기

diff1 = {} # diff1 딕셔너리 선언

for k in c1: # c1 딕셔너리의 각 키를 k에 대입하여 반복

if k not in c2: # c2에 k 키가 존재하지 않으면

diff1[k] = c1[k] # c1[k]를 diff1[k]에 저장

elif c1[k] > c2[k]: # c1[k]가 c2[k]보다 크다면

diff1[k] = c1[k] - c2[k] # (c1[k] - c2[k]) 값을 diff2[k]에 저장

# lt2가 더 많은 원소 찾기

diff2 = {} # diff2 딕셔너리 선언

for k in c2: # c2 딕셔너리의 각 키를 k에 대입하여 반복

if k not in c1: # c1에 k 키가 존재하지 않다면

diff2[k] = c2[k] # c2[k]를 diff2[k]에 저장

elif c2[k] > c1[k]: # c2[k]가 c1[k]보다 크면

diff2[k] = c2[k] - c1[k] # (c2[k] - c1[k]) 값을 diff2[k]에 저장

if diff1: # diff1에 값이 있다면 (lt1이 더 많은 원소가 있다면)

# diff1의 원소(k)와 개수(v)를 하나씩 꺼내 반복

for k, v in diff1.items():

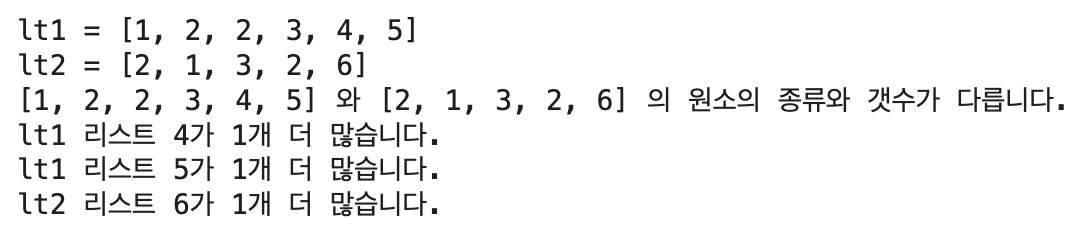
print(f"lt1 리스트 {k}가 {v}개 더 많습니다.")

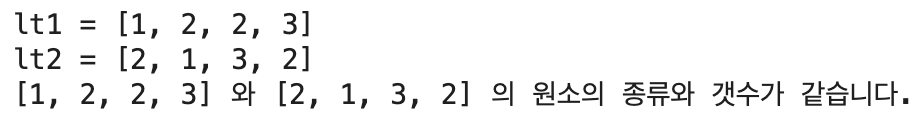
if diff2: # diff2에 값이 있다면 (lt2이 더 많은 원소가 있다면)

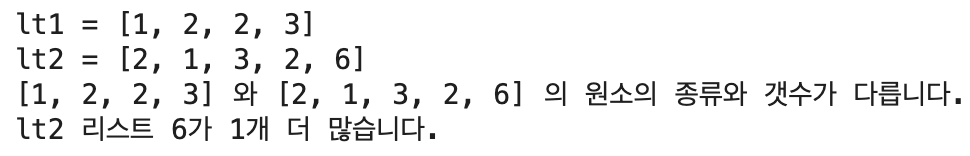
# diff2의 원소(k)와 개수(v)를 하나씩 꺼내 반복

for k, v in diff2.items():

print(f"lt2 리스트 {k}가 {v}개 더 많습니다.")







**문제 PY91-0010**

* 방법2) collections.Counter를 사용 o

# collections 모듈 안에 있는 Counter를 가져온다

from collections import Counter

import ast # ast 모듈을 가져온다

# ast.literal\_eval : 입력된 문자열을 안전하게 리스트로 변환

lt1 = ast.literal\_eval(input("lt1 = ")) # lt1 입력

lt2 = ast.literal\_eval(input("lt2 = ")) # lt2 입력

c1 = Counter(lt1) # c1에 lt1의 원소별 개수를 세어 저장

c2 = Counter(lt2) # c2에 lt2의 원소별 개수를 세어 저장

if c1 == c2: # c1과 c2이 같다면 (원소 종류와 개수가 모두 같음)

print(f"{lt1} 와 {lt2} 의 원소의 종류와 갯수가 같습니다.")

else: # c1, c2이 같지 않으면

print(f"{lt1} 와 {lt2} 의 원소의 종류와 갯수가 다릅니다.")

diff1 = c1 - c2 # (c1 - c2)를 diff1에 저장 (lt1에만 더 많은 원소들)

diff2 = c2 - c1 # (c2 - c1)diff2에 저장 (lt2에만 더 많은 원소들)

if diff1: # diff1에 값이 있다면 (lt1이 더 많은 원소가 있다면)

# diff1의 원소(k)와 개수(v)를 하나씩 꺼내 반복

for k, v in diff1.items():

