Chapter 5-2

Algorithmic Thinking

Contents

- Modularization
- Modularization Flexibility
- Value Returning Function
- Importing Module
- Computational Problem

Modularization

- Modularization: Function

Syntactic representation	Python syntax
module NAME() is ACTIONS endmodule	def NAME(): statements

Example of function definition

```
def grillSteak():
    steakTemp = 75
    minOnGrill = 0
    while steakTemp < 135:
        steakTemp = steakTemp + 13
        minOnGrill = minOnGrill + 3
```

Modularization

- Modularization: Function

Syntactic representation	Python syntax
module NAME() is ACTIONS endmodule	def NAME(): statements

Example of function call

```
grillSteak()
```

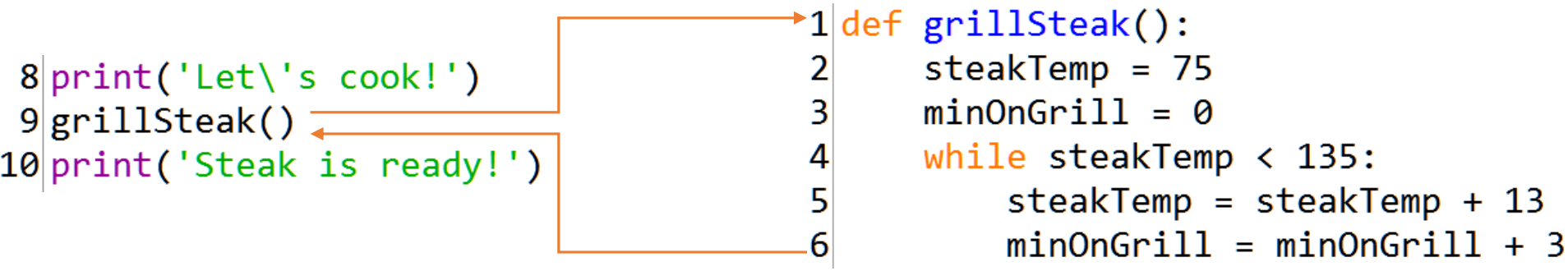
Modularization

- Modularization: Function
  - grillSteak()의 실행 과정

```
1  def grillSteak():
2      steakTemp = 75
3      minOnGrill = 0
4      while steakTemp < 135:
5          steakTemp = steakTemp + 13
6          minOnGrill = minOnGrill + 3
7
8  print('Let\'s girll a steak!')
9  grillSteak()
10 print('Steak is ready!')
```

Modularization

- Modularization: Function
  - grillSteak()의 실행 과정



  - grillSteak()를 호출 할 때마다 같은 코드가 실행

반복적으로 사용되는 코드
or
기능이 구분되는 코드

Define a function!!

Module Flexibility

- Module Flexibility: Function with Parameters

Syntactic representation	Python syntax
module NAME(V_1, V_2, …, V_n) is ACTIONS endmodule	def NAME(V_1, V_2, …, V_n): statements

Example of function definition

```
def grillSteak(steakTemp, targetTemp, increaseAmount):
    while steakTemp < targetTemp:
        steakTemp = steakTemp + increaseAmount
```

Module Flexibility

- Module Flexibility: Function with Parameters

Syntactic representation	Python syntax
module NAME(V_1, V_2, …, V_n) is ACTIONS endmodule	def NAME(V_1, V_2, …, V_n): statements

Example of function call

```
grillSteak(65, 130, 2)
```

65 130 2

```
def grillSteak(steakTemp, targetTemp, increaseAmount):
    while steakTemp < targetTemp:
        steakTemp = steakTemp + increaseAmount
```

Module Flexibility

- Module Flexibility: Function with Parameters

Syntactic representation	Python syntax
module NAME(V_1, V_2, …, V_n) is ACTIONS endmodule	def NAME(V_1, V_2, …, V_n): statements

Example of function call

```
grillSteak(94, 155, 13)
```

94 155 13

```
def grillSteak(steakTemp, targetTemp, increaseAmount):
    while steakTemp < targetTemp:
        steakTemp = steakTemp + increaseAmount
```

Module Flexibility

Exercise!

