

8. for 반복문


1. for 반복문의 기본 형태
2. 문자열과 for 반복문
3. 리스트와 for 반복문
4. 튜플과 for 반복문
5. 집합과 for 반복문
6. 사전과 for 반복문
7. range() 함수
8. for 반복문 내에서 break 사용하기
9. for 반복문에서 continue 사용하기
10. for 반복문에서 else 구문 사용하기
11. 중첩된 for 반복문
12. reversed() 함수에 대한 for 반복문
13. 정리

1. for 반복문의 기본 형태

◆ iterable 자료형

- for 반복문을 공부할 때 아주 중요함
- 컨테이너 자료형은 모두 iterable 자료형. (문자열, 리스트, 튜플, 집합, 사전)
- 내장 함수들 중에서 iterable 객체가 반환되는 함수들이 for 반복문에 유용하게 사용됨 (range(), reversed(), enumerate(), filter, map(), zip())

iterable 자료형에는 for 반복문이 아주 유용해요.



① 컨테이너 자료형

- 문자열 'hello world'
- 리스트 [80, 90, 70, 85]
- 튜플 ('red', 'blue', 'green')
- 집합 {7, 5, 13, 29, 113}
- 사전 {'Korea' : Seoul, 'Japan' : 'Tokyo', 'France' : Paris}

② iterable 객체를 반환하는 내장함수들

1. for 반복문의 기본 형태

컨테이너 자료형 : 문자열, 리스트, 튜플, 집합, 사전

iterable 객체 반환 함수 : range(), enumerate(), filter(), map(), reversed(), zip()

for 변수 in iterable 데이터 :

← 반드시 콜론으로 끝나야 합니다.



for 블록이라고 부릅니다. iterable 데이터에 포함된 원소가 하나씩 자동으로 차례대로 변수에 저장되어 for 블록을 수행합니다.

else:



else 블록이라고 부릅니다. else 블록은 있을 수도 있고 없는 경우도 있습니다.

2. 문자열과 for 반복문

■ 문자열과 while 반복문

```
string = 'star'
```

```
i = 0
```

```
while i < len(string):
```

```
    print(string[i])
```

```
    i += 1
```

[결과]

s

t

a

r

■ 문자열과 for 반복문

```
string = 'star'
```

```
for ch in string
```

```
    print(ch)
```

for 블록



```
for ch in
```

변수

s	t	a	r
---	---	---	---

첫 번째 루프에서 s가 ch에 저장되어 for 블록 수행함.
두 번째 루프에서 t가 ch에 저장되어 for 블록 수행함.
세 번째 루프에서 a가 ch에 저장되어 for 블록 수행함.
네 번째 루프에서 r가 ch에 저장되어 for 블록 수행함.

2. 문자열과 for 반복문

◆ 예제들

CODE 34 문자열 'AliCe'를 모두 대문자로 바꾸어 한 줄에 한 문자씩 출력하는 코드

```
name = 'AliCe'
for x in name:    # 문자열 name에서 문자 한 개씩 x에 넣고 for 블록 수행
    if x.isupper():    # if x.isupper()==True: 와 같습니다.
        print(x)
    else:            # x가 대문자가 아니라면, 대문자로 변환해서 출력합니다.
        print(x.upper())
```

[결과]

A
L
I
C
E

이 문제는 간단히 문자열의 upper() 메소드로 해결할 수 있음.

