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

2개의 문자열을 입력받고, 두 문자의 값을 바꾸는 코드를 완성하시오.

```
aa,bb = input().split()

print(aa,bb)

### 코드 추가 ######


####################
print(aa,bb)
```

- 조건
  - Trinket3.x
  - 공백을 중심으로 문자열을 구분한다.
  - 반복문 x
- 입출력 예
  - 입출력 결과와 똑같아야 합니다. (공백, 대소문자, 형식을 정확하게 맞춰주세요)
  - 입력 안내 메시지는 생략해주세요

입력1	출력1
3 6	3 6 6 3
입력2	출력2
Hello Python	Python Hello

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

print(aa,bb) # aa, bb 를 출력

### 코드 추가 ######
aa, bb = bb, aa # aa 를 bb 에 저장, bb 를 aa 에 저장
####################
print(aa,bb) # aa, bb 를 출력
```

```
3 6
3 6
6 3
```

```
Hello Python
Hello Python
Python Hello
```

문제 PY91-0002

다음 코드는 a, b 숫자를 바꾸는 swap 기능 코드의 일부이다. 사용자정의함수를 모두 구현하시오.

- 함수명 myswap1(), myswap2() myswap3()

```
a,b = 3,4
print(a,b)
a,b = myswap1(a,b)
print(a,b)
```

myswap1() 호출 예

```
a,b = 3,4
print(a,b)
myswap2(a,b)
print(a,b)
```

myswap2() 호출 예

```
a,b = 3,4
print(a,b)
myswap3()
print(a,b)
```

myswap3() 호출 예

- 입출력 예

입력1	출력1
없음	3 4 4 3

문제 PY91-0002

- myswap1()

```
def myswap1(a, b): # myswap1 사용자 정의 함수, 매개변수 a, b 를 받는다
  return b, a # a 와 b 의 값을 바꾸어 반환

a, b = 3, 4 # a 에는 3 을, b 에는 4 를 저장
print(a,b) # 바꾸기 전의 a, b 를 출력
a,b = myswap1(a,b) # myswap1 함수를 호출하여 a, b 값을 교환
print(a,b) # 바꾼 후의 a, b 를 출력
```

```
3 4
4 3
```

- myswap2()

```
def myswap2(a, b):   # myswap2 사용자 정의 함수, 매개변수 a, b 를 받는다
  a, b = b, a # a 를 b 에 저장, b 를 a 에 저장
  print(a, b) # 바뀐 a, b 를 출력

a, b = 3, 4 # a 에는 3 을, b 에는 4 를 저장
print(a,b) # 바꾸기 전의 a, b 를 출력
myswap2(a,b) # myswap2 함수를 호출하여 a, b 값을 교환
print(a,b) # 바꾼 후의 a, b 를 출력
```

```
3 4
4 3
3 4
```

- myswap3()

```
def myswap3(): # myswap3 사용자 정의 함수
    global a, b # 전역변수 a, b 선언
    a, b = b, a # a 에 b 를 저장, b 에 a 를 저장

a, b = 3, 4 # a 에 3 을 저장, b 에 4 를 저장
print(a, b) # a, b 를 출력
myswap3() # myswap3 함수를 호출
print(a, b) # 바꾼 후의 a, b 를 출력
```

```
3 4
4 3
```

문제 PY91-0003

다음 코드는 리스트 원소의 swap 기능 코드 일부이다. 사용자정의함수를 모두 구현하시오.

- 함수명 myswap4()
- 조건

```
lt= [3,4]

print(lt)
myswap4()
print(lt)
```

myswap4() 호출 예

- 입출력 예

입력1	출력1
없음	3 4 4 3

```
def myswap4():  # myswap3 사용자 정의 함수
  # lt 0 번째를 lt 1 번째에 저장, lt 1 번째를 lt 0 번째를 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
```

```
[3, 4]
[4, 3]
```

문제 PY91-0004

다음 프로그램은 함수 바깥쪽에서 만든 리스트의 값을 출력하는 프로그램이다.

- 조건
  - 출력을 아래와 같이 하도록 오류 발생원인을 찾고 해결을 해보자.

