

연산자, 조건문, 반복문

송지수

Operator

Overview

- ✦ 연산식 : 연산자를 이용해서 만든 수식
- ✦ 연산자(Operator) : 연산에 사용되는 기호
- ✦ 피연산자(Operand) : 연산의 대상이 되는 값
- ✦ 연산자의 종류
 - 산술, 비교, 논리, 비트 연산자 등



전체 연산자의 우선순위

(높다) `**` → `~` → `*`, `/`, `//`, `%` → `+`, `-` → `<<`, `>>` → `&` → `^` → `|` →
`<`, `<=`, `>`, `=>`, `!=`, `==` → `is`, `in` → `not` → `and` → `or` (낮다)

Operator

Overview

연산자	설명
**	지수
~	비트 단위 NOT(1의 보수)
+ -	양수, 음수(단항 연산자)
* / // %	곱하기, 나누기, 몫, 나머지
+ -	더하기, 빼기(이항 연산자)
<< >>	시프트 연산
&	비트 단위 AND
^	비트 단위 XOR

^	비트 단위 XOR
	비트 단위 OR
< <= > >= <> != ==	크기 비교
is, not is in, not in	신원(Identity) 확인 멤버 검사
not	논리 연산 not
and	논리 연산 and
or	논리 연산 or
Lambda	람다 표현식(함수 참조)

Operator

Assignment Operator

✦ 대입 연산자 ('=') !python에는 대입개념이 없음

- 변수에 데이터를 할당하기 위한 연산자
- 파이썬에서 변수는 별도로 선언할 필요가 없음

✦ 파이썬에서 표현 가능한 대입 연산자의 예

- 파이썬에서는 여러 개의 값을 한 줄에서 한꺼번에 대입할 수 있음
- 언팩킹 (Unpacking)
 - `number1, number2 = 100, 200`
- 파이썬에서는 콤마(,)로 구분된 여러 개의 값을 하나의 변수에 대입할 수 있으며, 튜플(Tuple)이라는 자료형으로 저장됨
- 팩킹 (Packing)
 - `number = 10, 20, 30`
- 여러 개의 변수를 하나의 값으로 대입할 수도 있음
 - `x = y = z = 10`

Operator

Arithmetic Operator

✦ 산술 연산자

- 사칙 연산 : + (덧셈), - (뺄셈), * (곱셈), / (나눗셈)
- 파이썬 v2와 v3의 차이점
 - 파이썬 v2
 - 나누기 연산에서 적어도 하나는 실수 이어야 실수 결과 값이 나타남
 - 두 수가 정수이면 무조건 정수 결과가 나타남
 - 예) $25 / 8 \rightarrow 3$, $25 / 8.0 \rightarrow 3.125$
 - 파이썬 v3
 - 나누기 연산에서 실수를 사용하지 않아도 실수 결과로 나타남

```
>>> a = 3; b = 4
>>> print(a + b, a - b, a * b, a / b)
7 -1 12 0.75
```

Operator

Arithmetic Operator

✦ 산술 연산자

- // 연산자 : 나눈 몫
- % 연산자 (modulus) : 나눈 나머지
- ** 지수 승 : $a**b \rightarrow a$ 의 b 승

```
>>> a = 3; b = 4
>>> print(a // b, a % b, a ** b)
0 3 81
```

```
>>> a = 13; b = 4
>>> print(divmod(a, b))
(3, 1)
```

- $y = 2x^2 + 3x + 1$
- $y = 2 * \text{pow}(x, 2) + 3 * x + 1$

※ 함수 : `divmod(a, b)`

a 나누기 b 의 몫과 나머지를 같이 계산

```
>>> 3 ** 2
9
>>> 3 ** -2
0.11111111111111111
>>> 2 ** 3 ** 2
512
>>> (2 ** 3) ** 2
64
```

지수 승의 우선순위 주의 (뒤에서 부터 연산)

우선 순위를 바꾸고자 할 경우 '()' 괄호를 사용함

Practice

Example 5-2

✦ 초 단위의 시간을 입력 받아 시, 분, 초로 표시하기

```
fullsec = int(input("전체 초 단위의 시간을 입력 : "))
```

```
hour = (fullsec // 60) // 60
```

```
minute = (fullsec // 60) % 60
```

```
seconds = fullsec % 60
```

```
print("%d초는" %fullsec, end=" ")
```

```
print("%d시간 %d분 %d초 입니다" %(hour, minute, seconds))
```

```
전체 초 단위의 시간을 입력 : 5234  
5234초는 1시간 27분 14초 입니다
```

Operator

Arithmetic Operator

✦ 축약 연산자

- 연산자의 축약 형을 사용해서 번거로운 코딩을 줄일 수 있음

축약 형	같은 의미의 수식
<code>a += b</code>	<code>a = a + b</code>
<code>a -= b</code>	<code>a = a - b</code>
<code>a *= b</code>	<code>a = a * b</code>
<code>a /= b</code>	<code>a = a / b</code>
<code>a %= b</code>	<code>a = a % b</code>
<code>a &= b</code>	<code>a = a & b</code>
<code>a = b</code>	<code>a = a b</code>
<code>a ^= b</code>	<code>a = a ^ b</code>
<code>a <<= b</code>	<code>a = a << b</code>
<code>a >>= b</code>	<code>a = a >> b</code>
<code>a **= b</code>	<code>a = a ** b</code>

```
>>> a = 10; b = 5
>>> a += b
>>> a
15
>>> a = 3; b = 2
>>> a **= b
>>> a
9
```

a = a
→ 자기 자신을 다시 대입

a = a + 1
→ 자신에 하나를 더해서 다시 대입

Practice

Example 5-3

✦ 금액을 입력하여 동전으로 교환하는 프로그램

```
money = int(input("교환할 금액을 입력 : "))
m500 = money // 500
money %= 500
m100 = money // 100
money %= 100
m50 = money // 50
money %= 50
m10 = money // 10
money %= 10
print("500원 %d개, 100원 %d개, 50원 %d개, 10원 %d개" %(m500, m100,
m50, m10))
```