2. 문자열과 for 반복문

◆ 예제들

CODE 35 문자열을 입력받아서 자음과 모음의 개수 구하는 코드

```
string = input('Enter string : ') # 문자열을 입력받습니다.
vowel_count = 0 # 모음의 개수
con_count = 0 # 자음의 개수
for s in string: # 입력받은 string의 문자가 하나씩 순서대로 s에 저장됩니다.
    if s.lower() in 'aeiou': # s를 소문자로 바꾸었을 때 'aeiou' 중에 하나인 경우.
        vowel_count += 1
    else: # s가 자음인 경우.
        con_count += 1
print('number of vowels : {}'.format(vowel_count)) # 결과 출력
print('number of consonants : {}'.format(con_count))
```

[결과 1]

```
Enter string : proGRAMMING
number of vowels : 3
number of consonants : 8
```

[결과 2]

```
Enter string : mississippi
number of vowels : 4
number of consonants : 7
```

2. 문자열과 for 반복문

◆ 예제들

CODE 36 문자열을 입력받아서 대문자는 소문자로, 소문자는 대문자로 바꾼 새로운 문자열 만드는 코드

```
string = input('Enter string : ')    # 문자열을 입력받습니다.
new_string = ''                     # 빈 문자열을 만들어 놓습니다.
for ch in string:                   # 문자열 string에서 한 문자씩 ch에 저장하고 루프를 수행합니다.
    if ch.islower():                # ch가 소문자인 경우
        new_string += ch.upper()   # 대문자로 변환하여 new_string에 추가합니다.
    else:                           # ch가 대문자인 경우
        new_string += ch.lower()   # 소문자로 변환하여 new_string에 추가합니다.
print(new_string)
```

[결과 1]

```
Enter string : heLLo wORld
HElLO WorLD
```

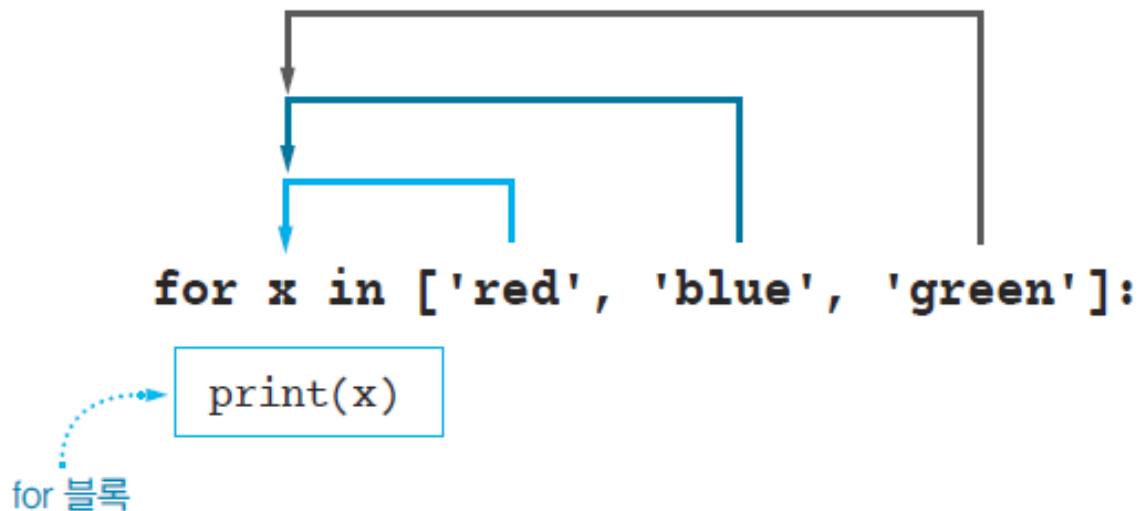
[결과 2]

```
Enter string : pyTHoN prograMMIng
PYThOn PROGRAMmiNG
```

3. 리스트와 for 반복문

◆ 문자열에 대한 for 반복문과 같음

<pre>for x in ['red', 'blue', 'white'] : print(x)</pre>	<p>[결과]</p> <p>red blue white</p>
---	---



첫 번째 루프에서 red가 x에 저장되어 for 블록 수행함.
두 번째 루프에서 blue가 x에 저장되어 for 블록 수행함.
세 번째 루프에서 green가 x에 저장되어 for 블록 수행함.