```
lt1 = ['AAA','BBB','AAA','CCC']

def delete_List():
  print(lt1)
  lt1 = ['DDD','BBB','FFF','CCC']
  print(lt1)

delete_List()
```

- 출력 예

```
['AAA', 'BBB', 'AAA', 'CCC']
['DDD', 'BBB', 'FFF', 'CCC']
```

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

def delete_List(): # delete_List 사용자 정의 함수
  global lt1 # lt1 을 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
```

```
['AAA', 'BBB', 'AAA', 'CCC']
['DDD', 'BBB', 'FFF', 'CCC']
```

문제 PY91-0005

다음 프로그램은 lt1의 내용을 lt2 에 복사하는 프로그램이다.

3개의 print() 결과가 일치하도록 코드를 완성하시오..

- 조건
  - lt2[0][1] 의 값을 변경했을 때, lt1 의 출력이 변경되지 않도록 하기 위한 코드를 작성하시오
  - 방법1,2 각각 작성하시오.
    - 방법1) deepcopy 사용 (모듈추가 코드 포함)
    - 방법2) LC, 슬라이스연산자 사용

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

## 코드 추가


####################
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]
```

- 입 출력 예

입력	출력
없음	[[3, 4], 5, 6] [] [[3, 4], 5, 6] [[3, 4], 5, 6] [[3, 4], 5, 6] [[3, 0], 5, 6]

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) # lt1 의 깊은 복사본을 lt2 에 저장 (내부 리스트까지 복사됨)
######################

print(lt1,lt2) #[[3, 4], 5, 6] [[3, 4], 5, 6]
lt2[0][1] = 0 # lt2 내부 리스트의 값 변경 (lt1 에는 영향 없음)

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

```
[[3, 4], 5, 6] []
[[3, 4], 5, 6] [[3, 4], 5, 6]
[[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. lt1 의 각 요소를 kk 로 하나씩 꺼낸다.
# 2. kk 가 리스트 타입이면 (type(kk) == list) -> kk[:]로 복사해서 사용한다.
# 3. list 타입이 아니면 (예: 숫자 같은 거) -> 그냥 kk 를 그대로 사용한다.
# 4. 이렇게 만들어진 값들을 모아서 lt2 를 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]
```

