

Chapter 7-3

Data Organization

Contents

- **Nested(Multi-dimensional) List**
- Sequence: List and String
- Iterating over Sequence
- Iterating over Nested Lists
- Computational Problem

Nested Lists

■ 중첩 리스트

- 리스트 안에 리스트를 요소로 저장하여 표와 같은 2차원 구조를 표현할 수 있음

table	
11	12
21	22
31	32

```
>>> table = [[11, 12], [21, 22], [31, 32]]
>>> table[0] ← 첫 번째 요소인 리스트 [11, 12]를 반환
[11, 12]
>>> table[1]
[21, 22]
>>> table[2]
[31, 32]
>>> table[0][0] ← 리스트 [11, 12]의 첫 번째 요소인
11                      11을 반환
```

Nested Lists

Nested List의 출력 예제 코드

```
score = [ ['Comp', 90, 95],  
          ['Math', 87, 91] ]  
col = 0  
row = 0  
  
print('Name\tMid\tFinal')  
while row < 2:  
    while col < 3:  
        print(score[row][col], end='\t')  
        col = col + 1  
    print()  
    row = row + 1  
    col = 0
```

출력결과

Name	Mid	Final
Comp	90	95
Math	87	91

>>>

Contents

- Nested(Multi-dimensional) List
- **Sequence: List and String**
- Iterating over Sequence
- Iterating over Nested Lists
- Computational Problem

Sequence: List and String

List

연속적으로 나열된 element로
표현된 구조



String

연속적으로 나열된 문자로
표현된 구조



Sequence Type

List 혹은 string과 같이 연속적인 자료로 이루어진 Python 자료 형
인덱스(Index)로 요소(element)를 접근 할 수 있음

```
>>> x = [1, 2, 3]
```

```
>>> x[0]
```

```
1
```

```
>>> x = 'Hello'
```

```
>>> x[0]
```

```
'H'
```

Sequence: List and String

- Sequence Operations: list와 string에 모두 적용 가능한 연산

Operation	x = [1, 2, 3]	x = 'hello'
len(x)	3	5
x[1]	2	'e'
x[1:3]	[2, 3]	'el'
2 in x	True	False
'el' in x	False	True
min(x)	1	'e'
max(x)	3	'o'
sum(x)	6	TypeError
x + [4, 5]	[1, 2, 3, 4, 5]	TypeError
x + ' world'	TypeError	'hello world'

Sequence: List and String

Exercise!

```
>>> x = 'Pineapple'  
>>> x[4:9]
```

```
>>> x = [-2, -1, 0, 1, 2]  
>>> x[2:4]
```

```
>>> 'Pine' in x
```

```
>>> '1' in x
```

```
>>> len(x)
```

```
>>> len(x)
```

```
>>> min(x)
```

```
>>> min(x)
```

```
>>> x[:4] + 'berry'
```

```
>>> x + [10, 20]
```


Contents

- Nested(Multi-dimensional) List
- Sequence: List and String
- **Iterating over Sequence**
- Iterating over Nested Lists
- Computational Problem

Iterating over Sequence

1. while loop을 이용하여 index 값으로 element를 접근

■ while loop example

```
lst = [10, 20, 30, 40, 50, 60]
i = 0
while i < len(lst):
    print(lst[i])
    i = i + 1
```

Result

10
20
30
40
50
60

2. for loop을 이용하여 각 element의 값을 사용

■ for loop example

```
lst = [10, 20, 30, 40, 50, 60]
for n in lst:
    print(n)
```

Iterating over Sequence

for 명령문은 리스트 혹은 문자열과 같이 연속된 값에 반복하여 접근할 때 자주 사용된다

Syntax	Example
<pre>for v in sequence: statements</pre>	<pre>lst = [2, 4, 6, 8, 10] for n in lst: print(lst)</pre>
	<pre>text = 'Hello' for c in text: print(c)</pre>

Iterating over Sequence

■ for 명령문의 또 다른 활용

- for 명령문은 특정 회수 만큼 반복을 시킬 때 유용

start**이상** end**미만**까지의 값을 차례로 생성

Syntax	Example
for x in range(start, end): statements	for n in range(1, 3): print('n =', n)

range(start, end)에서 생성한 값을 차례로 x에 할당

각 반복 시점에서의 n과 출력결과의 상태(state)

Iteration	n	output
1	1	n = 1
2	2	n = 2

Iterating over Sequence

■ for v in range(start, end)

- range(start, end): start 이상 end 미만까지의 sequence를 생성
- 특정한 회수 만큼 반복시킬 때 유용

예1) 0부터 4까지 출력

```
for n in range(0, 5):  
    print(n)
```

Result

0
1
2
3
4

예2) 1부터 100까지의 합 출력

```
sum = 0  
for i in range(1, 101):  
    sum = sum + i  
print(sum)
```