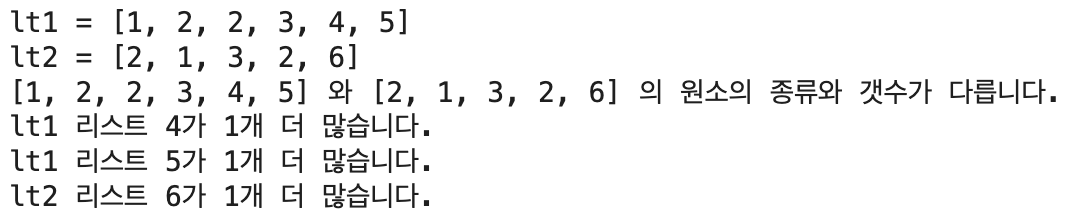
print(f"lt1 리스트 {k}가 {v}개 더 많습니다.")

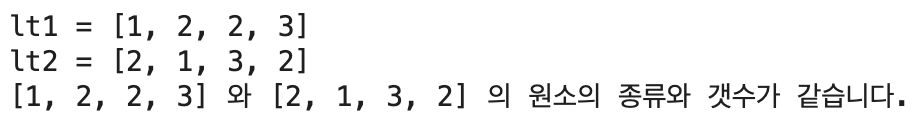
if diff2: # diff2에 값이 있다면 (lt2가 더 많은 원소가 있다면)

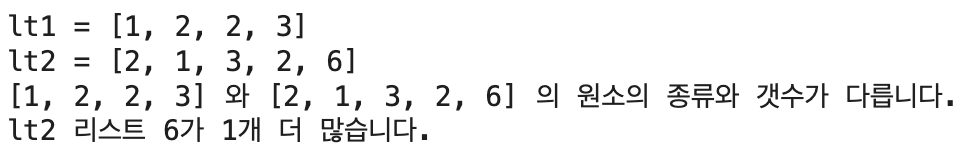
# diff2의 원소(k), 개수(v)를 하나씩 꺼내 반복

for k, v in diff2.items():

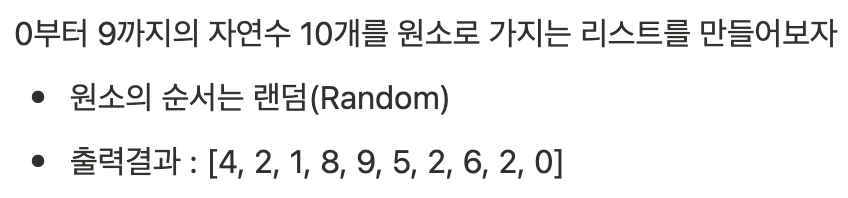
print(f"lt2 리스트 {k}가 {v}개 더 많습니다.")