```
[[3, 4], 5, 6] []
[[3, 4], 5, 6] [[3, 4], 5, 6]
[[3, 4], 5, 6] [[3, 0], 5, 6]
```

문제 PY91-0010

두 개의 리스트가 주어졌을 때, 두 리스트에 포함된 **원소의 종류와 각각의 개수가 동일한지** 판단하는 프로그램을 작성하시오.

두 리스트가 다를 경우, 어떤 원소가 얼마나 다른지까지 함께 출력하는 기능을 구현하시오.

조건

- 두 리스트는 정수로 이루어져 있다.
- 리스트 내 원소의 **종류와 개수가 모두 같아야** "같다"고 판단한다.
- 만약 다르다면,
  - 어떤 원소가 **lt1에만 더 많은지**, 혹은
  - 어떤 원소가 **lt2에만 더 많은지**를 알려줘야 한다.
- 방법1) collections.Counter 를 사용 x
- 방법2) collections.Counter 를 사용 o

입출력 예시

입력	출력결과
lt1 = [1, 2, 2, 3, 4, 5] lt2 = [2, 1, 3, 2, 6]	[1, 2, 2, 3, 4, 5] 와 [2, 1, 3, 2, 6] 의 원소의 종류와 갯수가 다릅니다. lt1 리스트 4 원소가 1 개 더 많습니다. lt1 리스트 5 원소가 1 개 더 많습니다. lt2 리스트 6 원소가 1 개 더 많습니다.
lt1 = [1, 2, 2, 3] lt2 = [2, 1, 3, 2]	[1, 2, 2, 3] 와 [2, 1, 3, 2] 의 원소의 종류와 갯수가 같습니다.
lt1 = [1, 2, 2, 3] lt2 = [2, 1, 3, 2, 6]	[1, 2, 2, 3] 와 [2, 1, 3, 2, 6] 의 원소의 종류와 갯수가 다릅니다. lt2 리스트 6 원소가 1 개 더 많습니다.

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

# lt1 에서 원소 개수 세기
c1 = {} # c1 딕셔너리 선언
for num in lt1: # lt1 의 각 문자를 num 에 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 1 을 저장한다 (처음 등장)

# lt2 에서 원소 개수 세기
c2 = {} # c2 딕셔너리 선언
for num in lt2: # lt2 의 각 문자를 num 에 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 1 을 저장한다 (처음 등장)

# 비교
if c1 == c2: # c1 과 c2 가 같다면 (원소 종류와 개수가 모두 같음)
  print(f"{lt1} 와 {lt2} 의 원소의 종류와 갯수가 같습니다.")
else: # c1 과 c2 가 다르면
  print(f"{lt1} 와 {lt2} 의 원소의 종류와 갯수가 다릅니다.")

  # lt1 이 더 많은 원소 찾기
  diff1 = {} # diff1 딕셔너리 선언
  for k in c1: # c1 딕셔너리의 각 키를 k 에 대입하여 반복
    if k not in c2: # c2 에 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

  # lt2 가 더 많은 원소 찾기
  diff2 = {} # diff2 딕셔너리 선언
  for k in c2: # c2 딕셔너리의 각 키를 k 에 대입하여 반복
```

```
    if k not in c1: # c1 에 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: # diff1 에 값이 있다면 (lt1 이 더 많은 원소가 있다면)
    # diff1 의 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: # diff2 에 값이 있다면 (lt2 이 더 많은 원소가 있다면)
    # diff2 의 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.")
```

```
lt1 = [1, 2, 2, 3, 4, 5]
lt2 = [2, 1, 3, 2, 6]
[1, 2, 2, 3, 4, 5] 와 [2, 1, 3, 2, 6] 의 원소의 종류와 갯수가 다릅니다.
lt1 리스트 4가 1개 더 많습니다.
lt1 리스트 5가 1개 더 많습니다.
lt2 리스트 6가 1개 더 많습니다.

lt1 = [1, 2, 2, 3]
lt2 = [2, 1, 3, 2]
[1, 2, 2, 3] 와 [2, 1, 3, 2] 의 원소의 종류와 갯수가 같습니다.

lt1 = [1, 2, 2, 3]
lt2 = [2, 1, 3, 2, 6]
[1, 2, 2, 3] 와 [2, 1, 3, 2, 6] 의 원소의 종류와 갯수가 다릅니다.
lt2 리스트 6가 1개 더 많습니다.
```

문제 PY91-0010

- 방법2) collections.Counter를 사용 o

```
# collections 모듈 안에 있는 Counter 를 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) # c1 에 lt1 의 원소별 개수를 세어 저장
c2 = Counter(lt2) # c2 에 lt2 의 원소별 개수를 세어 저장

if c1 == c2: # c1 과 c2 이 같다면 (원소 종류와 개수가 모두 같음)
  print(f"{lt1} 와 {lt2} 의 원소의 종류와 갯수가 같습니다.")
else: # c1, c2 이 같지 않으면
  print(f"{lt1} 와 {lt2} 의 원소의 종류와 갯수가 다릅니다.")

  diff1 = c1 - c2 # (c1 - c2)를 diff1 에 저장 (lt1 에만 더 많은 원소들)
  diff2 = c2 - c1 # (c2 - c1)diff2 에 저장 (lt2 에만 더 많은 원소들)

  if diff1: # diff1 에 값이 있다면 (lt1 이 더 많은 원소가 있다면)
    # diff1 의 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: # diff2 에 값이 있다면 (lt2 가 더 많은 원소가 있다면)
    # diff2 의 원소(k), 개수(v)를 하나씩 꺼내 반복
    for k, v in diff2.items():
      print(f"lt2 리스트 {k}가 {v}개 더 많습니다.")
```

```
lt1 = [1, 2, 2, 3, 4, 5]
lt2 = [2, 1, 3, 2, 6]
[1, 2, 2, 3, 4, 5] 와 [2, 1, 3, 2, 6] 의 원소의 종류와 갯수가 다릅니다.
lt1 리스트 4가 1개 더 많습니다.
lt1 리스트 5가 1개 더 많습니다.
lt2 리스트 6가 1개 더 많습니다.
```

```
lt1 = [1, 2, 2, 3]
lt2 = [2, 1, 3, 2]
[1, 2, 2, 3] 와 [2, 1, 3, 2] 의 원소의 종류와 갯수가 같습니다.
```

```
lt1 = [1, 2, 2, 3]
lt2 = [2, 1, 3, 2, 6]
[1, 2, 2, 3] 와 [2, 1, 3, 2, 6] 의 원소의 종류와 갯수가 다릅니다.
lt2 리스트 6가 1개 더 많습니다.
```