3. 리스트와 for 반복문

CODE 37 리스트 friends에 대해 for 반복문 수행하는 코드

```
friends = ['Paul', 'David', 'Alice', 'Jenny']  
for friend in friends:  
    print('Welcome! {}'.format(friend))
```

[결과]

```
Welcome! Paul.  
Welcome! David.  
Welcome! Alice.  
Welcome! Jenny.
```

CODE 38 리스트 books에 대해 각 단어의 첫 문자만을 대문자로 바꾸어 출력하는 코드

```
books = ['alice in wonderland',  
        'peter pan',  
        'snow white',  
        'sleeping beauty',  
        'goldilocks and the three bears']  
for book in books:  
    print(book.title())
```

[결과]

```
Alice In Wonderland  
Peter Pan  
Snow White  
Sleeping Beauty  
Goldilocks And The Three Bears
```

3. 리스트와 for 반복문

CODE 39 리스트 words에서 가장 긴 단어를 찾는 코드

```
words = ['hello', 'computer', 'bookshelf', 'chair', 'bicycle']
longest = len(words[0])    # longest - 첫 번째 단어('hello')의 길이를 갖습니다.
l_word = words[0]         # l_word - 첫 번째 단어('hello')를 갖습니다.

for w in words[1:]:       # words의 두 번째 단어부터 끝까지 차례로 w가 됩니다.
    if len(w) > longest:   # w의 길이가 longest보다 더 길다면,
        l_word = w        # w를 l_word로 대체합니다.
        longest = len(w)  # w의 길이가 longest가 됩니다.
print('longest word : {}'.format(l_word))
print('length : {}'.format(longest))
```

[결과]

```
longest word : bookshelf
length : 9
```

3. 리스트와 for 반복문

CODE 40 리스트에서 짝수의 합 구하는 코드

```
numbers = [4, 10, 9, -5, 5, 1, 8, 2, -2]
total = 0
for n in numbers:    # numbers의 원소들을 차례로 n에 저장합니다.
    if n%2 == 0:      # n이 짝수라면 total에 더해 줍니다.
        total += n
print('total : ', total)
```

[결과]
total : 22

3. 리스트와 for 반복문

CODE 41 리스트 원소에 번호 붙여 출력하는 코드

```
print('What is the capital of Korea ? ')
cities = ['Berlin', 'Seoul', 'Tokyo', 'Paris']
no = 1                                # 일련 번호를 붙이기 위한 변수입니다.
for city in cities:                   # 리스트 cities에 대해서 for 반복문을 수행합니다.
    print('{} . {}'.format(no, city))    # '번호. 도시'의 형태로 출력합니다.
    no += 1                            # 일련 번호를 1 증가시킵니다.
answer = int(input('Choose an answer : ')) # 답을 입력받습니다.
if answer == 2: print('Correct!')       # 2번이 답입니다.
else: print('Incorrect!')
```

What is the capital of Korea ?

1. Berlin

2. Seoul

3. Tokyo

4. Paris

Choose an answer : 1

Incorrect!

What is the capital of Korea ?

1. Berlin

2. Seoul

3. Tokyo

4. Paris


Choose an answer : 2

Correct!

4. 튜플과 for 반복문

- ◆ 튜플에도 문자열, 리스트와 같은 방법으로 for 반복문 적용함

<pre>T = (100, 200, 300) for x in T: print(x)</pre>	<p>[결과]</p> <pre>100 200 300</pre>
---	------------------------------------



```
for x in (100, 200, 300):
```

```
    print(x)
```

for 블록

첫 번째 루프에서 100 이 x에 저장되어 for 블록 수행함.
두 번째 루프에서 200 이 x에 저장되어 for 블록 수행함.
세 번째 루프에서 300 이 x에 저장되어 for 블록 수행함.

5. 집합과 for 반복문

- ◆ 집합은 시퀀스 자료형이 아님, 인덱스 개념이 없음
- ◆ 인덱스 개념이 없어서 while 반복문 적용은 어렵지만, for 반복문은 쉽게 적용할 수 있음

집합은 데이터가 주머니에 들어 있다고 생각하세요.