Result

5050

Iterating over Sequence

Exercise!

```
for n in [10, 20, 30]:  
    print(n)
```

```
for c in 'apple':  
    print(ord(c))
```

```
for s in ['apple', 'cherry']:  
    print(s)
```

```
lst = [10, 20, 30]  
for n in lst:  
    n = n + 1  
print(lst)
```

```
sum = 0  
for n in range(0, 10):  
    sum = sum + n  
print(sum)
```

```
for n in range(0, 10):  
    if n % 2 == 0:  
        print(n)
```

Contents

- Nested(Multi-dimensional) List
- Sequence: List and String
- Iterating over Sequence
- **Iterating over Nested Lists**
- Computational Problem

Iterating over Nested Lists

각 학생들의 과목 점수의 평균을 출력하기

Score Table

CS	Math	Art
98	87	78
78	67	91
86	83	89
63	59	60

List Representation

➡ `scores = [[98, 87, 78],
[78, 67, 91],
[86, 83, 89],
[63, 59, 60]]`

How many accesses are required to the list?

How many ***for*** loops are required?

Iterating over Nested Lists

각 학생들의 과목 점수의 평균을 출력하기

Score Table

	CS	Math	Art
row	98	87	79
	78	67	92
	86	83	89
	63	59	64

List Representation

scores = [[98, 87, 79],
[78, 67, 92],
[86, 83, 89],
[63, 59, 64]]

```
for row in scores:  
    print(row)
```

[98, 87, 79]
[78, 67, 92]
[86, 83, 89]
[63, 59, 64]

Iterating over Nested Lists

각 학생들의 과목 점수의 평균을 출력하기

Score Table

	CS	Math	Art
row	98	87	78
	78	67	91
	86	83	89
	63	59	60

col

List Representation

scores = [[98, 87, 79],
[78, 67, 92],
[86, 83, 89],
[63, 59, 64]]

```
for row in scores:  
    for col in row:  
        print(col, end=' ')  
    print()
```

98 87 79
78 67 92
86 83 89
63 59 64

Iterating over Nested Lists

각 학생들의 과목 점수의 평균을 출력하기

Score Table

	CS	Math	Art
row	98	87	78
	78	67	91
	86	83	89
	63	59	60

col

List Representation

scores = [[98, 87, 79],
[78, 67, 92],
[86, 83, 89],
[63, 59, 64]]

```
sum = 0
for row in scores:
    for col in row:
        sum = sum + col
    print(sum / len(row))
    sum = 0
```

88.0
79.0
86.0
62.0

Contents

- Nested(Multi-dimensional) List
- Sequence: List and String
- Iterating over Sequence
- Iterating over Nested Lists
- **Computational Problem**

Computational Problem

The Problem

Calculation of Average Score

아래 표에서 각 학생들의 평균(우측 Avg. 열)과 Mid 및 Final의 평균(하단 Avg. 행)을 완성하여 출력하세요

	Mid	Final	Avg.
John	95	55	
Mary	73	56	
David	91	88	
Susan	67	63	
Avg.			

Calculation of Average Score

Problem Analysis

표(table)의 성적 데이터 표현 : Nested list 이용

평균만 구하면 되므로 특별한 연산은 필요하지 않음

Calculation of Average Score

Data Representation

Score table data:

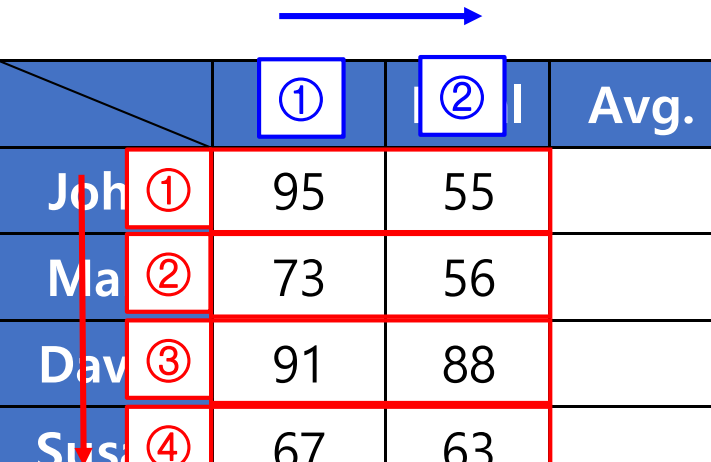
score

Calculation of Average Score

Algorithmic Thinking

각 학생의 시험점수 평균 구하기

John의 평균: ① 행에서 ①열과 ②열의 값을 더하여 열의 개수로 나눔
Mary의 평균: ② 행에서 ①열과 ②열의 값을 더하여 열의 개수로 나눔
David의 평균: ③ 행에서 ①열과 ②열의 값을 더하여 열의 개수로 나눔
Susan의 평균: ④ 행에서 ①열과 ②열의 값을 더하여 열의 개수로 나눔



