# Python

## 반복문

Spring 2025


AI융합학과
Seongbok Baik
sbbaik@dju.kr

# 00 Text Book

| 교재명 | 으뜸 파이썬 |
|---|---|
| 저자 | 박동규, 강영민 |
| 출판사 | 생능출판사 |
| 발행년 | 2024.06.14 |

# 학습목표

- for 반복문을 정의하고 사용하는 방법을 익힌다.
- for in 구문과 리스트에 대해 이해하고 활용할 수 있다.
- 이중 for 루프에 대해 알아보고 활용한다.
- while 반복문을 정의하고 사용하는 방법을 익힌다.
- for 문을 이용하여 작성한 반복문을 while 문으로 변경할 수 있다.
- break와 continue를 이용하여 반복문을 제어하는 방법을 익힌다.

# 4.1 for 반복문

- 특정한 작업을 여러 번 되풀이해서 수행하고 싶을 경우 사용
- 반복문에는 for 문과 while 문 두 종류가 있음
- for 문은 반복의 횟수가 미리 정해져 있는 경우, while 문은 반복 횟수는 알지 못하지만 반복하는 조건이 명확한 경우에 사용

# 4.1 for 반복문

- 반복문을 사용하지 않고 5번 반복해 출력하려면 다음과 같이 print() 함수를 5번에 걸쳐 사용
- 1000회 반복 시 매우 비효율적임

코드 4-1: print() 함수의 호출을 통한 반복적 수행

print_welcome.py

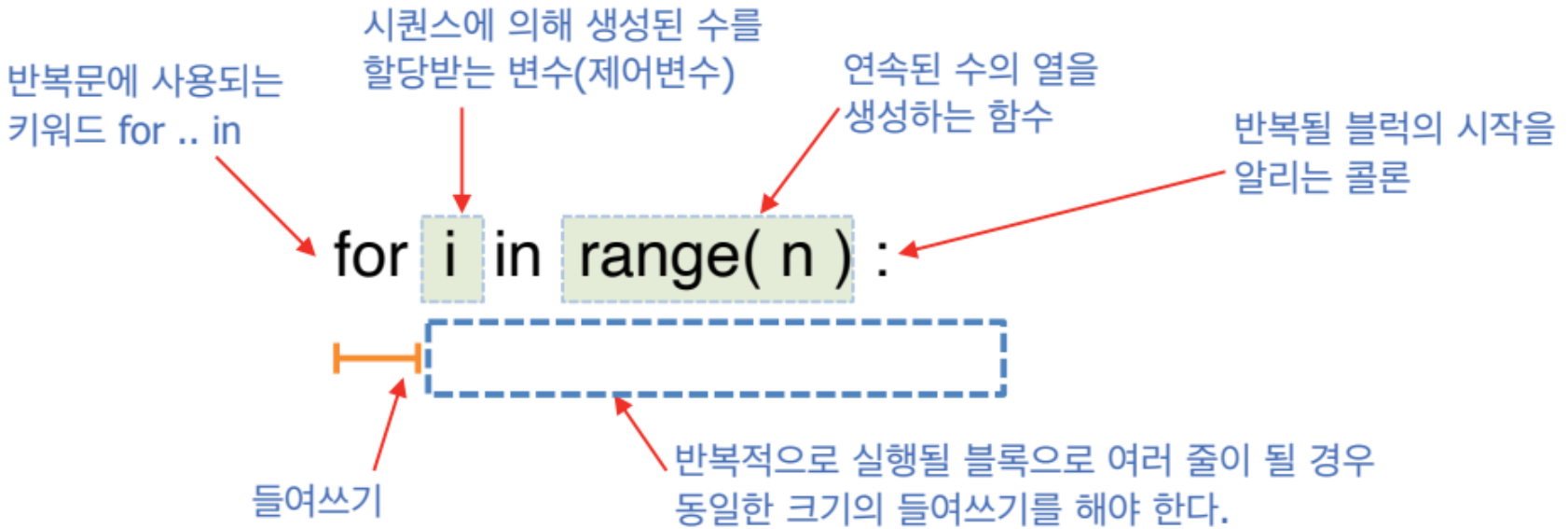
```
print('Welcome to everyone!!')
print('Welcome to everyone!!')
print('Welcome to everyone!!')
print('Welcome to everyone!!')
print('Welcome to everyone!!')
```