어느 순서로 변수에 저장될 지 알 수 없어요

for 변수 in



5. 집합과 for 반복문

◆ 집합은 순서 개념이 없음

```
sports = {'baseball', 'soccer', 'tennis', 'basketball'}  
for sport in sports:  
    print('{} is my favorite sports.'.format(sport))
```

[결과 1]

basketball is my favorite sports.
soccer is my favorite sports.
baseball is my favorite sports.
tennis is my favorite sports.

[결과 2]

basketball is my favorite sports.
baseball is my favorite sports.
soccer is my favorite sports.
tennis is my favorite sports.

[결과 3]

soccer is my favorite sports.
baseball is my favorite sports.
tennis is my favorite sports.
basketball is my favorite sports.

[결과 4]

tennis is my favorite sports.
baseball is my favorite sports.
soccer is my favorite sports.
basketball is my favorite sports.

5. 집합과 for 반복문

CODE 42 집합 A와 B의 교집합 찾는 코드

```
A = {'apple', 'orange', 'strawberry', 'watermelon'}  
B = {'strawberry', 'watermelon', 'kiwi'}  
for fruit in A:           # 집합 A에 있는 원소를 하나씩 변수 fruit에 넣습니다.  
    if fruit in B:        # 만약에 fruit이 집합 B에도 있는지 확인합니다.  
        print('We both like {}'.format(fruit))
```

[결과 1]

We both like strawberry.
We both like watermelon.

[결과 2]

We both like watermelon.
We both like strawberry.

6. 사전과 for 반복문

◆ 사전은 '키:값'의 형태로 데이터가 저장됨

```
info = {'name': 'Alice',  
        'age': 20,  
        'phone': '010-111-1111'}
```

```
for x in info:  
    print(x)
```

[결과]

```
name  
age  
phone
```

키만 변수에 차례대로 저장됩니다.

for 변수 in

```
'name' : 'Alice'  
'age' : 20  
'phone' : '010-111-1111'
```

7. range() 함수

- ◆ range() 함수는 for 반복문에서 사용하는 경우가 많음

인수	range() 함수	의미	예제
1개	range(a)	0부터 a-1까지의 정수	range(5) → 0, 1, 2, 3, 4
2개	range(a, b)	a부터 b-1까지의 정수	range(2, 7) → 2, 3, 4, 5, 6
3개	range(a, b, c)	a부터 b-1까지 c 간격의 정수	range(2, 10, 3) → 2, 5, 8

```
>>> a = range(5) # a는 range(5)의 반환값
>>> print(a)
range(0, 5)
>>> type(a)
<class 'range'>
```

```
>>> list(range(5)) # 리스트로 변환
[0, 1, 2, 3, 4]
>>> tuple(range(5)) # 튜플로 변환
(0, 1, 2, 3, 4)
>>> set(range(5)) # 집합으로 변환
{0, 1, 2, 3, 4}
```

7. range() 함수

◆ range() 예

>>> list(range(3))	# 0부터 2까지의 정수
[0, 1, 2]	
>>> list(range(11))	# 0부터 10까지의 정수
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]	
>>> list(range(5, 10))	# 5부터 9까지의 정수
[5, 6, 7, 8, 9]	
>>> list(range(2, 15, 3))	# 2부터 14까지 3칸 간격으로
[2, 5, 8, 11, 14]	
>>> list(range(10,5))	# 첫 번째 인수가 두 번째 인수보다 크면 [] 반환
[]	
>>> list(range(10, 5, -2))	# 첫 번째 인수가 두 번째 인수보다 클 때, # 세 번째 인수가 음수인 경우에는 결과가 있습니다.
[10, 8, 6]	

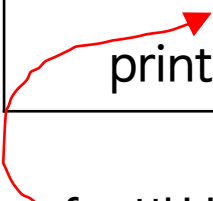
7. range() 함수

◆ range() 함수의 인수는 생략할 수 없음

```
>>> list(range(,10,2))      # list(range(0,10,2)) 이라고 해야 합니다.
SyntaxError: invalid syntax
>>> list(range(10,,-1))    # list(range(10,0,-1)) 이라고 해야 합니다.
SyntaxError: invalid syntax
```