**문제 PY92-0001**



import random

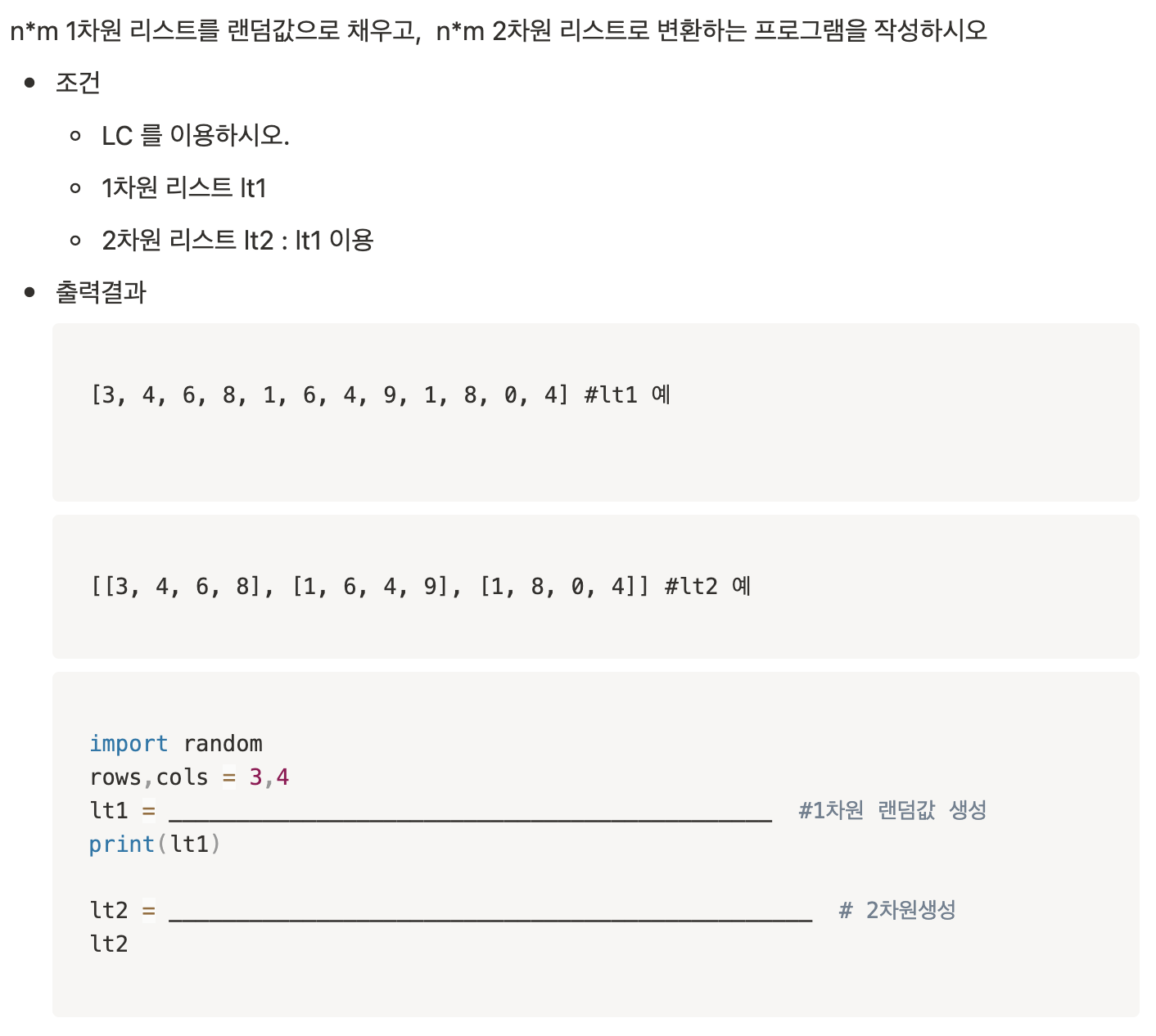
# 0~9까지 랜덤정수를 lt1 리스트 1~10까지 반복하여 저장

lt1 = [random.randint(0, 9) for kk in range(1, 10)]

print(lt1)



**문제 PY92-0002**



import random

rows,cols = 3,4 # 행(rows) 3개, 열(cols) 4개로 설정

# 0~9까지 랜덤 정수를 rows\*cols(12개)만큼 생성하여 lt1 리스트에 저장

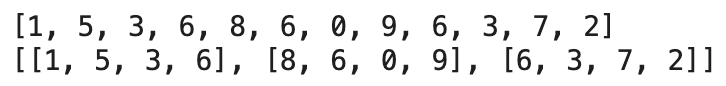
lt1 = [random.randint(0, 9) for kk in range(rows \* cols)] #1차원 랜덤값 생성

print(lt1) # lt1 출력

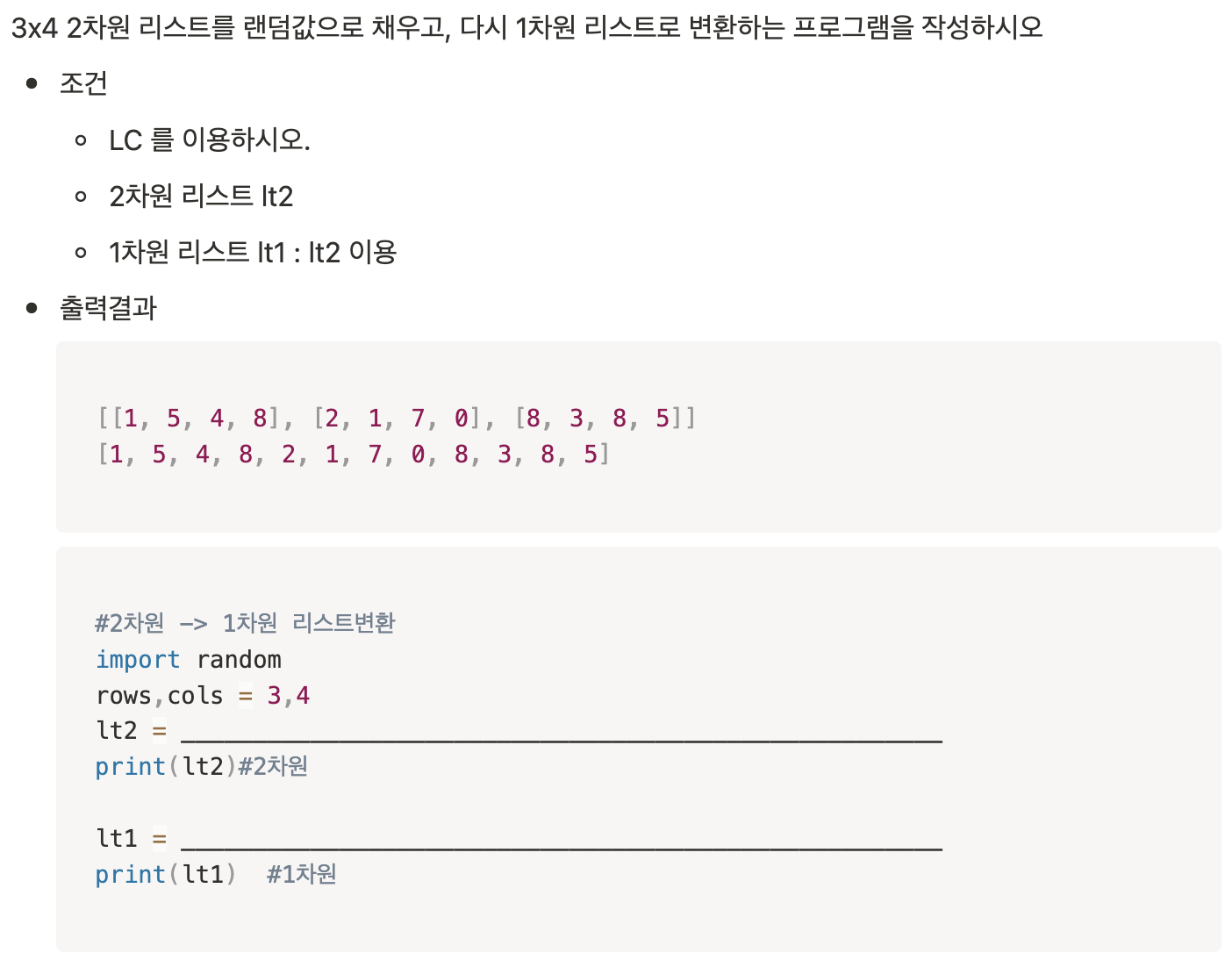
# lt1을 4개씩 끊어서 3행(row)짜리 2차원 리스트 lt2로 변환