교환할 금액을 입력 : 6870
500원 13개, 100원 3개, 50원 1개, 10원 2개

Operator

Comparison Operator

✦ 비교(관계) 연산자는 좌우 두 값이나 두 수식의 크고 작음을 비교함

✦ 연산 결과는 참이면 True, 거짓이면 False 중의 하나를 반환

✦ 연산자 종류

- <, >, <=, >=, ==, !=(같지 않다)

- '=='의 경우 같은 객체이냐를 비교하는 것이 아니라 객체의 값이 같으냐를 비교

```
>>> 6 > 7
```

```
False
```

```
>>> 6 == 7
```

```
False
```

```
>>> 6 != 7
```

```
True
```

```
>>> num = 5
```

```
>>> 0 < num < 10
```

```
True
```

```
>>> "korea" > "japan"      객체 크기의 비교 → 사전에서 먼저 나오는 것이 작음
```

```
True
```

```
>>> 'a' > 'A'      코드 값의 비교
```

```
True
```

Operator

Logical Operator

- ✦ 논리 연산자는 두 피연산자의 논리적인 판단에 따라 결과를 반환
- ✦ 논리 연산의 결과는 참(True), 거짓(False)을 반환함
- ✦ 0 또는 빈 객체이면 거짓 (none, 0, 0.0, [], {}, ())
- ✦ 논리 연산자의 종류
 - and : 논리곱 연산으로 두 조건이 모두 참이면 참
 - or : 논리합 연산으로 두 조건 중 하나만 참이더라도 참
 - not : 조건의 부정

```
>>> not 1
False
>>> not {}
True
>>>
```

피연산자1	피연산자2	and	or
True	True	True	True
True	False	False	True
False	True	False	True
False	false	False	False

Practice

Example 5-5

```
>>> theory = 95; practice = 85
>>> decision = theory >= 90 and practice >= 90
>>> print(decision)
False
>>> decision = theory >= 90 or practice >= 90
>>> print(decision)
True
```

Operator

Logical Operator

✦복합적인 and 연산과 or 연산

- 일반적으로 논리 연산자는 관계 연산자와 함께 사용함
- and 연산이 or 연산보다 우선 순위가 높음
- 관계 연산자는 논리연산자보다 우선순위가 높으므로 먼저 수행함
- 사용 예
 - 어떤 학교의 선생님이 학생의 성적에 따라 면담 여부를 결정하는 조건식
 - 성적이 90점에서 100점 사이의 점수를 가진 학생과 면담이 가능
 - 50점 이하의 학생도 면담이 가능

```
>>> score = 85
>>> decision = score <= 50 or score >= 90 and score <= 100
>>> print(decision) 4          1          2
                                   3

>>> score = 95
>>> decision = score <= 50 or 90 <= score <= 100
>>> print(decision)          5
```

Operator

Logical Operator

✦ 부울형 자료와 상수와의 논리 연산

- True and 10 → 10, False and 10 → False
- 10 and True → True, 10 and False → False
- True or 10 → True, False or 10 → 10
- 10 or True → 10, 10 or False → 10

✦ 상수와 상수와의 논리 연산

```
>>> number = 10 and 20
>>> number
20
>>> number = 10 or 20
>>> number
10
```

and 연산은 처음 조건이 False이면 다음 조건은 검사하지 않음
→ 무조건 False
or 연산은 처음 조건이 True이면 다음 조건은 검사하지 않음
→ 무조건 True

✦ and 연산과 or 연산을 이용한 값의 선택

- 두 피연산자가 상수인 논리 연산일 경우 두 개의 상수 중 하나를 선택
- and 연산은 좌우의 상수 중 우측 상수를 선택
- or 연산은 좌우의 상수 중 좌측 상수를 선택

Practice

Example 5-7

```
>>> count = 5
>>> number = count > 5 and 100 or 200
>>> print(number)

>>> count = 10
>>> number = count > 5 and 100 or 200
>>> print(number)
```

Operator

Function

✦eval("문자열")

- 문자열로 표현된 수식을 인수로 받아 인터프리터를 통해 문장을 수행함
- 단 연산 식만을 처리함

```
>>> eval("10+20")
30
>>> number = 10
>>> eval("number + 2")
12
>>> eval('print("Hello")')
Hello
```

```
>>> number = eval(input("수를 입력 : "))
수를 입력 : 100
>>> number
100
>>> number = eval(input("수를 입력 : "))
수를 입력 : 3.14
>>> number
3.14
```

```
>>> eval("number += 2")
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    eval("number += 2")
  File "<string>", line 1
    number += 2
              ^
SyntaxError: invalid syntax
>>>
>>> eval("print("Hello")")
SyntaxError: invalid syntax
```


Practice

Example 5-11

✦ 일반 수식을 문자열로 입력 받아 그대로 연산하기

```
expression = input("일반 수식을 그대로 입력 : ")  
answer = eval(expression)  
print("결과 : %.2f" %answer)
```

일반 수식을 그대로 입력 : (2+3) * (13/3)
결과 : 21.67

Operator

Function

✦ `exec("문자열")`

- 문자열로 표현된 파이썬 문장을 인수로 받아 컴파일 코드로 변환하고 인터프리터에 의해 실행

```
>>> number = 10
>>> exec("number += 2")
>>> number
12
```

```
>>> code = '''
radius = 3
pi = 3.141592
area = pi * (radius ** 2)
print("원 면적 :", round(area, 2))
'''
>>> exec(code)
원 면적 : 28.27
```

