CHAPTER 5

파이썬 기초 요소

학습 목표

기본 학습 목표

- 파이썬의 기본요소인 입출력 함수, 변수, 데이터 형을 이해할 수 있다.
- 파이썬의 연산자와 연산자의 우선순위를 이해할 수 있다.
- 파이썬의 선택문과 반복문을 이해할 수 있다.
- 파이썬의 기타 제어문과 주석문을 이해할 수 있다.

심화 학습 목표

- 파이썬의 기본요소를 종합적으로 활용하여 응용문제를 해결할 수 있다.

5.1 입출력

5.1.1 입력

- input()
- 계산을 위해 사용자로부터 입력 값을 받기 위한 명령어
- 예제

```
>>> input()
3
```

수행 결과

```
'3'
```

* 파이썬 IDLE : 파이썬을 다운로드(python.org)한 후, 패키지를 설치하면 기본적으로 제공되는 파이썬 명령어 처리기. '>>>' 는 명령어를 입력 받기 위한 문자열임.

5.1 입출력

- input() 함수 사용법
  - 괄호 안에 임의의 숫자와 한글, 영문 알파벳, 특수기호 등 사용 가능함.
  - 숫자를 제외한 나머지 문자들은 단일 따옴표(‘) 또는 이중 따옴표(“ “), 삼중 따옴표(“ “ “)로 감싸서 사용함.
  - 예제

```
>>> input('a = ')
a = 100
```

수행 결과

```
'100'
```

5.1 입출력

5.1.2 출력

- print()
- 괄호 안에는 0개 이상의 값들을 코마(,)로 분리하여 사용함.
- 예제

```
>>> print(input(), '을 입력하였습니다.')
파이썬 프로그래밍
```

수행 결과

```
파이썬 프로그래밍 을 입력하였습니다.
```

5.1 입출력

- 유의 사항
  - input() 함수의 결괏값은 일련의 문자들의 집합
  - print() 함수의 결괏값은 숫자이거나 문자들이거나 상관없이 해당하는 숫자와 문자들로만 표현.

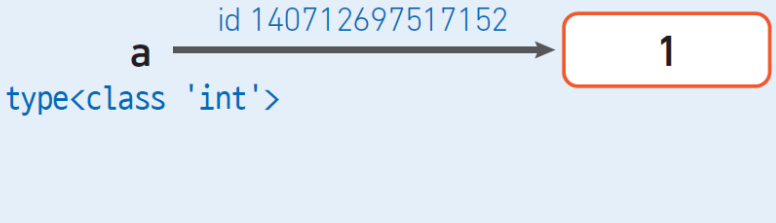
5.2 변수

- 정의
  - 파이썬 프로그램에서 특정 객체를 다루려면 해당 객체에 대한 레퍼런스가 필요함.
  - 레퍼런스(reference)는 객체 식별 및 참조에 사용되는 값.
  - 해당 객체에 접근할 수 있도록 객체에 붙여진 이름을 변수라 함.
- 생성 방법
  - 이름 = 식
    - ‘=’ 기호 우측의 식이 계산되어 값이 생성되면, 그 값을 어떤 객체가 가지게 되고, 그 객체에 붙여진 이름이 변수임.

5.2 변수

- 예제

```
>>> a = 1
>>> id(a)
140712697517152
>>> type(a)
<class 'int'>
```



- 변수의 속성
  - 변수 a의 고유식별값 id(a)
    - a로 명명된 객체의 고유식별값이 출력됨. 고유식별값은 주로 메모리 주소
  - 변수 a의 형 type(a)
    - A로 명명된 객체가 가진 값의 데이터 형. 위의 예제에서 a의 형은 정수(int).

5.2 변수

- 변수 명명 규칙
  - 변수의 이름은 알파벳 소문자와
  - 대문자(a~z, A~Z), 한글 등 각 나라의 알파벳 문자, 숫자(0~9), 언더바(_)로만 이루어짐.
  - 다른 특수 기호는 사용할 수 없음.
  - 변수 이름의 첫글자로 각 나라의 알파벳 문자 또는 언더바만 사용이 가능함.
  - 파이썬의 예약어(keywords)를 변수명으로 사용하면 안됨.

5.2 변수

- 변수명 사용 사례

적절한 변수 이름	부적절한 변수 이름
_a1	a*r
a3	_ 2
b_10	$a
d__	3a

- 파이썬 키워드

수행 결과

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break',
'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for',
'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or',
'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

5.2 변수

input() 함수와 print() 함수 사용

이번엔 input() 함수와 print() 함수를 함께 변수와 사용해보자. 사용자로부터 임의의 값을 입력받은 후, 이 값에 age라는 이름(변수 age)을 부여하고 변수 age를 이용하여 그 값을 출력해보자.