◆ for 반복문에 range() 함수 사용하기

```
for a in range(5): # for a in list(range(5)):
    print(a, end=' ')
```



[결과]

0 1 2 3 4

for 반복문에는 range() 함수를 바로 넣어도 됨
(range() 함수의 반환값은 iterable 자료형임)

7. range() 함수

◆ range() 함수 사용 예제

[1부터 10까지의 합 구하기]

[while 이용하기]	[for 와 range() 함수 이용하기]
<pre>total = 0 a = 1 while a <= 10: total += a a += 1 print(total)</pre>	<pre>total = 0 for a in range(11): total += a print(total)</pre>

<pre>total = 0 for x in range(10, 16): total += x print('10부터 15까지의 합 :', total)</pre>	<p>[결과]</p> <p>10부터 15까지의 합 : 75</p>
<pre>total = 0 for x in range(0, 101, 3): total += x print('100이하의 3의 배수의 합 :', total)</pre>	<p>[결과]</p> <p>100이하의 3의 배수의 합 : 1683</p>

7. range() 함수

◆ sum(), range() 함수 같이 사용하기

```
print(sum(range(11)))  
print(sum(range(10,16)))  
print(sum(range(0, 101, 3)))
```

7. range() 함수

CODE 43 하나의 양의 정수를 입력받아서 그 수의 약수를 모두 출력하는 코드

```
n = int(input('Enter one integer : ')) # 하나의 정수 n을 입력받습니다.  
for a in range(1,n+1): # 변수 a는 1부터 n까지 1씩 증가하면서 루프를 수행  
    if n % a == 0: # n%a가 0이면 a는 n의 약수입니다.  
        print(a, end = ' ')
```

[결과 1]

Enter one integer : 20

1 2 4 5 10 20

[결과 2]

Enter one integer : 30

1 2 3 5 6 10 15 30

7. range() 함수

[문자열에 for 반복문 : 아래 두 코드는 결과가 같음]

```
name = 'python'
for n in name:
    print(n)
```

```
name = 'python'
for x in range(len(name)): # range(6)
    print(name[x])
```

[리스트에 for 반복문 : 아래 두 코드는 결과가 같음]

```
score = [80, 90, 88, 93, 75]
for s in score:
    print(s)
```

```
score = [80, 90, 88, 93, 75]
for i in range(len(score)): # range(5)
    print(score[i])
```


7. range() 함수

[리스트에 있는 성적을 모두 5점 올리는 코드]

<p>[코드 1]</p> <pre>score = [80, 90, 88, 93, 75] for s in score: s += 5 print(score)</pre>	<p>[코드 2]</p> <pre>score = [80, 90, 88, 93, 75] for i in range(len(score)): score[i] += 5 print(score)</pre>
<p>[결과 1]</p> <p>[80, 90, 88, 93, 75]</p>	<p>[결과 2]</p> <p>[85, 95, 93, 98, 80]</p>

왼쪽 코드의 문제점?

8. for 반복문 내에서 break 사용하기

- ◆ for 반복문에서도 break는 루프를 끝내는 역할을 함

```
for x in [2, 9, -3, 8, 0, 10, -2, 1]:
```

```
    if x == 0:      # x에 0이 들어오면 for 반복문을 끝냅니다.
```

```
        break
```

```
    print(x)
```

[결과]

2

9

-3

8

9. for 반복문 내에서 continue 사용하기

- ◆ for 반복문에서도 continue는 while에서와 같음

```
for x in [5, 7, 0, 8, 0, 1]:  
    if x == 0: continue  
    print('x : {}'.format(x))    # x == 0이면 수행되지 않습니다.
```

[결과]

x:5
x:7
x:8
x:1

```
for x in [5, 7, 0, 8, 0, 1] :
```

```
    if x == 0 : continue
```

```
    print( ... )
```

continue를 만나면 다음 데이터를
x에 넣고 for 반복문을 수행합니다.