Operator

eval & exec

✦ eval("문자열")

- 문자열로 된 파이썬의 객체로 변환
- 문자열 형태를 가진 수식을 연산할 수 있음

✦ exec("문자열")

- 문자열로 된 파이썬 문장을 실행
- 바인딩(프로그램의 어떤 기본 다뤄가 가질 수 있는 구성요소의 구체적인 값, 성격을 확정하는 것) 가능

```
>>> number = 10
>>> eval("number+2")
12
>>> exec("number += 10")
>>> number
20
```

Control Statement

Overview

✦ 흐름 제어

- 프로그램의 흐름은 컴퓨터에게 내려지는 명령의 순서
 - "입력 받은 데이터를 저장하고 결과를 처리하라"
- 복잡한 명령에 따라 실행의 흐름을 바꿀 수 있음

✦ 조건문

- 참과 거짓의 조건 판단에 따라 두 종류의 방향으로 흐름을 분기하고 각각의 흐름에 대해 별개의 처리를 수행하는 문

✦ 반복문

- 특정 조건을 통해 같은 코드를 여러 번 반복할 수 있도록 하는 제어문
- 계수반복 : 정해진 횟수에 따라 반복을 수행
- 조건 반복 : 특정 조건이 만족할 때까지 반복을 수행

Control Statement

Code Block

✦코드 블록

- 제어문에서 수행해야 할 문장의 블록
- 수행할 문장은 하나일 수도 있으며, 여러 문장 일 수도 있음
- 코드를 보기 쉽게 만들어주며, 코드의 일관성을 유지
- 프로그래밍 언어마다 코드 블록을 표현 방법이 조금씩 다름
 - C언어의 경우 { } (중괄호)를 사용함
- 코드 블록은 함수, 클래스 등에서도 사용됨

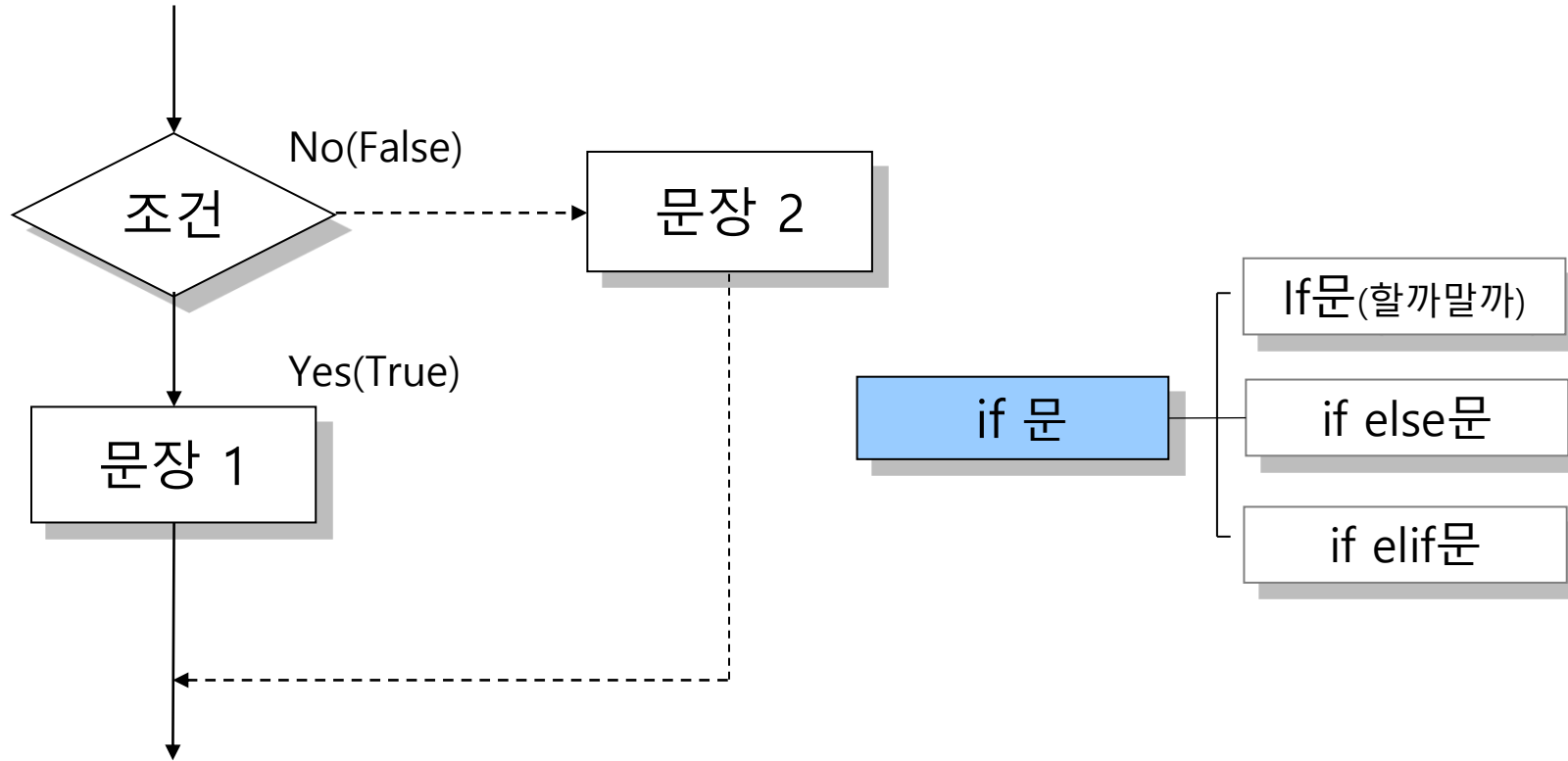
✦파이썬에서의 코드 블록

- 파이썬의 코드 블록은 들여쓰기를 사용
 - 스페이스바나 탭키 중 하나를 사용
- 잘못된 들여쓰기는 문법상 오류를 발생할 수 있으므로 주의

Condition Statement

Overview

- ✦ 조건에 따라 특정 문장을 수행 또는 수행하지 않도록 함
 - 조건은 오직 참(True) 또는 거짓(False)