lt2 = [lt1[kk\*cols:(kk+1)\*cols] for kk in range(rows)] # 2차원생성

print(lt2) # lt2 출력



**문제 PY92-0003**



#2차원 -> 1차원 리스트변환

import random

rows,cols = 3,4 # 행(rows) 3개, 열(cols) 4개로 설정

# 0~9까지 랜덤 정수를 열(cols) 개수만큼 생성한 리스트를 행(rows) 개수만큼 생성

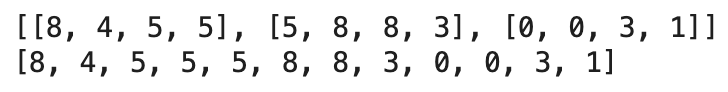
lt2 = [[random.randint(0, 9) for \_ in range(cols)] for \_ in range(rows)]

print(lt2)#2차원

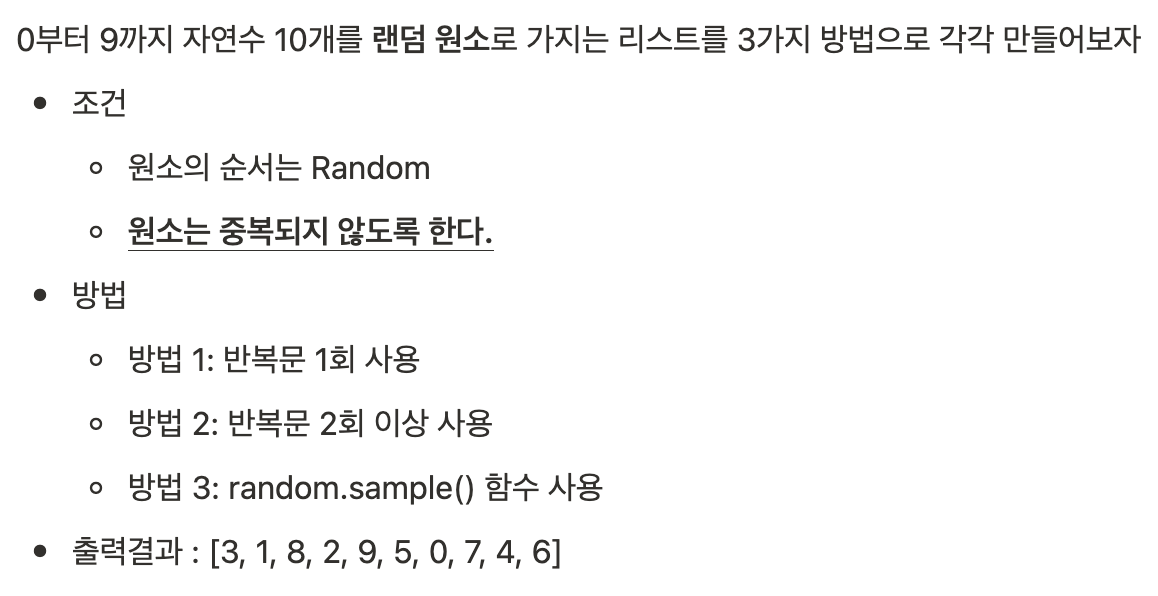
# lt2의 각 행(row)을 돌면서, 각 행 안의 num(원소)을 꺼내 1차원 리스트로 변환