✓ 다음 코드의 출력 결과는?

```
def introduce(name):
    print('Hello, My name is',name)

introduce('John')
```

✓ 다음 코드의 출력 결과는?

```
def wonToDollar(won, rate):
    print(format(won,','),'WON =',
          format(won / rate,',.2f'),
          'dollar(s)')

wonToDollar(1000000, 1050)
wonToDollar(1000000, 1200)
```

Value Returning Function

- Value Returning Function

Syntactic representation	Python syntax
module NAME(V_1, V_2, …, V_n) is ACTIONS return RESULT endmodule	def NAME(V_1, V_2, …, V_n): statements return expression

Example of value-returning function definition

```
def grillSteak(steakTemp, targetTemp, increaseAmount):
    while steakTemp < targetTemp:
        steakTemp = steakTemp + increaseAmount
    return steakTemp
```

Example Use of Value-Returning Function Call

```
finalSteakTemp = grillSteak(65, 130, 2)
```

Value Returning Function

- Value Returning Function

```
finalSteakTemp = 131
```


```
def grillSteak(steakTemp, targetTemp, increaseAmount):
    while steakTemp < targetTemp:
        steakTemp = steakTemp + increaseAmount
    return steakTemp
```

Function Execution

Value Returning Function

Exercise!

✓ 다음 코드의 출력 결과는?

```
def grillSteak(steakTemp, targetTemp, increaseAmount):
    while steakTemp < targetTemp:
        steakTemp = steakTemp + increaseAmount
        print('Current steak temperature is', steakTemp)
    return steakTemp

finalSteakTemp = grillSteak(65, 130, 2)

print('Final steak temperature is', finalSteakTemp)
```

```
Current steak temperature is 67
Current steak temperature is 69
Current steak temperature is 71
...
Current steak temperature is 129
Current steak temperature is 131
Final steak temperature is 131
```

Importing Module

- Python modules
  - Module: 변수 및 함수 등을 모아놓은 파일
  - Module 파일을 불러와서 이미 작성된 함수나 변수를 재사용할 수 있음 → 소프트웨어의 모듈화 및 개발 비용의 단축

1. import로 Module 파일명을 불러와서 모듈 사용 준비

```
import module_name
```

2. module_name에 .(dot)을 이용하여 모듈의 함수나 변수를 사용

```
module_name.func()
        or
module_name.variable
```

Importing Module

- Python modules
  - Python은 수학, 문자열, 난수, 날짜, 사운드 등 다양한 모듈 파일을 제공하고 있음 (참고: Python module list)

Example: math module의 pi변수(π)의 사용

```
>>> import math
>>> math.pi
3.141592653589793
```

Example: random module의 randint()함수를 이용하여 1 ~ 6사이의 난수 생성

```
>>> import random
>>> random.randint(1, 6)
3
>>> random.randint(1, 6)
5
```

Importing Module

Exercise!

```
>>> import math
>>> math.sqrt(81)

>>> math.ceil(8.1)

>>> math.floor(8.9)

>>> math.round(8.5)

>>> round(8.5)
```

```
>>> import random
>>> random.random()

>>> random.randint(0, 1)

>>> random.choice([1, 3, 5])

>> import datetime
>> datetime.date.today().year

>> datetime.date.today().month

>> datetime.date.today().day
```

Importing Module

Exercise!

For fun!!

```
>>> import winsound
>>> winsound.PlaySound("SystemExit", winsound.SND_ALIAS)

>>> import webbrowser
>>> webbrowser.open('https://www.python.org/')
```

Computational Problem

The Problem

The Rock, Paper, Scissors Game

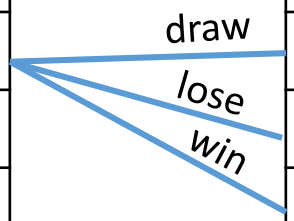
컴퓨터와 가위바위보 게임을 하여 승패를 출력하는 게임 프로그램을 구현합니다. 플레이어는 가위, 바위 또는 보를 1~3의 정수로 입력하고 컴퓨터는 무작위로 가위, 바위, 보를 결정하여 승패를 결정합니다.

The Rock, Paper, Scissors Game

Problem Analysis

- 프로그램이 시작하면 플레이어로부터 1에서 3사이의 정수를 하나 입력 받음
- 정수를 입력 받으면 컴퓨터는 1에서 3사이의 정수를 무작위로 하나 생성하고 결과를 출력(random 모듈 필요)
- 컴퓨터와 플레이어의 정수 값이 정해지면 두 개의 정수를 비교하여 승패를 판단하고 결과를 출력