예제

```
>>> age = input('나이? ')
나이? 3
print(age)
```

수행 결과

```
3
```

5.2 변수

- 묻고 답하기

Q 묻고 답하기

문 다음과 같이 파이썬 쉘에 입력하면 화면에 무엇이 출력될까?

```
>>> print('나이는', age, '입니다.')
```

5.3 데이터 형

- 파이썬의 기본 데이터 형

〈표 5-2〉 파이썬의 기본 데이터 형

데이터 형	의미	예제	변환함수
int	정수형	... -3, -2, -1, 0, 1, 2, 3, ...	int()
float	실수형	-2.54, 0.35, 4.2e5, 3.0e-5, ...	float()
str	문자열형	'a', "강아지", '''파이썬 프로그래밍'''(여러 줄에 걸친 문자열)	str()
bool	참, 거짓을 나타내는 논리형	True, False	bool()

5.3 데이터 형

- 정수형
  - 0, ±1, ±2, ...
  - 10진수(0~9)
  - 2진수(0, 1)
  - 8진수(0~7)
  - 16(0~9, A~F)진수
  - 2진수와 8진수, 16진수는 숫자 앞에 각각 0b, 0o와 0x가 달려 있어 10진수와 구별할 수 있음.
  - 예제
    - 십진수 20은 2진수 0b10100, 8진수 0o24, 16진수 0x14(또는 0X14)

5.3 데이터 형

- 실수형
  - 125000이라는 수는 실제 1.25×105. 이 값을 파이썬에서는 1.25e5로 표현하며 실수형으로 간주
  - 0.0000125는 1.25e-5로 표현

5.3 데이터 형

- 문자형
  - 알파벳으로 구성된 문자들의 집합이며 단일 따옴표, 이중 따옴표, 삼중 따옴표로 표현
  - 문자열 안에서 각 문자는 위치값을 갖는데, 이를 인덱싱(indexing)이라고 함.
  - 예를 들어, s = ‘Python’ 이라고 하면, s[0]은 ‘P ‘가 되고, s[1]은 ‘y ‘가 된다. s[-1]은 뒤로부터 위치를 계산해 ‘n’.

5.3 데이터 형

- 문자열의 슬라이싱
  - s = ‘Python programming’에 대해 슬라이싱을 정의

〈표 5-1〉 문자열의 슬라이싱

함수	결과
print(s[0:6])	Python
print(s[7:])	programming
print(s[:6])	Python
print(s[7:-4])	program

  - 묻고 답하기

Q 묻고 답하기

문 s[-1:]은?

5.3 데이터 형

- 리스트형(list type)
  - 원소(element)라고 부르는 값들의 집합인데, 각 값은 순서 정보를 가지며 대괄호로 묶어 표현함.
  - 값의 사이에는 코마(,)가 있다. list()라는 함수 또는 []로 빈(empty) 리스트를 만들 수 있음.
  - 예를 들어, alist = [10, 20, 30]이라고 정의한다면, 변수 alist는 값이 [10, 20, 30]인 리스트 객체를 레퍼런스하기 위한 이름. 즉, alist를 리스트 변수라함.
  - alist[0]의 값은 10, alist[1]의 값은 20, alist[2]의 값은 30이 됨.

5.3 데이터 형

- 리스트형

〈표 5-3〉 파이썬의 리스트형

데이터 형	함수	결과	해당 변수
리스트 a=[1, 2, 3] b=['a', 'b'] c = [[1,2], [20, 30]]	a.append(4) a.insert(2,10) a.remove(3) del a[0] i = a.index(10) d = a + b c[0][1] += c[1][1]	[1, 2, 3, 4] [1, 2, 10, 3, 4] [1, 2, 10, 4] [2, 10, 4] 1 [2, 10, 4, 'a', 'b'] [[1, 32], [20, 30]]	a a a a i d c

5.3 데이터 형

- 딕셔너리형(dictionary type)
  - 사전처럼 키(key)와 키에 대한 의미(value)의 쌍을 여러 개 가지고 있는 집합형
  - 쌍 사이에 코마(,)가 있고 함수 dict()로 초기화 됨.
  - 순서를 가지지 않고 키를 이용하여 값을 찾음.
  - 킷값이 중복되면 안되고, 리스트나 딕셔너리를 킷값으로 사용할 수 없음.
  - 예제
    - countries = { “대한민국” :” 서울” , “미국” :” 워싱턴DC” , “노르웨이” :” 오슬로” }
    - countries[“대한민국”]의 값은 “서울” 이고, countries[“미국”]의 값은 “워싱턴DC”