lt1 = [num for row in lt2 for num in row]

print(lt1) #1차원



**문제 PY93-0001**



* 방법1) 반복문 1회 사용

import random

s = set() # 중복을 허용하지 않는 집합(set) 선언

while len(s) < 10: # 집합 s의 원소 개수가 10개가 될 때까지 반복

s.add(random.randint(0, 9)) # 0~9 사이의 랜덤 정수를 중복 없이 추가

nums = list(s) # 집합(set)을 리스트로 변환

random.shuffle(nums) # 리스트 요소들의 순서를 랜덤하게 섞기

print(nums) # 섞은 nums 리스트 출력



**문제 PY93-0001**

* 방법2) 반복문 2회 사용

import random

nums = [] # 빈 리스트 선언

while len(nums) < 10: # nums 리스트의 원소 개수가 10개가 될 때까지 반복

num = random.randint(0, 9) # 0~9 사이의 랜덤 정수를 생성

is\_duplicate = False # 중복 여부를 저장할 변수 (초기값: 중복 없음)

for n in nums: # num 리스트 각 요소를 n에 대입하여 반복

if n == num: # n과 num이 같으면

is\_duplicate = True # 중복 있음 표시

break # 반복문 종료

if not is\_duplicate: # 중복이 아닌 경우

nums.append(num) # nums 리스트에 num 추가

print(nums) # nums 리스트 출력



* 방법3) random.sample() 함수 사용

import random

# 0~9 중에서 10개를 중복 없이 랜덤하게 선택

nums = random.sample(range(10), 10)

print(nums) # 선택된 nums 리스트 출력



**문제 PY93-0002**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**문제 PY93-0002**

* 방법1) randint() 함수를 사용

import random

result = [] # result 리스트 선언

for \_ in range(10): # 10번 반복

# 랜덤정수 0~1을 생성 -> 0이면 'H', 1이면 'T'생성하여 coin에 저장

coin = 'H' if random.randint(0, 1) == 0 else 'T'

dice = random.randint(1, 6) # 1~6까지 랜덤 정수를 생성하여 dice에 저장

result.append((coin, dice)) # (coin, dice) 튜플을 result 리스트에 추가

# 첫 번째 원소를 대괄호 [로 감싸서 출력, 콤마(,) 추가

print(f"[{result[0]},")

for i in range(1, len(result)): # 인덱스 1부터 result 끝까지 반복

if i != len(result) - 1: # 마지막 원소가 아니라면

# (문자, 숫자) 형태로 출력하고 콤마(,) 추가

print(f" ('{result[i][0]}', {result[i][1]}),")

else: # 마지막 원소라면

# (문자, 숫자) 형태로 출력하고 콤마 없이 대괄호로 닫고 출력

print(f" ('{result[i][0]}', {result[i][1]})]", end="")

폰트, 텍스트, 타이포그래피, 화이트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**문제 PY93-0002**

* 방법2) choices() 함수를 사용

import random

result = [] # result 리스트 선언

for \_ in range(10): # 10번 반복

# 'H', 'T'를 랜덤 선택하여 coin에 저장

coin = random.choices(['H', 'T'])[0]

# 1,2,3,4,5,6을 랜덤 선택하여 dice에 저장

dice = random.choices([1, 2, 3, 4, 5, 6])[0]

# (coin, dice) 튜플을 result 리스트에 추가

result.append((coin, dice))

# 첫 번째 원소를 대괄호 [로 감싸서 출력, 콤마(,) 추가

print(f"[{result[0]}, ")

for i in range(1, len(result)): # 인덱스 1부터 result 끝까지 반복

if i != len(result) - 1: # 마지막 원소가 아니라면

# (문자, 숫자) 형태로 출력하고 콤마(,) 추가

print(f" ('{result[i][0]}', {result[i][1]}), ")

else: # 마지막 원소라면

# (문자, 숫자) 형태로 출력하고 콤마 없이 대괄호로 닫고 출력

print(f" ('{result[i][0]}', {result[i][1]})]", end="")

텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**문제 PY93-0003**텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

