>>>>>>>>>>> 제어 구조의 설계 원리를 중심으로 배우는 >>>>>>>>>>>

프로그래밍의 정석

파이썬

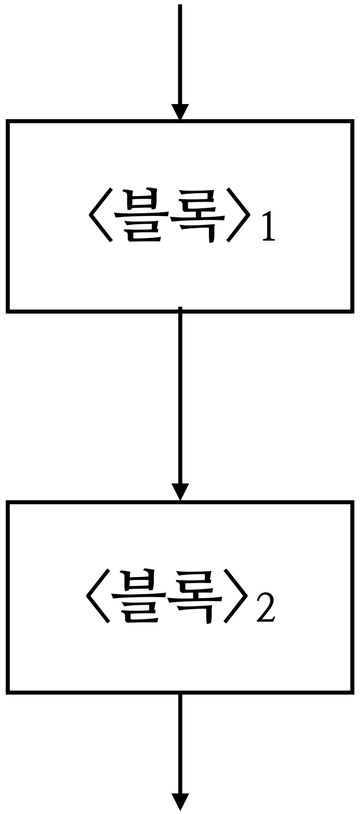
도경구 지음

생능출판

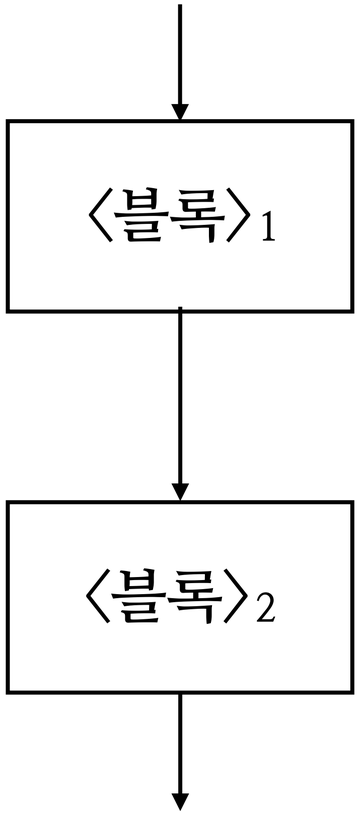
CHAPTER 3

제어 구조

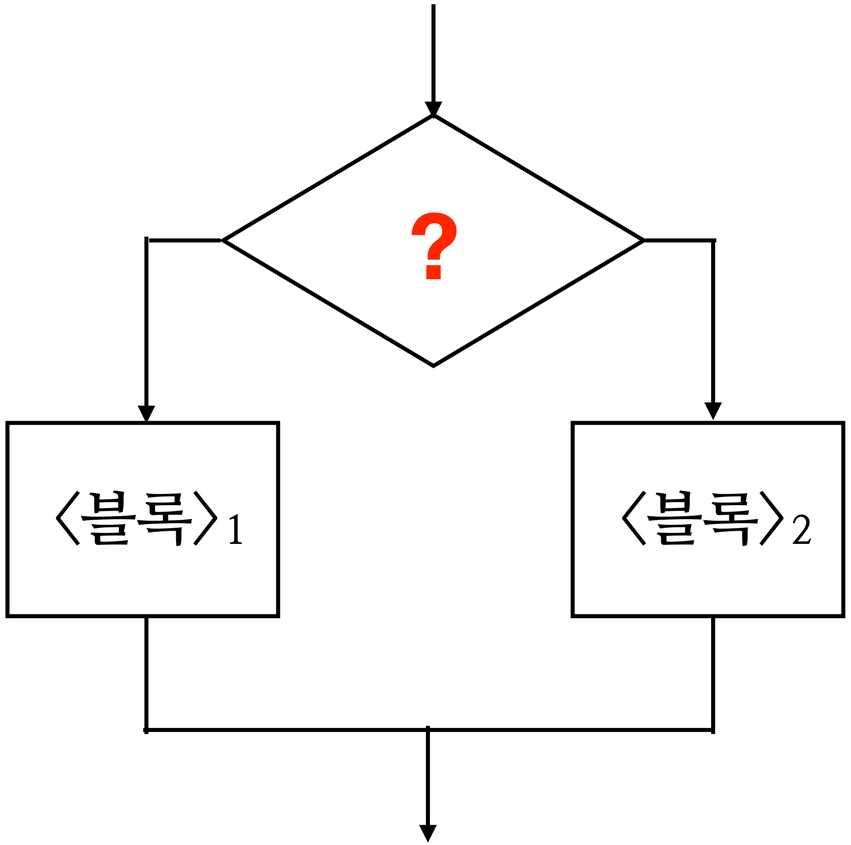
순차 제어



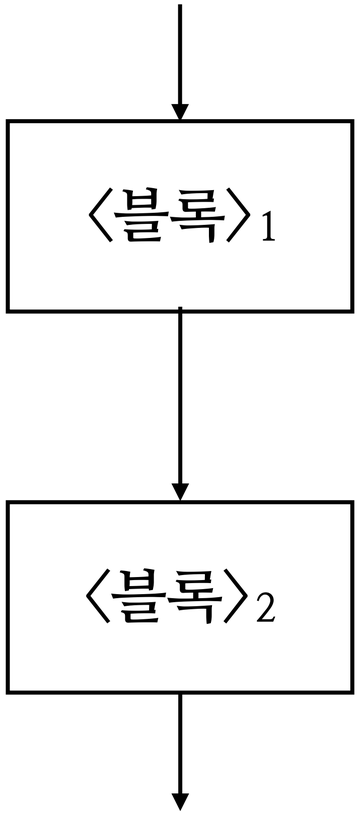
순차 제어



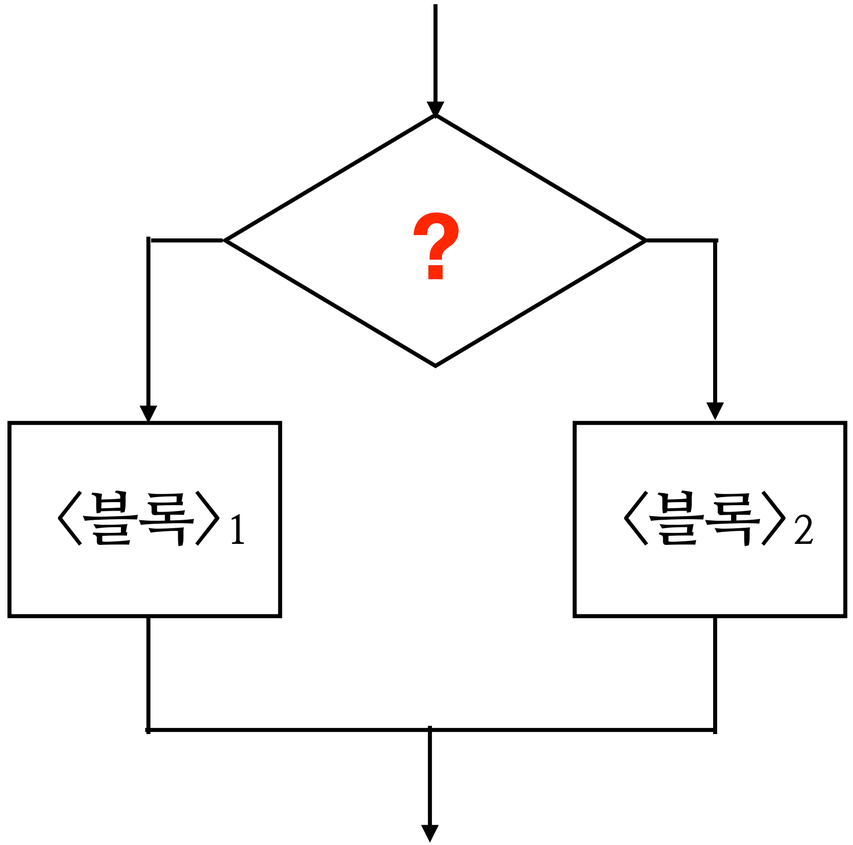
선택 제어



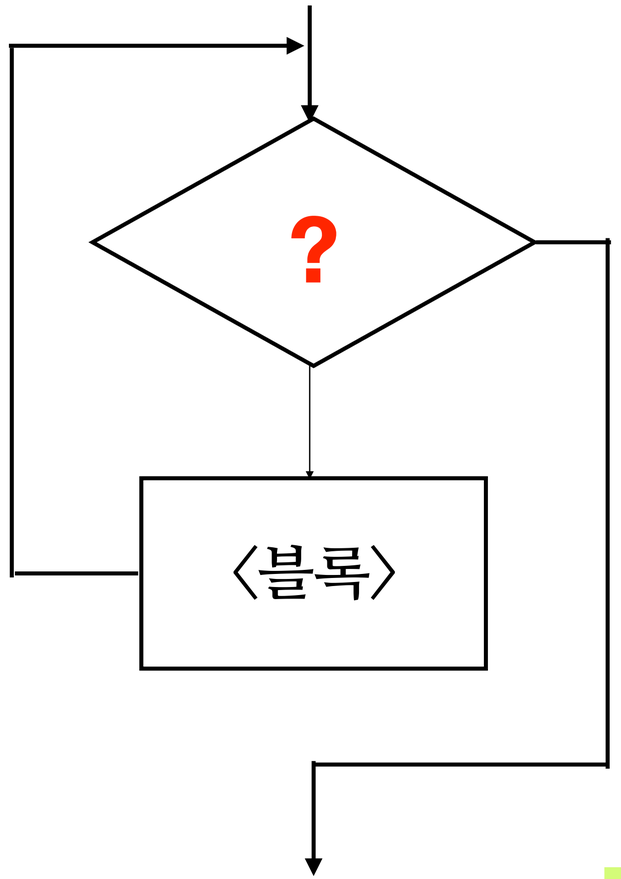
순차 제어



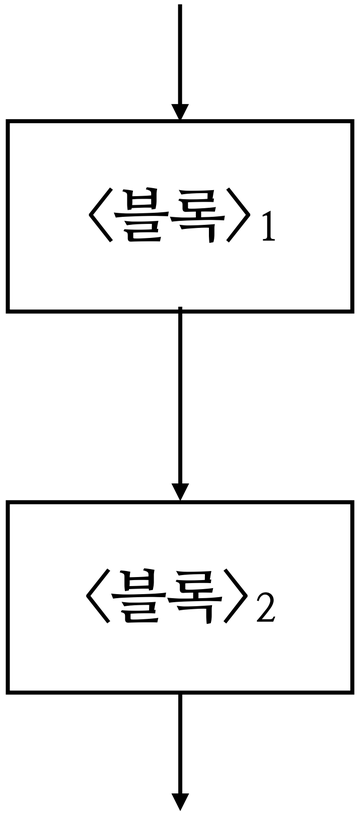
선택 제어



반복 제어

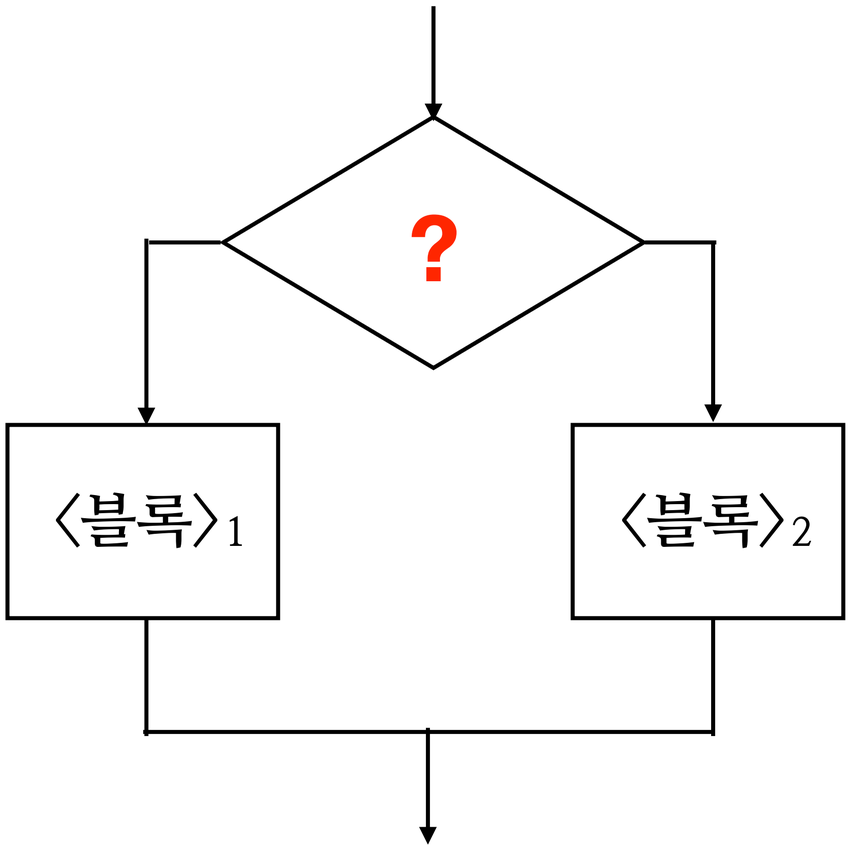


순차 제어



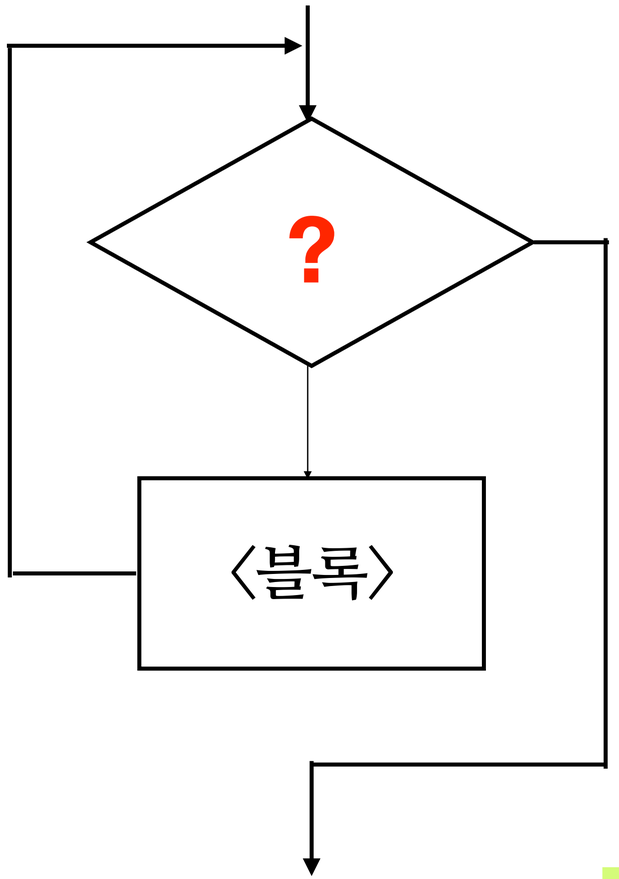
선택 제어

if-else

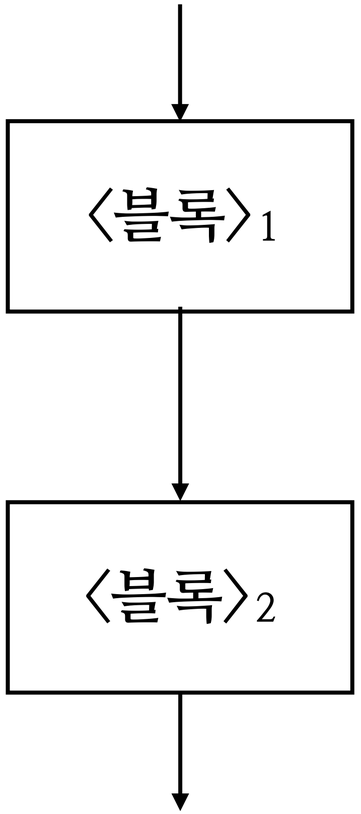


반복 제어

while

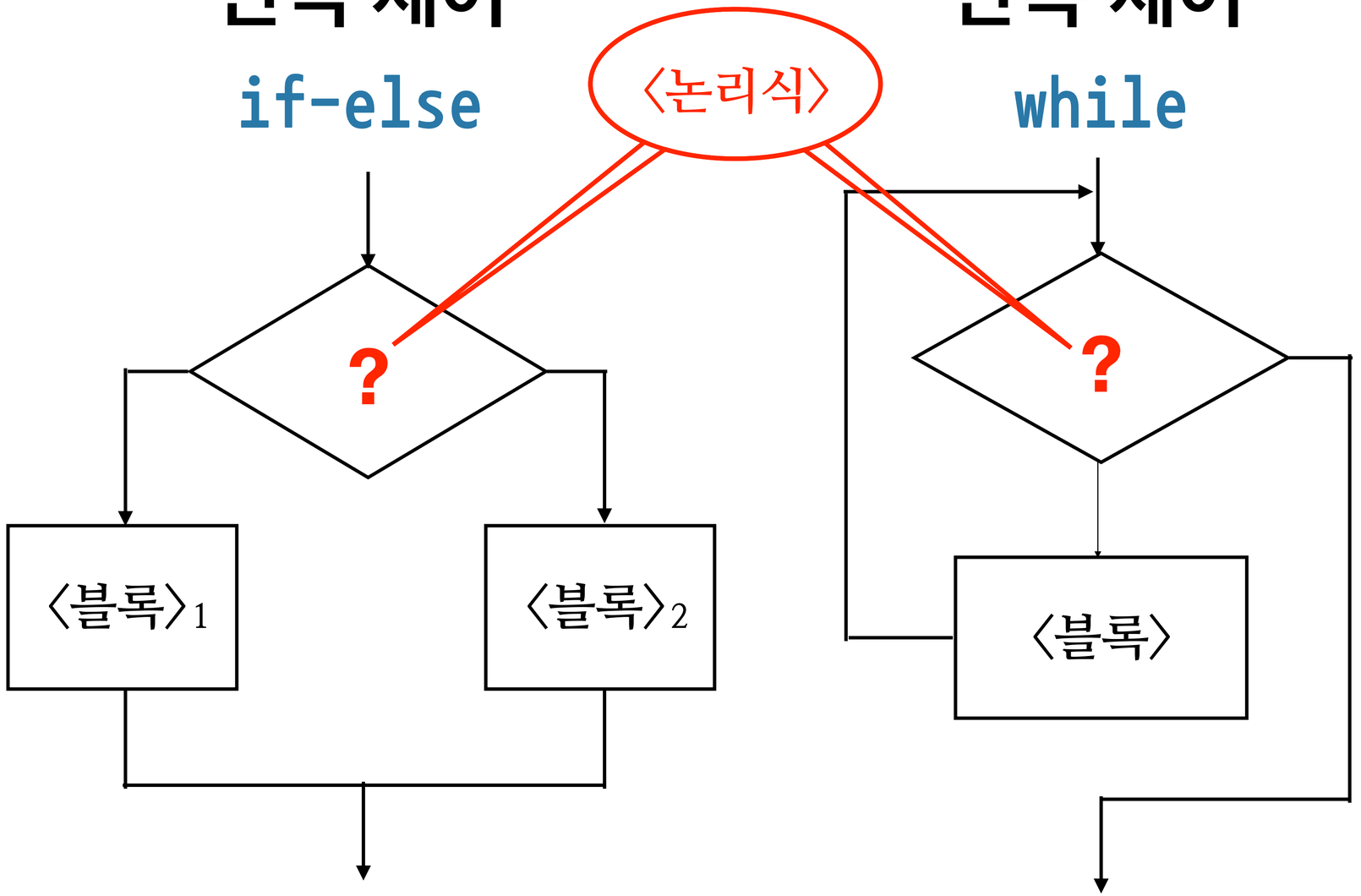


순차 제어



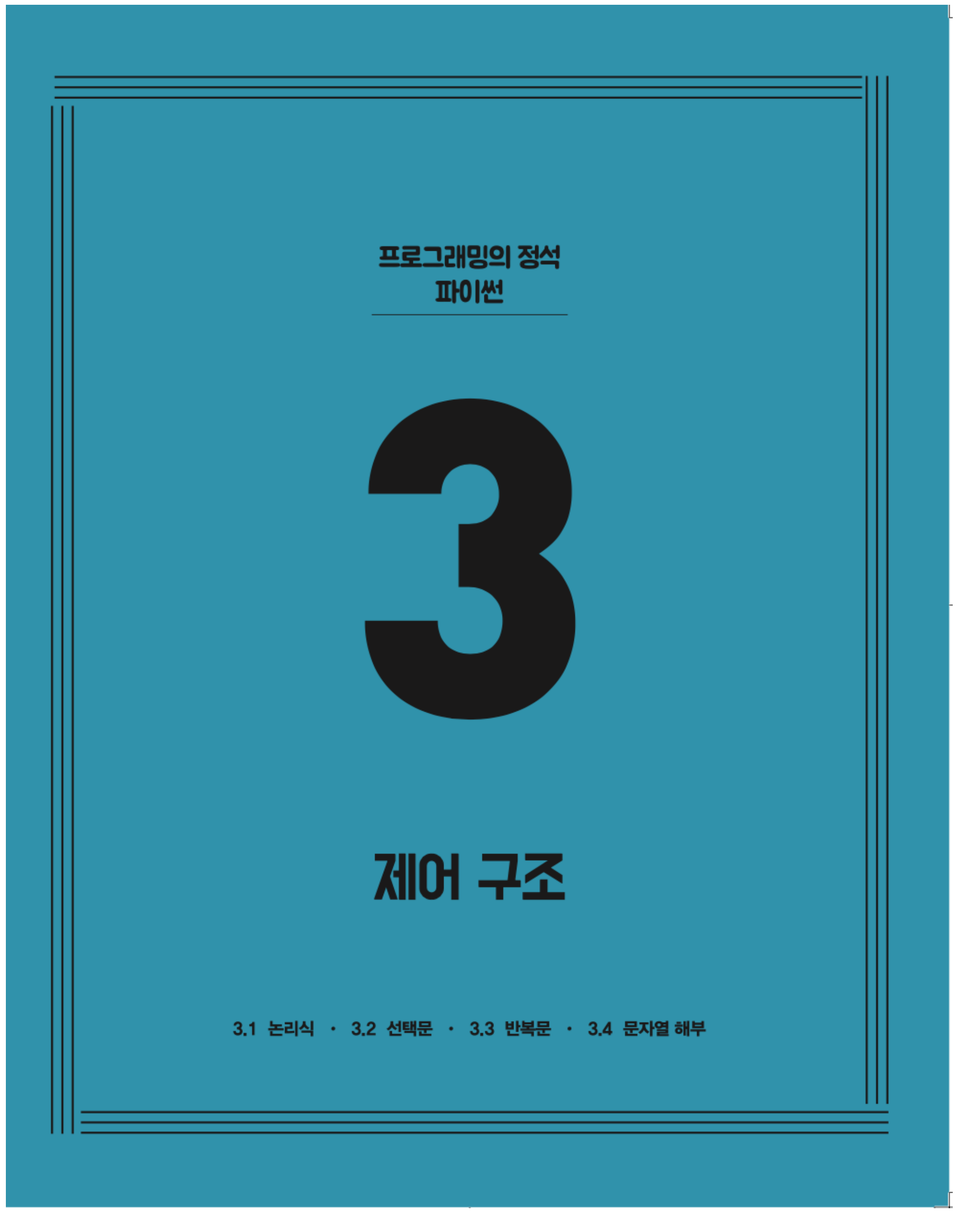
선택 제어

if-else



반복 제어

while



CHAPTER 3

제어 구조

- ✔ 3.1 논리식
- 3.2 선택문
- 3.3 반복문
- 3.4 문자열 해부

논리식

Logical Expression

Boolean Expression

논리값

Logical Value

Boolean Value

`True` `False`

논리 연산자

논리곱	논리합	논리역
and	or	not

〈논리식〉 and 〈논리식〉
〈논리식〉 or 〈논리식〉
not 〈논리식〉

진리표 Truth Table

p	q	p and q	p or q
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

p	not p
True	False
False	True

진리표 Truth Table

p	q	p and q	p or q
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

p	not p
True	False
False	True

진리표 Truth Table

p	q	p and q	p or q
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

p	not p
True	False
False	True

진리표 Truth Table

p	q	p and q	p or q
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

p	not p
True	False
False	True

연산자

연산자	우선순위	결합 순서
not	가장 높음	-
and	높음	-
or	낮음	-

연산자

연산자	우선순위	결합 순서
not	가장 높음	-
and	높음	-
or	낮음	-

```
(p and q) and r = p and (q and r)
(p or q) or r = p or (q or r)
```