10. for 반복문 내에서 else 구문 사용하기

◆ for 반복문에서도 else 블록은 while에서와 같음

- for 반복문이 끝나는 두 가지 경우

- ① 모든 데이터에 대해서 for 반복문이 수행된 경우에는 else 블록이 있으면 수행됩니다.
- ② for 블록 안에서 break를 만나서 for 반복문이 끝나는 경우, else 블록이 있더라도 수행되지 않습니다.

10. for 반복문 내에서 else 구문 사용하기

- ◆ for 반복문에서도 else 블록은 while에서와 같음

<p>[코드 1]</p> <pre>for x in [5, 7, 3, 8, 2, 1]: if x == 0: # break 걸리지 않음 break print('x : {}'.format(x)) else: print('this is else block')</pre>	<p>[코드 2]</p> <pre>for x in [5, 7, 0, 8, 0, 1]: if x == 0: # 여기에서 루프가 끝남 break print('x : {}'.format(x)) else: print('this is else block')</pre>
<p>[결과 1]</p> <pre>x: 5 x: 7 x: 3 x: 8 x: 2 x: 1 this is else block</pre>	<p>[결과 2]</p> <pre>x: 5 x: 7</pre>

11. 중첩된 for 반복문

- ◆ for 반복문 안에 다시 for 반복문이 중첩될 수도 있음
 - 이중 루프를 작성할 때 for 안에 while, 또는 while 안에 for도 가능함

```
for y in range(1, 11):  
    for x in range(1, 11):  
        print('{:4d}'.format(y*x), end='')  
    print()
```

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

11. 중첩된 for 반복문

CODE 44 for 이중 루프를 이용해서 두 집합의 교집합 구하는 코드
(CODE42와 같은 문제)

```
A = {'apple', 'orange', 'strawberry', 'watermelon', 'tomato'}
B = {'strawberry', 'tomato', 'kiwi'}
for fruit1 in A:      # 집합 A에 대해서 for 루프를 수행합니다.
    for fruit2 in B:  # 집합 A의 각 원소에 대해서 집합 B에 for 루프를 수행합니다.
        if fruit1 == fruit2:
            print('We both like {}'.format(fruit1))
```

[결과 1]

We both like strawberry.
We both like tomato.

[결과 2]

We both like tomato.
We both like strawberry.

11. 중첩된 for 반복문

CODE 45 for 이중 루프를 1부터 입력받은 수까지의 모든 약수를 출력하는 코드

```
n = int(input('Enter one integer : '))
for a in range(1, n+1):      # a는 1부터 n까지 변합니다.
    print('[{}] :'.format(a), end=' ')
    for b in range(1, a+1):  # b는 1부터 a까지 변합니다.
        if a % b == 0:
            print('{} '.format(b), end=' ')
    print()
```

[결과 1]

Enter one integer : 5

[1] : 1

[2] : 1 2

[3] : 1 3

[4] : 1 2 4

[5] : 1 5

[결과 2]

Enter one integer : 10

[1] : 1

[2] : 1 2

[3] : 1 3

[4] : 1 2 4

[5] : 1 5

[6] : 1 2 3 6

[7] : 1 7

[8] : 1 2 4 8

[9] : 1 3 9

[10] : 1 2 5 10

11. 중첩된 for 반복문

CODE 46 영문자와 숫자가 섞여 있는 문자열에서 영문자만 뽑아서 다시 문자열 구성하는 코드

<pre>L = ['heLLO123', '010-111-1234Alice', 'Y2018M10D11'] for word in L: # L에 있는 각 단어에 대해서 for 블록 수행함. new_word = '' # 빈 문자열 new_word가 필요합니다. for w in word: if w.isalpha(): # w가 영문자인지 판단합니다. new_word += w print(new_word)</pre>	<p>[결과]</p> <p>heLLO Alice YMD</p>
---	--

