REPORT

[과제 : 파이썬기초 문제지(4) 문제 풀이]

과 목 명	파이썬과학프로그래밍기초
교 수 명	김 병 정
학 번	20237107
작 성 자	하 태 영
제 출 일	2025.04.15

한림대학교

문제 PY32-0008

map()함수를 이용해서 입력받은 원소들의 제곱을 구하는 프로그램을 작성하시오.

- 조건
    - map()함수를 2 번이상 사용
    - 입력값이 2 와 8 사이의 값만 제곱한다. (2,8 포함)
        - 사용자정의함수를 map()함수에 이용한다

```
def square_if_valid(x): # 사용자 정의 함수 (조건에 따라 제곱)
  return x**2 if 2 <= x <= 8 else x # 2이상 8이하라면 제곱하고 그렇지 않으면 원래 숫자 그대로 반환.


data = input().split() # 입력값을 공백 기준으로 나눈 문자열 리스트
int_list = list(map(int, data)) # 입력값의 타입을 int로 변경하고, 리스트에 담는다.
result = list(map(square_if_valid, int_list)) # 입력받은 리스트를 제곱함수를 적용하고 결과 리스트에 담는다.
print(result) # 결과를 출력한다.
```

```
1 6 9 2         9 4 2 6
[1, 36, 9, 4] [9, 16, 4, 36]
```

문제 PY41-0001

1부터 100까지 자연수(100개) 의 평균,표준편차,분산,최대값,최소값 5개를 모두 출력하시오

- 조건
  - 입력값이 1 부터 100 이외의 값인 경우에도 정상 동작하도록 계산식에 상수를 직접 사용하지 않는다.
    - 오답 : (1+100)/2 (x) , avg = sum/100 (x)
  - **방법 1) for 문을 사용하시오.**
  - **방법 2) numpy 를 이용하시오**
- 입출력 예
  - 입출력 결과와 똑같아야 합니다. (공백, 대소문자, 형식을 정확하게 맞춰주세요)
  - 입력 안내 메시지는 생략해주세요

```
# 방법1 for

data = list(range(1, 101)) # 1 ~ 100까지 리스트로 변환


# 평균

total = 0 # 합계 초기화

for kk in data: # 리스트 요소 만큼 반복

  total += kk # 리스트 요소 합계

mean = total / len(data) # 평균 = 합계 / 리스트 길이


# 분산

var_sum = 0 # 합계 초기화

for kk in data: # 리스트 요소 만큼 반복

  var_sum += (kk - mean) ** 2 # (리스트 요소 - 평균)제곱의 합계

variance = var_sum / len(data) # 분산 = 합계 / 리스트 길이


# 표준편차

std_dev = variance ** 0.5 # 분산의 제곱근


# 최대값, 최소값
```

```
maximum = data[0] # 최대값 = 리스트0번째로 초기화
minimum = data[0] # 최소값 = 리스트0번째로 초기화
for kk in data: # 리스트 요소 만큼 반복
  if kk > maximum: # 리스트 요소가 최대값 보다 크다면
    maximum = kk # 최대값에 리스트 요소를 저장
  if kk < minimum: # 리스트 요소가 최소값 보다 작다면
    minimum = kk # 최소값에 리스트 요소를 저장

# 중간값
mid_index = len(data) // 2 # data 리스트 길이를 2로 나눈 몫의 정수 나누셈
if len(data) % 2 == 0: # data 리스트 길이가 짝수인 경우
    median = (data[mid_index - 1] + data[mid_index]) / 2 # 중간 값 = 중앙 두 값의 평균
else: # data 리스트 길이가 짝수가 아닌 경우
    median = data[mid_index] # 중간 값 = 데이터 중간 값

# 일반 출력
print(f"합계:{total}") # 합계 출력
print(f"평균:{mean}") # 평균 출력
print(f"분산:{variance}") # 분산 출력
print(f"표준편차:{std_dev:.2f}") # 표준편차 출력
print(f"최대값:{maximum}") # 최대값 출력
print(f"최소값:{minimum}") # 최소값 출력
```

```
합계:5050
평균:50.5
분산:833.25
표준편차:28.87
최대값:100
최소값:1
```

```python
# 방법2 numpy
import numpy as np   # numpy 라이브러리 불러오기


# NumPy Provides
#    1. 배열 객체 제공
#    2. 빠른 수학 연산
#    3. 선형대수, 푸리에변환, 난수 생성 등


data = np.arange(1, 101)   # 1 ~ 100까지 숫자를 생성하여 numpy 배열로 저장


# 통계 계산
mean = np.mean(data)                    # 평균
variance = np.var(data)                 # 분산
std_dev = np.std(data)                  # 표준편차
maximum = np.max(data)                  # 최대값
minimum = np.min(data)                  # 최소값
median = np.median(data)                # 중간값


# 일반 출력
print(f"합계:{np.sum(data)}")          # 합계 출력
print(f"평균:{mean}")                  # 평균 출력
print(f"분산:{variance}")              # 분산 출력
print(f"표준편차:{std_dev:.2f}")       # 표준편차 소수 둘째자리까지 출력
print(f"최대값:{maximum}")             # 최대값 출력
print(f"최소값:{minimum}")             # 최소값 출력
```