5.3 데이터 형

- 딕셔너리형

〈표 5-4〉 파이썬의 딕셔너리형

데이터 형	함수	결과
딕셔너리 {d='a':100, 'b':95}	d['c']=80	{'a': 100, 'b': 95, 'c': 80}
	del d['a']	{'b': 95, 'c': 80}
	d.keys()	dict_keys(['b', 'c'])
	d.values()	dict_values([95, 80])
	d.items()	dict_items([('b', 95), ('c', 80)])
	print(d.get('b'))	95

- 묻고 답하기

Q 묻고 답하기

문 〈표 5-4〉에서 dict_keys나 dict_values 값을 리스트형으로 바꾸려면?

5.3 데이터 형

- 집합형(*set type*)
  - 원소(element)라고 부르는 값들의 집합
  - 리스트와 다르게 순서가 없고, 원소 값이 중복되지 않음.
  - 중괄호{}로 묶어 표현하고 값의 사이에는 코마(,)가 있음.
  - 순서가 없으므로 원소의 위치(index)를 알 수 없음.
  - 집합 연산자
    - 교집합(&), 합집합(|), 차집합(-)
    - 원소를 추가하는 add()
    - 여러 원소를 함께 추가하는 update()
    - 원소를 삭제하는 remove()

5.3 데이터 형

〈표 5-5〉 파이썬의 집합형

데이터 형	함수	결과
집합(set) a={1, 2} b={2, 3}	a ¦ b 또는 a.union(b) a & b 또는 a.intersection(b) a – b 또는 a.difference(b) a.add(4) b.update([5, 6]) a.remove(2)	{1, 2, 3} {2} {1} {1, 2, 4} {2, 3, 5, 6} {1, 4}

5.3 데이터 형

- 투플형(*tuple type*)
  - 리스트형과 유사
  - 리스트는 여러 원소들을 대괄호([])로 묶어 표현하지만, 투플은 원소들을 소괄호(())로 묶어 표현하거나 코마로 나열하여 표현.
  - 예제
    - a = (1, 2, 3) 또는 a = 1, 2, 3으로 표현할
  - 원소의 값들은 새로운 원소값의 추가, 변경, 삭제가 가능하지만 투플은 변경할 수 없다.
  - 투플은 한 개의 원소를 갖는 경우 한 개의 원소 뒤에 코마를 찍어 표현한다.
  - 인덱싱, 슬라이싱, 더하기, 곱하기, 투플 원소 개수 구하기가 가능.

5.3 데이터 형

〈표 5-6〉 파이썬의 투플형

데이터 형	함수	결과
투플(tuple) a= 1, 2, 3 b = (10,)	a[1] a[0:2] a + b b * 3 len(a)	1 (1, 2) (1, 2, 3, 10) (10, 10, 10) 3

5.3 데이터 형

- 예제

```
>>> age = input("나이? ")
나이? 3
>>> type(age)
<class 'str'>
>>> age = int(age)
>>> type(age)
<class 'int'>
>>> age
3
```

- 변수 age의 값이 3
- 데이터 형을 구하기 위하여 새로운 함수 type()을 사용
- type(age)라고 하면, <class ‘str’>이라고 하는 결괏값을 받음.
- age로 명명된 객체에는 문자열형의 값이 저장되어 있는 것을 확인할 수 있다.

5.3 데이터 형

- 이차 리스트
  - 한 리스트의 원소값의 데이터 형이 리스트
- 리스트 예제

```
>>> numbers = [10, 20, 30]
>>> numbers[0]
10
>>> numbers[1]
20
>>> numbers[2]
30
>>> twolines = [[1, 2, 3], [10, 20, 30]]
>>> twolines[0]
[1, 2, 3]
>>> twolines[1]
[10, 20, 30]
>>> twolines[1][2]
30
```

5.3 데이터 형

- 딕셔너리 예제

```
>>> puppy = dict()
>>> puppy["이름"] = "밍키"
>>> puppy["나이"] = 3
>>> puppy["몸무게"] = 10.5
>>> puppy
{'이름': '밍키', '나이': 3, '몸무게': 10.5}
>>> apuppy = {"이름" : "버니", "나이" : 2, "몸무게" : 9.0}
>>> apuppy
{'이름': '버니', '나이': 2, '몸무게': 9.0}
```