문제 PY92-0001

0부터 9까지의 자연수 10개를 원소로 가지는 리스트를 만들어보자

- 원소의 순서는 랜덤(Random)
- 출력결과 : [4, 2, 1, 8, 9, 5, 2, 6, 2, 0]

```
import random

# 0~9 까지 랜덤정수를 lt1 리스트 1~10 까지 반복하여 저장
lt1 = [random.randint(0, 9) for kk in range(1, 10)]
print(lt1)
```

```
[2, 8, 9, 4, 1, 3, 7, 1, 8]
```

문제 PY92-0002

n*m 1차원 리스트를 랜덤값으로 채우고, n*m 2차원 리스트로 변환하는 프로그램을 작성하시오

- 조건
  - LC 를 이용하시오.
  - 1차원 리스트 lt1
  - 2차원 리스트 lt2 : lt1 이용
- 출력결과

```
[3, 4, 6, 8, 1, 6, 4, 9, 1, 8, 0, 4] #lt1 예
```

```
[[3, 4, 6, 8], [1, 6, 4, 9], [1, 8, 0, 4]] #lt2 예
```

```
import random
rows,cols = 3,4
lt1 = ____________________________________________________  #1차원 랜덤값 생성
print(lt1)

lt2 = ____________________________________________________  # 2차원생성
lt2
```

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

# 0~9 까지 랜덤 정수를 rows*cols(12 개)만큼 생성하여 lt1 리스트에 저장
lt1 = [random.randint(0, 9) for kk in range(rows * cols)] #1 차원 랜덤값 생성
print(lt1) # lt1 출력

# lt1 을 4 개씩 끊어서 3 행(row)짜리 2 차원 리스트 lt2 로 변환
lt2 = [lt1[kk*cols:(kk+1)*cols] for kk in range(rows)] # 2 차원생성
print(lt2) # lt2 출력
```

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

문제 PY92-0003

3x4 2차원 리스트를 랜덤값으로 채우고, 다시 1차원 리스트로 변환하는 프로그램을 작성하시오

- 조건
  - LC 를 이용하시오.
  - 2차원 리스트 lt2
  - 1차원 리스트 lt1 : lt2 이용
- 출력결과

```
[[1, 5, 4, 8], [2, 1, 7, 0], [8, 3, 8, 5]]
[1, 5, 4, 8, 2, 1, 7, 0, 8, 3, 8, 5]
```

```
#2차원 -> 1차원 리스트변환
import random
rows,cols = 3,4
lt2 = ____________________________________________________
print(lt2)#2차원

lt1 = ____________________________________________________
print(lt1)  #1차원
```

```
#2 차원 -> 1 차원 리스트변환
import random

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

# 0~9 까지 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)#2 차원

# lt2 의 각 행(row)을 돌면서, 각 행 안의 num(원소)을 꺼내 1 차원 리스트로 변환
lt1 = [num for row in lt2 for num in row]
print(lt1)  #1 차원
```

```
[[8, 4, 5, 5], [5, 8, 8, 3], [0, 0, 3, 1]]
[8, 4, 5, 5, 5, 8, 8, 3, 0, 0, 3, 1]
```

문제 PY93-0001

0부터 9까지 자연수 10개를 **랜덤 원소**로 가지는 리스트를 3가지 방법으로 각각 만들어보자

- 조건
  - 원소의 순서는 Random
  - **원소는 중복되지 않도록 한다.**
- 방법
  - 방법 1: 반복문 1회 사용
  - 방법 2: 반복문 2회 이상 사용
  - 방법 3: random.sample() 함수 사용
- 출력결과 : [3, 1, 8, 2, 9, 5, 0, 7, 4, 6]

- 방법1) 반복문 1회 사용