11. 중첩된 for 반복문

CODE 47 [CODE 46]의 코드를 while, for 섞어서 이중 루프로 작성한 코드

<p>while 안에 for 루프</p>	<pre> L = ['heLLO123', '010-111-1234Alice', 'Y2018M10D11'] i = 0 while i < len(L): # L[0], L[1], L[2], L[3] 차례로 루프를 수행합니다. new_word = '' # new_word에 문자만을 찾아서 계속 연결합니다. for w in L[i]: # 리스트 L에 저장된 각 단어에 대해서 for 루프를 수행합니다. if w.isalpha(): new_word += w print(new_word) i += 1 </pre>
<p>for 안에 while 루프</p>	<pre> L = ['heLLO123', '010-111-1234Alice', 'Y2018M10D11'] for word in L: # L에 저장된 단어들이 하나씩 word가 됩니다. new_word = '' i = 0 while i < len(word): if word[i].isalpha(): new_word += word[i] i += 1 print(new_word) </pre>

12. reversed() 함수에 대한 for 반복문

◆ iterable 객체를 반환하는 함수들

- enumerate(), filter(), map(), range(), reversed(), zip()
- 이런 함수들은 for 반복문에 바로 적용할 수 있음

◆ 문자열에 reversed() 함수 적용하기

```
>>> name = 'Alice'
>>> reversed(name) # 문자열에 reversed() 함수를 적용하면 reversed 객체를 반환함.
<reversed object at 0x01680B90>
>>> list(reversed(name)) # reversed() 결과에 list() 함수를 적용함.
['e', 'c', 'i', 'l', 'A']
>>> tuple(reversed(name)) # reversed() 결과에 tuple() 함수를 적용함.
('e', 'c', 'i', 'l', 'A')
```

```
name = 'Alice'
for c in reversed(name): # 바로 reversed(name)이라고 적습니다.
    print(c, end='')

```

[결과]

ecilA

12. reversed() 함수에 대한 for 반복문

◆ 리스트에 reversed() 함수 적용하기

```
>>> score = [80, 90, 93, 77]
>>> reversed(score)          # 리스트는 list_reverseiterator 객체를 반환함.
<list_reverseiterator object at 0x01B60B90>
>>> list(reversed(score))     # reversed() 결과에 list() 함수를 적용함.
[77, 93, 90, 80]
>>> tuple(reversed(score))    # reversed() 결과에 tuple() 함수를 적용함.
(77, 93, 90, 80)
```

```
score = [80, 90, 93, 77]
```

```
for s in reversed(score): # 바로 reversed(score)라고 적습니다.
    print(s, end=' ')
```

[결과]

77 93 90 80

12. reversed() 함수에 대한 for 반복문

◆ 튜플에 reversed() 함수 적용하기

```
>>> subjects = ('korean', 'english', 'math')
>>> reversed(subjects)      # 튜플은 문자열처럼 reversed 객체를 반환함.
<reversed object at 0x01B60B90>
>>> list(reversed(subjects)) # reversed() 결과에 list() 함수를 적용함.
['math', 'english', 'korean']
>>> tuple(reversed(subjects)) # reversed() 결과에 tuple() 함수를 적용함.
('math', 'english', 'korean')
```

```
subjects = ('korean', 'english', 'math')
for sub in reversed(subjects):
    print(sub, end=' ')
```

[결과]

math english korean

13. 정리

- ◆ 파이썬의 for 반복문에 대해서 학습하였음.
- ◆ For 반복문에는 iterable 자료형이 중요함.
- ◆ 컨테이너 자료형과 iterable 객체를 반환하는 함수를 for 반복문에 사용할 수 있음.
- ◆ for 반복문에서도 break, continue, else 구문을 사용할 수 있고, 기본적으로 while 반복문에서와 의미가 같음.
- ◆ 중첩된 for 반복문을 사용할 수 있음.