실행결과

```
Welcome to everyone!!
Welcome to everyone!!
Welcome to everyone!!
Welcome to everyone!!
Welcome to everyone!!
```

# 4.1 for 반복문

- range() 함수는 특정한 구간의 정수 **열**sequence을 반복해서 생성함
- for 문에서 순환을 위한 용도이다




[그림 4-1] for in range() 구문의 사용법

# 4.1 for 반복문

코드 4-2: for 문을 이용한 반복적 수행

print_welcome_with_for1.py

```
for i in range(5):
    print('Welcome to everyone!!')
```

실행결과

```
Welcome to everyone!!
Welcome to everyone!!
Welcome to everyone!!
Welcome to everyone!!
Welcome to everyone!!
```

10번 반복 시 range() 괄호 내의 값만 10으로 고쳐주면 됨

```
for i in range(10):
    print('Welcome to everyone!!')
```

# 4.1 for 반복문

### NOTE: 루프 제어변수의 익명화

위 반복문에서 새롭게 할당되는 변수 i는 실행문에서 사용되지 않는 루프변수이므로 다음과 같이 언더스코어(_)를 대신 넣어서 익명화시킬 수 있다.

```
for _ in range(10):
    print('Welcome to everyone!!!!')
```

# 4.1 for 반복문

코드 4-3: for 문을 이용한 반복적 수행 - 7회 수행

print_welcome_with_for2.py

```
for i in range(7):
    print(i, 'Welcome to everyone!!')
```

실행결과

```
0 Welcome to everyone!!
1 Welcome to everyone!!
2 Welcome to everyone!!
3 Welcome to everyone!!
4 Welcome to everyone!!
5 Welcome to everyone!!
6 Welcome to everyone!!
```

- 반복문에서 사용되는 변수는 i, j, k, l,... 와 같은 알파벳 문자를 할당
- 이러한 변수를 C나 Java에서는 **루프**loop **제어변수**라고 함

# 4.1 for 반복문

**LAB 4-1: 반복문을 이용해서 다음 코드를 작성해 보자.**

1. for 반복문에서 언더스코어 루프 제어변수를 사용하여 다음과 같이 Hello, Python!을 5번 출력하는 프로그램을 작성해 보자.

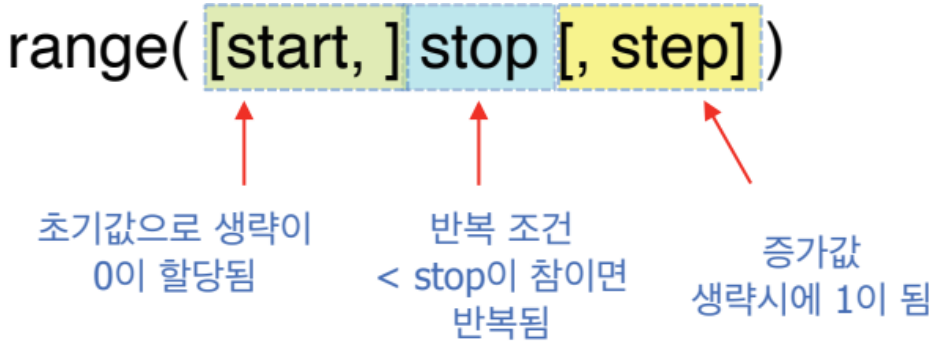
```
Hello, Python!
Hello, Python!
Hello, Python!
Hello, Python!
Hello, Python!
```

2. i라는 루프 제어변수를 사용해서 0에서 4까지의 정수를 출력하는 프로그램을 작성하시오.

```
0
1
2
3
4
```

## range() 함수의 사용법

- range(0, 5)와 같이 주어진 시작 값에서 마지막 값 사이의 연속적인 정수들을 생성할 수도 있으며, range(0, 5, 2)와 같이 마지막에 증가치 값을 넣어 줄 수도 있다.
- range(0, 5, 1)과 같이 호출할 경우 마지막의 1은 디폴트 간격(step) 값으로 1씩 더하면서 값을 변경하라는 의미




[그림 4-2] range() 함수의 사용법: [start]와 [step] 값은 생략할 수 있다

# 4.1 for 반복문

### NOTE: range() 함수의 인자

range() 함수는 실수 값을 인자로 가질 수 없다. 반면, 나중에 배우게 될 넘파이 모듈의 arange()라는 함수는 실수의 시작 값, 종료 값, 스텝 값을 가질 수 있다.