```
import random

s = set() # 중복을 허용하지 않는 집합(set) 선언
while len(s) < 10:  # 집합 s 의 원소 개수가 10 개가 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
```

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

문제 PY93-0001

- 방법2) 반복문 2회 사용

```
import random

nums = [] # 빈 리스트 선언

while len(nums) < 10: # nums 리스트의 원소 개수가 10 개가 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 n 에 대입하여 반복
    if n == num: # n 과 num 이 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
```

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

- 방법3) random.sample() 함수 사용

```
import random

# 0~9 중에서 10 개를 중복 없이 랜덤하게 선택
nums = random.sample(range(10), 10)
print(nums) # 선택된 nums 리스트 출력
```

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

문제 PY93-0002

주사위 1개와 동전1개를 10번 중복가능 추출한 결과를 출력하시오

- 조건
  - 방법1) randint() 함수를 사용
  - 방법2) choices() 함수를 사용
- 중복가능
- 출력예

```
[('H', 6),
 ('H', 5),
 ('H', 4),
 ('T', 6),
 ('T', 4),
 ('H', 4),
 ('H', 4),
 ('T', 1),
 ('T', 6),
 ('T', 4)]
```

문제 PY93-0002

- 방법1) randint() 함수를 사용

```
import random

result = [] # result 리스트 선언

for _ in range(10): # 10 번 반복
  # 랜덤정수 0~1 을 생성 -> 0 이면 'H', 1 이면 'T'생성하여 coin 에 저장
  coin = 'H' if random.randint(0, 1) == 0 else 'T'
  dice = random.randint(1, 6) # 1~6 까지 랜덤 정수를 생성하여 dice 에 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 1 부터 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="")
```

```
[('H', 2),
 ('T', 2),
 ('H', 3),
 ('T', 1),
 ('T', 6),
 ('T', 3),
 ('H', 4),
 ('H', 4),
 ('H', 5),
 ('T', 3)]
```

문제 PY93-0002

- 방법2) choices() 함수를 사용

```
import random

result = [] # result 리스트 선언

for _ in range(10): # 10 번 반복
  # 'H', 'T'를 랜덤 선택하여 coin 에 저장
  coin = random.choices(['H', 'T'])[0]
  # 1,2,3,4,5,6 을 랜덤 선택하여 dice 에 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 1 부터 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="")
```

```
[('H', 5),
 ('H', 2),
 ('T', 1),
 ('T', 4),
 ('H', 2),
 ('H', 3),
 ('T', 3),
 ('H', 5),
 ('H', 3),
 ('H', 4)]
```

문제 PY93-0003

주사위 1개와 동전1개를 10번 중복불가 추출한 결과를 출력하시오

- 조건
  - 방법1) randint() 함수를 사용
  - 방법2) choices() 함수를 사용
  - **중복불가**
- 출력예

```
[('H', 3),
 ('H', 6),
 ('T', 4),
 ('H', 5),
 ('T', 6),
 ('T', 2),
 ('T', 1),
 ('H', 2),
 ('H', 1),
 ('T', 5),
 ('H', 4),
 ('T', 3)]
```

- 방법1) randint() 함수를 사용