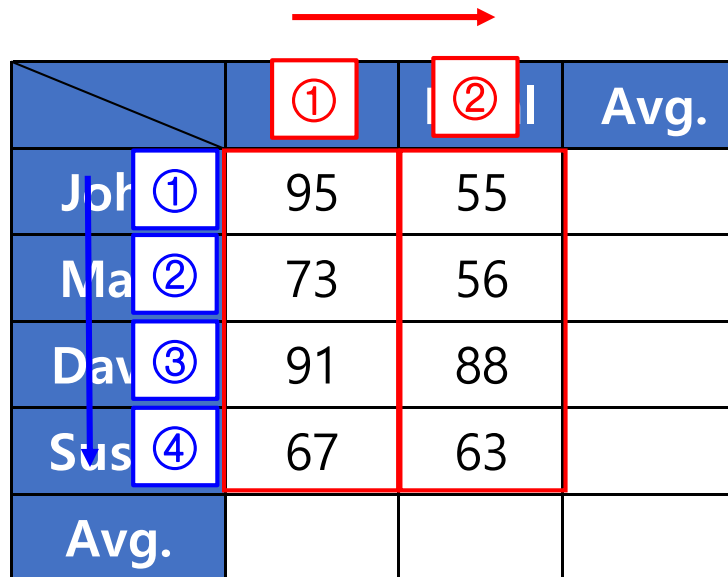
	①	②	Avg.
John	95	55	
Mary	73	56	
David	91	88	
Susan	67	63	
Avg.			

Calculation of Average Score

Algorithmic Thinking

각 시험의 평균 구하기

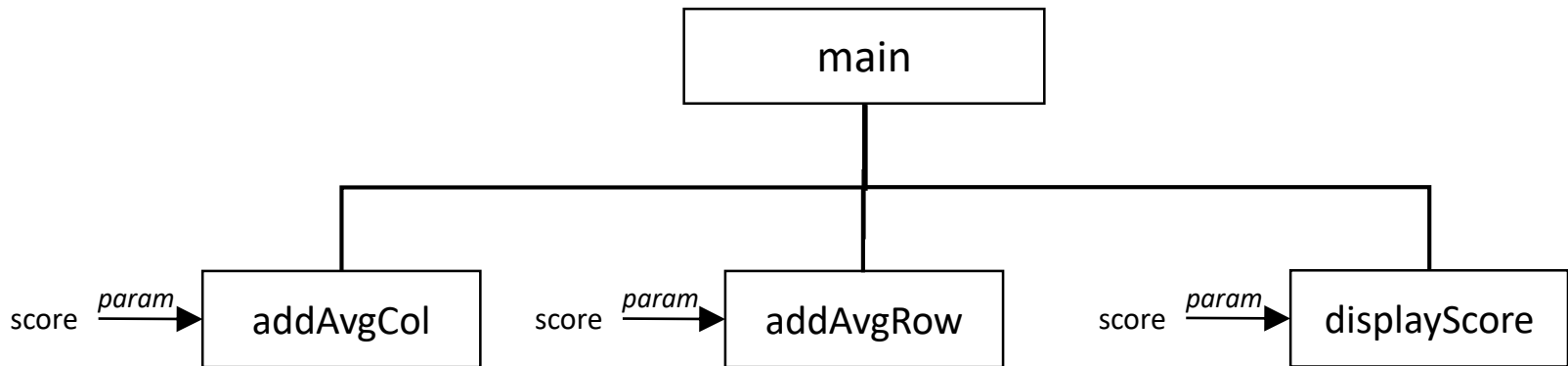
Mid의 평균: ① 열에서 ①, ②, ③, ④ 행의 값을 더하여 행의 개수로 나눔
Final의 평균: ② 열에서 ①, ②, ③, ④ 행의 값을 더하여 행의 개수로 나눔



		①	②	Avg.
John	①	95	55	
Martha	②	73	56	
Dave	③	91	88	
Susan	④	67	63	
Avg.				

Calculation of Average Score

Modularization



Calculation of Average Score

Program Design

Requirements

평균이 입력된 표를 출력

Calculation of Average Score

Program Implementation

```
1 score = [ ['John', 95, 55],  
2           ['Mary', 73, 56],  
3           ['David', 91, 88],  
4           ['Susan', 67, 63] ]  
5
```

```
29 addAvgCol(score)  
30 addAvgRow(score)  
31 displayScore(score)
```

Calculation of Average Score

Program Implementation

```
6 def addAvgCol(score):
7     for r in range(0, len(score)):
8         sum = 0
9         for c in range(1, len(score[r])):
10             sum = sum + score[r][c]
11         score[r].append(sum/2)
12
13 def addAvgRow(score):
14     col_5th = ['Avg.']
15     for c in range(1, len(score[0])):
16         sum = 0
17         for r in range(0, len(score)):
18             sum = sum + score[r][c]
19         col_5th.append(sum/4)
20     score.append(col_5th)
21
```

Calculation of Average Score

Program Implementation

```
22 def displayScore(score):
23     print('Name\tMid\tFinal\tAvg.')
24     for r in range(0, len(score)):
25         for c in range(0, len(score[r])):
26             print(score[r][c], end='\t')
27         print()
28
```

Calculation of Average Score

Program Execution

Name	Mid	Final	Avg.
John	95	55	75.0
Mary	73	56	64.5
David	91	88	89.5
Susan	67	63	65.0
Avg.	81.5	65.5	73.5

>>>