5.4 연산

- 정의
  - 피연산자를 이용하여 연산자의 정의에 맞게 계산한 후, 하나의 값을 결괏값으로 제시하는 과정
- 연산자
  - 산술(arithmetic)연산을 위한 **산술 연산자**(+, -, *, /, %, ** 등)
  - 값의 크기를 비교(comparison)하는 **비교 연산자**(<, <=, >, >=, ==, !=, is 등)
  - True와 False 값을 갖는 논리형의 피연산자를 대상으로 논리합, 논리곱, 논리 부정을 수행하는 **논리(logical) 연산자**(or, and, not)

5.4 연산

기본 연산자

연산자	의미(피연산자의 데이터 형)	예제	결과값
+	덧셈(정수형, 실수형)	7.5 + 3	10.5
+	문자열 잇기(문자열형)	'7.5' + '3'	'7.53'
-	뺄셈(정수형, 실수형)	7.5 - 3	4.5
*	곱셈(정수형, 실수형)	7.5 * 3	22.5
*	반복(문자열형, 정수형)	'7.5' * 3	'7.57.57.5'
**	지수(정수형, 실수형)	7.5 ** 3	421.875
/	나눗셈(정수형, 실수형)	7.5 / 3	2.5
//	나눗셈(정수형, 실수형)	7.5 // 3	2.0
%	나머지 구하기(정수형)	7 % 3	1

5.4 연산

연산자	의미(피연산자의 데이터 형)	예제	결과값
==	같다(딕셔너리 제외한 모든형)	4 == 4	True
!=	같지 않다(위와 같음)	4 != 4	False
<, <=	작다, 작거나 같다(위와 같음)	4 < 4	False
>, >=	크다, 크거나 같다(위와 같음)	4 >= 4	True
is	변수의 대상체(id)가 같다	4 is 4	True
in	연산자 왼쪽 값이 오른쪽	2 in [1, 2, 3]	True
or	논리합(부울형)	True or False	True
and	논리곱(부울형)	True and False	False
not	논리부정(부울형)	not True	False

5.4 연산

연산자	의미(피연산자의 데이터 형)	예제	결과값
&	비트 단위의 and 연산	5 & 3	1
|	비트 단위의 or 연산	5 | 3	7
^	비트 단위의 xor(exclusive-or) 연산	5 ^ 3	6
~	비트 단위 보수(complement) 연산	~5	-6
<<	비트 단위로 왼쪽으로 밀기	5<<3	40
>>	비트 단위로 오른쪽으로 밀기	5>>3	0

5.4 연산

■ 연산자의 우선순위 규칙

〈표 5-8〉 파이썬의 연산자 우선순위

연산자	기술
**	지수 연산자
~, +, -	보수, 단항 플러스, 단항 마이너스 연산자
*, /, %, //	곱하기, 나누기, 나머지, 몫 연산자
+, -	덧셈, 뺄셈 연산자
>>, <<	좌우 비트 밀기(shift) 연산자
&	비트 and 연산자
^, ¦	비트 XOR, 비트 or 연산자
<=, <, >, >=	비교 연산자
<>, ==, !=	비교 연산자
=, %=, /=, //=, -=, +=, *=, **=	치환 연산자, 축약 치환 연산자
is, is not	식별 연산자
in, not in	멤버(member) 연산자
not, and, or	논리 연산자

5.4 연산

- 식(expression)
  - 피연산자와 연산자들의 집합으로 이루어진 것을 연산식 또는 식이라 함.
  - 평가(evaluation)가 가능하며 하나의 값을 가짐.
  - 피연산자로 변수, 상수, 함수 등의 다양한 객체가 사용 가능함.
  - 예제
    - 변수 a가 5, 변수 b가 3, 변수 c가 3 일 때,
    - 식 a + b // 3은 a + (b // 3). 따라서, 5 + 1 = 6
    - a + b * c > a * b + 3은 (a + (b * c)) > ((a * b) + 3). 따라서, 14 > 18 ⇒ False
    - a < b or a – b and b > 0은 ((a < b) or ((a – b) and (b > 0)))
      - (False or (2 and True))⇒(False or True)⇒True