합계:5050
평균:50.5
분산:833.25
표준편차:28.87
최대값:100
최소값:1

문제 PY41-0002

1부터 100까지 자연수(100개) 의 평균,표준편차,분산,최대값,최소값,**중간값** 6개를 모두 출력하시오

- 조건
  - 입력값이 1 부터 100 이외의 값인 경우에도 정상 동작하도록 계산식에 상수를 직접 사용하지 않는다.
    - 오답 : (1+100)/2 (x) , avg = sum/100 (x)
  - 입력값이 랜덤하게 중복해서 섞여있을 수 있다.
  - **방법 1 : for 문을 사용하시오.**
    - sorted() 함수를 사용
  - **방법 2 : numpy 이용**
  - 입력
  - 입력값은 공백으로 구분한다.
- 출력
  - 일반 출력
- 입출력 예
  - 입출력 결과와 똑같아야 합니다. (공백, 대소문자, 형식을 정확하게 맞춰주세요)
  - 입력 안내 메시지는 생략해주세요

```
# 방법1 for

data = list(range(1, 101)) # 1 ~ 100까지 리스트로 변환


# 평균

total = 0 # 합계 초기화

for kk in data: # 리스트 요소 만큼 반복

  total += kk # 리스트 요소 합계

mean = total / len(data) # 평균 = 합계 / 리스트 길이


# 분산

var_sum = 0 # 합계 초기화

for kk in data: # 리스트 요소 만큼 반복

  var_sum += (kk - mean) ** 2 # (리스트 요소 - 평균)제곱의 합계

variance = var_sum / len(data) # 분산 = 합계 / 리스트 길이
```

```
# 표준편차
std_dev = variance ** 0.5 # 분산의 제곱근


# 최대값, 최소값
maximum = data[0] # 최대값 = 리스트0번째로 초기화
minimum = data[0] # 최소값 = 리스트0번째로 초기화
for kk in data: # 리스트 요소 만큼 반복
  if kk > maximum: # 리스트 요소가 최대값 보다 크다면
    maximum = kk # 최대값에 리스트 요소를 저장
  if kk < minimum: # 리스트 요소가 최소값 보다 작다면
    minimum = kk # 최소값에 리스트 요소를 저장


# 중간값
sorted_data = sorted(data) # 정렬된 data 리스트 = data 리스트 정렬한다.
n = len(sorted_data) # 정정렬된 data 리스트의 길이
if n % 2 == 0: # data 리스트 길이가 짝수인 경우
    # 중간 값 = 중앙 두 값의 평균(// : 나눗셈 몫의 정수 값을 계산해주는 연산자)
    median = (sorted_data[n//2 - 1] + sorted_data[n//2]) / 2
else: # data 리스트 길이가 짝수가 아닌 경우
    median = sorted_data[n//2] # 중간 값 = 데이터 중간 값


# 일반 출력
print(f"합계:{total}") # 합계 출력
print(f"평균:{mean}") # 평균 출력
print(f"분산:{variance}") # 분산 출력
print(f"표준편차:{std_dev:.2f}") # 표준편차 출력
print(f"최대값:{maximum}") # 최대값 출력
print(f"최소값:{minimum}") # 최소값 출력
```

```
print(f"중간값:{median}") # 중간값 출력
```

```
합계:5050
평균:50.5
분산:833.25
표준편차:28.87
최대값:100
최소값:1
중간값:50.5
```

```
# 방법2 numpy
import numpy as np   # numpy 라이브러리 불러오기


# NumPy Provides
#    1. 배열 객체 제공
#    2. 빠른 수학 연산
#    3. 선형대수, 푸리에변환, 난수 생성 등


data = np.arange(1, 101)   # 1 ~ 100까지 숫자를 생성하여 numpy 배열로 저장


# 통계 계산
mean = np.mean(data)   # 평균
variance = np.var(data) # 분산
std_dev = np.std(data)   # 표준편차
maximum = np.max(data)   # 최대값
minimum = np.min(data)   # 최소값
median = np.median(data)   # 중간값


# 일반 출력
print(f"합계:{np.sum(data)}") # 합계 출력
print(f"평균:{mean}") # 평균 출력
```

```
print(f"분산:{variance}") # 분산 출력
print(f"표준편차:{std_dev:.2f}")  # 표준편차 소수 둘째자리까지 출력
print(f"최대값:{maximum}")  # 최대값 출력
print(f"최소값:{minimum}")  # 최소값 출력
print(f"중간값:{median}") # 중간값 출력
```

```
합계:5050
평균:50.5
분산:833.25
표준편차:28.87
최대값:100
최소값:1
중간값:50.5
```