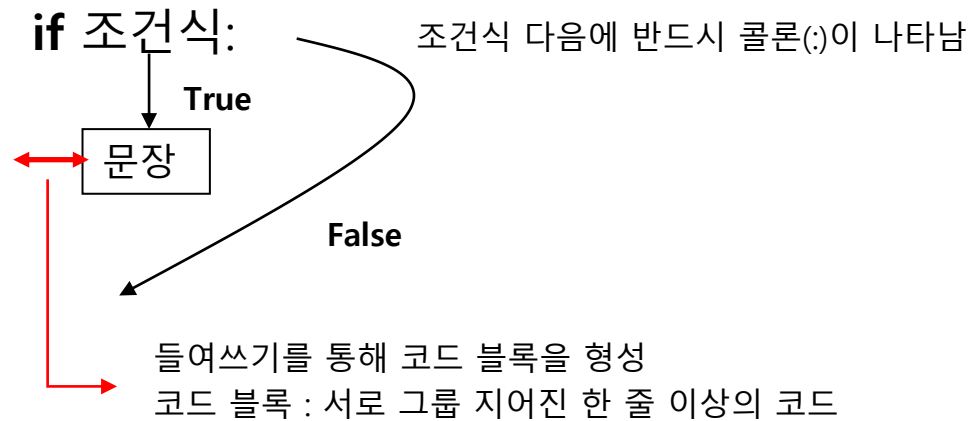
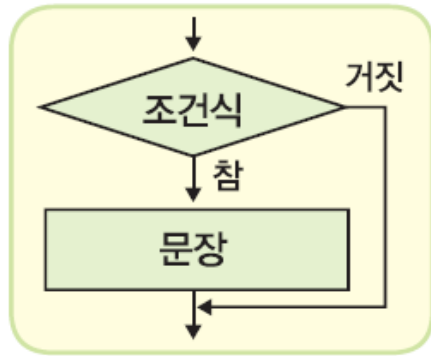


Condition Statement

Basic If

✦ 단순 if문

- if 다음의 조건식이 참이면 문장을 수행하고 거짓이면 수행하지 않음.



- if문의 작성규칙
 - 조건식 다음에 반드시 콜론(:)이 나타남
 - 코드 블록은 들여쓰기를 통해 형성함

Condition Statement

Basic if

✦조건식의 예

- `==` : 같은지 여부를 비교, `!=` : 다른지 여부를 비교
- `>`, `>=`, `<`, `<=` : 두 값의 크기를 비교
- 조건식은 범위의 형태로 표현이 가능함
 - `if 8 < age < 12` → age가 8에서 12사이일 경우
- `not`을 이용하여 비교의 의미를 반대로 사용
 - `if not score < 80` → scorer가 80보다 작지 않을 경우

```
num1 = 30
num2 = 20
```

```
if num1 > num2:    조건이 참일 때 수행함
    max = num1
```

```
max = num2    참 거짓 여부 관계없이 실행
print("max = %d" %max)
```

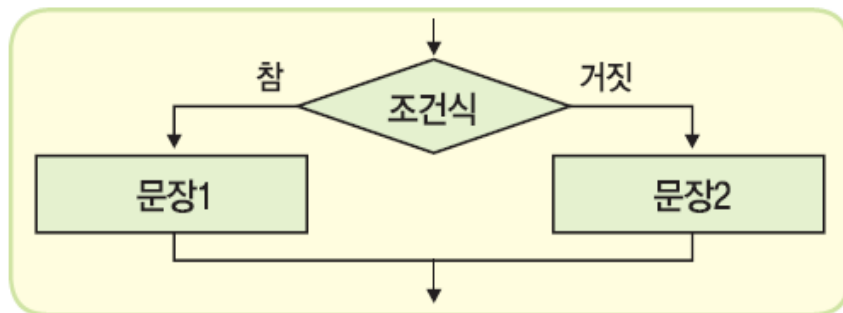
```
>>>
max = 20
```


Condition Statement

if else

✦ if 다음의 조건식을 판단하여 어느 쪽이든 한 개의 문장만 처리

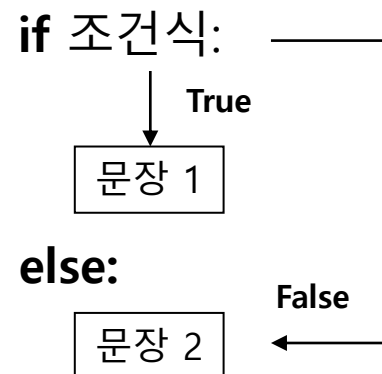
- 참이면 if 다음에 있는 문장을 수행하고 거짓이면 else 다음 문장을 수행



```
score = 70
```

```
if score >= 80:
    print("합격 입니다")
else:
    print("불합격 입니다")
```

```
>>>
불합격 입니다
```



```
if not score < 80:
    print("합격입니다")
else:
    print("불합격 입니다")
```

Practice

Example 6-3

✦ 수를 입력 받아 홀수 짝수 판별하기

```
if number % 2 == 0:  
    print("짝수")  
else:  
    print("홀수")
```