5.4 연산

유의사항

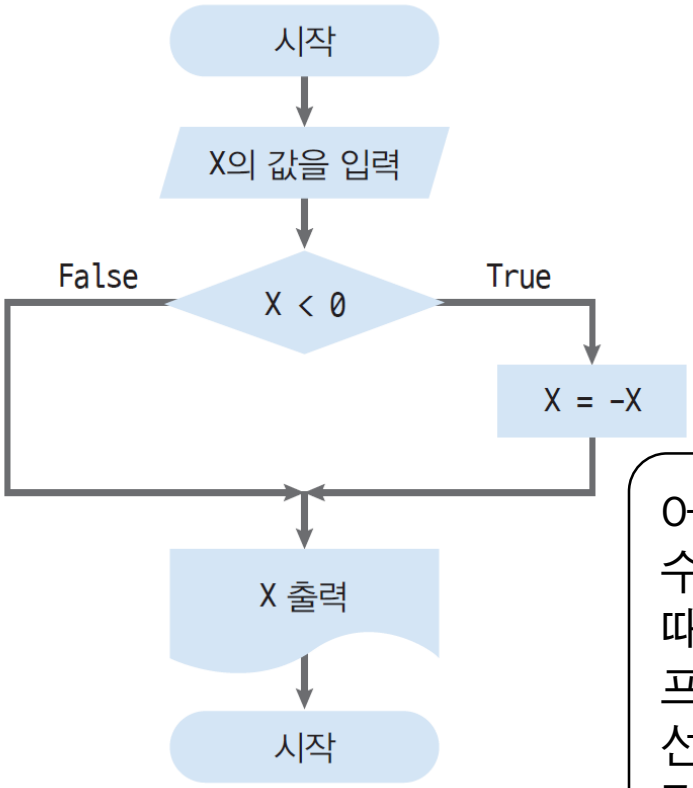
- 연산자의 우선순위를 정확하게 기억하지 못하면 연산의 결과를 보장할 수 있도록 소괄호(())를 사용하여야 한다. 소괄호는 어떤 연산자보다 높은 우선순위를 가진다.
- 연산자 == 와 =의 차이를 구별하자. 연산자 ==는 두 개의 피연산자값이 동일한지 여부를 판단하는 비교 연산자이다. 연산자 =는 오른쪽 피연산자의 값을 왼쪽 피연산자에게 부여하는 치환 연산자이다.

5.5 선택

- 구조적 프로그래밍
  - 1970년대 GOTO 문을 해결하기 위한 노력
  - 프로그램을 순차, 선택, 반복으로 표현가능하다고 주장.
- 프로그램
  - 프로그램이란 문장(statement)들이 순차적으로 기술된 문장들의 집합.
- 선택
  - 모든 문장을 순차적으로 처리하는 것이 아니라 선택적으로 처리함.

5.5 선택

- 예제 : 절대값 구하기



[그림 5-1] 절대값 구하는 순서

어떤 문장(x = -x)을 항상 수행하지 않고 조건이 만족할 때만 수행하도록 프로그래밍 해야 할 때, 선택문인 if, elif, else와 같은 키워드를 사용할 수 있다.

5.5 선택

- 선택문 종류

(가)

```
if 조건식 :
    문장-1
    문장-2
    ...
```

(나)

```
if 조건식 :
    문장-1
    문장-2
    ...
else :
    문장-1
    문장-2
    ...
```

(다)

```
if 조건식1 :
    문장-1
    문장-2
    ...
elif 조건식2 :
    문장-1
    문장-2
    ...
else :
    문장-1
    문장-2
    ...
```