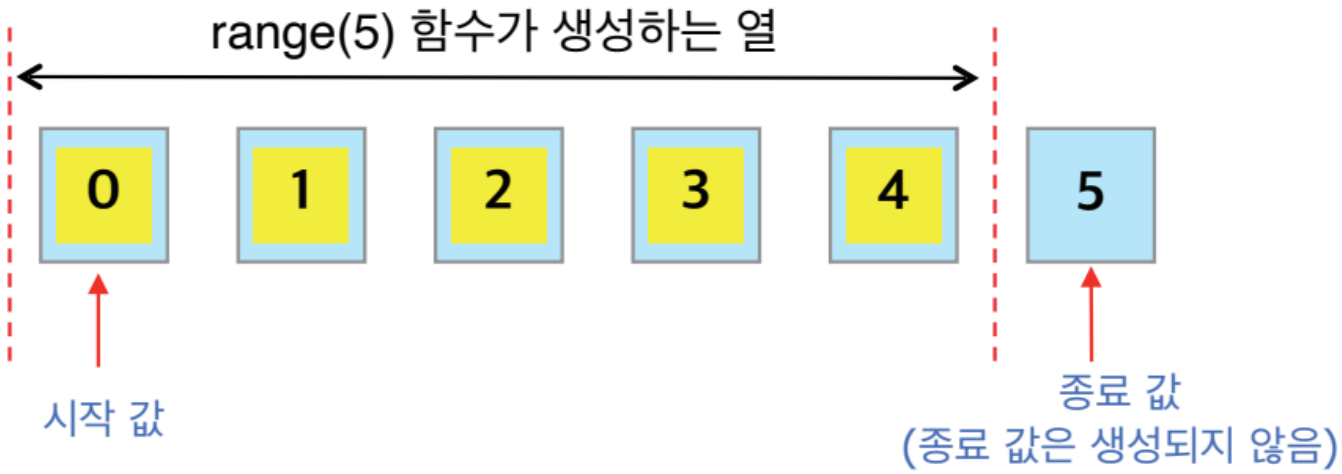
# 4.1 for 반복문

### 대화창 실습: range() 함수의 활용과 리스트

```
>>> list(range(5))                # 0에서 4 사이의 정수열을 생성
    [0, 1, 2, 3, 4]
>>> list(range(0, 5))             # list(range(5))와 동일한 결과
    [0, 1, 2, 3, 4]
>>> list(range(0, 5, 1))          # list(range(0, 5))와 동일한 결과
    [0, 1, 2, 3, 4]
>>> list(range(0, 5, 2))          # 생성하는 값을 2씩 증가시킴
    [0, 2, 4]
>>> list(range(2, 5))             # 2에서 5-1까지의 연속된 수 2, 3, 4를 생성
    [2, 3, 4]
>>> list(range(0, 10, 2))         # 0에서 9 사이의 짝수 리스트 생성
    [0, 2, 4, 6, 8]
>>> list(range(1, 10, 2))         # 0에서 9 사이의 홀수 리스트 생성
    [1, 3, 5, 7, 9]
>>> list(range(-2, -10, -2))      # 음수 간격 값을 이용하여 -2, -4, -6, -8을 생성
    [-2, -4, -6, -8]
```

