

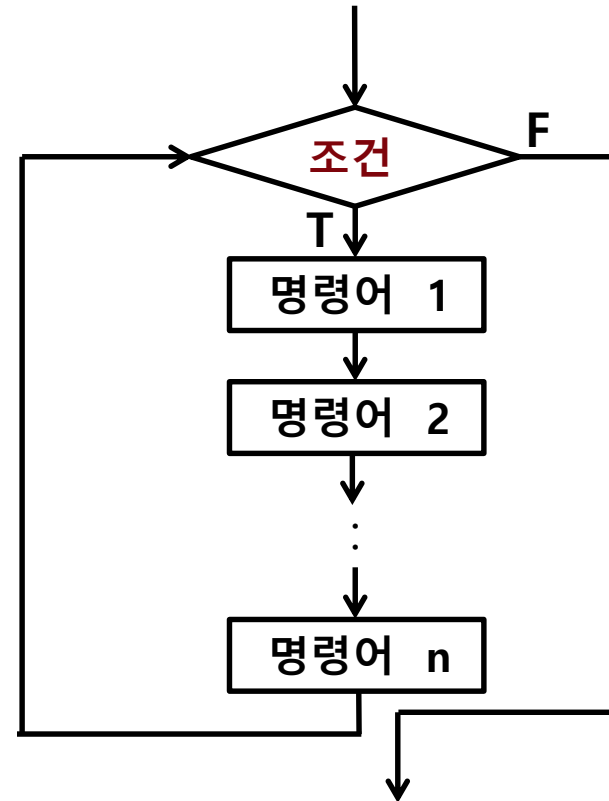
IT 개론

6장. 반복문

3. 반복 논리의 기본 개념

◆ 반복 논리

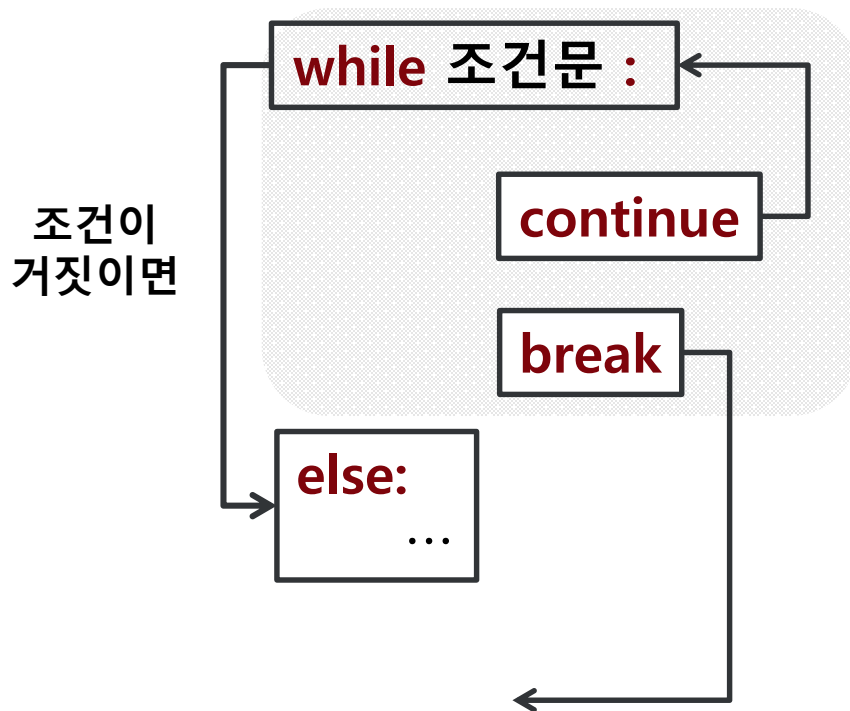
- 어떤 문제를 해결하는 과정에서 특정 명령 또는 연산을 반복적으로 수행해야 하는 경우가 발생한다.
- 순서도에서는 반복을 제어할 판단문과 반복적으로 실행해야 하는 명령문으로 반복 논리를 나타낸다.