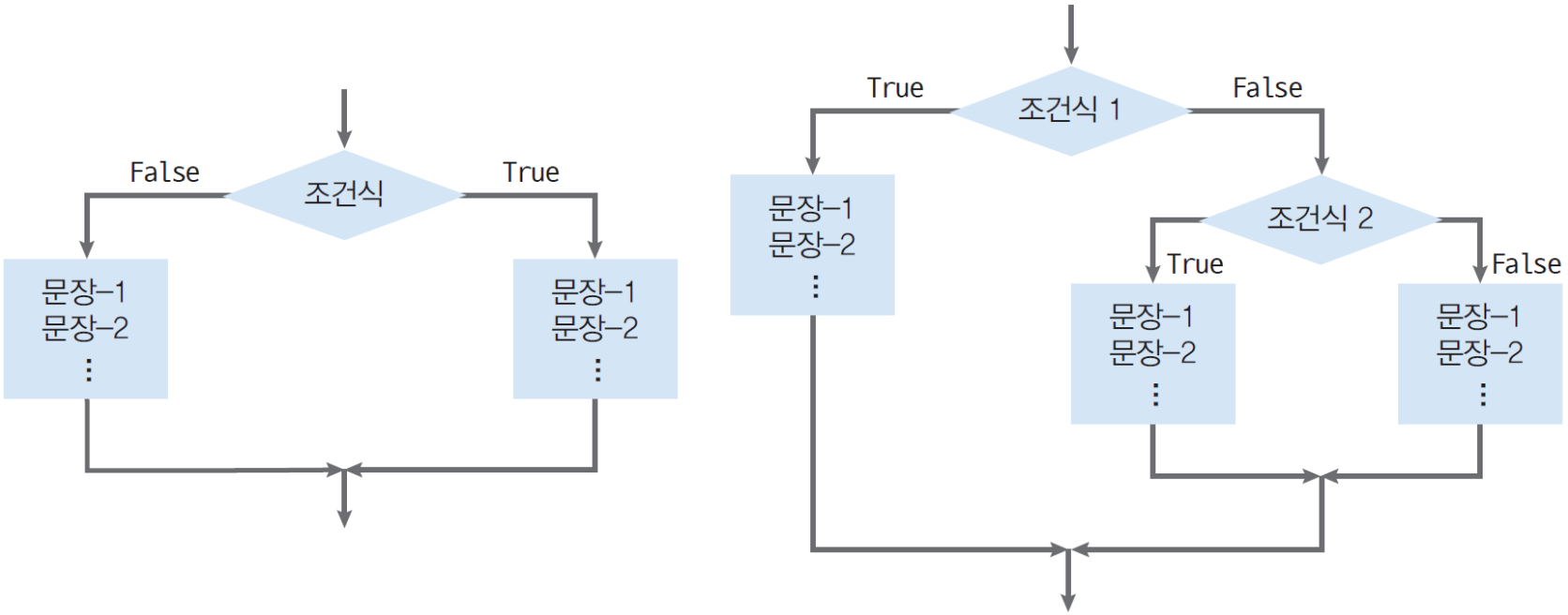
(라)

```
if 조건식1 :
    문장-1
    문장-2
    ...
elif 조건식2 :
    문장-1
    문장-2
    ...
elif 조건식n :
    문장-1
    문장-2
    ...
else :
    문장-1
    문장-2
    ...
```