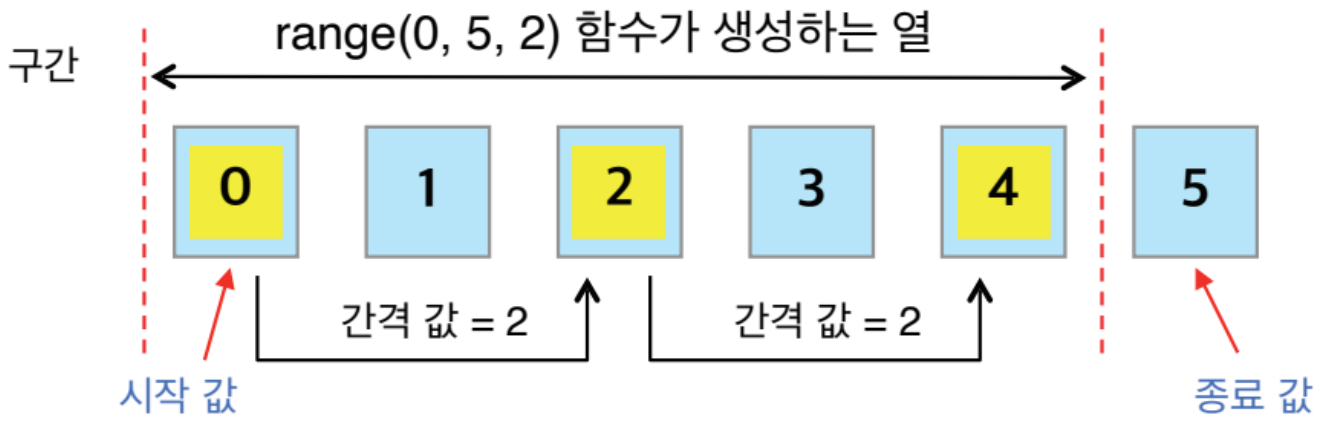
# 4.1 for 반복문

- range(5)의 시작 값과 종료 값, 그리고 반환되는 열



# 4.1 for 반복문

- range(2, 5)는 2 이상 5 미만의 정수열 [2, 3, 4]를 생성하며, range(0, 5, 2)는 0에서 5 미만의 정수들을 생성할 때 2씩 증가시키므로 [0, 2, 4]의 정수를 생성
- 양수 간격 값을 사용할 때는 range(0, 5, 1)과 같이 반드시 시작 값이 종료 값보다 작아야 하며, 음수 간격 값을 사용할 때는 range(-2, -10, -2)와 같이 시작 값이 반드시 종료 값보다 커야 한다.



# 4.1 for 반복문

## LAB 4-2: range() 함수의 응용

1. range() 함수와 list() 함수를 사용하여 1 이상 100 이하의 자연수 리스트를 다음과 같이 만드시오.

```
[1, 2, 3, 4, 5, (생략), 100]
```

2. 1 이상 100 이하의 짝수 리스트를 만드시오.

3. 1 이상 100 이하의 홀수 리스트를 만드시오.

4. -100보다 크고 0보다 작은 음수 리스트를 만드시오.

# 4.1 for 반복문

코드 4-4: range() 함수를 이용한 for 문의 제어

for_in_range_test1.py

```
# for in range 표현식
for i in range(5):
    print(i)
```

실행결과

```
0
1
2
3
4
```

- print(i)에서 수를 출력한 후 매번 줄 바꿈을 하므로 결과를 보는 것이 불편함
- 아래와 같은 방법을 사용하여 디폴트 문자를 공백문자로 변경

```
print(i, end = ' ')
```

# 4.1 for 반복문

**코드 4-5: for 문의 제어와 print() 함수의 end 키워드 인자 사용방법**

for_in_range_test2.py

```
# for in range 표현식 1
for i in range(5):
    print(i, end = ' ')
# for in range 표현식 2
for i in range(0, 5):
    print(i, end = ' ')
```

실행결과

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

# 4.1 for 반복문

코드 4-6: for 문의 제어와 print() 함수의 end 인자 사용 방법

for_in_range_test3.py

```
# for in range 표현식 1
for i in range(5):
    print(i, end = ' ')
print()
# for in range 표현식 2
for i in range(0, 5):
    print(i, end = ' ')
print()
```

실행결과

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

# 4.1 for 반복문

**코드 4-7: range() 함수를 이용한 for 문의 제어와 간격값 사용법**

for_in_range_test4.py

```
# 표현식 1: 2에서 5-1까지 연속값 2, 3, 4 출력
for i in range(2, 5):
    print(i, end = ' ')
print()
# 표현식 2: 간격 값을 사용하여 0, 2, 4, 6, 8 출력
for i in range(0, 10, 2):
    print(i, end = ' ')
print()
# 표현식 3: 음수 간격 값 사용, -2, -4, -6, -8 출력
for i in range(-2, -10, -2):
    print(i, end = ' ')
print()
```

