for문 형식

for문의 일반 형식은 다음과 같다.

```
for 변수 in 순회형:
    for-명령문
else:
    else-명령문
```

다음과 같은 특징이 있다.

순회형에 들어 있는 객체들을 처음부터 마지막까지 추출해서

차례로 ***변수***에 할당한 후 ***for-명령문***을 실행한다.

변수는 ***순회형***에서 가져온 객체를 참조하는

- 하나의 변수일 수도 있고,
- 복합 변수일 수도 있다.
  - 복합 변수면 주로 튜플을 사용한다.

for문을 종료한 후, **else**문의 ***else-명령문***을 실행한다.

- 즉, **for**문이 정상적으로 종료되었다면 **else**문은 반드시 실행된다.

else-명령문을 실행한 후, **for**문이 종료된다.

다음 두 가지 경우에는 **else**문이 실행되지 않는다.

- **while**문이 **break** 또는 **return**에 의해 종료될 때
- **while**문 실행 중 **예외**(exception)가 발생할 때

else문은 선택사항이다.

for와 **else**의 마지막에는 쌍점(**:**)이 온다.

for문의 실행 순서는 다음과 같다.

1. ***변수***는 ***순회형*** 자료의 제일 앞에 있는 객체를 할당 받는다.
2. ***for-명령문***을 실행한다.
3. ***변수***는 이전에 받은 객체의 바로 다음 객체를 할당 받는다.
4. ***for-명령문***을 실행한다.
5. ... 반복 ...
6. 최종적으로 ***순회형***의 마지막 객체까지 ***변수***에 할당되면, **for**문을 종료하고 **else**문의 ***else-명령문***을 실행한다.
7. ***else-명령문***을 실행한 후, **for**문을 종료한다.

단순 for문

for문만 있고 **else**문이 없는 순환문을 작성해보자.

다음 예는 문자열을 순회하면서 문자를 하나씩 꺼내어 출력한다.

In [ ]:

```
for i in ['a', 'b', 'c', 'd', 'e']:
    print(i)
```

NOTE

- **for 변수 in 순회형:** 구문에서 ***변수***는 ***순회형***에 들어있는 객체의 실제 값(value)을 가져온다.
- ***변수***가 인덱스 값을 가져오는 것이 아님을 명심하자.

range() 클래스

range()는 **for**문에서 흔히 사용하는 클래스다.

특히, 특정 횟수만큼 **for**문을 반복해서 실행할 때 매우 유용하게 쓰인다.

range()는 다음과 같은 경우에 많이 사용한다.

- 정수를 담은 리스트 또는 튜플을 생성할 때
- **for** 순환문에서 순회할 횟수를 정해야 할 때

range()는 지정한 범위 안에서 정수를 생성할 수 있는 순회자(iterator)를 반환한다.

따라해보기

In [ ]:

```
range(10)       # 10개의 정수를 담은 순회자(iterator)를 반환한다.
```

따라서 **range()**의 전달인자는 반드시 정수여야 하며,

전달인자 개수에 따라 처리하는 정수의 범위도 달라 진다.

range(*끝번호*)

- 하나의 정수를 지정하면, **0**부터 ***끝번호* - 1**까지의 정수를 담은 순회자를 반환한다.
- 즉, 0, 1, 2, 3, ..., ***끝번호* - 1**을 반환한다.

따라해보기

In [ ]:

```
# 순회자를 반환하기 때문에 생성한 정수를 사용하려면 형변환을 해야 한다.
list(range(10))  # 0부터 10 - 1까지
```

range(*시작번호*, *끝번호*)

- 두 개의 정수를 지정하면, ***시작번호***부터 ***끝번호* - 1**까지의 정수를 담은 순회자를 반환한다.
- 즉, ***시작번호***, ***시작번호* + 1**, ..., ***끝번호* - 1**을 반환한다.

따라해보기

In [ ]:

```
list(range(1, 10))  # 1부터 10 - 1 까지
```

range(*시작번호*, *끝번호*, *폭*)

- 세 개의 정수를 지정하면, ***시작번호***부터 ***끝번호* - 1**까지 ***폭***만큼 간격으로 생성한 정수를 담은 순회자를 반환한다.
- 즉, ***시작번호***, ***시작번호* + *폭***, ..., ***끝번호* - 1**을 반환한다.
  - 이때 ***폭***의 크기에 따라 ***끝번호* - 1**은 포함하지 않을 수도 있다.