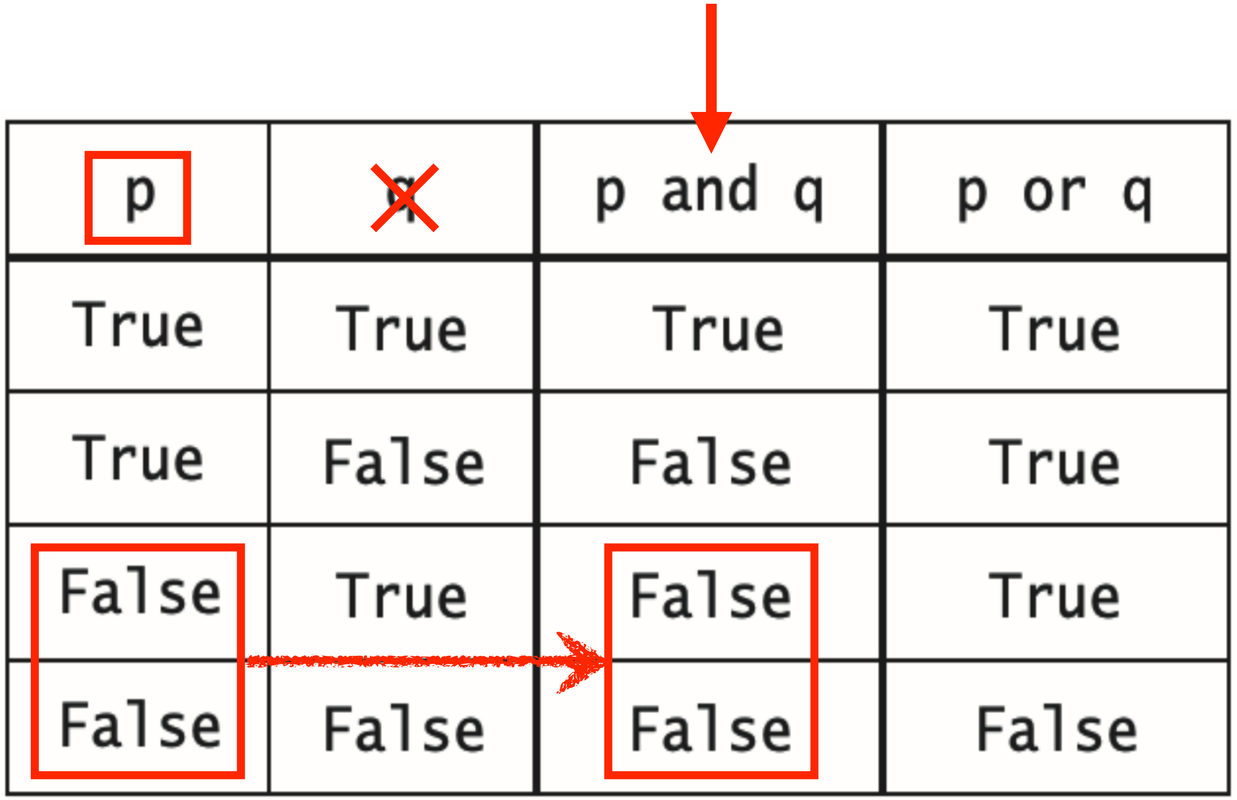
단축 계산

Short-circuit Evaluation

p	q	p and q	p or q
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

단축 계산

Short-circuit Evaluation



p	q	p and q	p or q
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

단축 계산

Short-circuit Evaluation

p	q	p and q	p or q
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

비교 논리식

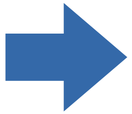
비교 연산자

같다	다르다	크다	작다	크거나 같다	작거나 같다
==	!=	>	<	>=	<=

〈식〉 == 〈식〉
〈식〉 != 〈식〉

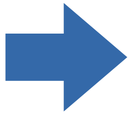
〈식〉 > 〈식〉
〈식〉 < 〈식〉

〈식〉 >= 〈식〉
〈식〉 <= 〈식〉



비교 기준

수	크기
논리값	`True`는 1로, `False`는 0으로 처리
문자열	ASCII 코드 / Unicode 값



ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

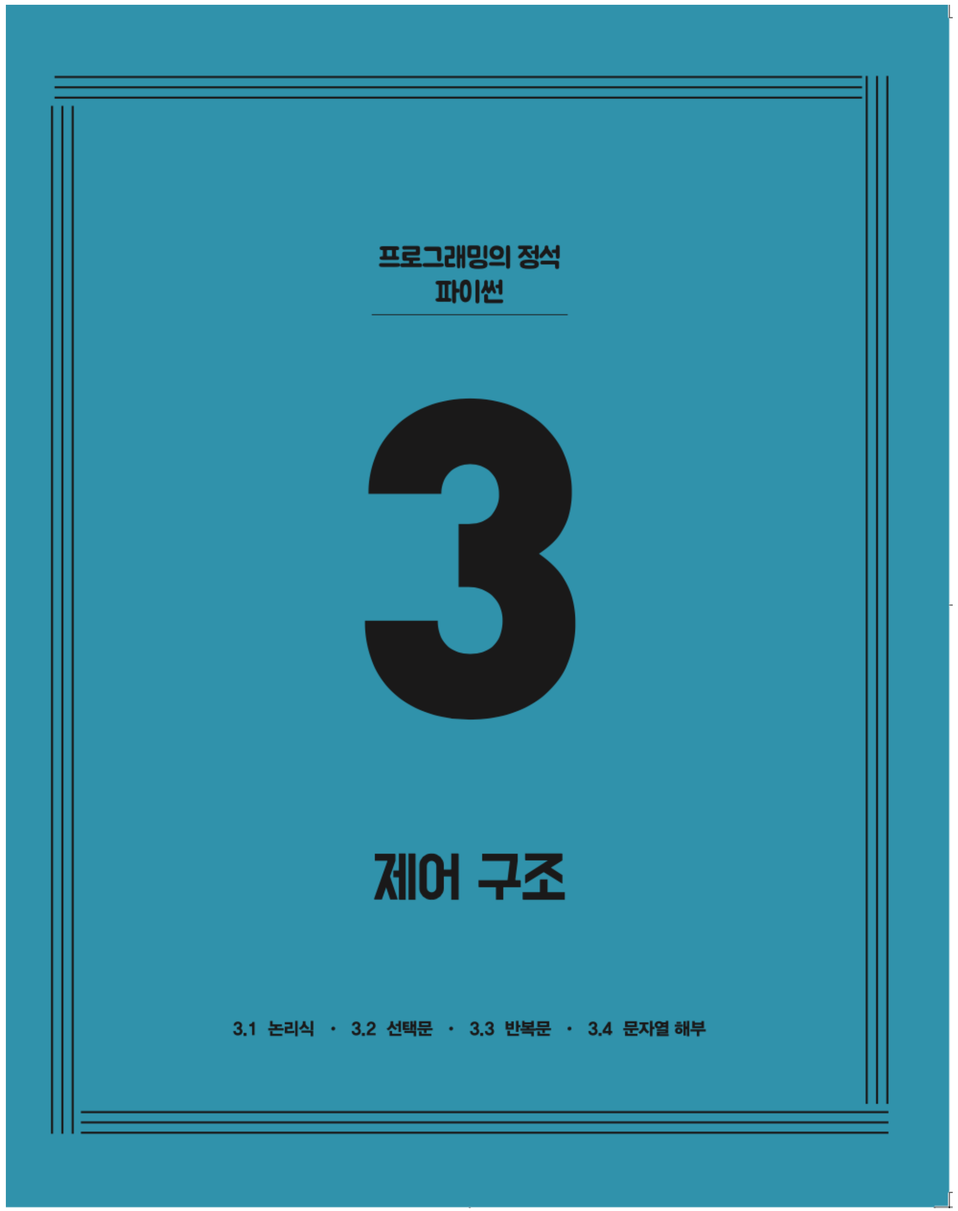
실습 3.1 짝수 확인 함수

정수를 인수로 받아서 짝수이면 True를, 홀수이면 False를 내주는 함수 even을 아래 형식에 맞추어 작성하자. 짝수이면 참이 되는 논리식을 만들어 리턴하면 된다.

code : 3-1.py

```
1  def even(n):
2      return None # Write your Boolean expression here.
3
4  print(even(13)) # prints False
5  print(even(26)) # prints True
```

힌트 짝수를 2로 나누면 나머지는 항상 0이다.