* 방법1) randint() 함수를 사용

import random

result = set() # result 집합 선언(중복제거)

while len(result) < 10: # 10개가 될 떄까지 반복

# 랜덤정수 0~1을 생성 -> 0이면 'H', 1이면 'T'생성하여 coin에 저장

coin = 'H' if random.randint(0, 1) == 0 else 'T'

dice = random.randint(1, 6) # 1~6까지 랜덤 정수를 생성하여 dice에 저장

result.add((coin, dice)) # (coin, dice) 튜플을 result 집합에 추가

result = list(result) # result를 list로 변환

# 첫 번째 원소를 대괄호 [로 감싸서 출력, 콤마(,) 추가

print(f"[('{result[0][0]}', {result[0][1]}),")

for i in range(1, len(result)): # 인덱스 1부터 result 끝까지 반복

if i != len(result) - 1: # 마지막 원소가 아니라면

# (문자, 숫자) 형태로 출력하고 콤마(,) 추가

print(f" ('{result[i][0]}', {result[i][1]}),")

else: # 마지막 원소라면

# (문자, 숫자) 형태로 출력하고 콤마 없이 대괄호로 닫고 출력

print(f" ('{result[i][0]}', {result[i][1]})]", end="")

텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**문제 PY93-0003**

* 방법2) choices() 함수를 사용

import random

result = set() # result 집합 선언(중복제거)

while len(result) < 10: # 10개가 될 떄까지 반복

# 'H', 'T'를 랜덤 선택하여 coin에 저장

coin = random.choices(['H', 'T'])[0]

# 1,2,3,4,5,6을 랜덤 선택하여 dice에 저장

dice = random.choices([1, 2, 3, 4, 5, 6])[0]

# (coin, dice) 튜플을 result 리스트에 추가

result.add((coin, dice))

result = list(result) # result를 list로 변환

# 첫 번째 원소를 대괄호 [로 감싸서 출력, 콤마(,) 추가

print(f"[('{result[0][0]}', {result[0][1]}),")

for i in range(1, len(result)): # 인덱스 1부터 result 끝까지 반복

if i != len(result) - 1: # 마지막 원소가 아니라면

# (문자, 숫자) 형태로 출력하고 콤마(,) 추가

print(f" ('{result[i][0]}', {result[i][1]}),")

else: # 마지막 원소라면

# (문자, 숫자) 형태로 출력하고 콤마 없이 대괄호로 닫고 출력

print(f" ('{result[i][0]}', {result[i][1]})]", end="")

폰트, 타이포그래피, 화이트, 텍스트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**문제 PY94-0001**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

# mk\_lt 사용자 정의 함수(행, 열, 초기값)

def mk\_lt(row, col, init\_val=0):

# 열 개수만큼 반복하고, 행 개수만큼 반복

# 행(row) 수만큼 반복하면서, 각 행에 대해 열(col) 수만큼 초기값(init\_val)을 채운 리스트를 생성

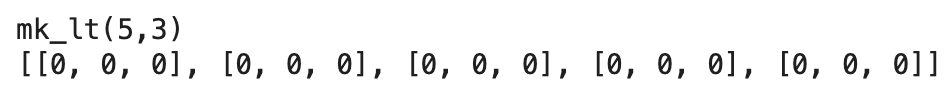
return [[init\_val for \_ in range(col)] for \_ in range(row)]

user\_input = input() # 사용자로부터 입력값을 문자열로 받음

print(eval(user\_input)) # 입력받은 문자열을 실제 코드처럼 실행해서 결과 출력

폰트, 텍스트, 화이트, 타이포그래피이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.



**문제 PY94-0002**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**문제 PY94-0002**

# lt\_reshape 사용자 정의 함수 선언(리스트, 행 수, 열 수, 펼침 여부)

def lt\_reshape(lst, row=None, col=None, flatten=False):

if flatten: # flatten이 True라면 (리스트를 펼친다)

result = [] # result 리스트 선언

# lst 안의 각 서브리스트를 sublist로 가져와서

for sublist in lst:

# sublist 안의 요소들을 result 리스트에 이어붙인다

result.extend(sublist)

return result # 1차원으로 펼친 result를 반환

else: # flatten이 False라면 (리스트 모양을 (row, col)로 바꾼다)

# 만약 lst[0]이 리스트라면 (2차원 리스트라면)

if isinstance(lst[0], list):

flattened = [] # flattened 리스트 선언

# lst 안의 각 서브리스트를 sublist로 가져와서

for sublist in lst:

# sublist 안의 요소들을 flattened에 이어붙인다

flattened.extend(sublist)

# lst를 펼친 리스트로 교체

lst = flattened

result = [] # result 리스트 선언

for r in range(row): # 행(row) 수만큼 반복

# r번째 행에 해당하는 열(col)만큼 슬라이싱하여 추가

result.append(lst[r\*col:(r+1)\*col])

return result # 2차원으로 재구성한 result 반환

rows, cols = 3,4

lt1 = list(range(rows\*cols)) # 0부터 11까지 숫자를 담은 리스트 생성

print(lt1) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

# 3행 4열로 변환

lt2 = lt\_reshape(lt1,3,4) # [[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]]

print(lt2)

print()