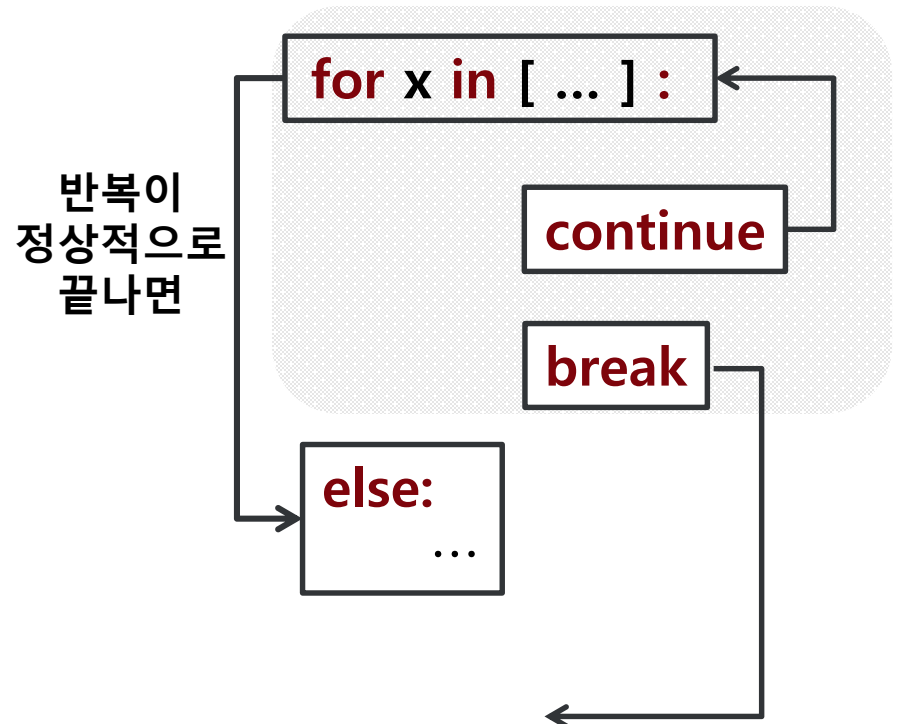
3. 반복 논리의 기본 개념

◆ 파이썬의 두 가지 반복 유형

while 반복문



for 반복문



else, continue, break 는 있을 수도 있고 없을 수도 있다.

3. 반복 논리의 기본 개념

코드 for.py

```
patterns = ['가위', '보', '보', '가위', '가위', '가위', '보', '가위', '바위', '보']  
for pattern in patterns:  
    print(pattern)
```

가위

보

보

가위

가위

가위

보

가위

가위

보

patterns의 element가 순서대로 출력되었다

3. 반복 논리의 기본 개념



```
for pattern in patterns:  
    print(pattern)
```

patterns로부터 값을 하나씩 가져와
pattern이라는 '새로운 변수'에 대입한다.

3. 반복 논리의 기본 개념

내용을 대입할 변수 이름
리스트의 원소

반복하려는 내용
리스트



```
for pattern in patterns:  
    print(pattern)
```

반복할 코드 블록

3. 반복 논리의 기본 개념

코드

```
for i in range(5):    # range(5)가 나타내는 범위 안에 있는 i를  
    print(i)         # 출력한다
```

0

1

2

3

4

`range(a)`

0부터 시작하여 a 바로 직전까지, 즉 a-1까지의 범위가
다

`range(a, b)`

a부터 b-1까지

0부터 999까지 표현하고자 하면, `range(1000)`으로 나타내면 된다.

3. 반복 논리의 기본 개념

코드

```
names = ['철수', '영희', '바둑이', '귀도']  
  
for i in range(4):  
    name = names[i]  
    print('{}번: {}'.format(i + 1, name))
```

1번: 철수
2번: 영희
3번: 바둑이
4번: 귀도

리스트의 크기는 중간에 변할 수 있으므로, 4, 5와 같은 상수 값을 range에 사용하는 것은 좋은 프로그래밍이 아니다.