CHAPTER 3

제어 구조

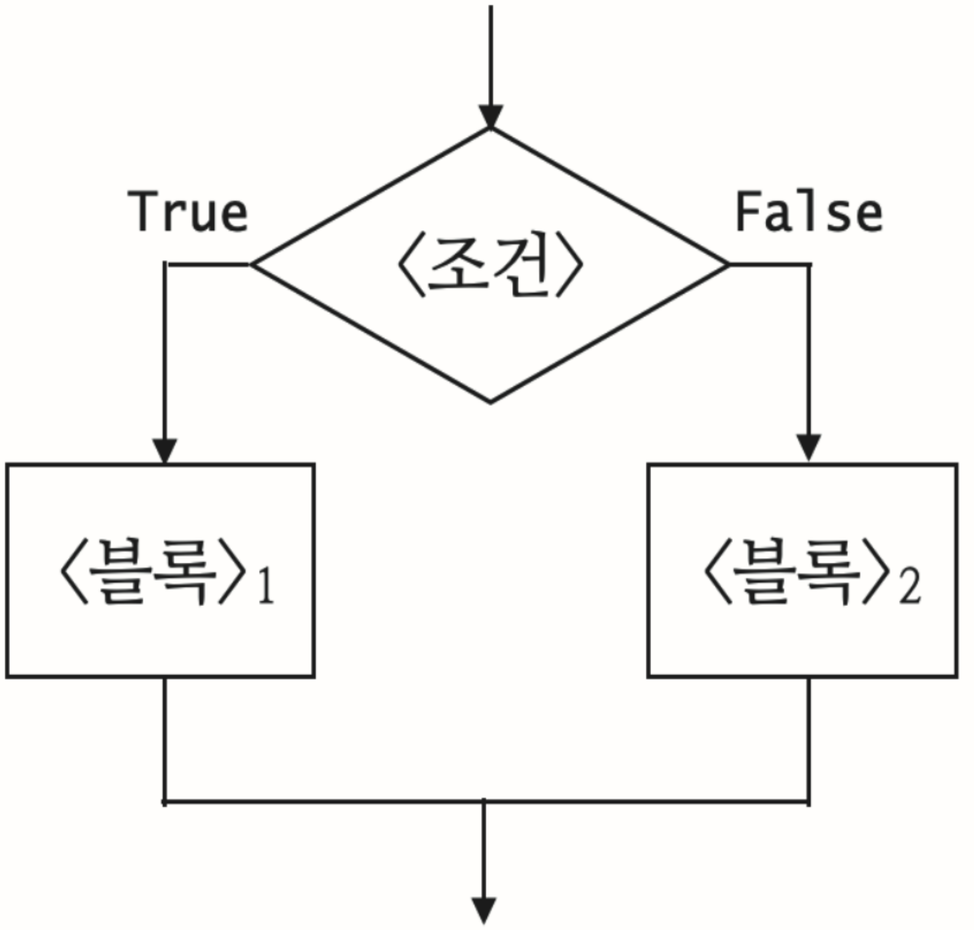
선택문

Conditional Statement

구문

```
if 〈조건〉:
    〈블록〉1
else:
    〈블록〉2
```

의미

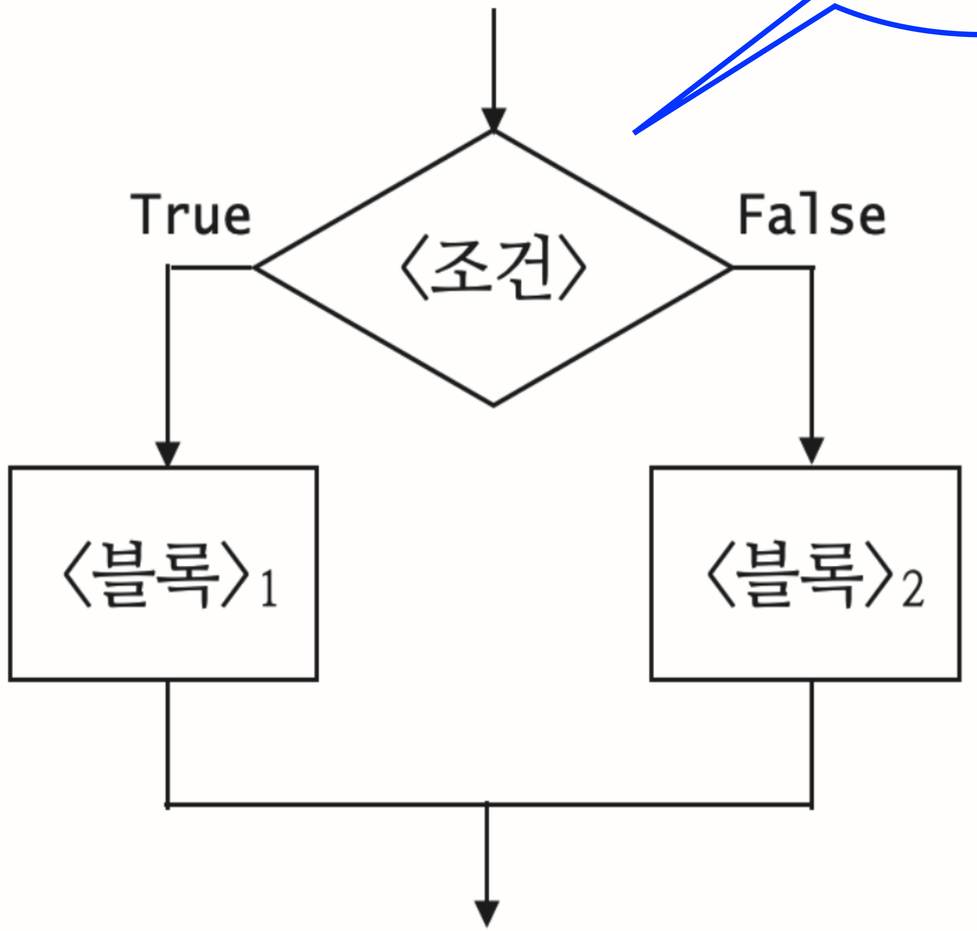


구문

```
if 〈조건〉:
    〈블록〉1
else:
    〈블록〉2
```

의미

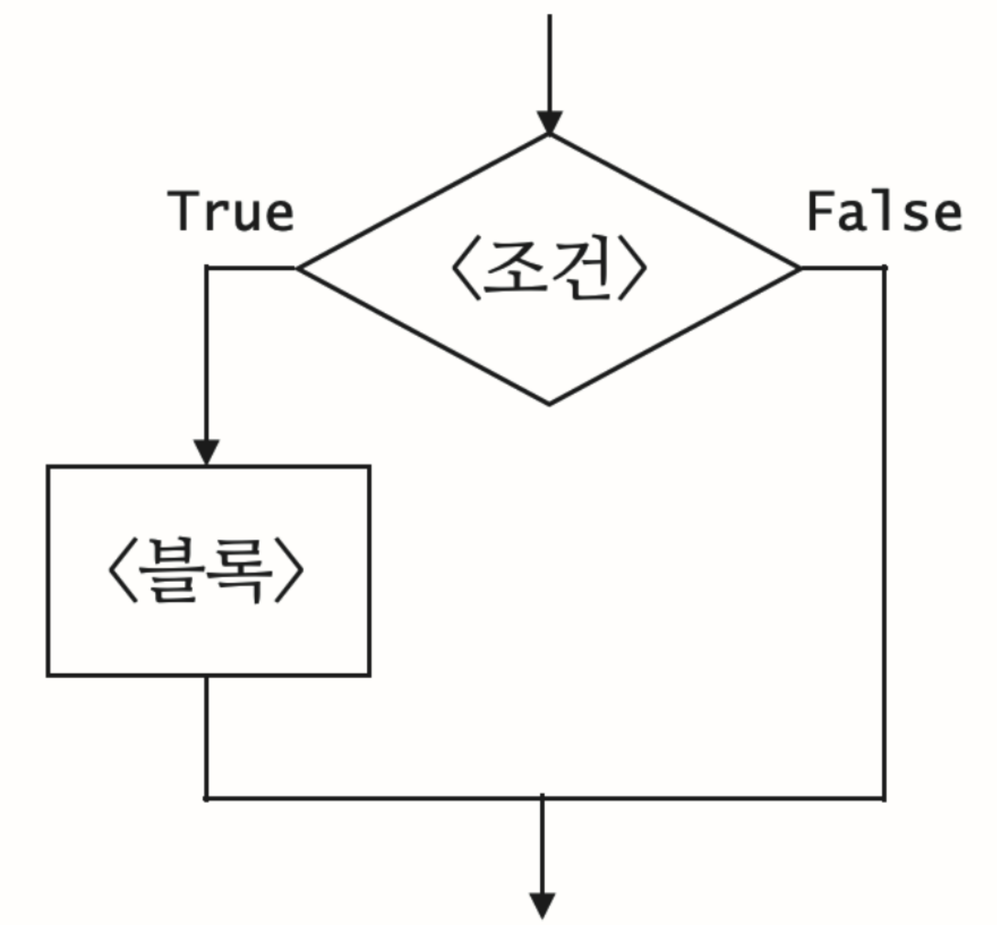
흐름도
Flow Chart



구문

```
if 〈조건〉:
    〈블록〉
```

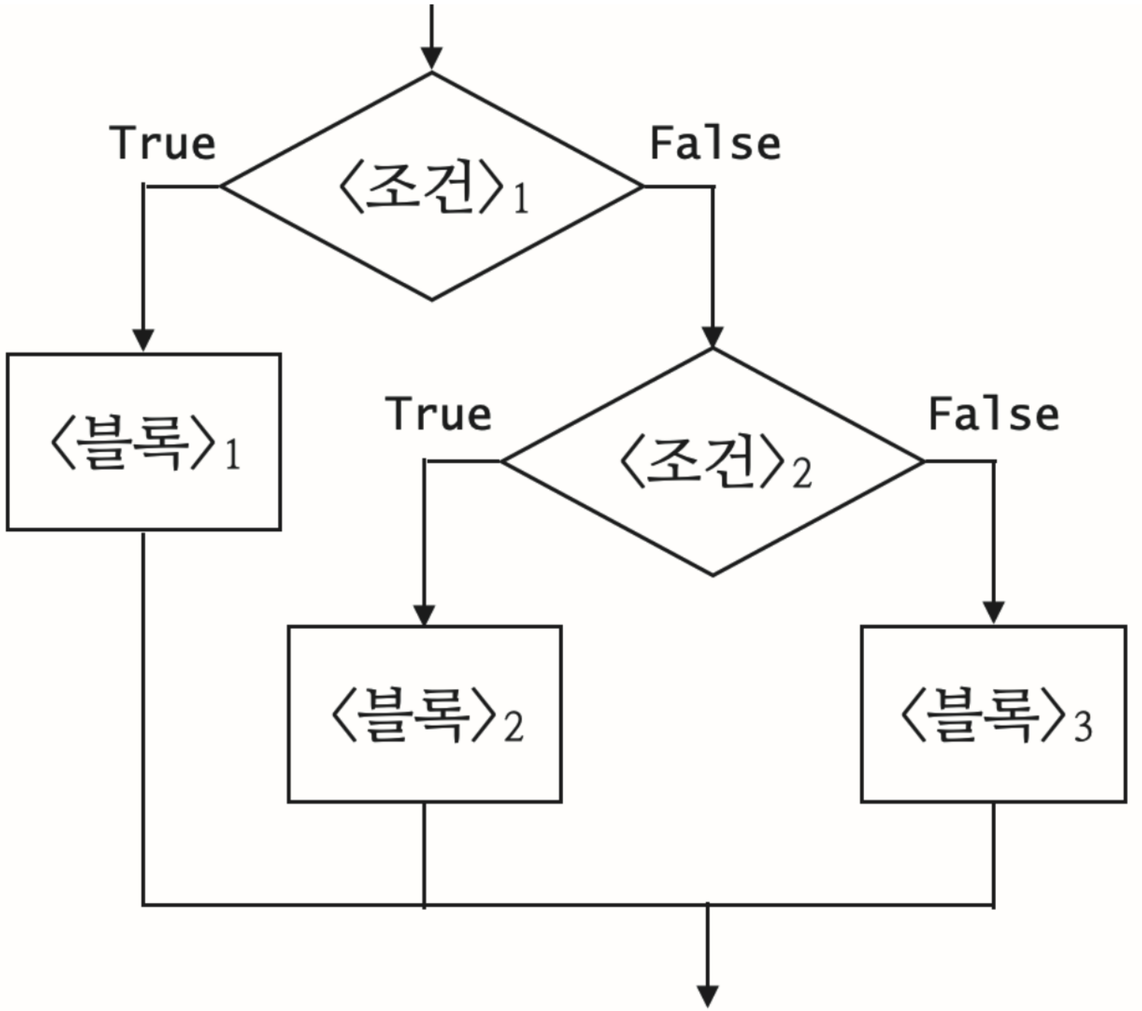
의미



구문

```
if 〈조건〉1:
    〈블록〉1
elif 〈조건〉2:
    〈블록〉2
else:
    〈블록〉3
```

의미



```
if 〈조건〉1:
    〈블록〉1
elif 〈조건〉2:
    〈블록〉2
elif …
else:
    〈블록〉3
```

```
if 〈조건〉1:
    〈블록〉1
elif 〈조건〉2:
    〈블록〉2
elif …
else:
    〈블록〉3
```

```
if 〈조건〉1:
    〈블록〉1
elif 〈조건〉2:
    〈블록〉2
elif …
```

실습 3.2 흐름도 그리기

다음 프로그램을 눈으로 읽어서 이해하고 흐름도를 그려보자.

code : 3-4.py

```
score = int(input('Enter your score : '))
if 90 <= score <= 100:
    print('A')
elif 80 <= score <= 89:
    print('B')
elif 70 <= score <= 79:
    print('C')
elif 60 <= score <= 69:
    print('D')
elif 0 <= score <= 59:
    print('F')
else:
    print('Your score is out of range.')
```

완성한 흐름도를 잘 살펴보면 이 프로그램의 실행 가능 경로는 모두 6개이다. 프로그램의 실행 경로를 모두 섭렵하는 6개의 다른 입력 값을 찾아서 각각 실행하여 기대한 대로 결과를 프린트하는지 확인하자.

선택문의 중첩

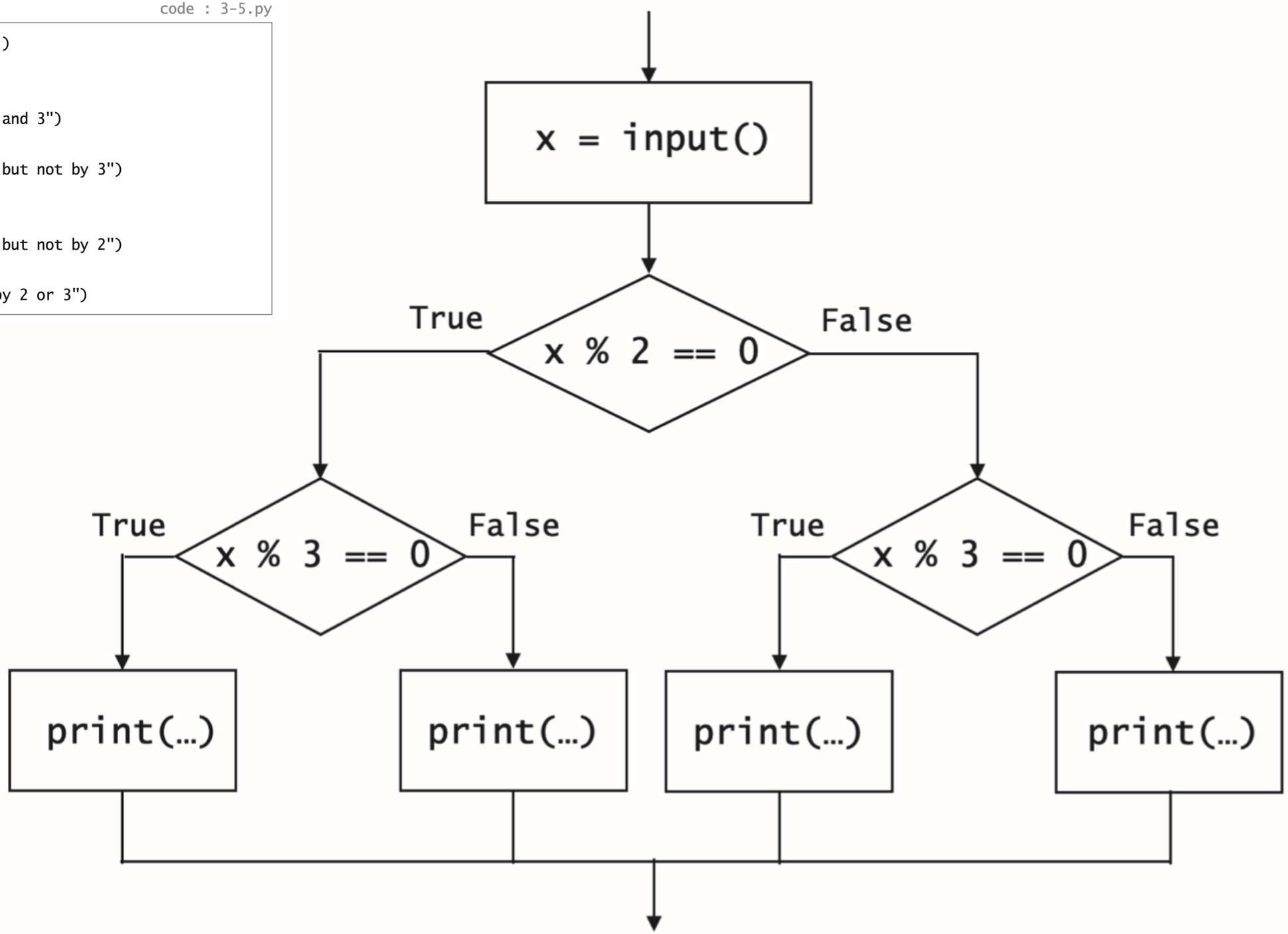
nested

code : 3-5.py

```
1  x = int(input("Enter your number: "))
2  if x % 2 == 0:
3      if x % 3 == 0:
4          print(x, "is divisible by 2 and 3")
5      else:
6          print(x, "is divisible by 2 but not by 3")
7  else:
8      if x % 3 == 0:
9          print(x, "is divisible by 3 but not by 2")
10     else:
11         print(x, "is not divisible by 2 or 3")
```

code : 3-5.py

```
x = int(input("Enter your number: "))
if x % 2 == 0:
    if x % 3 == 0:
        print(x, "is divisible by 2 and 3")
    else:
        print(x, "is divisible by 2 but not by 3")
else:
    if x % 3 == 0:
        print(x, "is divisible by 3 but not by 2")
    else:
        print(x, "is not divisible by 2 or 3")
```



code : 3-5.py

```
x = int(input("Enter your number: "))
if x % 2 == 0:
    if x % 3 == 0:
        print(x, "is divisible by 2 and 3")
    else:
        print(x, "is divisible by 2 but not by 3")
else:
    if x % 3 == 0:
        print(x, "is divisible by 3 but not by 2")
    else:
        print(x, "is not divisible by 2 or 3")
```

and 연산자로 중첩 선택문 펼치기

code : 3-6.py

```
x = int(input("Enter your number: "))
if x % 2 == 0 and x % 3 == 0:
    print(x, "is divisible by 2 and 3")
elif x % 2 == 0 and not(x % 3 == 0):
    print(x, "is divisible by 2 but not by 3")
elif not(x % 2 == 0) and x % 3 == 0:
    print(x, "is divisible by 3 but not by 2")
else:
    print(x, "is not divisible by 2 or 3")
```

code : 3-5.py

```
x = int(input("Enter your number: "))
if x % 2 == 0:
    if x % 3 == 0:
        print(x, "is divisible by 2 and 3")
    else:
        print(x, "is divisible by 2 but not by 3")
else:
    if x % 3 == 0:
        print(x, "is divisible by 3 but not by 2")
    else:
        print(x, "is not divisible by 2 or 3")
```

p.108

실습 3.3 중첩 선택문의 실행 경로 따져보기

위 프로그램의 실행 경로별로 그 경로를 따라 실행할 입력 값을 각각 찾아보자.

pp.108~109

실습 3.4 둘 중 작은 수 찾기 함수

정수 2개를 인수로 받아서, 그중 작은 수를 리턴해 주는 함수 smaller를 작성하자.

code : 3-7.py

```
1 def smaller(x,y):
2     pass # Write your conditional here.
3
4 # Test code
5 print(smaller(3,5)) # 3
6 print(smaller(5,3)) # 3
7 print(smaller(3,3)) # 3
```

실습 3.5 셋 중 가장 작은 수 찾기 함수

정수 3개를 인수로 받아서, 그중 가장 작은 수를 리턴해 주는 함수 smallest를 작성하자.

code : 3-8.py

```
def smallest(x,y,z):
    pass # Write your nested conditional here.

# Test code
print(smallest(3,5,9)) # 3
print(smallest(5,3,9)) # 3
print(smallest(5,9,3)) # 3
print(smallest(3,9,5)) # 3
print(smallest(9,3,5)) # 3
print(smallest(9,5,3)) # 3
print(smallest(3,3,5)) # 3
print(smallest(5,3,3)) # 3
print(smallest(3,5,3)) # 3
print(smallest(3,3,3)) # 3
```