문제 PY42-0001

아래 코드는 데이터열 조작 CRUD 예제 중 일부이다.

삭제할 문자열을 검색하고, 문자열을 삭제하는 delete_List() 함수를 완성하시오.

- 조건
  - 검색하고자 하는 데이터열을 **집합**으로 바꾸고 delete_List() 함수를 완성하시오.
  - 지우려고 할 때 데이터가 없는 경우 에러 발생→ 데이터가 없다는 메시지 출력
  - 복수개의 데이터 삭제 불가 → 중복된 데이터가 있다면 함께 삭제 되어야 함
  - 방법 택 1
    - 방법 1 : 데이터가 없는 조건식을 **교집합**으로 판단

```
lt1 = []   #데이터열 생성 Create


def append_List():
   key = input('Append Data :')
   lt1.append(key)
   print(lt1)


def print_List():
   print(lt1)
   pass


def delete_List():
   print(lt1)
   key = input('Delete Data :') # 삭제할 문자열 입력
   if set([key]) & set(lt1): #검색하고자 하는 데이터열을 집합으로 바꾸고 교집합인 경우
      while key in lt1: # lt1리스트 요소 동안 반복
         lt1.remove(key) # lt1리스트 요소 삭제
   else: # 삭제하고자 하는 데이터가 없다면
      print("데이터가 없습니다.") # 데이터가 없습니다. 출력
```

```
    print(lt1)


while True:
    print('1. Append List') #데이터열 조작(추가) Update
    print('2. Print List') #데이터열 읽기 Read
    print('3. Delete List') #데이터열 조작(검색+추가) Update
    print('4. Exit')   #데이터열 삭제 Delete

    key = input('Select Number [1~4] :') #IO 처리 (1)
    if key == '1':
        append_List()
    elif key == '2':
        print_List()
    elif key == '3':
        delete_List()
    else:
        break

print('\nThanks.')
```

```
1. Append List
2. Print List
3. Delete List
4. Exit
Select Number [1~4] :1
Append Data :AAA
['AAA']
1. Append List
2. Print List
3. Delete List
4. Exit
Select Number [1~4] :1
Append Data :BBB
['AAA', 'BBB']
1. Append List
2. Print List
3. Delete List
4. Exit
Select Number [1~4] :1
Append Data :CCC
['AAA', 'BBB', 'CCC']
1. Append List
2. Print List
3. Delete List
4. Exit
Select Number [1~4] :1
Append Data :AAA
['AAA', 'BBB', 'CCC', 'AAA']
```

```
1. Append List
2. Print List
3. Delete List
4. Exit
Select Number [1~4] :2
['AAA', 'BBB', 'CCC', 'AAA']
1. Append List
2. Print List
3. Delete List
4. Exit
Select Number [1~4] :3
['AAA', 'BBB', 'CCC', 'AAA']
Delete Data :AAA
['BBB', 'CCC']
1. Append List
2. Print List
3. Delete List
4. Exit
Select Number [1~4] :4

Thanks.
```

문제 PY42-0002

다음 프로그램은 lt1 의 원소를 삭제하는 프로그램의 일부이다.

아래 코드는 'AAA' 원소를 삭제하는 경우 'AAA' 모든 원소가 삭제되지 않는 오류가 있다.

코드 내용을 수정해서 올바른 입출력을 갖는 프로그램을 작성하시오.