따라해보기

In [ ]:

```
list(range(1, 10, 2))
```

In [ ]:

```
list(range(9, 0, -1))
```

In [ ]:

```
tuple(range(15, -21, -5))
```

for문에서 순회형 값을 처리할 때 **range()**를 사용하면 코드가 훨씬 간결해진다.

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

In [ ]:

```
for i in range(5):
    print('안녕 파이썬')
```

In [ ]:

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

for-else문

다음 코드는 **for-else**문을 사용한 예다.

따라해보기

for문을 이용해 1부터 10까지 숫자를 더한 후 그 결과를 출력해보자.

In [ ]:

```
total = 0
for i in range(1, 11): # 1부터 10까지 정수를 생성해서 차례로 i에 할당한다.
    total += i         # i를 차례로 합산한다.
else:                  # 결과를 출력한다.
    print('1부터 10까지의 합은 {}입니다.'.format(total))
```

In [ ]:

```
# --- while문
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
for item in integers:  # 리스트의 모든 객체를 처음부터 끝까지 하나씩 꺼내 item에 할당한다.
    total += item
else:                  # 결과를 출력한다.
    print('튜플에 들어있는 숫자의 합은 {}입니다.'.format(total))
```

In [ ]:

```
# --- while문
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:
    print('튜플에 들어있는 숫자의 합은 {}입니다.'.format(total))
```

enumerate() 함수

enumerate() 함수를 사용하는 형식은 다음과 같다.

enumerate(순회형, start=0)

다음과 같은 특징이 있다.

- ***순회형*** 자료를 전달인자로 받아, ***순회형***이 담은 개별 객체와 각 객체의 순서(인덱스 번호)를 **(순번, 객체)** 형식의 쌍으로 순회할 수 있는 열거형(enumerate) 객체를 반환한다.
- ***start***는 첫 번째 객체의 **시작 번호**다.
  - 선택해서 사용할 수 있는 ***start***의 기본값은 **0**이지만 다른 값으로 대체할 수 있다.

따라해보기

In [ ]:

```
L = ['a', 'b', 'c']
for i, k in enumerate(L):
    print(i, k)
```

In [ ]:

```
items = tuple('abcdefg')
print(items)
```

In [ ]:

```
x = enumerate(items)  # 열거형 객체(순회자)를 반환한다.
type(x)
```

In [ ]:

```
# --- items = 'a', 'b', 'c', 'd', 'e', 'f', 'g'
for i in enumerate(items):  # 시작 번호는 기본값인 0으로 한다.
    print(i)                # (인덱스, 객체)의 튜플 쌍으로 순화한다.
```

In [ ]:

```
# --- items = 'a', 'b', 'c', 'd', 'e', 'f', 'g'
for index, item in enumerate(items):  # 튜플 언패킹
    print(index, ':', item)
```

In [ ]:

```
# --- items = 'a', 'b', 'c', 'd', 'e', 'f', 'g'
for index, item in enumerate(items, start=11):  # enumerate(items, 11)
    print(index, item)
```

In [ ]:

```
# --- items = 'a', 'b', 'c', 'd', 'e', 'f', 'g'
L = list(enumerate(items, 101))
print(L)
```

In [ ]:

```
# --- items = 'a', 'b', 'c', 'd', 'e', 'f', 'g'
d = dict(enumerate(items, 101))
print(d)
```

for문과 튜플대입(언패킹)

따라해보기

In [ ]:

```
display_stand = [
    ('사과', 21),
    ('바나나', 27),
    ('블루베리', 35),
    ('딸기', 29),
    ('포도', 38)
]
```

In [ ]:

```
# 과일 진열대의 개별 객체(튜플)를 출력한다.
for i in display_stand:
    print(i)
```

In [ ]:

```
# 튜플 언패킹해서 원하는 형식으로 출력한다.
for fruit, number in display_stand:
    print('진열대 번호:', number, '==> 과일명:', fruit)
```

In [ ]:

```
# for문과 if문
k = [('a', 1), ('b', 2), ('c', 3), ('d', 4), ('e', 5)]

for x, y in k:
    if y % 2 == 1:
        print(x)    # 번호가 홀수인 튜플의 첫 번째 객체를 출력한다.
```

for문과 딕셔너리 순회

따라해보기

In [ ]:

```
koreng = {
    '사과': 'apple',
    '블루베리': 'blueberry',
    '딸기': 'strawberry',
    '키위':'kiwi',
    '바나나': 'banana',
    '포도': 'grape',
    '자두': 'plum'
}
```

In [ ]:

```
for i in koreng:
    print(i)
```

In [ ]:

```
for i in koreng.items():
    print(i)
```

In [ ]:

```
for k, v in koreng.items():
    print(k, ':', v)
```