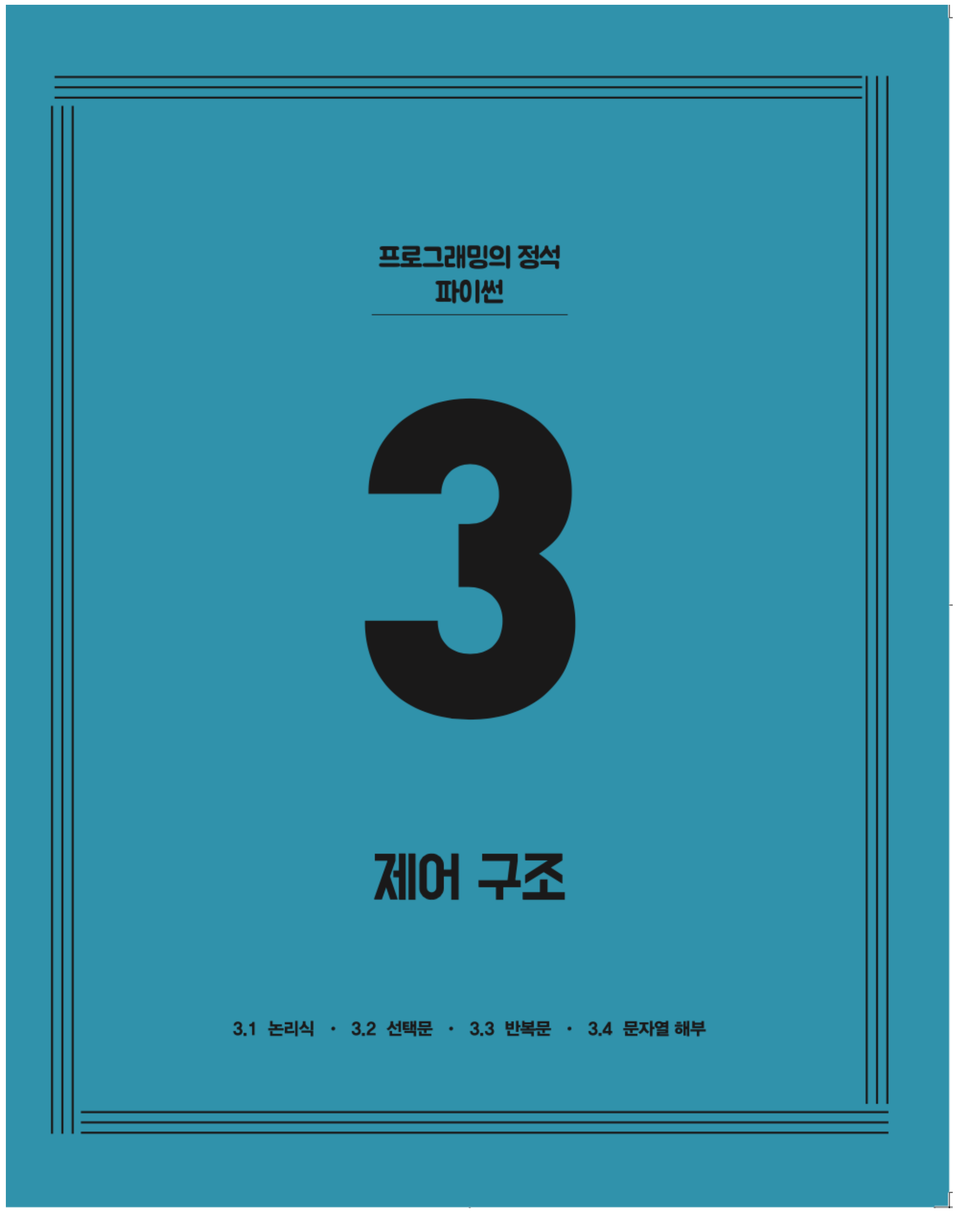
실습 3.6 셋 중 가장 작은 수 찾기 함수 (smaller 활용)

선택문을 사용하지 않고 smaller 함수 호출을 활용하여 smallest 함수를 작성하자.

힌트 smaller 함수를 두 번만 호출하면 충분하다.

code : 3-9.py

```
def smallest(x,y,z):
    return None # Write your expression here.

# Test code
print(smallest(3,5,9)) # 3
print(smallest(5,3,9)) # 3
print(smallest(5,9,3)) # 3
print(smallest(3,9,5)) # 3
print(smallest(9,3,5)) # 3
print(smallest(9,5,3)) # 3
print(smallest(3,3,5)) # 3
print(smallest(5,3,3)) # 3
print(smallest(3,5,3)) # 3
print(smallest(3,3,3)) # 3
```



CHAPTER 3

제어 구조

반복문

Loop

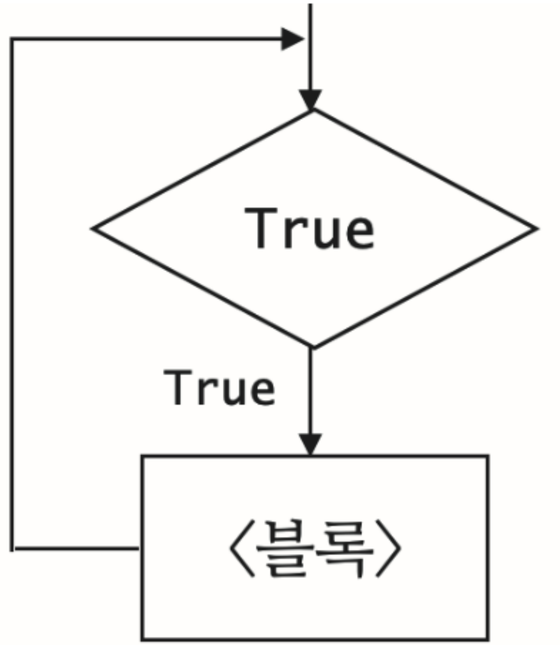
무조건 반복

Infinite Loop

구문

```
while True:
    〈블록〉
```

의미



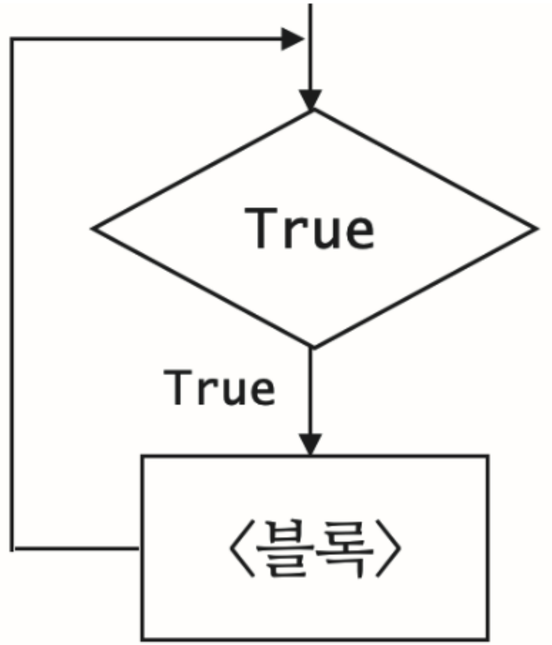
무조건 반복

Infinite Loop

구문

```
while True:
    〈블록〉
```

의미



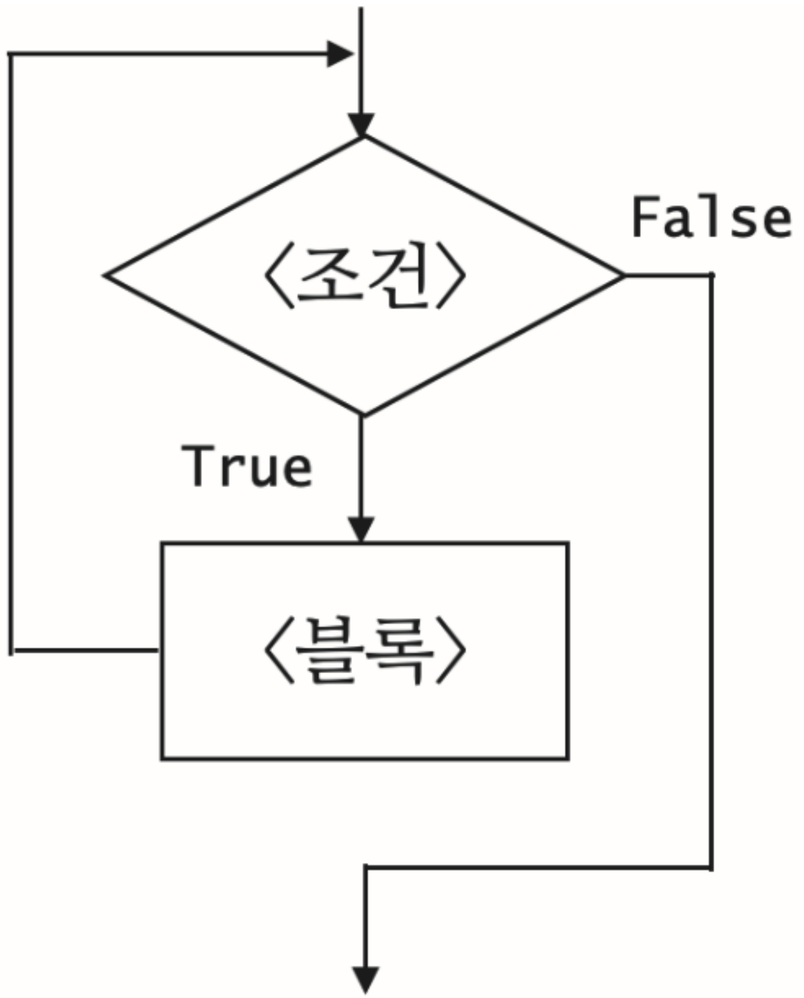
강제 종료 : Ctrl + C

조건 반복

구문

```
while 〈조건〉:
    〈블록〉
```

의미



실습 3.7 수강과목 평균 점수 계산 서비스

수강과목의 평균 점수를 계산하는 프로그램을 만들자. 키보드 입력으로 사용자가 제공할 정보는 과목의 개수와 각 과목의 점수이다. 실행창에서 다음과 같이 차례로 입력받아 평균 점수를 계산하여 화면에 프린트하도록 한다.

```
Score Average Calculator
How many classes? 5
Enter your score: 86
Enter your score: 92
Enter your score: 94
Enter your score: 82
Enter your score: 79
Your average score = 86.6
```

평균 점수는 반올림하여 소수점 아래 첫째 자리까지만 표시한다.

수강과목 수가 0인 경우에는 점수 입력을 받지 않고 다음과 같이 프린트해야 한다.

```
Score Average Calculator
How many classes? 0
Your average score = 0.0
```

평균을 계산할 때 과목의 수로 나누어야 하는데, 이 경우 나누기 0 오류가 발생하지 않도록 해야 한다.

프로그램 실행 절차는 다음과 같다.

1. 평균 점수 계산 서비스 메시지를 프린트한다.
2. 수강과목 개수를 입력받아 number 변수로 지정한다.
3. 점수의 누적 합을 기억할 total 변수를 0으로 지정한다.
4. 입력받은 점수의 개수를 세는 count 변수를 0으로 지정한다.
5. 수강과목 개수만큼 다음을 반복한다.
   (a) 점수를 입력받는다.
   (b) 점수를 total 변수에 누적한다.
   (c) count 변수의 값을 1 증가한다.
6. count 변수의 값이 0보다 크면, 평균을 계산하여, `Your average score = …` 형식으로 프린트한다.
7. count 변수의 값이 0이면, `Your average score = 0.0`을 프린트한다.

이와 같이 프로그램 실행 절차를 기술해놓은 것을 알고리즘algorithm이라고 한다.

이 알고리즘을 참고하여 다음 뼈대코드 형식에 맞추어 프로그램을 완성하자.

code : 3-13.py

```
1  print("Score Average Calculator")
2  number = input("How many classes? ")
3  total = 0
4  count = 0
5  while None: # Write your loop condition
6      pass # Get and accumulate scores
7
8
9  if count > 0:
10     pass # Compute and print the average
11 else:
12     print("Your average score = 0.0")
```

코딩 가이드

1. 키보드 입력으로 받은 값은 모두 문자열이다. 수식 계산에 쓰려면 입력 문자열을 정수로 타입 변환해야 한다.
2. 라인 5의 None 부분에 반복조건을 채워야 한다. 반복조건은 “키보드 입력으로 읽어 들인 점수의 개수가 수강과목의 개수보다 작다.”이다.

사례 학습

입력 확인

Input Validation

문자열은 객체이다!