- 조건
  - Jupyterlab 을 이용하시오.
  - 두가지 방법 모두 작성하시오
    - (방법 1) for 문을 사용하지 말것
    - (방법 2) for 문을 사용할것

```
# (방법1) for 문을 사용하지 말것
lt1 = ['BBB','AAA','AAA'] #데이터열 생성 Create


def delete_List():
  print(lt1)
  key = input('Delete Data :') #IO 처리 (2)

  # 데이터열내에서 데이터 유무 검사
  if set(lt1) & {key} == set(): #공집합
    print(f'[{key}] No data in list')
    return

  # 데이터열 검색 + 삭제
  # lt1[:] = 기존 리스트의 내용을 슬라이싱으로 전부 바꾼다
  # lt1에서 key 값이 아닌 요소만 남김 (key와 일치하는 값은 모두 제거)
  lt1[:] = [kk for kk in lt1 if kk != key]

  print(lt1) # lt1리스트 출력
```

```
delete_List()
```

```
['BBB', 'AAA', 'AAA']
Delete Data :AAA
['BBB']
```

```
['BBB', 'AAA', 'AAA']
Delete Data :BBB
['AAA', 'AAA']
```

```
# (방법2) for 문을 사용할것
lt1 = ['BBB','AAA','AAA'] #데이터열 생성 Create


def delete_List():
  print(lt1)
  key = input('Delete Data :') #IO 처리 (2)

  # 데이터열내에서 데이터 유무 검사
  if set(lt1) & {key} == set(): #공집합
    print(f'[{key}] No data in list')
    return

  # 데이터열 검색 + 삭제
  # (앞에서 삭제하면 인덱스 밀림 오류가 발생할 수 있으므로 역순 사용)
  for i in range(len(lt1) -1, -1, -1): # len-1부터 0까지 역순 반복
    if lt1[i] == key: # 현재 요소가 삭제 대상인 경우
      del lt1[i] # 해당 요소 삭제

  print(lt1)

delete_List()
```

```
['BBB', 'AAA', 'AAA'] ['BBB', 'AAA', 'AAA']
Delete Data :AAA      Delete Data :BBB
['BBB']               ['AAA', 'AAA']
```

문제 PY43-0001

슬라이스 연산자를 이용해서 다음과 같이 출력되도록 프로그램 하시오.

- 입출력 예
    - 입출력 결과와 똑같아야 합니다. (공백, 대소문자, 형식을 정확하게 맞춰주세요)
    - 입력 안내 메시지는 생략해주세요

```
str1 = "123456789"

str1 = str1[7:0:-2] # str1[7]=8 ~ str1[0]=1 까지 역순으로 2개씩 자르기

print(str1)
```

```
8642
```

문제 PY43-0002

다음은 문자열을 입력받고, 슬라이스 연산자를 이용한 출력 결과를 보여 주는 코드의 일부이다.

코드에 맞는 출력 결과를 작성하시오.

- 오류이거나 출력이 없는 경우 x 표시하시오.

```
str1 = '123456789'


result1 = str1[::2]

result2 = str1[7:-4:-1]

result3 = str1[7:4:-1]


print(f"str1[::2] = {result1}")

print(f"str1[7:-4:-1] = {result2}")

print(f"str1[7:4:-1] = {result3}")
```

```
str1[::2] = 13579
str1[7:-4:-1] = 87
str1[7:4:-1] = 876
```

문제 PY43-0003

입력 문자열의 홀수번째 문자열을 앞쪽에 배치하고, 짝수번째 문자열을 이어서 배치해보자.

- 조건
  - Trinket3.x
  - 슬라이싱 연산자를 이용하시오.
- 입력
  - 공백 기준으로 입력
- 출력
  - 입출력 예 참고
- 입출력 예
  - 입출력 결과와 똑같아야 합니다. (공백, 대소문자, 형식을 정확하게 맞춰주세요)
  - 입력 안내 메시지는 생략해주세요

```
str1 = input()

result = str1[::2] + str1[1::2]

print(result)
```

```
1234567890
1357924680
```

문제 PY43-0004

입력 문자열의 모든 단어들의 순서를 뒤집어보자.

- 조건
  - Trinket3.x
  - 아래방법 모두 작성하시오.
    - 방법 1) for 반복문 이용하시오
    - 방법 2) LC 사용
- 입출력 예
  - 입출력 결과와 똑같아야 합니다. (공백, 대소문자, 형식을 정확하게 맞춰주세요)
  - 입력 안내 메시지는 생략해주세요

```
# 방법1) for 반복문 이용하시오

words = input().split()  # 입력받은 문자열을 공백 기준으로 단어 리스트로 분리

reversed_words = [] # 결과를 저장할 리스트



for kk in words: # # 각 단어를 순회하면서

    reversed_words.append(kk[::-1]) # 단어를 뒤집어서 결과 리스트에 추가



print(" ".join(reversed_words)) # 공백 기준으로 연결하여 출력
```

```
Hello Python World
olleH nohtyP dlroW
```

```
# 방법2) LC 사용

words = input().split() # 입력받은 문자열을 공백 기준으로 단어 리스트로 분리

reversed_words = [word[::-1] for word in words]  # 각 단어를 뒤집어서 리스트로 저장

print(" ".join(reversed_words)) # 공백 기준으로 연결하여 출력
```

```
Hello Python World
olleH nohtyP dlroW
```

문제 PY43-0005

문장속에 등장하는 문자들을 알파벳순으로 정렬된 문자열을 만드시오.