[그림 5-2] 선택문의 종류

5.5 선택

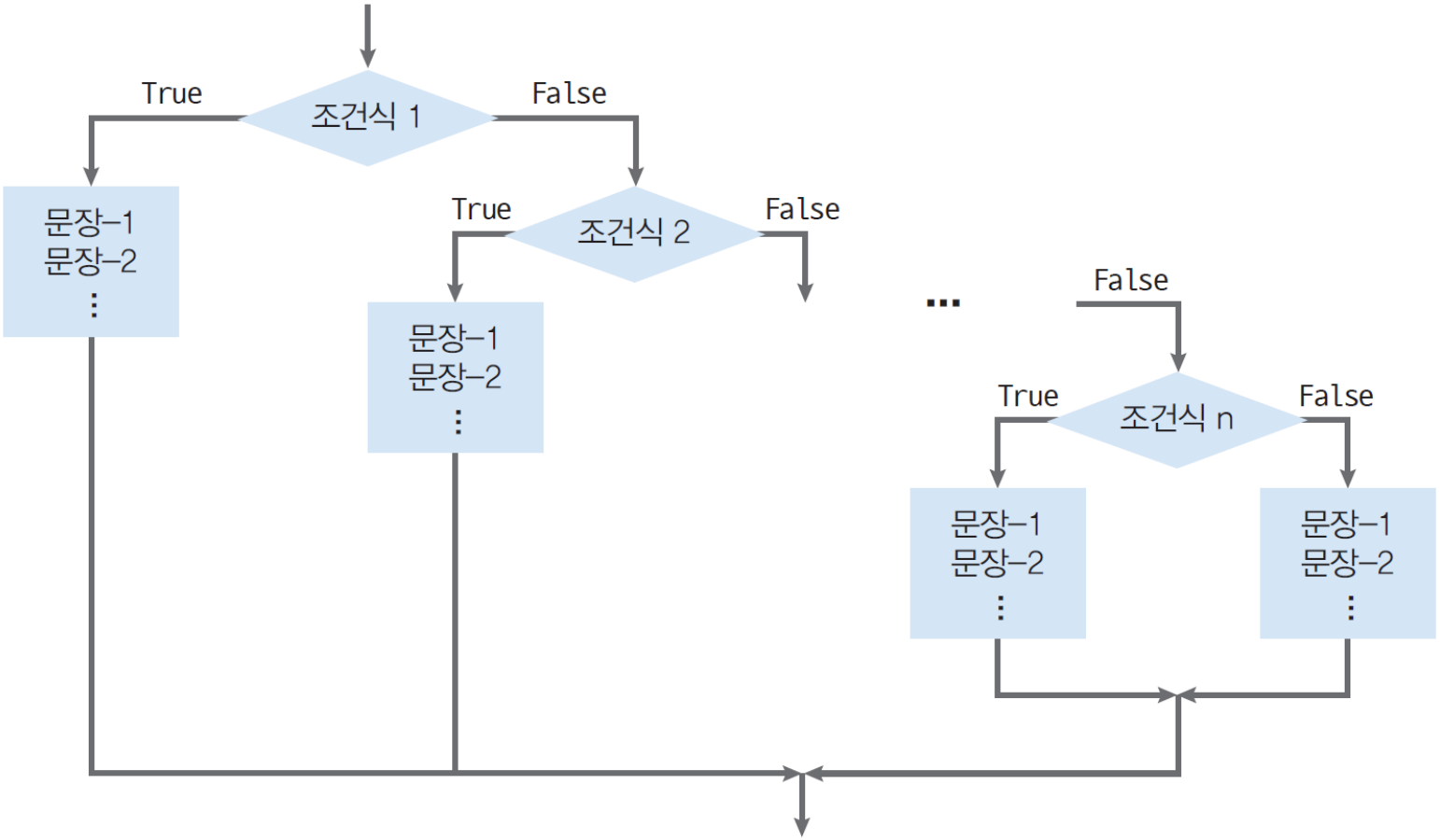
- 선택문 순서도



(가) if-else 문 (나) if-elif-else 문

[그림 5-3] 선택문 순서도

5.5 선택



(다) 여러 개의 elif 문을 갖는 if 문

[그림 5-3] 선택문 순서도

5.5 선택

■ 선택문의 예제

다음은 if 문만 사용하는 사례를 살펴보자. "한국공원의 입장료는 1만원이다. 그런데, 만약 나이가 65세 이상이면 입장료의 20%를 할인받는다." 사람의 나이를 입력받아, 입장료를 계산하는 프로그램을 작성하여보자. 지금부터는 파일에 파이썬 문장들을 저장하기로 한다.

```
age = int(input("나이? "))
fee = 10000
if (age >= 65) :
    fee = int(fee * 0.8)

print("입장료는 ", fee, "원입니다.")
```

5.5 선택

- 선택문의 다른 예제
  - 윤년 계산하기

윤년이란 2월이 28일이 아니고 29일인 해이다. 윤년을 계산하는 방식은 400으로 나누어지는 해이거나 4로 나누어지나 100으로는 나누어지지 않는 해로 정의된다.

```
year = int(input("년도? "))
if (year % 400 == 0 or (year % 4 == 0 and year % 100)) :
    print(year, "년은 윤년이다.")
else:
    print(year, "년은 윤년이 아니다.")
```

수행 결과

```
년도? 2019
2019 년은 윤년이 아니다.
-----------------------------------------------------------------------
년도? 2020
2020 년은 윤년이다.
```

5.5 선택

- 연습문제
  - 한국대학교에서 학년과 성별에 따라 기숙사를 배정하고자 한다. 만일, 여성이면 모두 진리관에 배정한다. 만일, 남성이면서 1, 2 학년이면 정의관, 남성이면서 3, 4학년이면 창조관에 배정한다. 그 밖의 입력값은 오류 처리한다. 기숙사를 배정하는 프로그램은?

```
gender = input("성별 (여, 남) >> ")
year = int(input("학년 >> "))

if (gender == "여"):
    print("진리관에 배정이 되었습니다.")
elif (gender == "남" and (year == 1 or year == 2)):
    print("정의관에 배정이 되었습니다.")
elif (gender == "남" and (year == 3 or year == 4)):
    print("창조관에 배정이 되었습니다.")
else:
    print("입력 오류입니다.")
```

5.6 반복

- 정의
  - 특정 문장집합을 여러 번 반복적으로 사용할 때 사용됨.
- 종류 : for와 while

```
for 변수 in 리스트(또는 투플, 문자열):
    문장-1
    문장-2
    ...
```

(가) for 문

```
while <조건식>:
    문장-1
    문장-2
    ...
```

(나) while 문

[그림 5-4] 반복문 구문

5.6 반복

5.6.1 for 문

- 예제
  - 리스트에 저장된 킷값들을 대상으로 사용자로부터 입력을 받아서 원하는 킷값이 있는지 탐색해보자.

```
numbers = [24, 1, 98, 20, 10, 33]
userno = int(input("찾고자 하는 킷값 >> "))
find = False
for no in numbers:
    if no == userno :
        print("키값 %d : 찾았습니다." % userno)
        find = True
        break

if find == False:
    print("키값 %d : 목록에 없습니다." % userno)
```

5.6 반복

- range() 함수
  - 일련의 정수들로 이루어진 리스트를 결괏값으로 갖는다.
  - range(5)라고 하면 [0,1, 2, 3, 4]를 생성한다. 5는 포함되지 않는다.
  - for 문장과 함께 잘 사용이 된다.