# 2행 6열로 변환

lt2 = lt\_reshape(lt1,2,6) # [[0, 1, 2, 3, 4, 5], [6, 7, 8, 9, 10, 11]]

print(lt2)

print()

# 4행 3열로 다시 변환

lt3 = lt\_reshape(lt2,4,3)

print(lt3) #[[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11]]

# 다시 1차원 리스트로 펼침

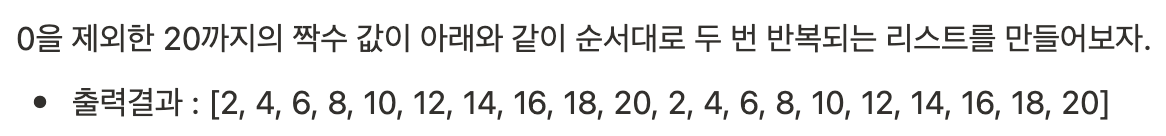
lt4 = lt\_reshape(lt3,flatten=True)

print(lt4) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

텍스트, 폰트, 화이트, 타이포그래피이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**문제 PY95-0001**



result = [] # result 리스트 선언

for \_ in range(2): # 전체 과정을 2번 반복

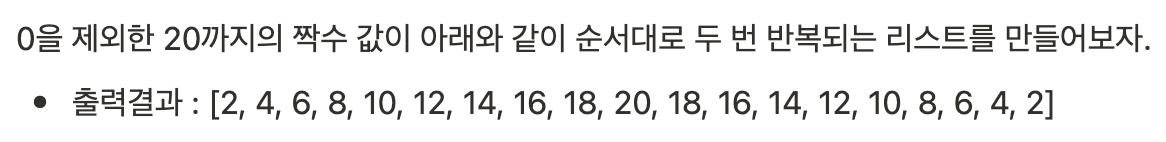
for kk in range(2, 21, 2): # 1부터 20까지 2씩 증가

result.append(kk) # result 리스트에 추가

print(result) # 최종 result 리스트 출력



**문제 PY95-0002**



result = [] # result 리스트 선언

for kk in range(2, 21, 2): # 2부터 20까지 2씩 증가

result.append(kk) # result 리스트에 추가

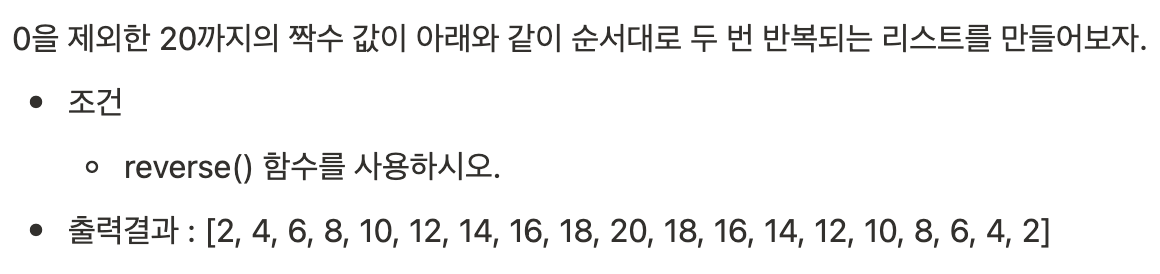
for kk in range(18, 1, -2): # 18부터 1까지 2씩 감소