수 입력 : 12
짝수

Practice

Example 6-4

★가격 할인 행사에 다른 상품 구입 가격 구하기

- 상품 구입 가격이 10만원 이상이면 20%, 그렇지 않으면 10%를 할인함

```
price = float(input("상품 가격을 입력 (만원) : "))
```

```
if price >= 10:
```

```
    discount = price * 0.2
```

```
else:
```

```
    discount = price * 0.1
```

```
final_price = price - discount
```

```
print("최종 가격 : %.2f만원" %final_price)
```

```
상품 가격을 입력 (만원) : 15
```

```
최종 가격 : 12.00만원
```

Practice

Example 6-5

✦근무시간에 따른 시간당 임금 구하기

- 단 근무시간이 7시간을 넘을 경우, 남은 시간에 대해서는 시간당 임금의 50%를 추가로 지급함

```
hours = float(input("근무 시간 입력 : "))  
pay = 8000
```

```
if hours > 7:
```

```
    salary = pay * hours
```

```
    salary = salary + (pay * (hours-7)) * 0.5
```

```
else:
```

```
    salary = pay * hours
```

```
print("임금 : %.2f원" %salary)
```

근무 시간 입력 : 9
임금 : 80000.00원

Practice

Example 6-6

★여자 축구단을 모집하는 프로그램

- 단 나이는 10~12살 사이의 여성이어야 함, 성별은 m과 f로 받아들임

```
gender = input("성별을 입력 (남:m, 여:f) : ")
```

```
if gender.lower() == 'f':  
    age = int(input("나이 입력 : "))  
    if 10 <= age <= 12:  
        print("입단이 가능합니다")  
    else:  
        print("나이가 조건에 안맞습니다")
```

```
else:  
    print("여자만 모집합니다")
```

```
성별을 입력 (남:m, 여:f) : F  
나이 입력 : 11  
입단이 가능합니다
```

Condition Statement

if elif

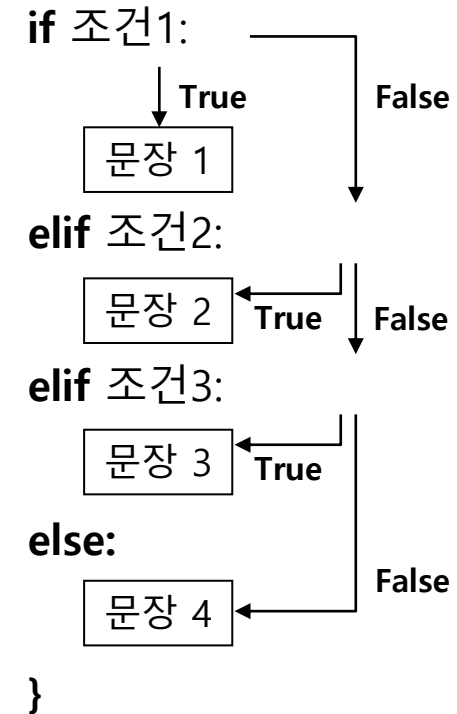
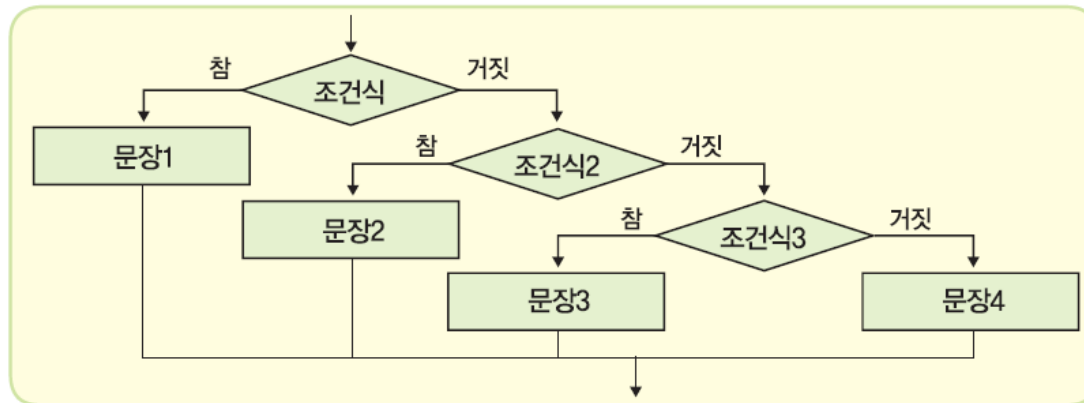
✦ 중첩된 if elif문은 else문에 다시 if문으로 조건을 검사하는 문장

✦ elif문

- 2개 이상의 조건을 처리하며, elif를 통해 다음 조건을 검사

✦ else문

- 어떠한 조건에도 해당하지 않는 경우
- 가장 마지막에만 사용 가능



Condition Statement

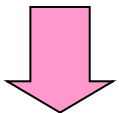
if elif

✦ 어떤 표현이 좀더 효율적일까?

- If문 내부에 다시 if문을 사용하는 것보다 논리 연산자를 이용하여 조건식을 만드는 것이 좋은 표현

```
if menu == 1:  
    if point < price:  
        print("포인트가 부족합니다")
```

동일한 표현이지만 논리적으로 좀더 이해하기 쉬운 코드로 작성하자



```
if menu == 1 and point < price:  
    print("포인트가 부족합니다")
```

Practice

Example 6-7

✦ 점수를 입력 받아 학점을 출력하는 프로그램

90점 이상 100점 이하 A, 80점 이상 90점 미만 B, 70점 이상 80점 미만 C, 나머지 F

```
score = int(input("점수를 입력 : "))
```

```
if 90 <= score <= 100:
```

```
    print("A")
```

```
elif 80 <= score < 90:
```

```
    print("B")
```

```
elif 70 <= score < 80:
```

```
    print("C")
```

```
else:
```

```
    print("F")
```