실행결과

```
2 3 4
0 2 4 6 8
-2 -4 -6 -8
```

## 4.1.1 반복문의 활용

- 1에서 10까지의 정수의 합 구하기

코드 4-8: 연속적인 값의 생성과 누적 덧셈

for_sum_ex1.py

```
s = 0
for i in range(1, 11):
    s = s + i
print('1에서 10까지의 합:', s)
```

실행결과

```
1에서 10까지의 합: 55
```

# 4.1 for 반복문

**주의 - sum() 함수와 이름의 중복 사용**

위의 코드에서 sum이라는 변수를 사용하게 되면 파이썬 내장함수 sum()과 헷갈릴 수 있다. 동일한 모듈 내에 sum = 0이라는 변수와 sum(numbers)라는 함수를 호출하게 되면 오류가 발생한다. 따라서 sum() 함수와 중복되지 않도록 변수의 이름을 s나 total로 사용할 것을 권장한다.

# 4.1 for 반복문

**NOTE: 반복문에서 초기화의 중요성, 누적 연산**

[코드 4-8]의 프로그램에서 for 문이 반복되면 다음과 같이 i 값과 s 값이 변하게 된다. i는 매 반복 루프가 돌 때마다 그 값이 변하며, 이 때문에 s 값도 증가하게 된다. i는 매번 1씩 증가하여 1, 2, 3, ...이 된다. 그리고 초기 상태의 s = 0이므로 첫 번째 반복문 수행 시 0+1이 s에 할당되지만, 매번 s + i 연산을 수행하기 때문에 다음과 같이 0+1, 0+1+2, 0+1+2+3, ... 연산을 수행하게 된다. 마침내 10번째 반복이 수행되면 최종적으로 s는 55 값을 가지며, 반복은 중단된다.

| 반복 | i 값 | s 값 | 반복 여부 | s의 계산 과정 |
|---|---|---|---|---|
| 1번째 | 1 | 1 | 반복 | 0+1 |
| 2번째 | 2 | 3 | 반복 | 0+1+2 |
| 3번째 | 3 | 6 | 반복 | 0+1+2+3 |
| 4번째 | 4 | 10 | 반복 | 0+1+2+3+4 |
| 5번째 | 5 | 15 | 반복 | 0+1+2+3+4+5 |
| 6번째 | 6 | 21 | 반복 | 0+1+2+3+4+5+6 |
| 7번째 | 7 | 28 | 반복 | 0+1+2+3+4+5+6+7 |
| 8번째 | 8 | 36 | 반복 | 0+1+2+3+4+5+6+7+8 |
| 9번째 | 9 | 45 | 반복 | 0+1+2+3+4+5+6+7+8+9 |
| 10번째 | 10 | 55 | 반복 중단 | 0+1+2+3+4+5+6+7+8+9+10 |

이전 연산의 결과가 누적되어 더해지게 된다.

이전 연산의 결과가 누적되어 더해지게 된다.

이전 연산의 결과가 누적되어 더해지게 된다.

...

# 4.1 for 반복문

코드 4-9: 누적 덧셈의 중간 결과 출력하기

for_sum_ex2.py

```
s = 0
for i in range(1, 11):
    s = s + i
    print('i = {}, s = {}'.format(i, s) )
print('1에서 10까지의 합:', s)
```

실행결과

```
i = 1, s = 1
i = 2, s = 3
i = 3, s = 6
i = 4, s = 10
i = 5, s = 15
i = 6, s = 21
i = 7, s = 28
i = 8, s = 36
i = 9, s = 45
i = 10, s = 55
1에서 10까지의 합: 55
```

- 코드 for_sum_ex1.py에 비해
  전체 수행과정을 살펴볼 수 있어 이해하기 편하다.

# 4.1 for 반복문

## LAB 4-3: 누적 덧셈의 응용

1. 앞서 배운 누적 덧셈을 응용하여 1에서 100까지 정수의 합을 구하여 출력하여라(힌트: range()의 구간이 1에서 100까지가 되도록 하자).

2. range() 함수의 step 값을 이용하여 1에서 100까지 정수 중에서 짝수의 합을 구하여 출력하여라(힌트: range() 함수의 시작 값은 0으로 하고 step 값을 2로 하여라).

3. range() 함수의 step 값을 이용하여 1에서 100까지 정수 중에서 홀수의 합을 구하여 출력하여라(힌트: range() 함수의 시작 값은 1로 하고 step 값을 2로 하여라).