5.6 반복

- 예제

```
for value in range(5) :
        print(value)
```

```
for value in range(5, 10, 2) :
        print(value)
```

수행 결과

```
0
1
2
3
4
```

```
5
7
9
```

5.6 반복

- 예제
  - A교사는 3학년 전교생의 키를 조사하여 가장 작은 학생, 가장 큰 학생, 평균 키를 구한다고 가정하자. 데이터를 리스트로 가지고 있으며 이를 자동으로 프로그래밍하려 한다.

5.6 반복

- 답

```
minh=3.0
maxh=0.0
avgh=0.0
height=[1.50, 1.44, 1.38, 1.55, 1.58, 1.65, 1.35, 1.48]
for i in range(len(height)):
    if height[i] < minh :
        minh = height[i]
    elif height[i] > maxh :
        maxh = height[i]
    avgh += height[i]
avgh = avgh/len(height)

print("가장 작은 키 : %5.2f, 가장 큰 키 : %5.2f, 평균 키 : %5.2f" %
      (minh, maxh, avgh))
```

- len()이라는 함수는 매개변수의 원소 개수를 결괏값으로 반환함.
- 옆 예에서, len(height)는 height 리스트 원소 개수인 8을 값으로 함.
- Print() 함수에서 사용된 (1) “~%5.2f ~ %5.2f ~ %5.2f”, (2) %, (3) (minh, maxh, avgh)는 각각 (1) 포맷문자열, (2) 분리자, (3) 출력할 변수들을 나타냄.
- 포맷문자열 안의 %5.2f는 다섯개의 자리수를 최소 출력할 수 있는 자리를 확보하여 그중 두 자리는 소수점 이하 실수를 위한 자리로 사용하라는 의미를 나타낸다. F는 실수를 의미함.

5.6 반복

5.6.2 while 문

- 앞의 예를 while문으로 변경하기

```
 minh=3.0
 maxh=0.0
 avgh=0.0
 height=[1.50, 1.44, 1.38, 1.55, 1.58, 1.65, 1.35, 1.48]
 i = 0
nodata = len(height)
while (i < nodata) :
    if height[i] < minh :
        minh = height[i]
    elif height[i] > maxh :
        maxh = height[i]
    avgh += height[i]
    i += 1
avgh = avgh/len(height)

print("가장 작은 키 : %5.2f, 가장 큰 키 : %5.2f, 평균 키 : %5.2f" %
      (minh, maxh, avgh))
```

5.6 반복

5.7 기타 제어

5.7.1 continue와 break 문

예제

```
numbers = 5
data = [10, -20, 40, 30, -5]
sum = 0
for n in data :
    if n < 0 :
        continue
    sum += n

print("sum = ", sum)
```

```
numbers = 5
data = [10, -20, 49, 30, -5]
sum = 0
for n in data :
    if n < 0 :
        break
    sum += n

print("sum = ", sum)
```

수행 결과

```
sum = 80
```

```
sum = 10
```

5.6 반복

5.7.2 pass 문

pass 문은 어떤 조건에 대해서는 아무 일도 처리하지 않고 그냥 다음을 처리할 때 사용한다. 예를 들어, 어떤 정수형 변수 n의 값이 음수는 처리하지 않고 0과 양수만 처리하려할 때 또는 3부에서 소개될 함수 이름만 정의해 두고자 할 때 사용한다.

```
if  n < 0 :
    pass
else: ...
```

```
def functionA() :
    pass
```

5.6 반복

5.7.3 주석문

파이썬에서 프로그램을 설명하기 위해서 한 줄 이내의 주석문을 사용할 때에는 # 기호를 사용하여 표현한다.

```
no = int(input("몇 명 ? "))   # 정렬할 데이터의 개수
```

여러 줄에 걸친 주석을 나타낼 때에는 '''를 사용하거나 """를 사용하면 된다.