

Chapter 7-1

Data Organization

Contents

- List
- List Operations
- Computational Problem

List

■ List

- C와 같은 고전 프로그래밍 언어에서는 리스트 구조와 기능을 개발자가 직접 프로그래밍하여 사용하여야 했음
- Python에서는 list 자료구조를 기본으로 지원
- Python의 list는 7장의 리스트 자료구조(Chapter 7, List data organization) 개념과 기능을 프로그래밍 언어로 구현한 것
- 여러 개의 자료를 순서대로 나열하여 표현할 때 편리
ex) 수업을 듣는 학생들의 학번, 일년간의 일별 온도, 10년간의 경제성장률 등

List Operations

■ 리스트 표현

- 리스트 표현: [] 안에 쉼표로 구분된 아이템들을 나열
- 변수에 리스트 할당: identifier = [item1, item2...]

numbers = [10, 20, 30, 40, 50, 60]

numbers	
Index	item
0	10
1	20
2	30
3	40
4	50
5	60

List Operations

- 리스트 값의 접근
 - identifier[index]: index 위치의 아이템에 접근


numbers		
Index	item	
0	10	← numbers[0]
1	20	← numbers[1]
2	30	← numbers[2]
3	40	← numbers[3]
4	50	← numbers[4]
5	60	← numbers[5]

List Operations

■ 리스트 값의 범위 접근

- identifier[start:end] : start 인덱스 위치 부터 end 인덱스 위치 이전까지의 아이템들을 리스트로 반환

numbers	
Index	item
0	10
1	20
2	30
3	40
4	50
5	60



numbers[:2] == [10, 20]

numbers[2:5] == [30, 40, 50]

numbers[4:] == [50, 60]

List Operations

- 리스트 값의 갱신
 - 리스트의 인덱스 위치에 새로운 아이템을 할당

number	
Index	item
0	10
1	20 25
2	30
3	40
4	50
5	60 'Hi '

`numbers[1] = 25`

`numbers[5] = 'Hi '`

List Operations

■ 리스트 값의 추가

- `identifier.append(item)`: 리스트의 가장 마지막 아이템 다음에 새로운 아이템을 추가

numbers	
Index	item
0	10
1	25
2	30
3	40
4	50
5	'Hi'
6	70

`numbers.append(70)`

List Operations

■ 리스트 값의 삽입

- `identifier.insert(index, element)`: 인덱스 위치 앞에 새로운 아이템을 삽입

numbers	
Index	item
0	10
1	15
2	25
3	30
4	40
5	50
6	'Hi'
7	70

`numbers.insert(1, 15)`

List Operations

- 리스트 값의 삭제
 - `del identifier[index]`: 인덱스 위치의 아이템을 제거

numbers	
Index	item
0	10
1	15
2	25
3	30
4	40
5	50
6	'Hi'
6	70

`del numbers[6]`

List Operations

Exercise!

```
>>> lst = [1, 2, 3, 4, 5]
>>> lst
```

```
>>> lst.insert(3, 'New')
>>> lst
```

```
>>> lst[0]
```

```
>>> del lst[3]
>>> lst
```

```
>>> lst[1] = 2.5
>>> lst
```

```
>>> lst2 = [2, 4, 6]
>>> new_list = lst + lst2
>>> new_list
```

```
>>> lst.append(6)
>>> lst
```

List Operations

Exercise!

사용자로부터 5개의 과일 이름을 입력 받아 리스트에 저장하고, 입력 받은 모든 과일 이름을 출력하는 프로그램을 작성하세요

```
fruitList = []

i = 0
while i < 5:
    fruit = input('Enter a fruit: ')
    fruitList.append(fruit)
    i = i + 1

print(fruitList)
```

```
Enter a fruit: banana
Enter a fruit: apple
Enter a fruit: kiwi
Enter a fruit: orange
Enter a fruit: cherry
['banana', 'apple', 'kiwi', 'orange', 'cherry']
>>>
```