- 조건
  - Trinket3.x
  - while 반복문 이용
- 입력
  - 입력된 문자열의 스페이스 ,Tab 은 제거한다.
- 출력
  - 입출력 예 참고
- 입출력 예
  - 입출력 결과와 똑같아야 합니다. (공백, 대소문자, 형식을 정확하게 맞춰주세요)
  - 입력 안내 메시지는 생략해주세요

```
str1 = input()  # 문자열 입력 받기
str1 = str1.replace(" ", "").replace("\t", "")  # 문자열에서 공백과 탭 제거
chars = list(str1)  # 문자열을 문자 하나씩 분리하여 리스트로 변환


# 버블 정렬 알고리즘을 사용해 리스트를 오름차순 정렬 (while 반복문 사용)
i = 0
while i < len(chars):  # 리스트 전체를 순회하며
    j = 0
    while j < len(chars) - 1:  # 인접한 문자들끼리 비교
        if chars[j] > chars[j + 1]:  # 앞 문자가 뒤 문자보다 크면 (알파벳 순서상 뒤)
            chars[j], chars[j + 1] = chars[j + 1], chars[j]  # 위치 교환
        j += 1  # 다음 인덱스로 이동
    i += 1  # 외부 루프 반복 (여러 번 순회 필요)


# 정렬된 문자 리스트를 하나의 문자열로 합쳐서 출력
print("".join(chars))
```

```
Hello Python World
HPWdehlllnooorty
```

문제 PY43-0006

8 자리 이하의 문자열 하나를 입력받고, 작업 후 8 자리 문자열을 출력하고자 한다.

문자열이 숫자로만 이뤄진 경우 문자열 나머지 공간은 '0' 으로 채우고,

숫자이외의 값이 포함된 경우 '-' 로 자리를 채우는 프로그램을 작성해보자.

- 조건
  - Trinket3.x
  - 문자열 함수 사용 가능
- 입력
  - 입출력 예 참고
- 출력
  - 입출력 예 참고
- 입출력 예
  - 입출력 결과와 똑같아야 합니다. (공백, 대소문자, 형식을 정확하게 맞춰주세요)
  - 입력 안내 메시지는 생략해주세요

```
lt = input() # 문자열 입력 받기


if lt.isdigit(): # 문자열이 숫자라면
  print(lt.rjust(8, '0')) # 전체 길이를 8로 맞추고, 왼쪽을 '0'으로 채워서 출력
else: # 문자열이 숫자가 아니라면
  print(lt.rjust(8, '-')) # 전체 길이를 8로 맞추고, 왼쪽을 '-'로 채워서 출력
```

```
123         123x        xyz
00000123    ----123x    -----xyz
```

문제 PY43-0007

문자열의 일부를 수정할 수 있는 str.replace() 와 같은 기능을 하는 myReplace 함수를 만들어 보자.

- 조건
  - Trinket3.x
  - str.replace() 함수를 사용하지 마시오

```
# myReplace 사용자 정의 함수
def myReplace(text, x, y): # 입력파라미터 3개를 받는다 (원본 문자열, 바꿀 문자열, 새 문자열)
  result = "" # 결과 문자열 변수 초기화
  i = 0 # 시작 인덱스를 0으로 초기화

  while i < len(text): # i가 text 문자열 길이보다 작을 경우 반복한다

    if text[i:i+len(x)] == x: # 현재 인덱스부터 x의 길이만큼 자른 부분이 x와 같다면
      result += y # 결과 문자열에 y를 추가한다
      i += len(x) # x의 길이만큼 인덱스를 건너뛴다
    else: # 그렇지 않다면
      result += text[i] # 현재 문자 그대로 결과에 추가
      i += 1 # 인덱스를 1 증가

  return result # 결과 문자열 반환

str1 = "Hello Python World Hello Python World"
str2 = myReplace(str1, "llo", "xx")
print(str2)
# Hexx Python World Hexx Python World

str1 = "Hello Python World Hello Python World"
```

```
str2 = myReplace(str1, "Py", "1234")

print(str2)

# Hello 1234thon World Hello 1234thon World
```

```
Hexx Python World Hexx Python World
Hello 1234thon World Hello 1234thon World
```

문제 PY43-0008

menu1 은 공백을 포함하는 문자열 리스트이다.

아래 조건을 만족하는 menu2fctname() 함수를 완성하시오.