객체 **object**

```
"365"

"365.0"

"-17"


"freedom"
```

클래스 class

객체 object

"365"

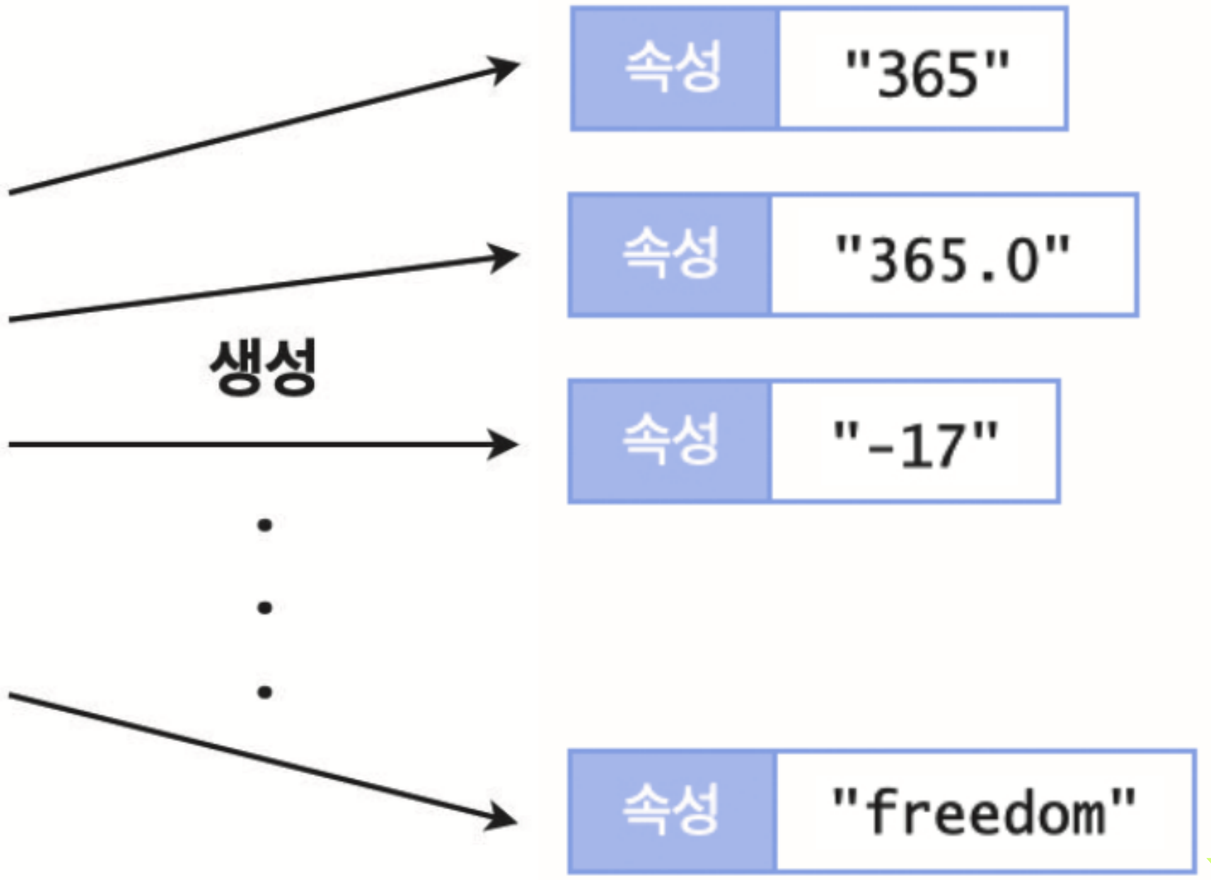
"365.0"

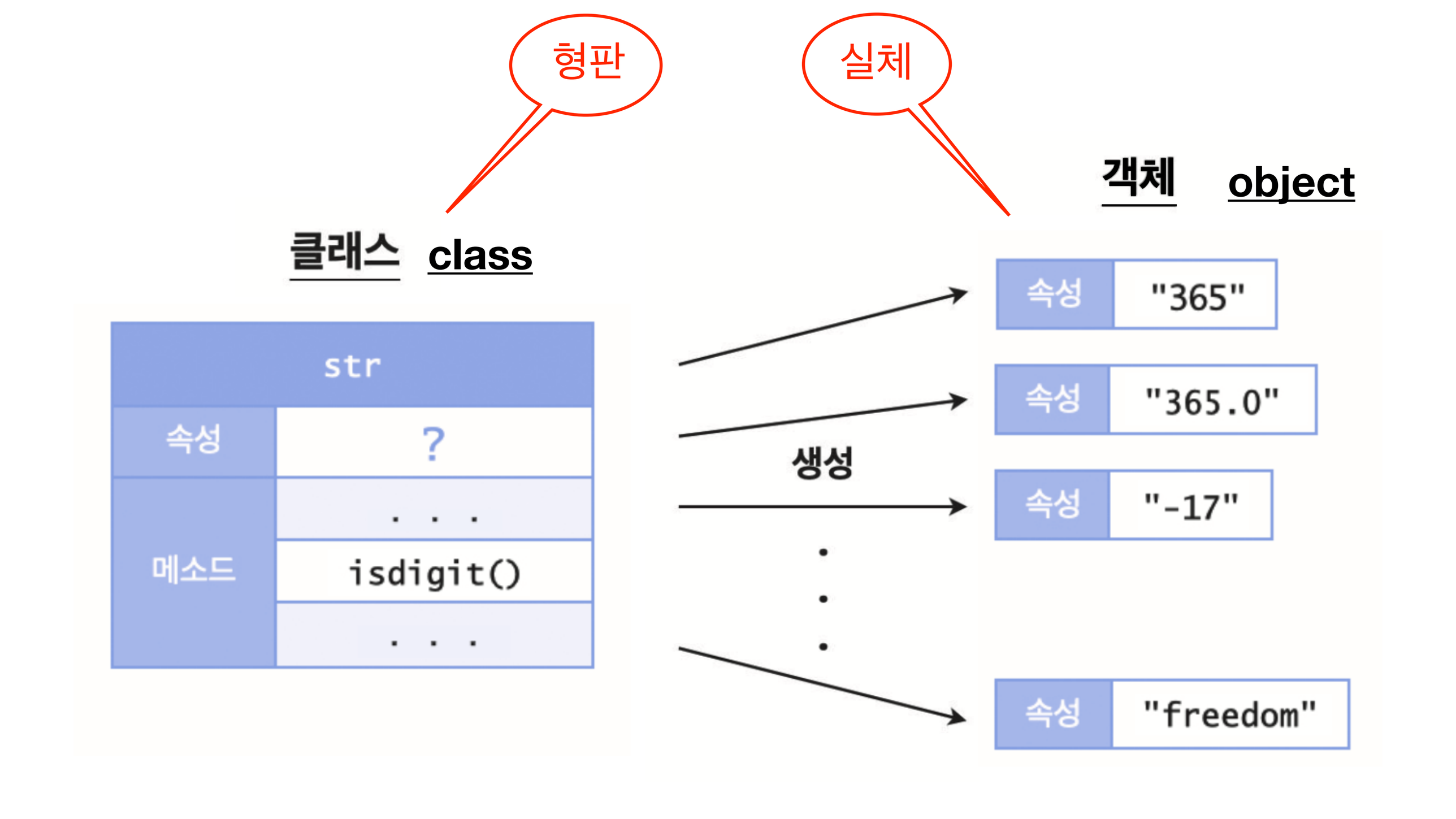
"-17"

"freedom"

클래스 class

객체 object




형판
실체
클래스 class
객체 object
str
속성
?
메소드
. . .
isdigit()
. . .
생성
속성 "365"
속성 "365.0"
속성 "-17"
속성 "freedom"

메소드

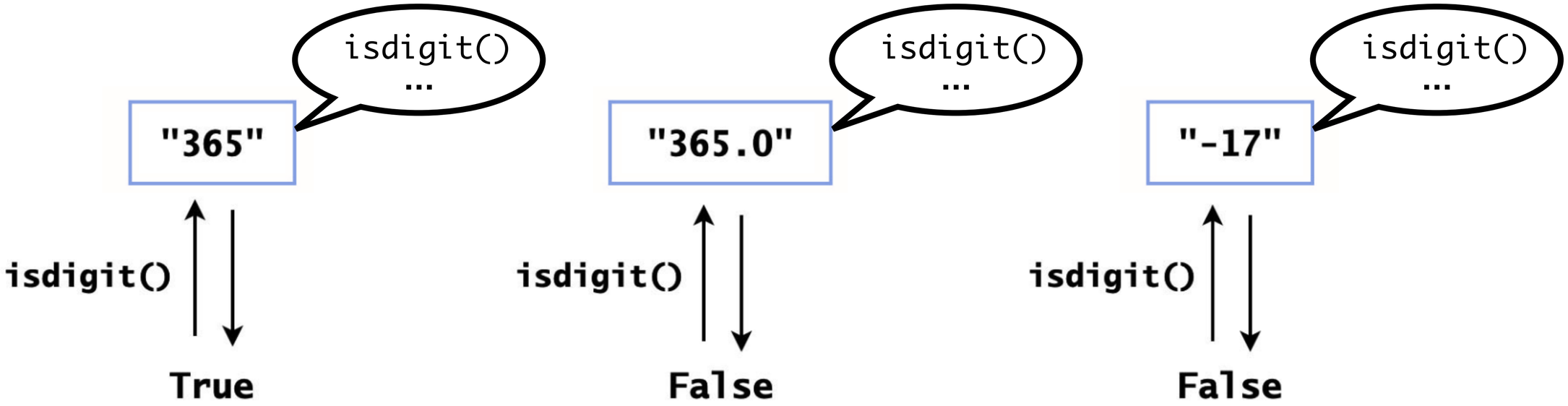
method

〈문자열〉.isdigit()

메소드

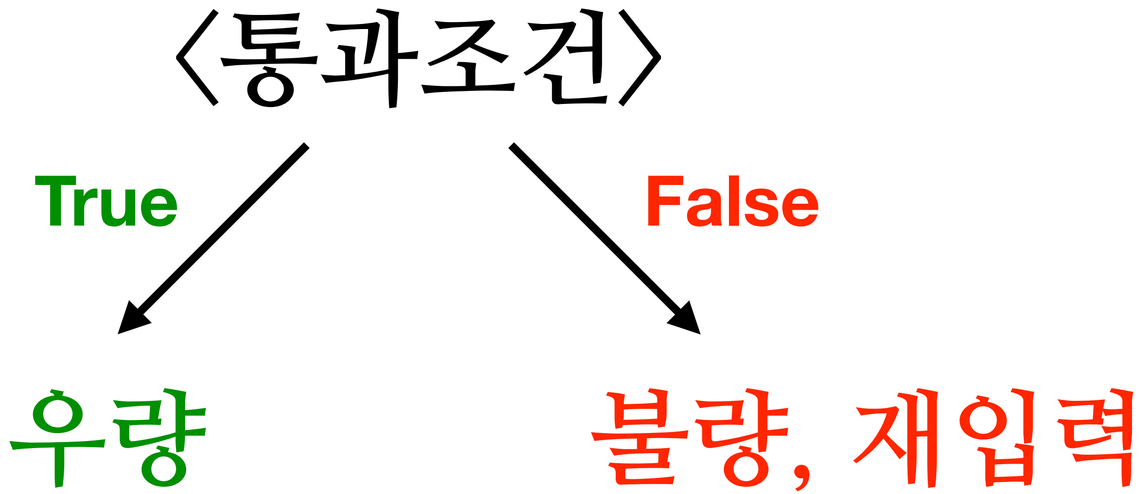
method

〈문자열〉.isdigit()



입력 확인

Input Validation



입력 확인 코드 패턴

```
x = input()

while not 〈통과조건〉:

    x = input()
```

입력 확인 코드 패턴

```
x = input()

while not 〈통과조건〉:

    x = input()
```

x.isdigit()

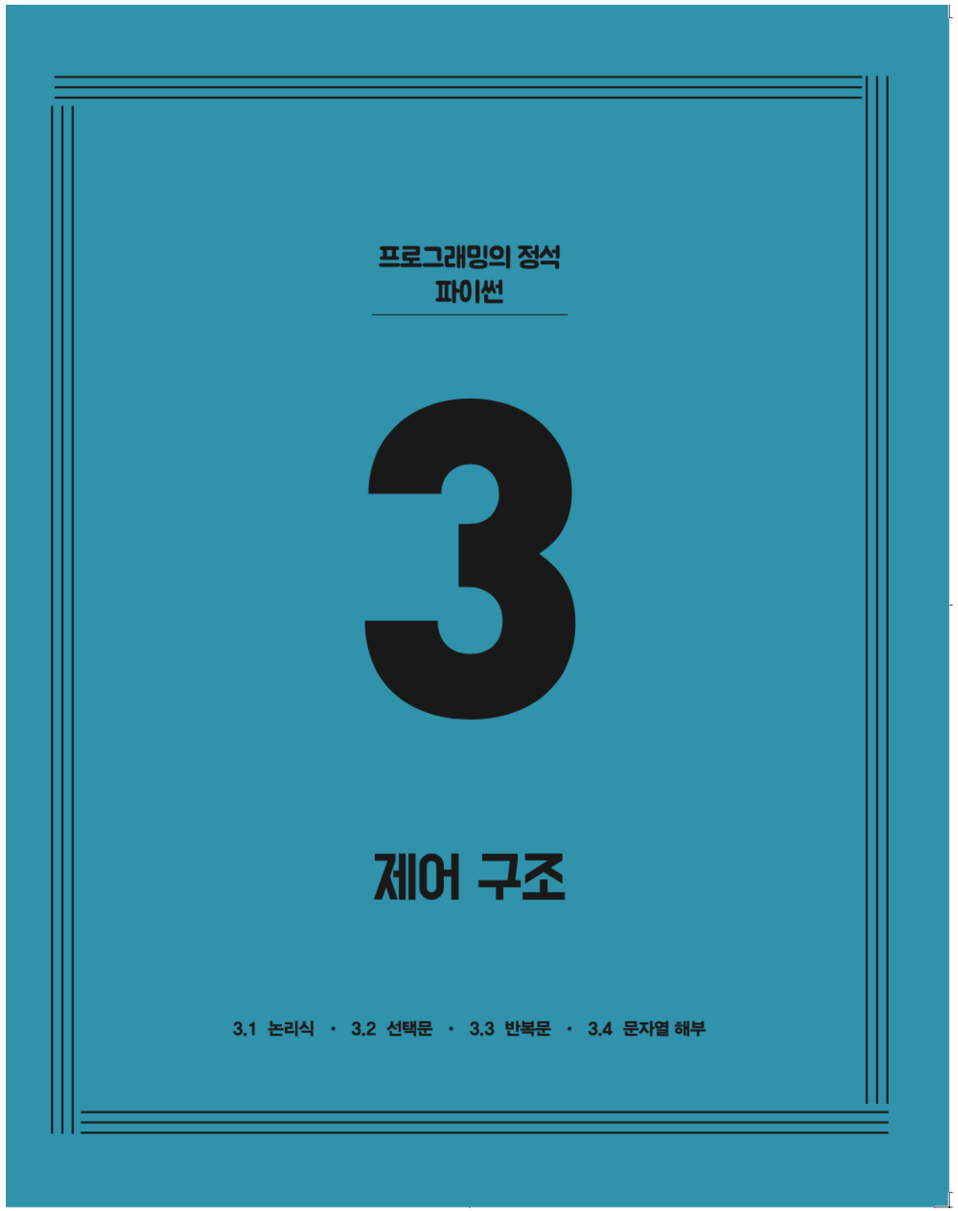
code : 3-14.py

```
from math import pi

def area_circle(radius, n):
    if radius > 0:
        area = pi * radius ** 2
        return round(area, n)
    else:
        return 0.0

print("Circle Area Calculator")
more = 'y'
while more == 'y':
    r = input("Radius? ")
    while not r.isdigit():
        r = input("Radius? ")
    p = input("Precision? ")
    while not p.isdigit():
        p = input("Precision? ")
    area = area_circle(int(r),int(p))
    print("The area of a circle with radius", r, "is", area, ".")
    more = input("Press y to continue, any other key to exit. ")
print("Please come back again.")
```