3. 반복 논리의 기본 개념

코드

```
names = ['철수', '영희', '바둑이', '귀도', '비단뱀']  
  
for i in range(len(names)):  
    name = names[i]  
    print('{}번: {}'.format(i + 1, name))
```

1번: 철수
2번: 영희
3번: 바둑이
4번: 귀도
5번: 비단뱀

`len()` : **length**. 실행인자로 전달받은 변수에 항목이 몇 개 있는지를 되돌려준다. 즉, 전달받은 변수의 길이를 전달하는 것으로 이해하면 쉽다.

3. 반복 논리의 기본 개념

코드

```
for i, name in enumerate(names):  
    print('{}번: {}'.format(i + 1, name))
```

1번: 철수
2번: 영희
3번: 바둑이
4번: 귀도
5번: 비단뱀

`enumerate`는 순서 `i`와 변수 `name`을 둘다 generate한다.

3. 반복 논리의 기본 개념

각 방법은 언제 사용하는가?

1. 이미 사용할 값의 **목록이 정해져 있고**, 그 목록에서 값을 하나씩 꺼내 쓰기만 하면 된다 → *for in list*
2. **횟수가 정해져** 있거나 일정하게 증가하는 숫자를 써야 한다
→ *for in range*

5. for 반복문

◆ for 구문을 이용한 반복문

for **x** **in** [.....] :
명령어 1
명령어 2
...
명령어 n
else :
명령어 n+1

변수

range() 함수
문자열
리스트
튜플
집합
사전

데이터 집합에서 데이터를 하나씩
변수 x에 넣고 반복을 수행한다.

else 구문이 있으면 for 반복을 끝내
고 나갈 때 else 부분이 수행된다.

5. for 반복문

◆ range() 함수

`range(b)` – 0부터 `b-1`까지의 수를 반환한다.

`range(a,b)` – `a`부터 `b-1`까지의 수를 반환한다.

`range(a,b,n)` – `a`부터 `b-1`까지의 `n` 간격의 수를 반환한다

```
>>> range(5)
range(0, 5)
>>> range(10)
range(0, 10)
>>> print(range(5))
range(0, 5)
```


```
>>> list(range(5))
[0, 1, 2, 3, 4]
>>> list(range(5,10))
[5, 6, 7, 8, 9]
>>> list(range(3,10,2))
[3, 5, 7, 9]
>>> list(range(10,1,-2))
[10, 8, 6, 4, 2]
```

5. for 반복문

◆ range() 함수에 대해서 for 구문 수행하기

예제 1

```
for x in range(5): # range 함수 사용  
    print(x)
```



range(5) 자리에 [0,1,2,3,4]가 와서 차례대로 x값이 되어 for 블록을 수행한다.

```
>>>  
0  
1  
2  
3  
4
```

예제 2

```
for data in range(1,10,2):  
    print(data+1)  
    print(data*2)
```

```
2  
2  
4  
6  
6  
10  
8  
14  
10  
18
```

5. for 반복문

◆ 문자열에 대해서 for 구문 수행하기

예제 3

```
for letter in 'PYTHON': # 문자열 사용  
    print(letter)
```

↑
'PYTHON' 이 하나씩 letter 값이 되어
서 for 블록을 수행한다.

```
>>>  
P  
Y  
T  
H  
O  
N
```

예제 4

```
for alpha in 'sogang':  
    y = alpha.upper()  
    print(y)
```

```
S  
O  
G  
A  
N  
G
```