- 조건
  - Trinket3.x
  - 원소들의 첫 문자는 소문자로,
  - 공백은 '_' 로 치환

```
def menu2fctname(menu1): # 사용자 지정 함수, 문자열 리스트 menu1 을 입력받음
  result = [] # 결과를 저장할 리스트 초기화
  for kk in menu1: # menu1 리스트의 각 요소에 대해 반복
    converted = kk.replace(" ", "_") # 요소 내 공백을 "_"로 치환
    converted = converted[0].lower() + converted[1:] # 첫 글자를 소문자로 변환
    result.append(converted) # 결과 리스트에 추가
  return result # 결과 리스트 반환

menu1 = ['Append List','Print List','Delete List','Rename List','Exit']

fct1 = menu2fctname(menu1)

print(fct1)
# ['append_List', 'print_List', 'delete_List', 'rename_List', 'exit']
```

```
['append_List', 'print_List', 'delete_List', 'rename_List', 'exit']
```

문제 PY44-0001

복수개의 숫자를 입력받고, 복수개의 숫자를 리스트로 출력하는 프로그램을 작성하시오.

- 조건
  - Trinket3.x
  - 문자열 함수 사용 가능
- 입력
  - 입력값은 공백으로 구분한다.
- 출력
  - 입출력 예 참고
- 입출력 예
  - 입출력 결과와 똑같아야 합니다. (공백, 대소문자, 형식을 정확하게 맞춰주세요)
  - 입력 안내 메시지는 생략해주세요

```
text = input().split() # 입력된 문자열을 공백 기준으로 나누어 리스트로 저장

numbers = list(map(int, text)) # 문자열 리스트의 각 요소를 정수로 변환하여 저장

print(numbers) # 정수 리스트 출력
```

```
135 42
[135, 42]
```

문제 PY44-0002

복수개의 실수를 입력받고, 합을 구하시오 (단, 소수점 첫째 자리까지 반올림하시오)

- 조건
  - Trinket3.x
- 입력
  - 입력값은 공백으로 구분한다.
- 출력
  - 입출력 예 참고
- 입출력 예
  - 입출력 결과와 똑같아야 합니다. (공백, 대소문자, 형식을 정확하게 맞춰주세요)
  - 입력 안내 메시지는 생략해주세요

```
text = input().split() # 입력된 문자열을 공백 기준으로 나누어 리스트로 저장

floats = list(map(float, text)) # 문자열 리스트 각 요소를 실수로 변환하여 저장

total = 0 # 합계를 저장할 변수 초기화

for kk in floats: # floats 리스트의 각 요소에 대해 반복

  total += kk # 각 요소를 합계에 누적

print(round(total, 1)) # 합계의 소수점 1자리 반올림하여 출력
```

```
3.14 3.14
6.3
3.14 3.1
6.2
135 42
177.0
```

문제 PY44-0003

실수 2 개를 입력받고, 소수점 자리와 정수 자리를 구분하시오

- 조건
  - Trinket3.x
  - 정수부 실수부 분리 (LC 사용하시오)
- 입력
  - 입력값은 공백으로 구분한다.
- 출력
  - 입출력 예 참고
- 입출력 예
  - 입출력 결과와 똑같아야 합니다. (공백, 대소문자, 형식을 정확하게 맞춰주세요)
  - 입력 안내 메시지는 생략해주세요

```
text = input().split() # 입력받은 문자열을 공백 기준 분리

floats = list(map(float, text)) # 문자열을 실수로 변환


# # 각 실수를 문자열로 변환 → '.' 기준 분리 → 정수로 변환 → 튜플로 묶어 리스트 생성

split_numbers = [tuple(map(int,str(kk).split("."))) for kk in floats]


print(split_numbers)  # 최종 결과 출력
```

```
3.14 6.28
[(3, 14), (6, 28)]
```

```
2.414 6.28
[(2, 414), (6, 28)]
```

문제 PY44-0004

다음과 같은 2차원 리스트를 입력으로 하는 myfct() 를 작성하시오.