# 4.1 for 반복문

## 임의의 양의 정수를 입력받아 1부터 n까지의 정수 출력

**코드 4-10: 사용자로부터 입력을 받은 후 누적 합계 값 구하기**

for_sum_input1.py

```
n = int(input('합계를 구할 수를 입력하세요 : '))
s = 0
for i in range(0, n) :
    s = s + (i+1)
print('1부터 {}까지의 합은 {}'.format(n, s))
```

실행결과

```
합계를 구할 수를 입력하세요 : 100
1부터 100까지의 합은 5050
```

- s = s + (i+1) 은 매번 루프 실행때 마다 (i + 1)을 수행함, 코드도 난해함
- 여러 가지 방식 중에서 가장 **이해하기 쉽고** 다른 사람이 **읽기 좋은 코딩 방식**을 사용

# 4.1 for 반복문

임의의 양의 정수를 입력받아 1부터 n까지의 정수 출력

코드 4-11: 사용자로부터 입력을 받은 후 누적 합계 값 구하기

for_sum_input2.py

```
n = int(input('합계를 구할 수를 입력하세요 : '))
s = 0
for i in range(1, n+1) :
    s = s + i
print('1부터 {}까지의 합은 {}'.format(n, s))
```

실행결과

```
합계를 구할 수를 입력하세요 : 100
1부터 100까지의 합은 5050
```

- range() 함수의 시작 값을 1로 하고 마지막 값을 n+1로 하여 1부터 n까지의 정수를 더하기
- s = s + i 를 사용하여 이해하기 편리함

- 4.1.2 팩토리얼factorial 구하기
  - n이 임의의 자연수일 때 1에서 n까지의 모든 자연수를 곱한 값이다.

$$n! = \prod_{k=1}^{n} k = n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 3 \cdot 2 \cdot 1$$

## • 4.1.2 팩토리얼factorial 구하기

코드 4-12: for 반복문을 이용한 5 팩토리얼(5!) 계산

for_factorial.py

```
n = int(input('수를 입력하세요 : '))
fact = 1
for i in range(1, n+1):
    fact = fact * i
print('{}! = {}'.format(n, fact))
```

실행결과

```
수를 입력하세요 : 5
5! = 120
```

- 변수 fact의 초기 값은 1로 두고 덧셈(+) 연산 대신 곱셈(*) 연산을 for문 안에서 사용

# 4.1 for 반복문

## NOTE: 파이썬과 정수 표현의 한계

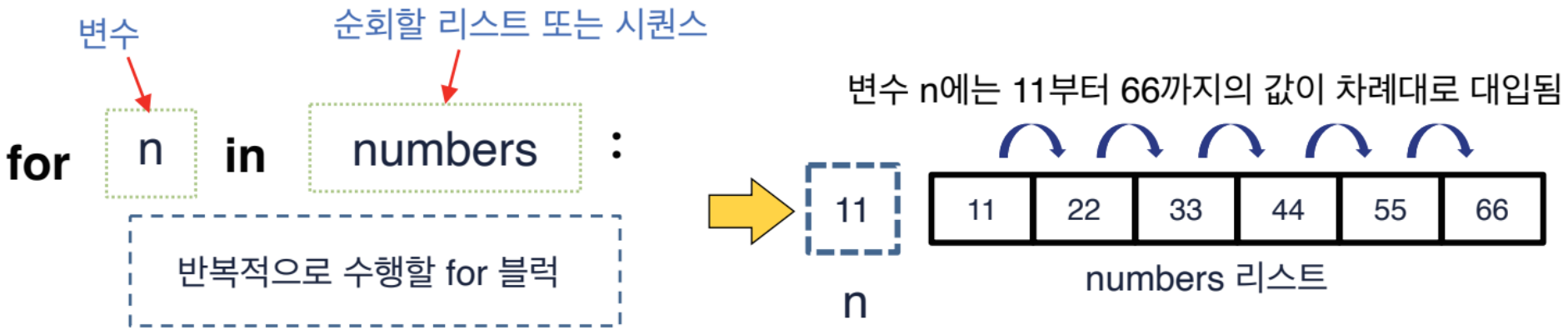
팩토리얼을 구하는 **[코드 4-12]**를 수정하여 n 값을 50으로 바꾼다면 다음과 같은 엄청나게 큰 수를 출력한다.