5. for 반복문

◆ 리스트에 대해서 for 구문 수행하기

예제 5

```
for x in [1,3,5,7,9]: # 리스트 사용
    print(x)
```

↑
[1,3,5,7,9] 원소가 하나씩 x값이 되어 for 블록을 수행한다.

```
>>>
1
3
5
7
9
```

예제 6

```
for x in [1,2,3,4,5]:
    for y in [2,4,6,8,10]:
        print(x*y, end=' ')
    print()
```

```
2 4 6 8 10
4 8 12 16 20
6 12 18 24 30
8 16 24 32 40
10 20 30 40 50
```


5. for 반복문

◆ 튜플에 대해서 for 구문 수행하기

예제 7

```
for a in (6,2,9,8): # 튜플 사용
    print(a)
```

(6,2,9,8) 원소가 하나씩 a값이 되어
for 블록을 수행한다.

```
>>>
6
2
9
8
```

예제 8

```
for a in (1,2,3,4):
    for b in (5,6,7):
        print(a+b, end=' ')
    print()
```

```
6 7 8
7 8 9
8 9 10
9 10 11
```

5. for 반복문

◆ 집합에 대해서 for 구문 수행하기

예제 9

```
for a in {3,6,4,2,8}: # 집합 사용
    print(a)
```

8
2
3
4
6

{3,6,4,2,8} 원소가 하나씩 a값이
되어 for 블록을 수행한다.

예제 10

```
for a in {1,2,3}:
    for b in {4,5}:
        print(a,b)
```

1 4
1 5
2 4
2 5
3 4
3 5

5. for 반복문

◆ 사전에 대해서 for 구문 수행하기 (1)

사전 score는 {번호:성적}으로 구성된다.

```
score = {4:90, 2:80, 1:95, 5:88, 3:92}
```

예제 11 `for x in score: # 사전 사용
 print(x)`

예제 12 `for x in score.keys():
 print(x)`



1
2
3
4
5

5. for 반복문

◆ 사전에 대해서 for 구문 수행하기 (2)

```
score = {4:90, 2:80, 1:95, 5:88, 3:92}
```

예제 13

```
for x in score.values():  
    print(x)
```

```
95  
80  
92  
90  
88
```

예제 14

```
for x,y in score.items():  
    print(x,y)
```

```
1 95  
2 80  
3 92  
4 90  
5 88
```

5. for 반복문

◆ 다양한 형태로 for 구문 사용하기

```
L1 = []  
for x in range(1,5):  
    L1.append(x)  
print(L1)
```

```
M1 = [x for x in range(1,5)]
```

```
L2 = []  
for i in range(1,4):  
    for j in range(6,8):  
        L2.append((i,j))  
print(L2)
```

```
M2 = [(i,j) for i in range(1,4) for j in range(6,8)]
```

5. for 반복문

◆ 다양한 형태로 for 구문 사용하기

```
>>> colors = [ "red", "green", "yellow", "blue" ]
>>> things = [ "house", "car", "tree" ]
>>> colored_things = [ (x,y) for x in colors for y in things ]

>>> print(colored_things)
[('red', 'house'), ('red', 'car'), ('red', 'tree'), ('green', 'house'), ('green', 'car'), ('green', 'tree'), ('yellow', 'house'), ('yellow', 'car'), ('yellow', 'tree'), ('blue', 'house'), ('blue', 'car'), ('blue', 'tree')]
```

```
>>> [(x,y,z) for x in range(1,30) for y in range(x,30) for z in range(y,30) if x**2 + y**2 == z**2]
```

```
[(3, 4, 5), (5, 12, 13), (6, 8, 10), (7, 24, 25), (8, 15, 17), (9, 12, 15), (10, 24, 26), (12, 16, 20), (15, 20, 25), (20, 21, 29)]
```

5. for 반복문

◆ 다양한 형태로 for 구문 사용하기

$$A = \{ x^2 : x \in \{0, 1, 2, \dots, 9\} \}$$
$$B = \{ 1, 2, 2^2, 2^3, \dots, 2^{20} \}$$
$$C = \{ x \mid x \in A \text{ and } x \text{ is even} \}$$

```
>>> A = [ x**2 for x in range(10)]
```

```
>>> print(A)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
>>> B = [2**i for i in range(21)]
```

```
>>> print(B)
```

```
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288, 1048576]
```

```
>>> C = [x for x in A if x%2 == 0]
```

```
>>> print(C)
```

```
[0, 4, 16, 36, 64]
```