실습 3.8 수강과목 평균 점수 계산 서비스 (입력 확인)

〈실습 3.7〉에서 완성한 프로그램은 입력 확인을 하지 않아서 정수가 아닌 입력에 타입 오류가 발생한다. 게다가 일반적으로 점수는 100점 만점으로 계산하기 때문에 0~100 사이의 정수로 입력을 제한시켜야 합리적이다. 0~100 범위의 정수가 아닌 사용자 입력은 무시하고 재입력을 요청하도록, 입력 확인 기능을 추가하여 프로그램을 보완하자. 다음과 같은 형식으로 재입력을 요청해야 한다.

```
Score Average Calculator
How many classes? five
Enter a positive number: -5
Enter a positive number: 5
The number of classes = 5
Enter your score: 86
Your score = 86
Enter your score: A
Enter your score between 0 and 100: 200
Enter your score between 0 and 100: 92
Your score = 92
Enter your score: 94
Your score = 94
Enter your score: 82
Your score = 82
Enter your score: 79
Your score = 79
Your average score = 86.6
```



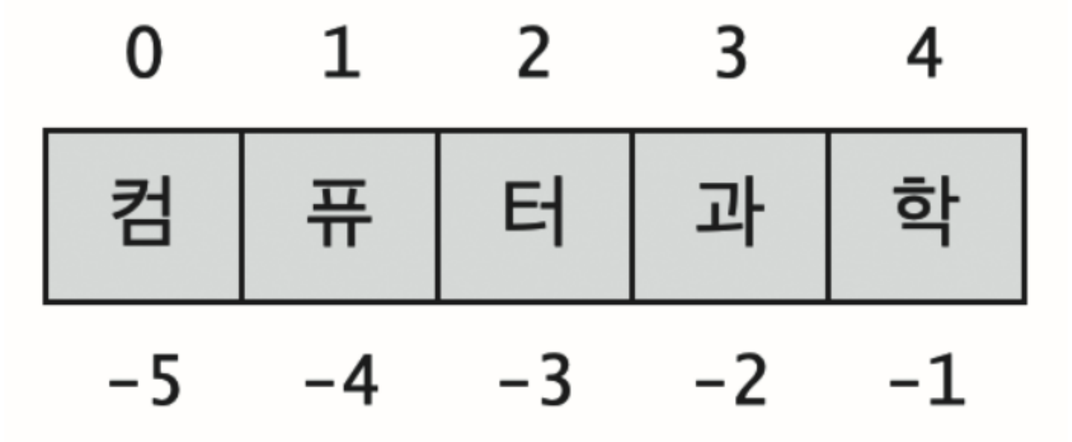
CHAPTER 3

제어 구조

인덱스

index

"컴퓨터과학"



인덱스의 범위

0 ~ n-1

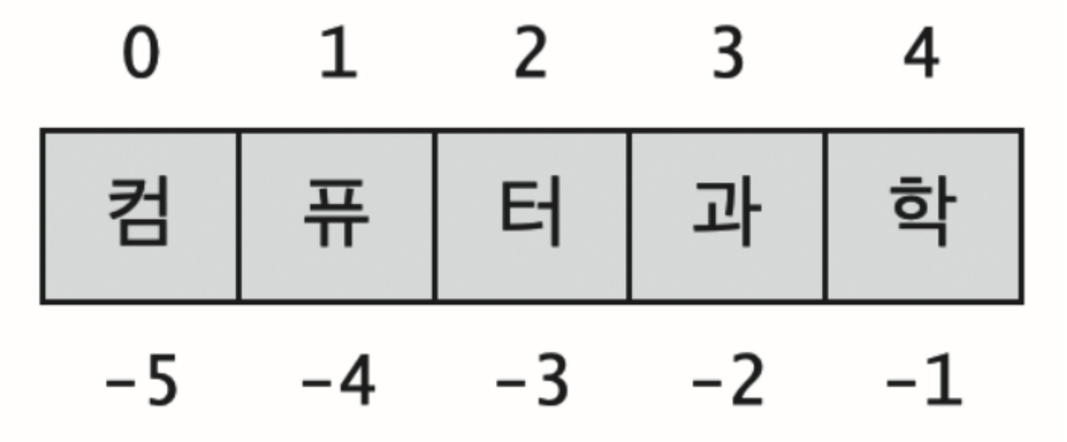
-1 ~ -n

n = 문자열의 길이

인덱스

index

"컴퓨터과학"

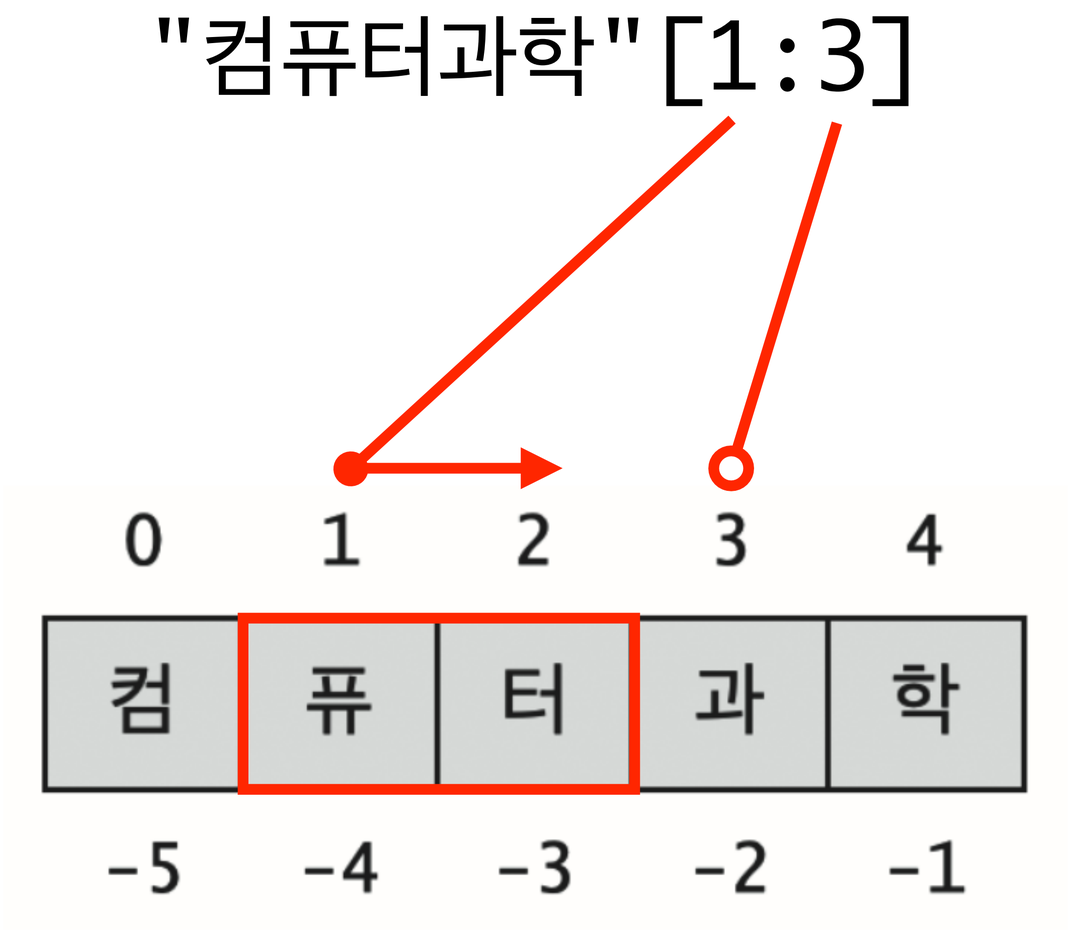


"컴퓨터과학"[0]

"컴퓨터과학"[2]

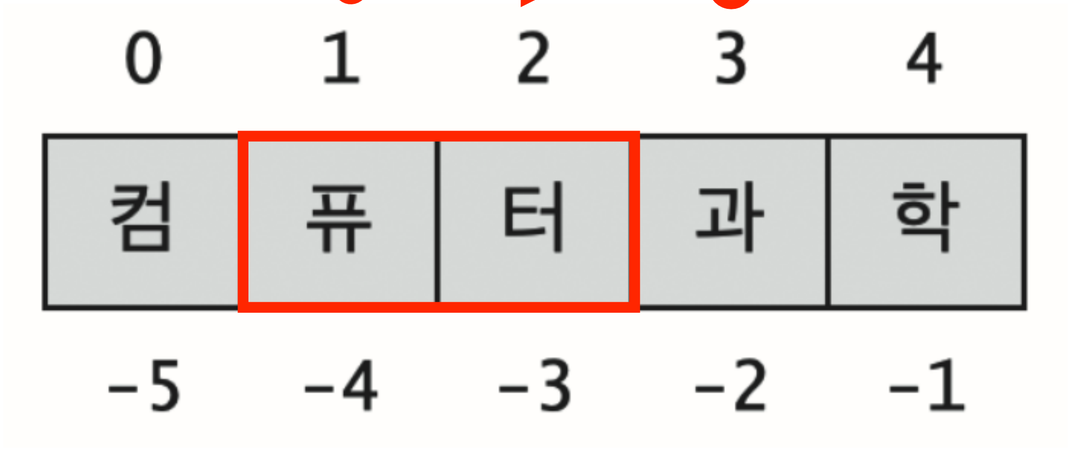
"컴퓨터과학"[-1]

문자열 조각 복제

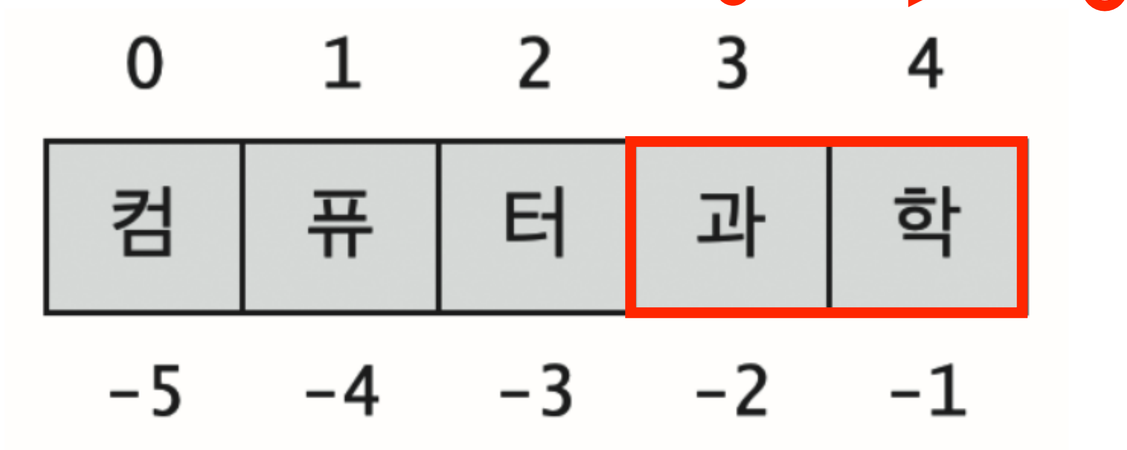


문자열 조각 복제

"컴퓨터과학"[1:3]