result.append(kk) # result 리스트에 추가

print(result) # 최종 result 리스트 출력



**문제 PY95-0003**



result = [] # result 리스트 선언

for kk in range(2, 21, 2): # 2부터 20까지 2씩 증가

result.append(kk) # result 리스트에 추가

temp = result[:-1] # result 리스트 마지막 요소(20) 제외한 복사본 만들기

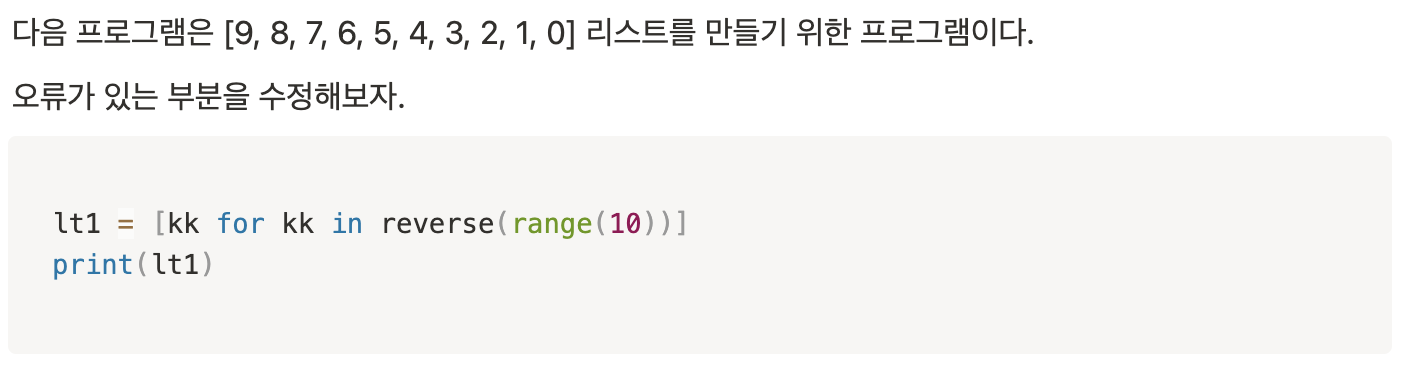
temp.reverse() # temp를 뒤집기

result.extend(temp) # result에 뒤집어진 temp를 이어붙이기

print(result) # 최종 result 리스트 출력



**문제 PY95-0004**

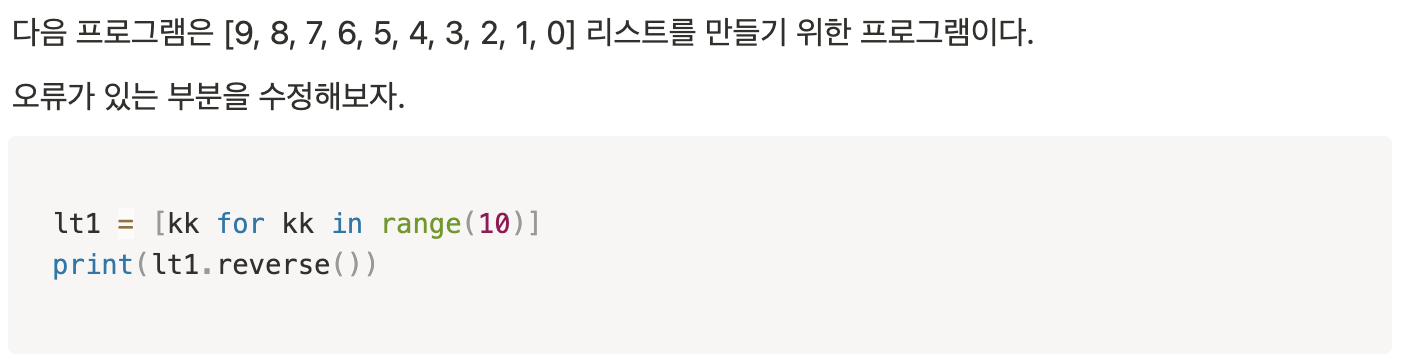


lt1 = [kk for kk in reversed(range(10))] # 0부터 10까지 lt1에 뒤집어서 저장

print(lt1) # lt1 출력



**문제 PY95-0005**



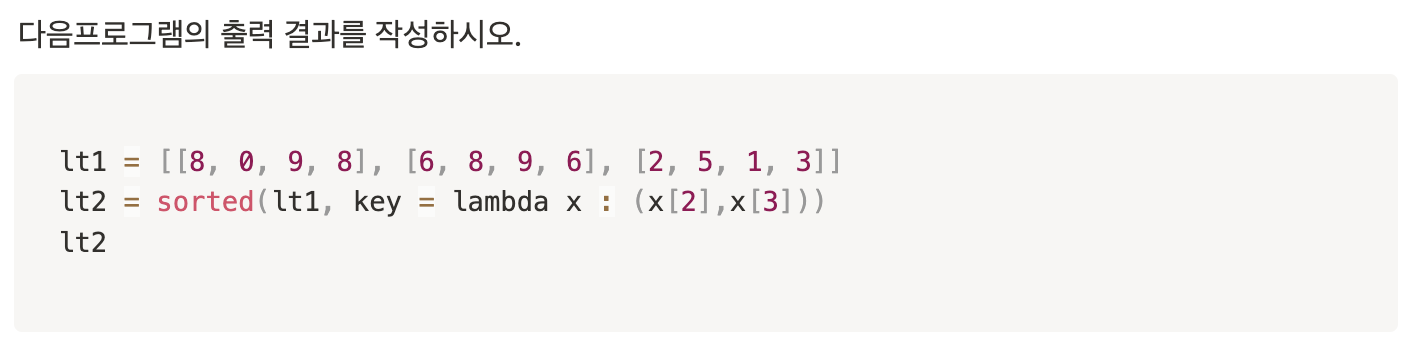
lt1 = [kk for kk in range(10)] # 1부터 10까지 lt1에 저장

lt1.reverse() # lt1 뒤집기

print(lt1) # lt1 출력



**문제 PY96-0001**



lt1 = [[8, 0, 9, 8], [6, 8, 9, 6], [2, 5, 1, 3]]

lt2 = sorted(lt1, key = lambda x : (x[2],x[3]))

lt2