```
import random

result = set() # result 집합 선언(중복제거)

while len(result) < 10: # 10 개가 될 때까지 반복
  # 랜덤정수 0~1 을 생성 -> 0 이면 'H', 1 이면 'T'생성하여 coin 에 저장
  coin = 'H' if random.randint(0, 1) == 0 else 'T'
  dice = random.randint(1, 6) # 1~6 까지 랜덤 정수를 생성하여 dice 에 저장
  result.add((coin, dice)) # (coin, dice) 튜플을 result 집합에 추가

result = list(result) # result 를 list 로 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 1 부터 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="")
```

```
[('H', 1),
 ('H', 3),
 ('T', 6),
 ('H', 6),
 ('T', 2),
 ('H', 2),
 ('T', 5),
 ('H', 5),
 ('T', 4),
 ('T', 1)]
```

문제 PY93-0003

- 방법2) choices() 함수를 사용

```
import random

result = set() # result 집합 선언(중복제거)

while len(result) < 10: # 10 개가 될 때까지 반복
  # 'H', 'T'를 랜덤 선택하여 coin 에 저장
  coin = random.choices(['H', 'T'])[0]
  # 1,2,3,4,5,6 을 랜덤 선택하여 dice 에 저장
  dice = random.choices([1, 2, 3, 4, 5, 6])[0]
  # (coin, dice) 튜플을 result 리스트에 추가
  result.add((coin, dice))

result = list(result) # result 를 list 로 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 1 부터 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="")
```

```
[('H', 1),
 ('H', 4),
 ('T', 3),
 ('H', 3),
 ('T', 6),
 ('T', 2),
 ('H', 2),
 ('T', 5),
 ('H', 5),
 ('T', 1)]
```

문제 PY94-0001

다음과 같이 행과 열을 입력받고, 2차원 리스트를 만드는 함수 mk_lt() 를 만드시오.

- 조건
  - mk_lt(행, 열, 초기값)
- 출력결과 :

```
mk_lt(2,3,1)
[[1, 1, 1], [1, 1, 1]]

mk_lt(5,3)
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

```
# 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
```

```
mk_lt(2,3,1)
[[1, 1, 1], [1, 1, 1]]

mk_lt(5,3)
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

문제 PY94-0002

다음과 같이 리스트, 행, 열을 정보를 입력받고, 리스트의 모양을 바꿀수 있는 함수를 만드시오.

- 조건
  - lt_reshape(리스트, 행, 열, 평탄화)
- 출력결과 :

```
rows, cols = 3,4
lt1 = list(range(rows*cols))

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

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

lt3 = lt_reshape(lt2,4,3)
print(lt3) #[[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11]]

lt4 = lt_reshape(lt3,flatten=True)
print(lt4) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
[[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]]

[[0, 1, 2, 3, 4, 5], [6, 7, 8, 9, 10, 11]]

[[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11]]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

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: # flatten 이 True 라면 (리스트를 펼친다)
    result = [] # result 리스트 선언
    # lst 안의 각 서브리스트를 sublist 로 가져와서
    for sublist in lst:
      # sublist 안의 요소들을 result 리스트에 이어붙인다
      result.extend(sublist)
    return result # 1 차원으로 펼친 result 를 반환

  else: # flatten 이 False 라면 (리스트 모양을 (row, col)로 바꾼다)
    # 만약 lst[0]이 리스트라면 (2 차원 리스트라면)
    if isinstance(lst[0], list):
      flattened = []  # flattened 리스트 선언
      # lst 안의 각 서브리스트를 sublist 로 가져와서
      for sublist in lst:
        # sublist 안의 요소들을 flattened 에 이어붙인다
        flattened.extend(sublist)
      # lst 를 펼친 리스트로 교체
      lst = flattened

    result = [] # result 리스트 선언
    for r in range(row):  # 행(row) 수만큼 반복
      # r 번째 행에 해당하는 열(col)만큼 슬라이싱하여 추가
      result.append(lst[r*col:(r+1)*col])
    return result # 2 차원으로 재구성한 result 반환

rows, cols = 3,4
lt1 = list(range(rows*cols)) # 0 부터 11 까지 숫자를 담은 리스트 생성

print(lt1)     # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
# 3 행 4 열로 변환
lt2 = lt_reshape(lt1,3,4) # [[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]]
print(lt2)
print()