50! = 30414093201713378043612608166064768844377641568960512000000000000.

C나 Java와 같은 프로그래밍 언어는 정수의 크기가 4바이트 형으로 정해져 있어서 일정한 크기 이상의 정수는 표현할 수 없다. 예를 들어, Java의 long형 정수의 범위는 -9223372036854775808~9223372036854775807까지이다. 이와 달리 **파이썬은 정수 표현의 한계가 없다**. 이 점이 파이썬의 또 다른 큰 장점이다.

## 4.1.3 for 문과 리스트

- for in 구문
  - 반복문 키워드 for와 in 사이에 계속 새롭게 할당할 변수 n을 선언
  - in 뒤에 리스트 자료형을 넣어 리스트를 차례대로 순회하는 실행이 가능




[그림 4-5] for - in 구문에서 적용되는 numbers 리스트의 순회 방문 원리

- 4.1.3 for 문과 리스트

코드 4-13: for 문을 이용한 리스트의 정수 객체 순회
for_in_numbers.py

```
numbers = [11, 22, 33, 44, 55, 66]
for n in numbers:
    print(n, end = ' ')
```

실행결과

```
11 22 33 44 55 66
```

코드 4-14: for 문을 이용한 리스트의 실수 객체 순회
for_in_f_numbers.py

```
f_numbers = [1.1, 2.5, 3.7, 5.6, 9.2, 11.3, 6.8]
for f in f_numbers:
    print(f, end = ' ')
```

실행결과

```
1.1 2.5 3.7 5.6 9.2 11.3 6.8
```

- for in 구문의 in 다음에 범위를 지정하는 함수 range()가 아닌 numbers 라는 리스트가 있음
- 실수 리스트도 for in문을 통해 순회가 가능

## 4.1.3 for 문과 리스트

- for문을 이용하며 문자열을 원소로 가지는 리스트의 원소들을 출력

코드 4-15: for 문을 이용한 리스트의 문자열 객체 순회

for_in_str_list.py

```
summer_fruits = ['수박', '참외', '체리', '포도']

for fruit in summer_fruits:
    print(fruit, end = ' ')
```

실행결과

```
수박 참외 체리 포도
```

## 4.1.3 for 문과 리스트

- 누적 덧셈의 기능을 활용하여 리스트 내에 있는 정수 항목 값들의 합을 구하는 프로그램

코드 4-16: 리스트 항목 내 정수 값들의 누적 덧셈

for_sum1.py

```
numbers = [10, 20, 30, 40, 50]
s = 0

for n in numbers:
    s = s + n

print('리스트 항목 값의 합 :', s)
```

실행결과

```
리스트 항목 값의 합 : 150
```

## 4.1.3 for 문과 리스트

- 리스트 원소들의 합은 for 문을 사용하지 않고 내장함수 sum()을 사용하여 간편하게 합계를 구하는 것도 가능

코드 4-17: sum() 함수의 사용

for_sum2.py

```
numbers = [10, 20, 30, 40, 50]

print('리스트 항목 값의 합 :', sum(numbers))
```

실행결과

```
리스트 항목 값의 합 : 150
```

## 4.1.3 for 문과 리스트

**대화창 실습: 대화형 모드를 통한 1에서 100까지의 합**

```
>>> print('1에서 100까지의 합 :', sum(range(1, 101)))
1에서 100까지의 합 : 5050
```

- 문자열 자료형은 list() 함수를 이용하여 리스트 객체로 만드는 것이 가능

**대화창 실습: 대화형 모드를 통한 str 자료형의 리스트화**

```
>>> st = 'Hello'
>>> list(st)
['H', 'e', 'l', 'l', 'o']
```

## 4.1.3 for 문과 리스트

- 문자열 'Hello'를 list() 함수로 형 변환시켜 문자 원소들을 리스트로 만드는 것이 가능
- for in 구문 뒤에 문자열을 위치시켜 문자 단위로 분리해 순회할 수 있다.

코드 4-18: for 반복문에서 문자열의 사용

for_in_hello.py

```
for ch in 'Hello':
    print(ch, end = ' ')
```

실행결과

```
H e l l o
```

Leistung ist nicht alles / Keinen Studierenden zurücklassen


DAEJEON UNIVERSITY
SINCE 1980
대전대학교