순환문 개요

순환문(loop)이란?

- 특정 조건을 충족하는 동안 작업을 계속 반복 실행하는 일련의 명령이다.
- '반복문' 이라고도 부른다.

일반적으로 순환문은 미리 정해 놓은 조건을 만족하면 일련의 명령문들을 실행한다.

그리고 다시 돌아가서 조건을 만족하는지 검사한다.

조건을 만족하면 다시 같은 작업을 반복 실행한다.

만약 더 이상 조건을 만족하지 않으면 순환문을 빠져나가게 된다.

파이썬에서 사용하는 순환문 형태로는 두 가지가 있다.

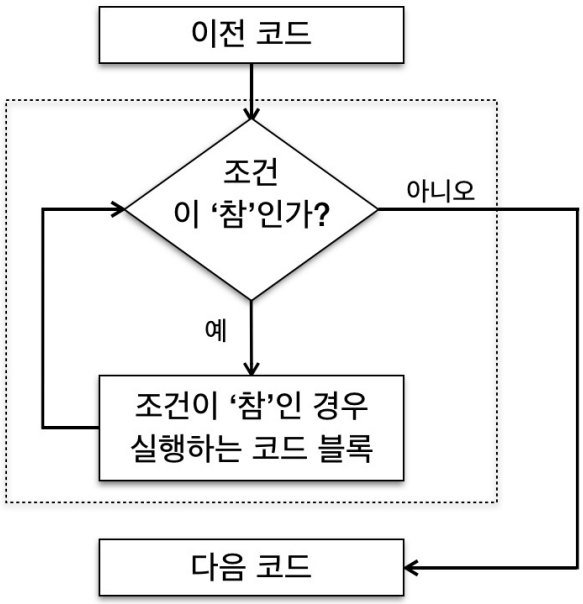
while문

- 주어진 조건이 '참'인 동안 **while**문에 속한 명령문을 반복적으로 실행한다.

for문

- 주어진 조건이 '참'인 동안 **for**문에 속한 명령문을 반복적으로 실행한다.

조건문의 기본적인 흐름도는 다음 그림과 같다.



while문 형식

while문의 일반 형식은 다음과 같다.

```
while 불린-표현식:
    while-명령문
else:
    else-명령문
```

다음과 같은 특징이 있다.

while문에 있는 ***불린-표현식***이 '참(**True**)'이면, ***while-명령문***을 실행한다.

- ***불린-표현식***이 '거짓(**False**)'일 때 까지 ***while-명령문***을 반복해서 실행한다.

while문에 있는 ***불린-표현식***이 '거짓(**False**)'이면, **while**문을 종료하고 **else**문의 ***else-명령문***을 실행한다.

- 즉, **while**문을 정상적으로 종료했다면 **else**문은 반드시 실행된다.

else-명령문을 실행한 후, **while**문이 종료된다.

다음 두 가지 경우에는 **else**문이 실행되지 않는다.

- **while**문이 **break** 또는 **return**에 의해 종료된다.
- **while**문 실행 중 **예외**(exception)가 발생한다.

else문 실행 규칙은 아래 명령문에 모두 적용된다.

- **while**문
- **for**문
- **try-except**문

else문은 선택사항이다.

while과 **else**의 마지막에는 쌍점(**:**)이 온다.

단순 while문

while문만 있고 **else**문이 없는 순환문을 작성해보자.

다음 예는 **while**문을 사용해 '안녕 파이썬'을 다섯 번 출력한다.

In [ ]:

```
i = 0
while i < 5:
    print('안녕 파이썬')
    i += 1
```

따라해보기

다음 예는 **while**문을 10번 반복해서 실행하면서 숫자를 출력한다.

In [ ]:

```
i = 0          # 센티널 변수를 0으로 초기화한다.
while i < 10:
    print(i)
    i += 1     # 센티널 값을 1 증가시킨다.
```

while문을 이용해 시퀀스형이 담고 있는 객체들을 꺼집어 내어 처리할 수도 있다.

따라해보기

다음 예는 튜플을 순회하면서 객체를 하나씩 꺼내어 출력한다.

In [ ]:

```
t = tuple('abcdefg')
print(t)
```

In [ ]:

```
i = 0  # 센티널 변수로 리스트의 인덱스로 사용한다.
while i < len(t):
    print(t[i])
    i += 1
```

while-else문

이번에는 **while-else**문을 작성해보자.

다음 예는 **while**문을 이용해 1부터 10까지 숫자를 더한 후 그 결과를 출력하는 코드다.

In [ ]:

```
total = 0
i = 0
while i < 10:  # 명령문을 10회 반복해서 수행한 후 종료한다.
    i += 1
    total += i
else:          # 결과를 출력한다.
    print('1부터 10까지의 합은 {}입니다.'.format(total))
```

다음 예는 튜플에 들어있는 정수들의 합을 구해 출력한다.

In [ ]:

```
integers = 2, 8, 3, 5, 10, 27
total = 0
i = 0
while i < len(integers):     # 리스트의 모든 항목을 처리한 후 종료한다.
    total += integers[i]     # 리스트 인덱스 0부터 len(integers) - 1, 즉 전체 객체의 합을 구한다.
    i += 1                   # 리스트 인덱스 번호를 1 증가시킨다.
else:                        # 결과를 출력한다.
    print('튜플에 들어있는 숫자의 합은 {}입니다.'.format(total))
```

다음 예는 **while**문을 이용해 메뉴를 선택해서 실행할 수 있게 한다.

In [ ]:

```
menu = '''1. 학생 추가
2. 학생 삭제
3. 학생 목록 보기
4. 종료'''

number = ''
while number != '4':
    print(menu)
    number = input('메뉴 번호를 입력하세요: ')
    if number in '1234':
        print('{}번 메뉴를 선택했습니다.\n'.format(number))
    else:
        print('{}번은 없는 메뉴 번호입니다.\n'.format(number))
else:
    print('Bye~~')
```

따라해보기

지금까지의 예는 특정 횟수까지 **while**문을 반복한 후 종료했다면,

이번에는 횟수에 상관없이 조건이 '거짓'이면 실행을 중단하고 **while**문을 빠져 나가는 프로그램을 작성해보자.

In [ ]:

```
import random
fruits = ['사과', '딸기', '바나나', '블루베리', '포도']
fruit = ''
while fruit != '블루베리':
    # 리스트가 담고 있는 객체 중 하나를 무작위로 선택한다.
    fruit = random.choice(fruits)
    print(fruit)
else:
    print('블루베리가 선택되어 프로그램을 종료합니다.')
```

따라해보기

이번에는 **break** 명령어를 사용해서

임의로 선택한 항목이 '블루베리'인 경우 이것을 출력하지 않고 바로 **while**문을 종료하는 프로그램을 작성해보자.

이 경우 **while**문이 **break**에 의해 종료되기 때문에 **else**문은 실행되지 않는다.

In [ ]:

```
import random
fruits = ['사과', '딸기', '바나나', '블루베리', '포도']
fruit = ''
while fruit != '블루베리':
    fruit = random.choice(fruits)
    print(fruit)
    if fruit == '블루베리':
        break
else:  # else문은 절대 실행되지 않는다.
    print('블루베리가 선택되어 프로그램을 종료합니다.')
```