5. for 반복문

◆ 다양한 형태로 for 구문 사용하기

```
>>> A = [j for i in range(2, 8) for j in range(i*2, 100, i)]
>>> B = [x for x in range(2, 100) if x not in A]

>>> print(B)
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59,
61, 67, 71, 73, 79, 83, 89, 97]
```


3. 반복 논리의 기본 개념

◆ 반복 논리 예제 - 1부터 10까지의 합

$$1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = ?$$

```
a = 1  
sum = 0
```

```
a <= 10 인 동안 다음은 반복한다.  
    sum += a  
    a += 1
```

반복이 끝나면 sum을 출력한다.

```
a = 1  
sum = 0  
while a <= 10 :  
    sum += a  
    a += 1  
print('sum :', sum)
```

4. while 반복문

◆ 1부터 10까지의 합 구하기

```
a = 1
sum = 0
while a <= 10 :
    sum += a
    a += 1
print('sum :', sum)
```

```
>>>
sum : 55
```

4. while 반복문

◆ 1부터 10까지 짝수의 합 구하기

방법 1 : 2부터 시작하여 2씩 증가시켜 나가면서 합을 구한다.

방법 2 : 1부터 시작하여 1씩 증가시켜 나가면서 짝수인지 판단하여 짝수이면 합을 구한다.

```
a = 2
sum = 0
while a <= 10 :
    sum += a
    a += 2
print('sum :', sum)
```

```
a = 1
sum = 0
while a <= 10 :
    if a % 2 == 0 :
        sum += a
    a += 1
print('sum :', sum)
```

4. while 반복문

◆ else 구문이 있는 while 구문

- while의 조건이 False가 되어 반복을 끝낼 때, else 구문이 있으면 수행한다.
- 예제 - 1부터 30까지 3의 배수만을 출력한다.

```
a = 1

while a <= 30 :
    if a%3 == 0 :
        print(a)
    a += 1
else :
    print('exit from while...')

print('after while')
```

```
3
6
9
12
15
18
21
24
27
30
exit from while...
after while
```

4. while 반복문

◆ while 구문에 break 사용하기

- break 는 반복문 안에서 사용하면 반복문을 끝내도록 한다.

```
a = 1

while a <= 10 :
    print('a =', a)
    if a >= 5 :
        break
    a += 1

print('after while')
```

```
>>>
a = 1
a = 2
a = 3
a = 4
a = 5
after while
```

4. while 반복문

◆ while 반복문에 else, break 모두 있는 경우 (1)

- while 반복문이 break로 반복문을 끝내는 경우에는 else 구문이 있더라도 실행되지 않는다.

```
a = 1
while a <= 5 :
    print('a =',a)
    if a >= 3 : break
    a += 1
else:
    print('else in while')
print('after while')
```

```
>>>
a = 1
a = 2
a = 3
after while
```

4. while 반복문

◆ while 반복문에 else, break 모두 있는 경우 (2)

```
a = 1

while a <= 5 :
    print('a =',a)
    if a >= 10 : break
    a += 1
else:
    print('else in while')

print('after while')
```

```
a = 1
a = 2
a = 3
a = 4
a = 5
else in while
after while
```

4. while 반복문

◆ while 반복문에 continue 사용하기

- continue 는 반복문 안에서 사용하면 반복문의 조건으로 제어가 간다.

```
a = 0

while a <= 10 :
    a += 1
    if a%3 == 0 :
        continue
    print('a =',a)

print('after while')
```

```
>>>
a = 1
a = 2
a = 4
a = 5
a = 7
a = 8
a = 10
a = 11
after while
```

continue 구문을 사용하면 continue 조건이 참인 경우 continue 아래 부분은 실행하지 않는다.