점수를 입력 : 70
C

Practice

Example 6-8

★ 달의 일수를 구하기

```
month = int(input("달을 입력 : "))
```

```
if month == 2:
```

```
    print("28일")
```

```
elif month == 4 or month == 6 or month == 10:
```

```
    print("30일")
```

```
else:
```

```
    print("31일")
```

```
달을 입력 : 4  
30일
```

Practice

Example 6-11

✦ 사칙 계산기

```
print("1.덧셈, 2.뺄셈, 3.곱셈, 4.나눗셈")  
menu = int(input("메뉴 선택 : "))
```

```
1.덧셈, 2.뺄셈, 3.곱셈, 4.나눗셈  
메뉴 선택 : 3  
첫 번째 수를 입력 : 20  
두 번째 수를 입력 : 3  
결과 : 60
```

Loop Statement

Overview

- ✦ 같은 코드를 여러 번 반복할 수 있도록 하는 제어문
- ✦ 코드 블록을 특정 조건이 만족하는 동안에 반복 실행

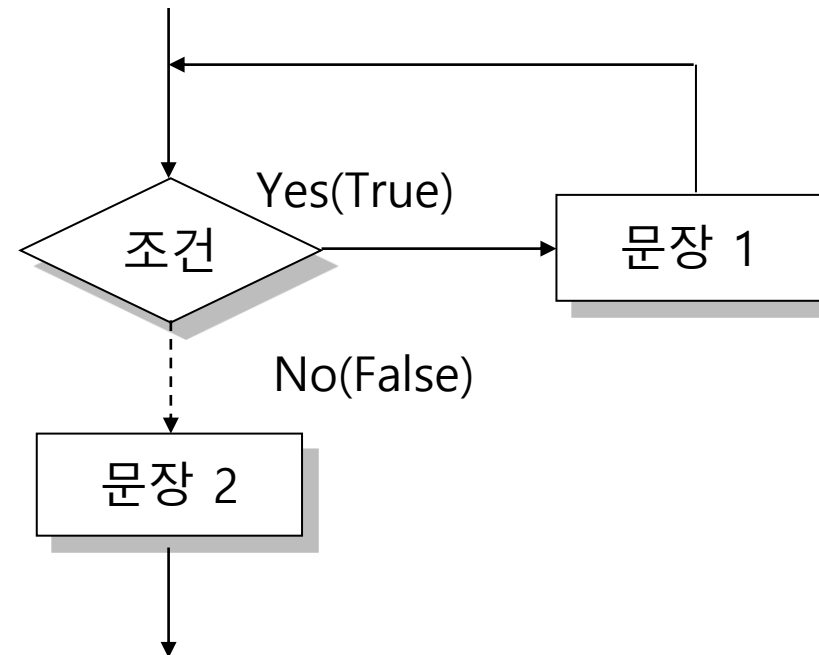
List나 Dictionary 구조에서 반드시 필요함

✦ 반복의 종류

- 계수반복(Counting Loop)
 - 특정 횟수만큼 반복
 - 반복 횟수를 알고 있음
- 조건 반복(Conditional Loop)
 - 특정한 일이 일어나기 전까지 반복
 - 얼마나 반복할지 모름

✦ 종류

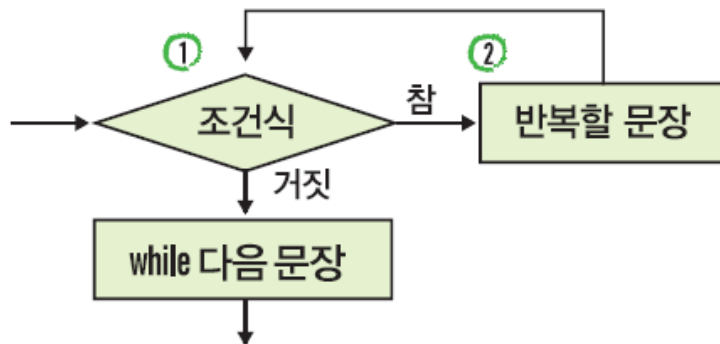
- While문, for문



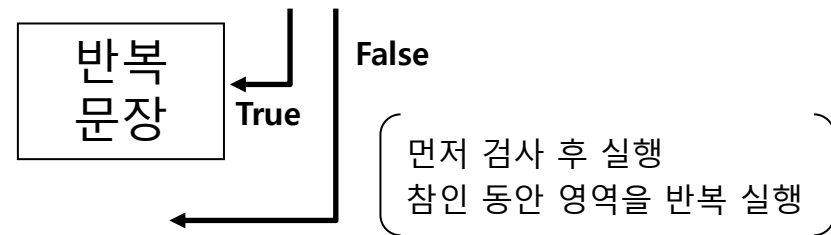
Loop Statement

while

- ✦ while문의 조건식이 참인 동안 아래의 코드 블록 문장을 계속 반복하며, 거짓일 경우 while문을 빠져 나감
- ✦ 조건식에서 언제 반복을 중단할 것인지를 결정함
 - 조건식은 주로 관계 연산자나 논리 연산자를 사용
 - 조건식 다음에는 반드시 ':' 을 넣어야 함
 - 반복으로 수행할 문장은 들여쓰기로 코드 블록을 작성



while 반복조건:



Loop Statement

while

✦계수 반복

- 특정 횟수만큼 반복하므로 횟수를 기록하는 변수와 횟수를 증가 시킴

```
count = 1
while count <= 10:
    print("hello python")
    count += 1
```

✦무한 반복

- 반복의 조건이 무조건 참이므로 계속 반복하는 코드

```
count = 1
while True:
    print("hello python")
    if count == 10:
        break
    count += 1
```