- 조건
  - 반복문사용
  - 배열의 값을 코드에 직접사용 x
  - 배열의 위치 사용

```
def myfct(lt): # 사용자 정의 함수 (2차원 리스트 lt를 입력받음)
    result = [] # 결과를 저장할 리스트 초기화
    for i in range(0, len(lt)): # 0부터 lt의 행 수 만큼 반복
        row = i # 행 인덱스
        col = len(lt[0]) - 1 - i # 열 인덱스: 열 개수 - 1 - 행 인덱스
        if 0 <= col < len(lt[0]):  # 열이 0이상 리스트 열 개수 미만이면(유효 열 범위 확인)
            result.append(lt[row][col]) # 해당 위치 값을 결과 리스트에 추가
    return result # 최종 결과 반환


lt1 = [[1,2,3,1],
       [4,5,6,4],
       [7,8,9,7]]


lt2 = [[1,2,3,4,1],
       [4,5,7,6,4],
       [7,9,8,9,7],
       [7,9,8,9,7]]


print(myfct(lt1)) #[1, 6, 8] 출력      [lt1[0][0], lt1[1][2], lt1[2][1]]
print(myfct(lt2)) #[1, 6, 8, 9] 출력  [lt2[0][0], lt2[1][3], lt2[2][2], lt2[3][3]
```

```
[1, 6, 8]
[1, 6, 8, 9]
```

문제 PY44-0005

학생들의 시험 점수가 2차원 리스트로 주어집니다.

각 학생마다 과목 점수가 하나의 리스트에 저장되어 있습니다.

가장 높은 평균 점수를 받은 학생의 인덱스를 구하세요.

조건

- scores 는 학생 수 × 과목 수 크기의 2 차원 리스트입니다.
- 각 내부 리스트는 한 학생의 점수 리스트입니다.
- 평균이 가장 높은 학생의 인덱스를 출력하세요.
- 평균이 같은 경우 먼저 나온 학생을 선택합니다.
- 풀이 1) 리스트이용
- 풀이 2) numpy 이용
- 풀이 3) pandas 이용

```
import numpy as np # numpy 라이브러리 불러와서 이름을 np로 수정
import pandas as pd # pandas 라이브러리 불러와서 이름을 pd로 수정


# 1. List 이용
def best_student_index(scores): # 사용자 정의 함수 (입력: 학생 점수 2차원 리스트)
  max_avg = -1 # 최대 평균 초기화 (평균이 0점일수 도 있으므로 -1로 시작)
  max_index = -1 # 최대 평균을 가진 학생의 인덱스 초기화
  for i, student in enumerate(scores): # scores 리스트를 인덱스와 함께 반복
    avg = sum(student) / len(student) # 평균 = 총합 / 과목수
    if avg > max_avg: # 현재 평균이 최대 평균 보다 크면
      max_avg = avg # 최대 평균 갱신
      max_index = i # 해당 학생의 인덱스 저장
  return max_index # 평균이 가장 높은 학생의 인덱스 반환


# 2. NumPy 이용
def best_student_index_np(scores):  # 사용자 정의 함수 (입력: 학생 점수 2차원 리스트)
    arr = np.array(scores)  # 2차원 리스트를 NumPy 배열로 변환
```

```
    # axis=0 : 열(세로) 단위 연산, axis=1 : 열(가로) 단위 연산
    avg_scores = arr.mean(axis=1)  # 각 학생(행 기준) 평균
    # np.argmax : 가장 큰 값이 있는 위치를 찾아주는 함수
    return int(np.argmax(avg_scores))  # 평균이 가장 높은 학생의 인덱스 반환

# 3. Pandas 이용
def best_student_index_pd(scores):  # 사용자 정의 함수 (입력: 학생 점수 2차원 리스트)
    # DataFrame() : 엑셀 스프레드시트와 비슷한 형태의 2차원 데이터 구조
    df = pd.DataFrame(scores) # 2차원 리스트를 표 형태 DataFrame으로 변환
    avg_scores = df.mean(axis=1)  # 각 행(학생) 평균
    # idxmax() : 가장 큰 값을 가진 인덱스를 반환하는 함수
    return int(avg_scores.idxmax())  # 평균이 가장 높은 학생의 인덱스 반환

scores = eval(input()) # 문자열 형태로 입력된 리스트를 실제 리스트로 변환
print(best_student_index(scores))          # List 방식 결과 출력
print(best_student_index_np(scores))      # NumPy 방식 결과 출력
print(best_student_index_pd(scores))      # Pandas 방식 결과 출력
```

```
[[70, 80, 90], [85, 85, 85], [100, 60, 70]]
1
1
1
[[50, 60], [90, 95], [80, 85]]
1
1
1
```

문제 PY44-0006

도시별 온도가 2차원 리스트로 주어집니다.

각 행은 한 도시의 일주일간 기온을 나타냅니다.

평균 기온이 t도 이상인 도시의 인덱스를 모두 출력하세요.