# 2 행 6 열로 변환
lt2 = lt_reshape(lt1,2,6) # [[0, 1, 2, 3, 4, 5], [6, 7, 8, 9, 10, 11]]
print(lt2)
print()
```

```
# 4 행 3 열로 다시 변환
lt3 = lt_reshape(lt2,4,3)
print(lt3) #[[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11]]

# 다시 1 차원 리스트로 펼침
lt4 = lt_reshape(lt3,flatten=True)
print(lt4) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
[[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]]

[[0, 1, 2, 3, 4, 5], [6, 7, 8, 9, 10, 11]]

[[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11]]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

문제 PY95-0001

0을 제외한 20까지의 짝수 값이 아래와 같이 순서대로 두 번 반복되는 리스트를 만들어보자.

- 출력결과 : [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

```
result = [] # result 리스트 선언

for _ in range(2):   # 전체 과정을 2 번 반복
  for kk in range(2, 21, 2): # 1 부터 20 까지 2 씩 증가
    result.append(kk) # result 리스트에 추가

print(result) # 최종 result 리스트 출력
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

문제 PY95-0002

0을 제외한 20까지의 짝수 값이 아래와 같이 순서대로 두 번 반복되는 리스트를 만들어보자.

- 출력결과 : [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 18, 16, 14, 12, 10, 8, 6, 4, 2]

```
result = [] # result 리스트 선언

for kk in range(2, 21, 2): # 2 부터 20 까지 2 씩 증가
  result.append(kk) # result 리스트에 추가
for kk in range(18, 1, -2): # 18 부터 1 까지 2 씩 감소
  result.append(kk) # result 리스트에 추가

print(result) # 최종 result 리스트 출력
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 18, 16, 14, 12, 10, 8, 6, 4, 2]
```

문제 PY95-0003

0을 제외한 20까지의 짝수 값이 아래와 같이 순서대로 두 번 반복되는 리스트를 만들어보자.

- 조건
  - reverse() 함수를 사용하시오.
- 출력결과 : [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 18, 16, 14, 12, 10, 8, 6, 4, 2]

```
result = [] # result 리스트 선언

for kk in range(2, 21, 2): # 2 부터 20 까지 2 씩 증가
  result.append(kk) # result 리스트에 추가

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

result.extend(temp) # result 에 뒤집어진 temp 를 이어붙이기

print(result) # 최종 result 리스트 출력
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 18, 16, 14, 12, 10, 8, 6, 4, 2]
```

문제 PY95-0004

다음 프로그램은 [9, 8, 7, 6, 5, 4, 3, 2, 1, 0] 리스트를 만들기 위한 프로그램이다.

오류가 있는 부분을 수정해보자.

```
lt1 = [kk for kk in reverse(range(10))]
print(lt1)
```

```
lt1 = [kk for kk in reversed(range(10))] # 0 부터 10 까지 lt1 에 뒤집어서 저장
print(lt1) # lt1 출력
```

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

문제 PY95-0005

다음 프로그램은 [9, 8, 7, 6, 5, 4, 3, 2, 1, 0] 리스트를 만들기 위한 프로그램이다.

오류가 있는 부분을 수정해보자.

```
lt1 = [kk for kk in range(10)]
print(lt1.reverse())
```

```
lt1 = [kk for kk in range(10)]   # 1 부터 10 까지 lt1 에 저장
lt1.reverse() # lt1 뒤집기
print(lt1)   # lt1 출력
```

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

문제 PY96-0001

다음프로그램의 출력 결과를 작성하시오.

```
lt1 = [[8, 0, 9, 8], [6, 8, 9, 6], [2, 5, 1, 3]]
lt2 = sorted(lt1, key = lambda x : (x[2],x[3]))
lt2
```

```
lt1 = [[8, 0, 9, 8], [6, 8, 9, 6], [2, 5, 1, 3]]
lt2 = sorted(lt1, key = lambda x : (x[2],x[3]))
lt2
```

```
[[2, 5, 1, 3], [6, 8, 9, 6], [8, 0, 9, 8]]
```