Computational Problem

The Problem

Lotto Number Generator

Lotto 판매기에 설치할 Lotto 일련 번호 자동 생성 프로그램을 작성하세요. Lotto 일련 번호는 1에서 45 사이의 중복되지 않는 정수 6개로 구성되며 순서는 상관 없습니다. 프로그램을 실행하면 자동으로 1개의 Lotto 일련 번호를 출력합니다

Lotto Number Generator

Problem Analysis

- ① 프로그램이 시작되면 Lotto 번호들을 저장할 빈 리스트를 준비
- ② 무작위로 1에서 45사이의 1개의 정수를 생성하여 리스트에 추가
- ③ 만약 리스트에 삽입하려는 정수가 존재하면 다른 정수를 생성하여 추가
- ④ 리스트에 6개의 정수가 채워질 때 까지 2번과 3번을 반복
- ⑤ 리스트에 번호가 6개가 되면 리스트 내용을 화면에 출력

Lotto Number Generator

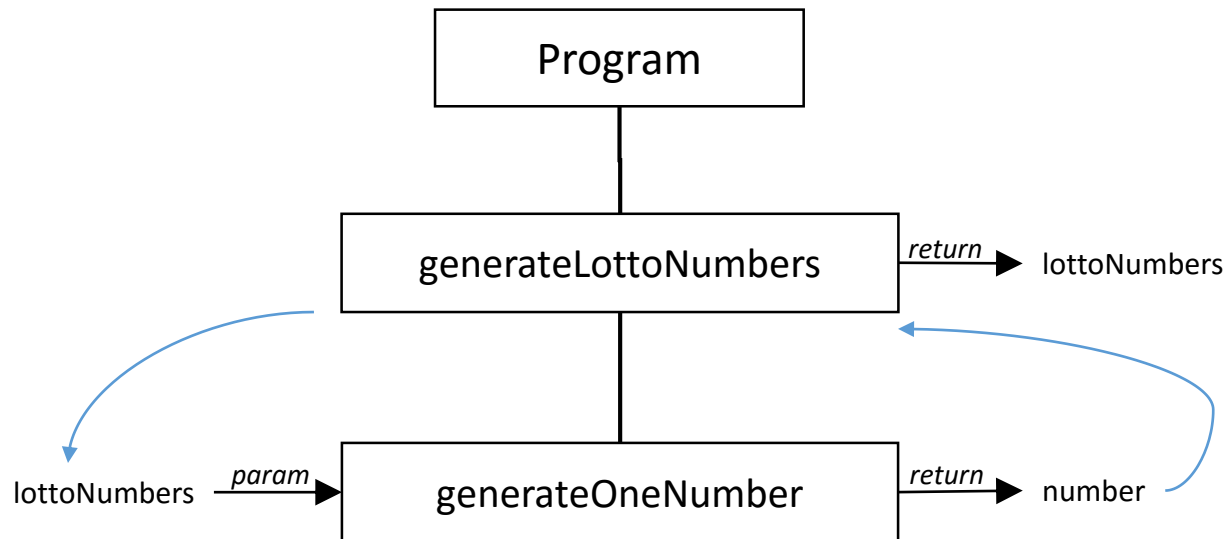
Data Representation

Lotto 일련 번호를 저장하기 위한 리스트:

lottoNumbers

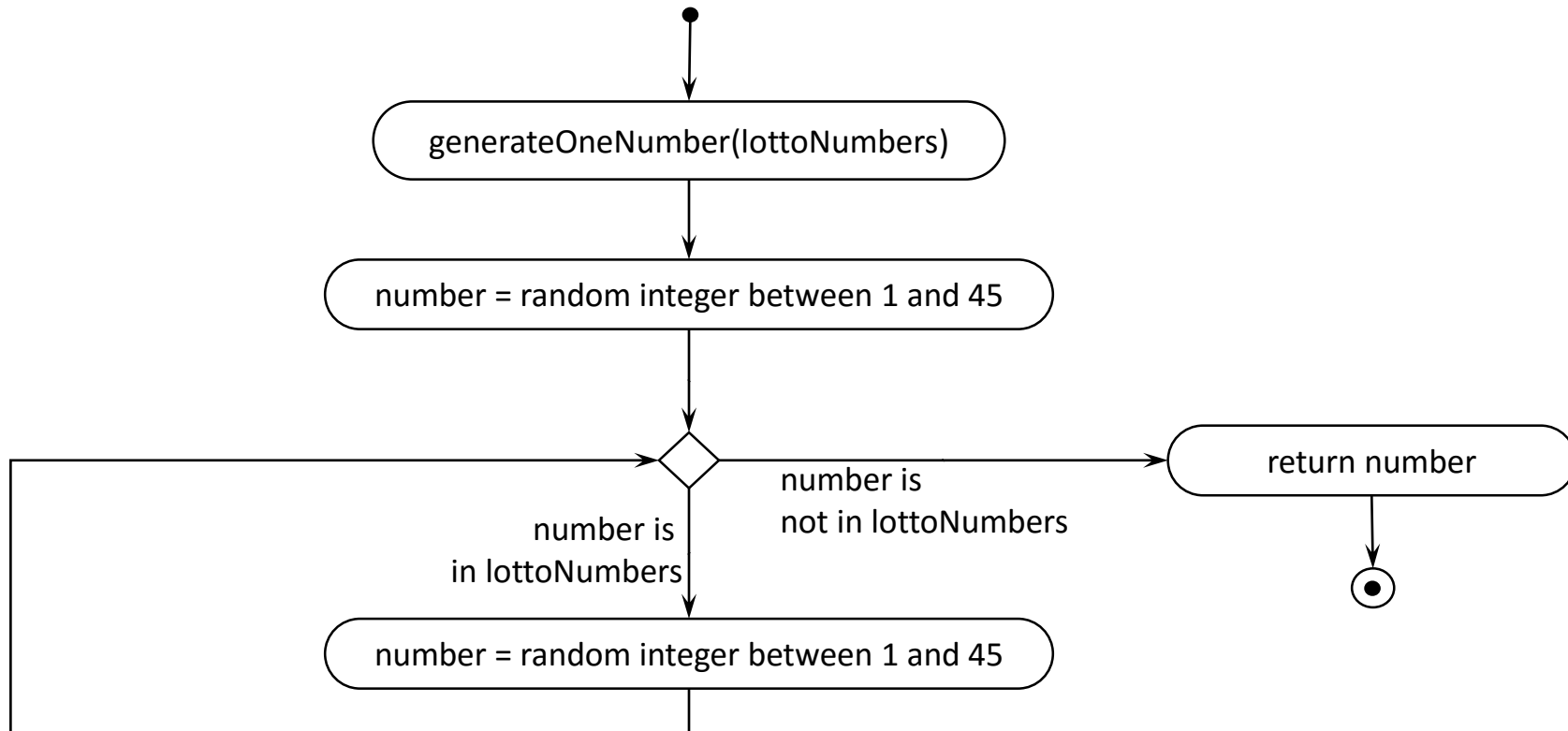
Lotto Number Generator

Decomposition



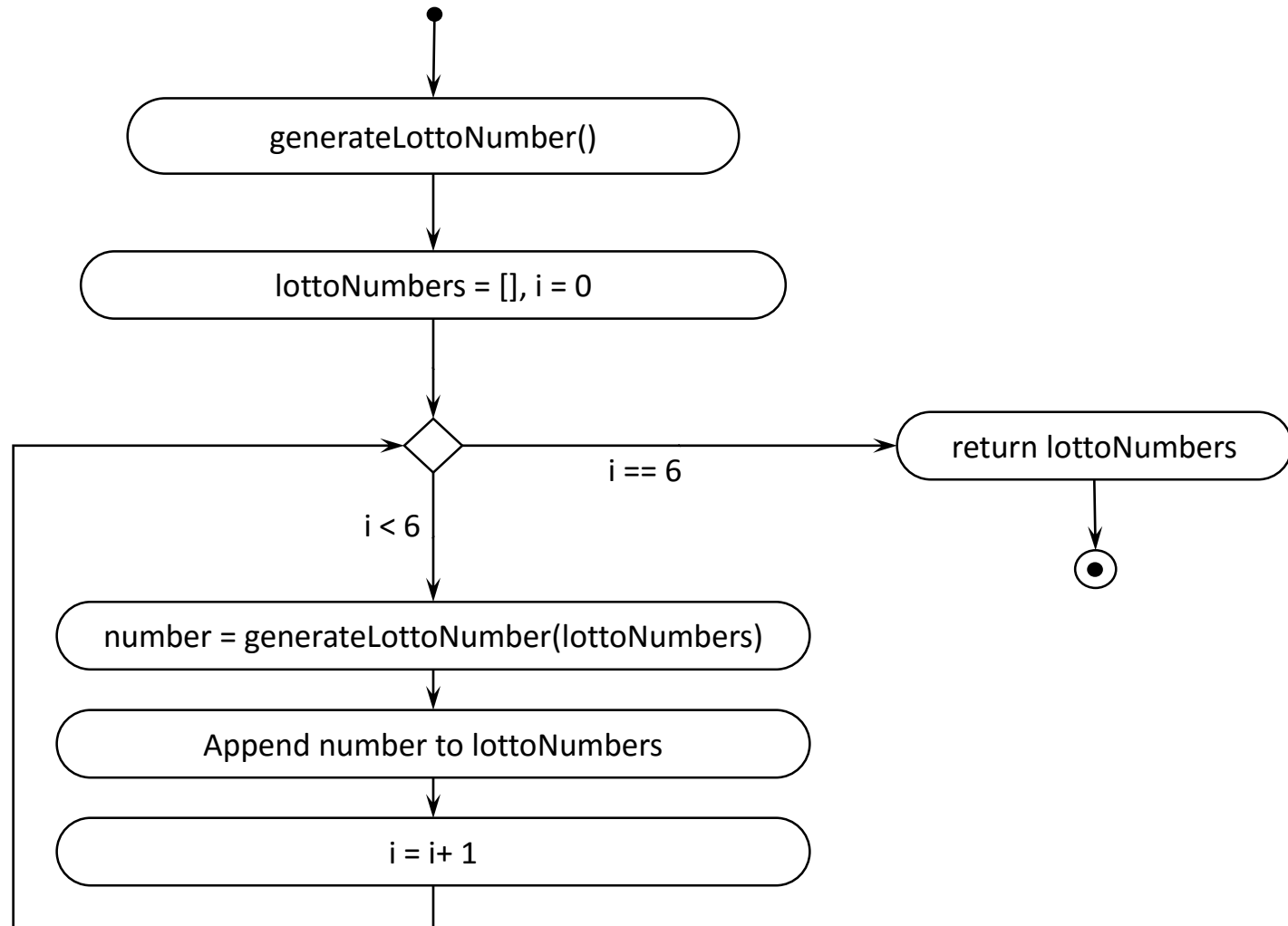
Lotto Number Generator

Algorithmic Thinking



Lotto Number Generator

Algorithmic Thinking



Lotto Number Generator

Implementation

```
import random

def generateOneNumber(lottoNumbers):
    number = random.randint(1, 45)
    while number in lottoNumbers:
        number = random.randint(1, 45)
    return number

def generateLottoNumbers():
    lottoNumbers = []
    i = 0
    while i < 6:
        number = generateOneNumber(lottoNumbers)
        lottoNumbers.append(number)
        i = i + 1
    return lottoNumbers

lottoNumbers = generateLottoNumbers()
print('Your lotto number:', lottoNumbers)
```

Lotto Number Generator

Program Execution

```
Your lotto number: [17, 2, 32, 14, 12, 38]
```

```
>>>
```