조건

- temperatures는 도시 수 × 7 크기의 2 차원 리스트입니다.
- 정수 t가 주어집니다.
- 평균이 t 이상인 도시의 인덱스를 리스트로 출력하세요.
- 도시 인덱스는 오름차순으로 정렬되어야 합니다.
- 풀이 1) 리스트이용
- 풀이 2) numpy 이용
- 풀이 3) pandas 이용

```
import numpy as np
import pandas as pd


# 문제에서 고정된 기준 온도 t
t = 22


# 1. List 방식
def hot_cities_list(temperatures): # 사용자 정의 함수(입력 : 2차원 리스트)
    result = [] # 결과 리스트 초기화
    for i, city in enumerate(temperatures): # temperatures 리스트를 인덱스와 함께 반복
        avg = sum(city) / len(city) # 도시별 평균 기온 계산 (총합 / 일수)
        if avg >= t:  # 평균이 기준 온도 이상이면
            result.append(i)  # 결과 리스트에 해당 도시 인덱스를 추가
    return result # 평균이 기준 이상인 도시들의 인덱스 리스트 반환


# 2. NumPy 방식
def hot_cities_numpy(temperatures): # 사용자 정의 함수(입력 : 2차원 리스트)
```

```
    arr = np.array(temperatures)  # 2차원 리스트를 NumPy 배열로 변환
    avg = arr.mean(axis=1)  # 각 도시(행 기준)의 평균 기온 계산
    # np.where(조건) : 조건에 만족하는 인덱스들을 튜플로 반환
    return list(map(int, np.where(avg >= t)[0])) # 조건을 만족하는 인덱스 배열의 첫 번째 요소를 리스트로 변환하여 반환

# 3. Pandas 방식
def hot_cities_pandas(temperatures): # 사용자 정의 함수(입력 : 2차원 리스트)
    df = pd.DataFrame(temperatures) # 2차원 리스트를 표 형태 DataFrame으로 변환
    avg = df.mean(axis=1) # 각 행(도시별)의 평균 기온 계산
    return avg[avg >= t].index.tolist() # 평균이 기준 이상인 도시의 인덱스를 리스트로 반환

temperatures = eval(input()) # 입력 받은 문자열을 리스트로 반환

print(hot_cities_list(temperatures)) # List 방식 결과 출력
print(hot_cities_numpy(temperatures)) # NumPy 방식 결과 출력
print(hot_cities_pandas(temperatures)) # Pandas 방식 결과 출력
```

```
[[22, 24, 25, 23, 21, 22, 20], [18, 19, 17, 20, 21, 22, 23], [25, 26, 27, 28, 29, 30, 31]]
[0, 2]
[0, 2]
[0, 2]
```

```
import numpy as np
import pandas as pd

# 문제에서 고정된 기준 온도 t
t = 20

# 1. List 방식
def hot_cities_list(temperatures): # 사용자 정의 함수(입력 : 2차원 리스트)
```

```
    result = [] # 결과 리스트 초기화
    for i, city in enumerate(temperatures): # temperatures 리스트를 인덱스와 함께 반복
        avg = sum(city) / len(city) # 도시별 평균 기온 계산 (총합 / 일수)
        if avg >= t:  # 평균이 기준 온도 이상이면
            result.append(i)  # 결과 리스트에 해당 도시 인덱스를 추가
    return result # 평균이 기준 이상인 도시들의 인덱스 리스트 반환

# 2. NumPy 방식
def hot_cities_numpy(temperatures): # 사용자 정의 함수(입력 : 2차원 리스트)
    arr = np.array(temperatures)  # 2차원 리스트를 NumPy 배열로 변환
    avg = arr.mean(axis=1)  # 각 도시(행 기준)의 평균 기온 계산
    # np.where(조건) : 조건에 만족하는 인덱스들을 튜플로 반환
    return list(map(int, np.where(avg >= t)[0])) # 조건을 만족하는 인덱스 배열의 첫 번째 요소를 리스트로 변환하여 반환

# 3. Pandas 방식
def hot_cities_pandas(temperatures): # 사용자 정의 함수(입력 : 2차원 리스트)
    df = pd.DataFrame(temperatures) # 2차원 리스트를 표 형태 DataFrame으로 변환
    avg = df.mean(axis=1) # 각 행(도시별)의 평균 기온 계산
    return avg[avg >= t].index.tolist() # 평균이 기준 이상인 도시의 인덱스를 리스트로 반환

temperatures = eval(input()) # 입력 받은 문자열을 리스트로 반환

print(hot_cities_list(temperatures)) # List 방식 결과 출력
print(hot_cities_numpy(temperatures)) # NumPy 방식 결과 출력
```

```
print(hot_cities_pandas(temperatures)) # Pandas 방식 결과 출력
```

```
[[15, 16, 14, 13, 12, 11, 10], [24, 24, 24, 24, 24, 24, 24]]
[1]
[1]
[1]
```