플레이어 선택	player 변수값
Rock	1
Paper	2
Scissors	3



컴퓨터 선택	computer 변수값
Rock	1
Paper	2
Scissors	3

The Rock, Paper, Scissors Game

Data Representation

컴퓨터의 선택 값

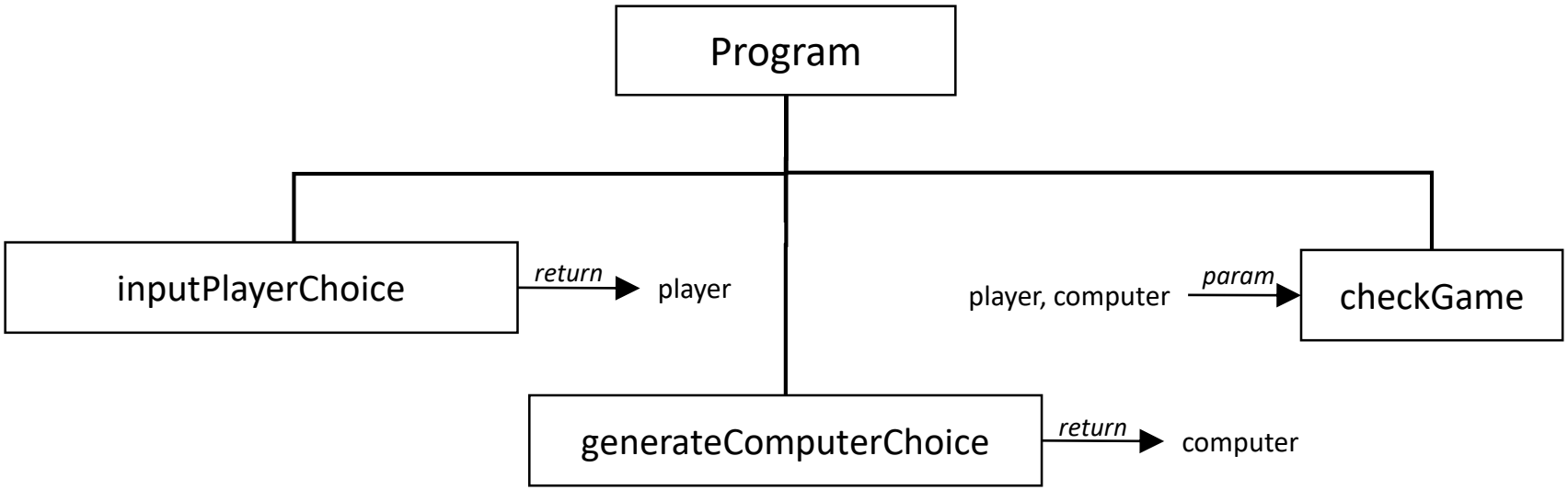
computer

플레이어의 선택 값

player

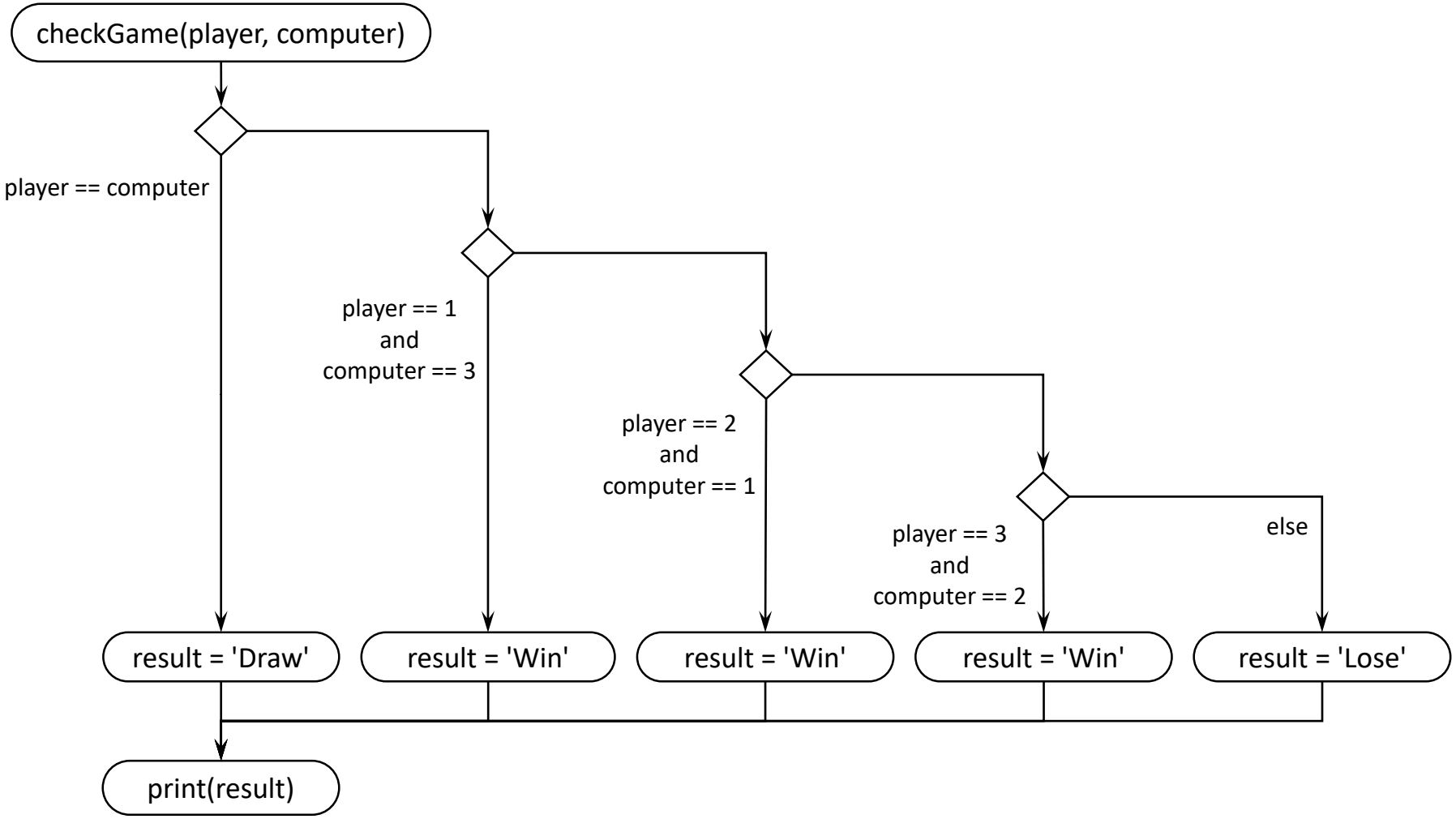
The Rock, Paper, Scissors Game

Decomposition



The Rock, Paper, Scissors Game

Algorithmic Thinking



The Rock, Paper, Scissors Game

Program Design

Program Requirements

inputPlayerChoice() function 구현
generateComputerChoice() function 구현
checkGame() function 구현

사용자가 Rock, Paper, Scissors에 해당하는 정수를 입력

Player won, Player lost, Draw를 출력

The Number Guessing Game

Program Design

Introduction

프로그램 소개문 출력

Input

사용자가 1에서 3사이의 정수 입력
컴퓨터의 정수 생성

Process

플레이어와 컴퓨터의 승패 비교

Output

결과 출력

The Rock, Paper, Scissors Game

Program Implementation

```
 1 import random
 2 def inputPlayerChoice():
 3     player = int(input('Your choice (1)Rock (2)Paper (3)Scissor : '))
 4     return player
 5
 6 def generateComputerChoice():
 7     computer = random.randint(1, 3)
 8     if computer == 1:
 9         print('Computer\'s choice: Rock')
10     elif computer == 2:
11         print('Computer\'s choice: Paper')
12     elif computer == 3:
13         print('Computer\'s choice: Scissor')
14     return computer
15
```

The Rock, Paper, Scissors Game

Program Implementation

```
16 def checkGame(player, computer):
17     if player == computer:
18         result = 'Draw'
19     elif player == 1 and computer == 3:
20         result = 'Player won'
21     elif player == 2 and computer == 1:
22         result = 'Player won'
23     elif player == 3 and computer == 2:
24         result = 'Player won'
25     else:
26         result = 'Player lost'
27     print(result)
28
29 player = inputPlayerChoice()
30 computer = generateComputerChoice()
31 checkGame(player, computer)
```

The Rock, Paper, Scissors Game

Program Execution

```
Your choice (1)Rock (2)Paper (3)Scissor : 2
Computer's choice: Paper
Draw
>>>

Your choice (1)Rock (2)Paper (3)Scissor : 1
Computer's choice: Scissor
Player won
>>>

Your choice (1)Rock (2)Paper (3)Scissor : 1
Computer's choice: Paper
Player lost
>>>
```

Importing Module

Exercise!

사용자 Module 만들기 및 불러오기

- 사용자가 작성한 파일도 module로 재사용할 수 있음!

mymodule.py

```
def sumTwoNumbers(a, b)
    return a + b
```

importing

main.py

```
import mymodule

result = mymodule.sumTwoNumbers(2, 3)
print('Result is', result)
```

output

```
Result is 5
```