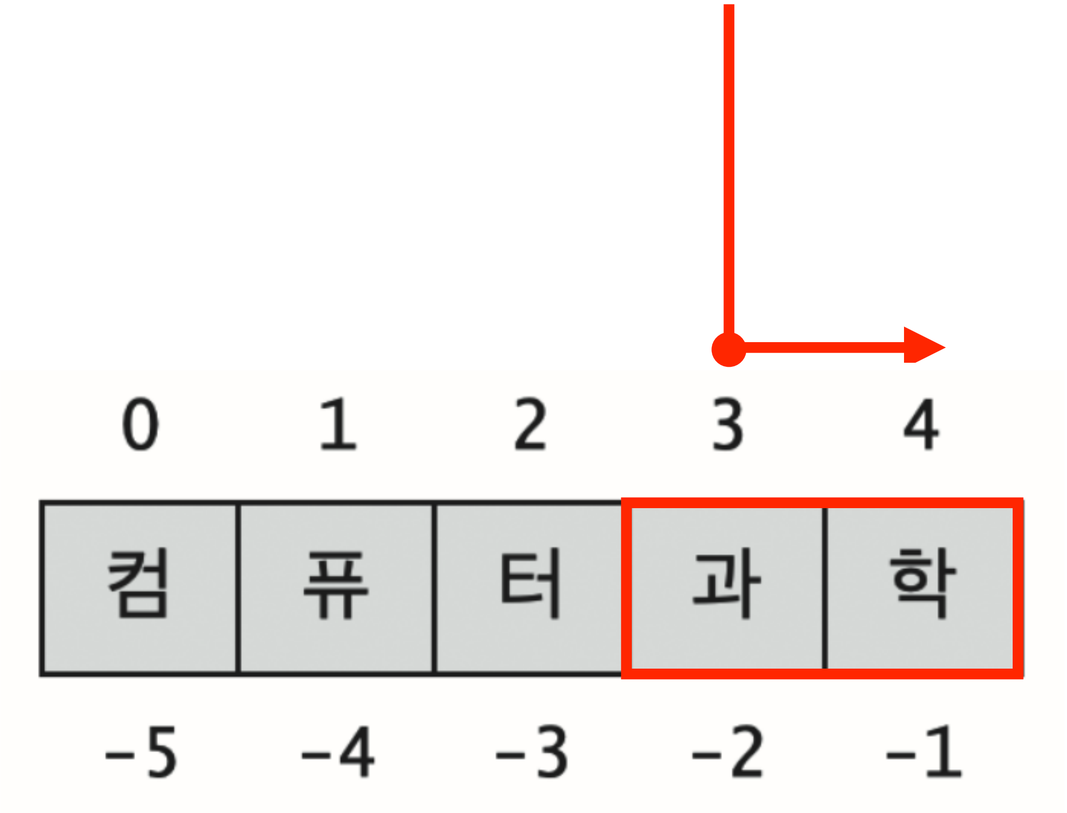


"컴퓨터과학"[3:5]



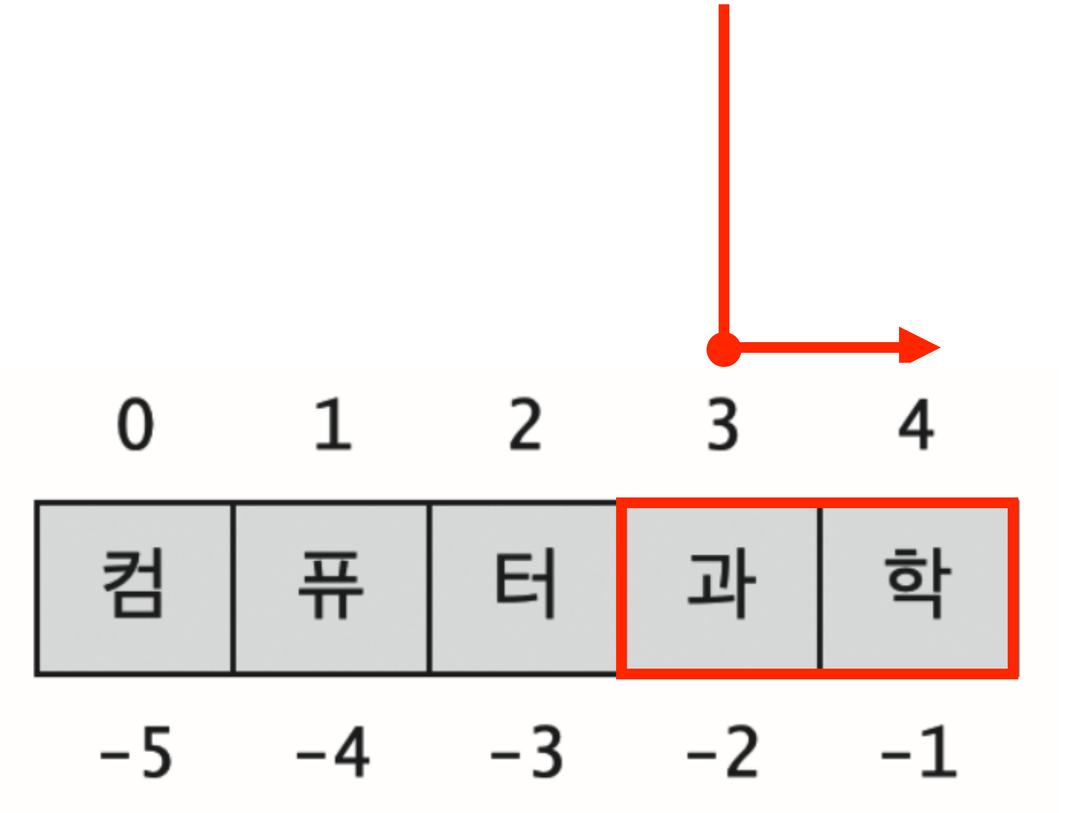
문자열 조각 복제

"컴퓨터과학"[3:]

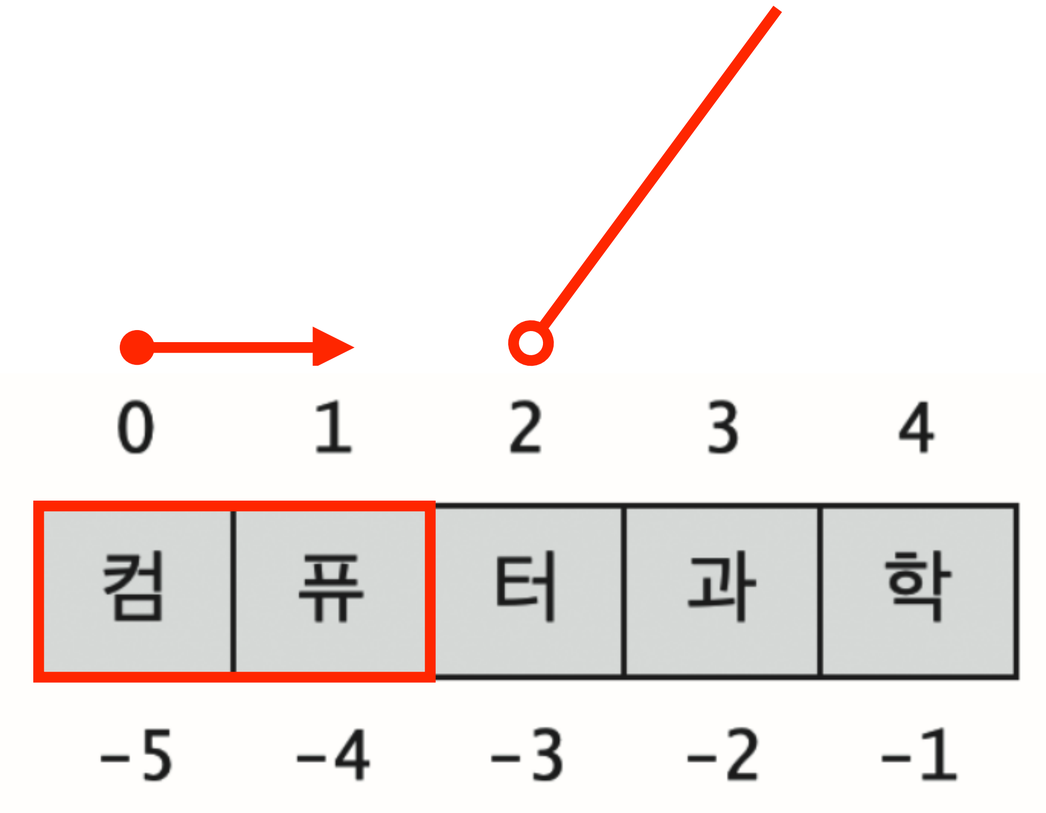


문자열 조각 복제

"컴퓨터과학"[3:]

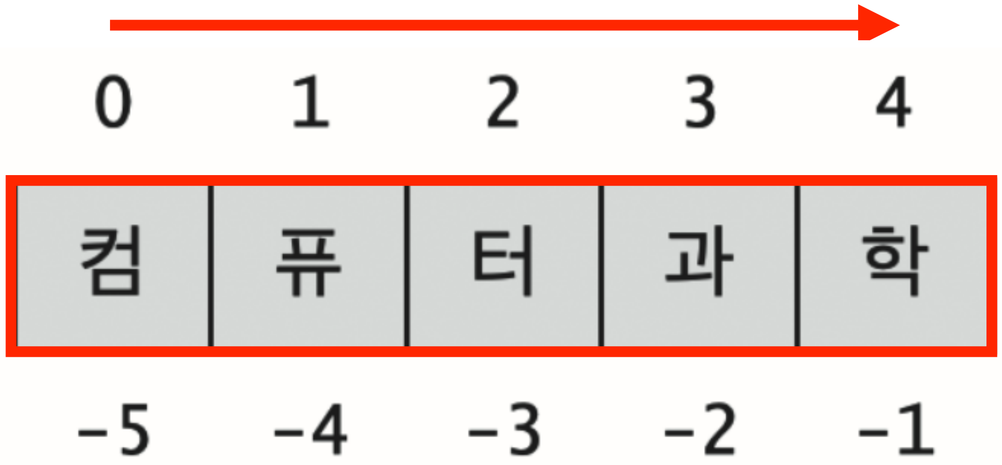


"컴퓨터과학"[:2]



문자열 복제

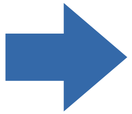
"컴퓨터과학"[:]




```
"3.14159".partition(".")
```

```
("3", ".", "14159")
```

튜플(tuple)



pp.127~128

실습 3.9 정수 문자열 확인 함수 (음수 포함)

isdigit() 메소드 함수는 문자열이 숫자만으로 구성되어 있는지 확인한다. 즉, 자연수 문자열인지는 이 메소드만으로 확인 가능하다. 그런데 정수는 음수를 포함하기 때문에, 음수 기호가 앞에 붙는다. 음수도 포함한 정수 문자열인지 확인하려면, 문자열은 다음 두 조건 중 하나를 만족해야 한다.

- 1개 이상의 이어진 숫자로 구성
- '-' 문자가 맨 앞에 있고, 이어서 1개 이상의 이어진 숫자로 구성

(음수도 포함하는) 정수 문자열을 인수로 받아서 정수 형식에 맞으면 True, 그렇지 않으면 False를 내주는 isinteger 함수는 다음과 같이 작성할 수 있다.

code : 3-15.py

```
1  def isinteger(s):
2      return s.isdigit() or \
3             s != '' and s[0] == '-' and s[1:].isdigit()
4
5  # Test code
6  print(isinteger("55"))    # True
7  print(isinteger("5o5"))   # False
8  print(isinteger("-55"))   # True
9  print(isinteger("--55"))  # False
10 print(isinteger("---55")) # False
11 print(isinteger("5-5"))   # False
12 print(isinteger("55-"))   # False
13 print(isinteger("+55"))   # False
14 print(isinteger("five"))  # False
15 print(isinteger("-"))     # False
16 print(isinteger(""))      # False
```

- 문제 1 : 라인 2, 3의 논리식이 완벽한 통과조건임을 확인하자.
- 문제 2 : 라인 3의 첫 비교 논리식 s != ''을 빼면 통과조건이 완벽하지 않다. 일부 불량 입력에 대해서 재입력을 유도하는 대신 오류가 발생한다. 오류를 발생시키는 불량 입력을 찾아보자.

pp.128~129

실습 3.10 실수 문자열 확인 함수 (음수 포함)

문자열이 실수를 고정소수점 표시 방식으로 정확히 표현하고 있는지 확인하기 위해서, 입력 문자열은 다음 두 조건 중 하나를 만족해야 한다.

- 0개 이상의 숫자, 이어서 '.' 문자, 이어서 0개 이상의 숫자로 구성 (단, '.' 만 있는 경우는 허용하지 않음)
- '-' 문자가 맨 앞에 있고, 0개 이상의 숫자, 이어서 '.' 문자, 이어서 0개 이상의 숫자로 구성 (단, '-.'는 허용하지 않음)

문자열을 인수로 받아서 고정소수점 수 형식에 맞으면 True, 그렇지 않으면 False를 내주는 isfloat 함수를 아래 뼈대코드에 맞추어 작성하자.

- 코딩 가이드 1 : 먼저 partition('.') 메소드 호출을 활용하여 문자열을 삼등분 낸다.
- 코딩 가이드 2 : 앞에서 공부한 isinteger 함수를 적극 활용하면 프로그램이 간결해진다.

```
def isinteger(s):
    return s.isdigit() or \
           s != '' and s[0] == '-' and s[1:].isdigit()

def isfloat(s):
    (left, dot, right) = s.partition(".")
    return None # Replace None with Boolean expression.

# Test code
print(isfloat(".112"))    # True
print(isfloat("-.112"))   # True
print(isfloat("3.14"))    # True
print(isfloat("-3.14"))   # True
print(isfloat("5."))      # True
print(isfloat("5.0"))     # True
print(isfloat("-777.0"))  # True
print(isfloat("-777."))   # True
print(isfloat("."))       # False
print(isfloat(".."))      # False
```

>>>>>>>>>>> 제어 구조의 설계 원리를 중심으로 배우는 >>>>>>>>>>>

프로그래밍의 정석

파이썬

도경구 지음

생능출판

CHAPTER 3

제어 구조