무한 루프

조건 반복 : 조건이 참일 경우 break 문을 통해 반복을 빠져 나옴

Practice

Example 6-12

✦ 1부터 10까지의 합 구하기

total = 0

count = 1

합 : 55

Practice

Example 6-15

✦ 두 수의 중간 값 구하기

```
minNumber = int(input("작은 수 입력 : "))  
maxNumber = int(input("큰 수 입력 : "))
```

```
while True:  
    if minNumber >= maxNumber:  
        break  
    minNumber += 1  
    maxNumber -= 1
```

```
print("중간값 :", minNumber)
```

```
작은 수 입력 : 150  
큰 수 입력 : 480  
중간값 : 315
```

Loop Statement

while else

✦ while문에서 else문 사용이 가능함

- 모든 반복을 다 수행하였을 경우 else문을 수행
- 반복을 다 수행하지 못하고 빠져나올 경우 else문은 수행하지 않음

```
count = 1
while count <= 5:
    if count == 3:
        break
    print("python", count)
    count += 1
else:
    print("감사합니다")
```

```
python 1
python 2
```

```
count = 1
while count <= 5:
    print("python", count)
    count += 1
else:
    print("감사합니다")
```

```
python 1
python 2
python 3
python 4
python 5
감사합니다
```


Practice

Example 6-17

✦ 비밀번호 입력 프로그램

- 단 비밀번호가 5회 틀릴 경우 메시지를 출력

```
passwd = "python1234"
tries = 1

while tries < 6:
    guess = input("비밀번호 : ");
    if guess == passwd:
        print("비밀번호가 일치합니다")
        break
    else:
        print("비밀번호가 일치하지 않습니다")
        if tries != 5:
            print("비밀번호 %d회 오류입니다" %tries)
            print("총 %d회 남았습니다" %(5-tries))

        tries += 1

else:
    print("처음부터 다시 시작하세요")
```

비밀번호 : python1234
비밀번호가 일치합니다

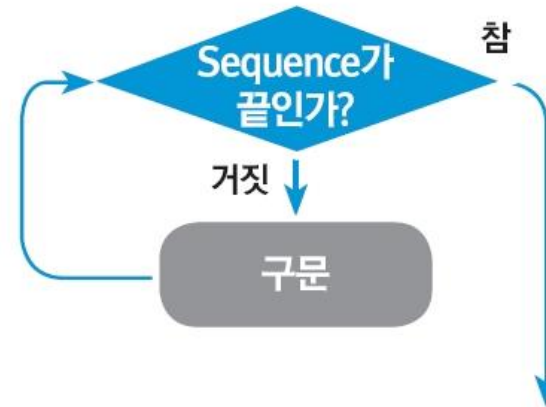
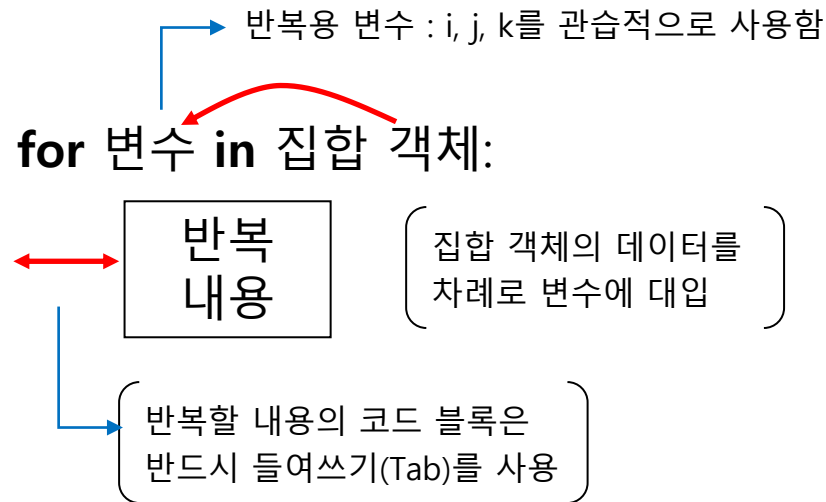
비밀번호 : 1235
비밀번호가 일치하지 않습니다
비밀번호 4회 오류입니다
총 1회 남았습니다
비밀번호 : 5632
비밀번호가 일치하지 않습니다
처음부터 다시 시작하세요

Loop Statement

for

✦ 순차 타입의 객체를 차례로 순회하여 반복함

- 순차 객체의 요소들을 차례로 for문 변수에 할당하여 처리함
- 모든 요소들을 순회하거나 break를 만나면 for문이 종료
- for문의 마지막에 콜론(:)을 사용하며, 반복문장은 코드블록을 사용



Loop Statement

for

✦ 사용 예

- in 다음의 리스트 순차 자료 개수 만큼 반복하여 "python" 문자열 출력

```
for number in [1, 2, 3, 4, 5]:  
    print("python")
```

- 리스트의 개수만큼 차례로 반복하면서 score 변수에 하나씩 저장

```
for score in [100, 98, 85, 70]:  
    print(score)
```

- in 다음의 문자열 길이 만큼 반복하면서 각 문자를 letter 변수에 저장

```
for letter in "hello":  
    print(letter)
```

- in 다음의 리스트의 문자열 수만큼 반복하여 문자열을 변수에 저장

```
for animal in ["dog", "cat", "pig", "lion"]:  
    print("동물 :", animal)
```

Practice

Example 6-18

✦ 5개 성적의 합과 평균 구하기

```
total = 0
for score in [100, 98, 85, 97, 74]:
    total += score
average = total / 5;
```

```
print("합 : %d" %total)
print("평균 : %.2f" %average)
```

```
합 : 454
평균 : 90.80
```

Loop Statement

for & range()

✦ 함수 : range(start, end, step)

- start : 시작 값, end : 종료 값, step : 증가 값
- 해당 범위의 숫자 리스트를 반환
- for문과 같은 제어문에서 많이 사용함
- 시작 값을 생략할 경우 0부터 진행함
- 종료 값 end는 포함 안됨
- 예) range(1, 5) → [1, 2, 3, 4]

```
for count in [0, 1, 2, 3, 4]:  
    print(count)
```

count는 0에서 4까지 반복할때마다
0에서 4까지 하나씩 출력

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

0에서 4까지 데이터 출력

```
for count in range(1, 10, 2):  
    print(count)
```

1, 3, 5, 7, 9 출력 (10은 출력하지 않음)

```
for count in range(10, 1, -1):  
    print(count)
```

9, 8, 7, 6, 5, 4, 3, 2 출력

Practice

Example 6-20

✦ 1에서부터 사용자가 입력한 수까지 합 구하기

```
number = int(input("수를 입력 : "))  
total = 0  
for count in range(1, number+1):  
    total += count  
print("합 :", total)
```

```
수를 입력 : 100  
합 : 5050
```

Practice

Example 6-22

✦ 1에서 100까지 3의 배수이면서 7의 배수인 값 구하기

```
for number in range(1, 101):  
    if number % 3 == 0 and number % 7 == 0:  
        print(number, end=" ")
```

21 42 63 84

Loop Statement

for else

✦for문에 else문 사용할 수 경우

- 모든 반복을 다 수행했을 경우 else문을 수행
- 반복 수행 중 중간에 중지되는 경우 else문을 수행하지 않음

Practice

Example 6-23

✦ 1에서 입력한 수까지 합을 구할 때 합이 10000이 넘는지 확인하기

```
number = int(input("수를 입력 : "))
total = 0
for count in range(1, number+1):
    total += count
    if total > 10000:
        print("합이 10000을 넘는 수 :", count)
        break
else:
    print("합이 10000을 넘지 않습니다")
```

```
수를 입력 : 100
합이 10000을 넘지 않습니다
```

```
수를 입력 : 1000
합이 10000을 넘는 수 : 141
```

Loop Statement

Nested Loop

✦ 중첩된 반복문

- 반복문안에 다시 반복문을 사용하여 표현 한 것
- 내부의 반복문은 외부의 반복문이 새롭게 수행될 때마다 다시 수행

```
count1 = 1
while 조건1:
    count2 = 1
    while 조건2:
        반복 문장
        count2 += 1
    count1 += 1
```

```
for 반복변수1 in 순차타입의 객체:
    for 반복변수2 in 순차타입의 객체:
        반복 문장
```

Practice

Example 6-24, 6-25

✦ 중첩된 반복문을 이용한 구구단 작성

```
dan = 2
while dan <= 9:
    su = 1
    while su <= 9:
        print("%d X %d = %d" %(dan, su, dan*su))
        su += 1
    dan += 1
```

```
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
```

```
for dan in range(2, 10):
    for su in range(1, 10):
        print("%d X %d = %d" %(dan, su, dan*su))
```

```
9 X 6 = 54
9 X 7 = 63
9 X 8 = 72
9 X 9 = 81
```

Practice

Example 6-27

✦ 라인과 별의 수를 입력하여 별 블록 생성하기

```
line = int(input("라인 개수 : "))  
star = int(input("별의 개수 : "))
```

```
for lcount in range(line):  
    for scount in range(star):  
        print("*", end="")  
    print()
```

```
라인 개수 : 3  
별의 개수 : 5  
*****  
*****  
*****
```

Practice

Example 6-28

✦ 별을 이용하여 삼각형 그리기

```
for line in range(1, 6):  
    for star in range(line):  
        print("*", end="")  
    print()
```

```
for line in range(1, 5):  
    for star in range(line, 5):  
        print("*", end="")  
    print()
```

```
*  
* *  
* * *  
* * * *  
* * * * *  
* * * *  
* * *  
* *  
*  
*
```


Control Statement

Break

✦ Break 문

- 반복구조에서 내부의 특정 조건에 따라 루프를 탈출하는 용도로 사용
- Break를 만나면 반복을 즉시 종료하고 해당 반복문 끝으로 제어를 옮김
- else문이 있을 경우 수행하지 않음

```
while True:           무한 루프
    .....
    if x < 0:          조건이 참일 경우
        break;
    .....
else:
    while문을 빠져 나와 다음 문장으로
```



```
for letter in "python":
    if letter == "t":
        break
    print(letter, end="")
else:
    print("출력완료")
```

결과 : py

Control Statement

Continue

✦continue 문

- 현재 수행중인 반복을 중단하고 다음 반복으로 곧장 넘어감
- 반복문의 시작 부분으로 돌아가 루프를 계속 수행
- 루프의 나머지 부분은 건너 띄고 넘어감
- 다시 조건을 검사하여 루프를 계속 수행
- else문이 있을 경우 결국 반복을 다 수행한 것이므로 else문을 수행함

```
for letter in "python":  
    if letter == "t":  
        continue  
    print(letter, end="")  
else:  
    print("출력완료")
```

결과 : pyhon출력완료

Practice

Example 6-30

✦수를 입력하여 소수(prime) 판별하기

```
while True:
    prime = True
    number = int(input("수를 입력 (0:종료) : "))

    if number == 0:
        break
    if number == 1:
        print("값을 다시 입력하세요")
        continue

    for count in range(2, number):
        if number % count == 0:
            prime = False
            break

    if prime:
        print("소수입니다")
    else:
        print("소수가 아닙니다")
```

```
수를 입력 (0:종료) : 11
소수입니다
수를 입력 (0:종료) : 16
소수가 아닙니다
수를 입력 (0:종료) : 0
```