

# for 명령문 (for loop)

- 반복(iteration)은 동일한 명령어들을 여러 번 실행하는 구조
- 컴퓨터는 인간과 다르게 반복적인 작업을 실수 없이 빠르게 할 수 있으며, 이것이 컴퓨터의 가장 큰 장점임
- 만약 같은 명령어를 1000번 반복해야 한다면 다음과 같이 작성

```
print("방문을 환영합니다!")  
print("방문을 환영합니다!")  
print("방문을 환영합니다!")  
print("방문을 환영합니다!")  
print("방문을 환영합니다!")  
...  
...  
print("방문을 환영합니다!")
```

**반복문 사용**

```
for i in range(1000):  
    print("방문을 환영합니다!")
```

# for 명령문 (for loop)

- 파이썬에서는 2가지 형태의 반복문 지원  
for 문 - 정해진 횟수만큼 반복할 때  
while 문 - 어떤 조건이 만족되는 동안 반복할 때
- for 명령문 형식



**iterable :** string, list, range( ) 등 셀 수 있는 데이터 모임(순서열 형식 데이터)

**a :** iterable의 원소 값을 갖는 변수

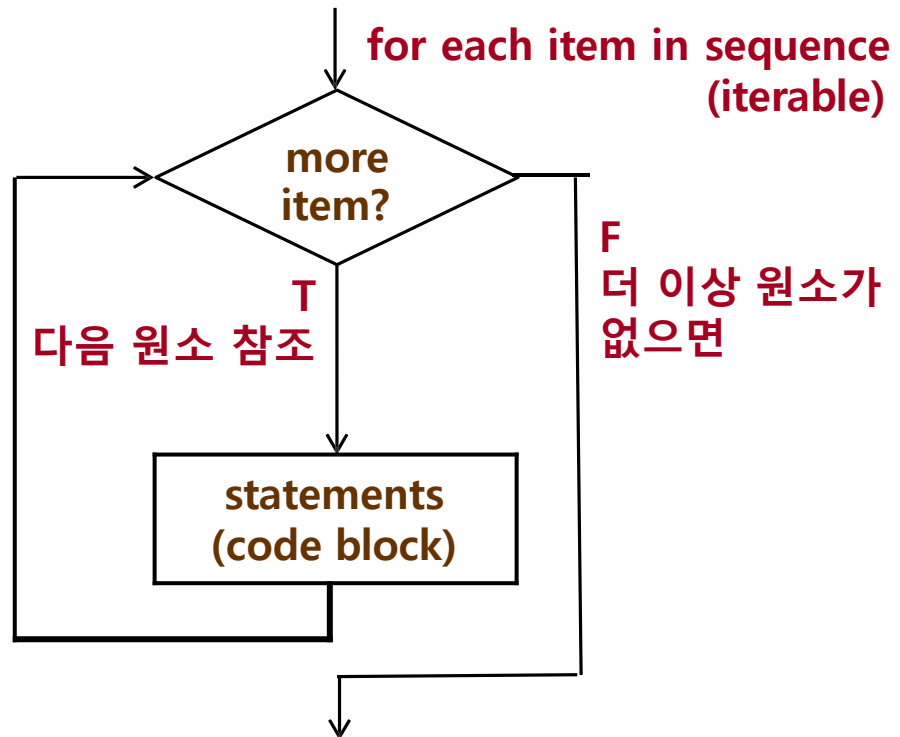
**for 문 :** iterable의 모든 원소 값을 차례로 a로 참조하여, for 내의 명령을 반복 수행

# for 명령문 (for loop)

- 수행 순서

- 반복
- ① 예약어 **in** 다음에 지정되어 있는 iterable object에서 차례로 데이터를 참조
  - ② 참조한 데이터를 예약어 **for** 다음에 명시한 변수 a에 저장함
  - ③ 코드 블록을 수행, iterable object 에서 참조할 더 이상의 원소가 없으면 반복을 종료

```
for a in iterable :  
    any statements
```



# for 명령문과 리스트

- 리스트 원소들에 대한 반복

리스트에서 원소를 하나씩 순서대로 참조

```
for name in ["길동", "철수", "영희", "유신"] :  
    print("안녕! " + name)
```

- ① 변수 name에 첫번째 값 "길동" 할당
- ② 코드 블록의 출력문 실행
- ③ 리스트에 있는 원소 값이 차례로 변수 name에 할당되어 출력문 실행을 반복
- ④ "유신"까지 할당하여 출력문 실행 후 리스트에 더 이상 남아있는 항목이 없으므로 for 문 종료 함

출력

```
안녕! 길동  
안녕! 철수  
안녕! 영희  
안녕! 유신
```

## for 명령문과 리스트

```
L = [1, 3, 6]
for x in L :
    print(x, end=" ") # 1 3 6 한 칸씩 띄어 출력
print("") # 줄 바꿈
```

end=" " : 출력 후 빈칸 추가 출력

```
for x in L :
    x = x * x
print(L) # [1,3,6] L 값은 바뀌지 않음
```

x는 차례로 1, 3, 6을 가리키는 변수  
x를 제공해서 x에 할당. 리스트의 원소 값은 바뀌지 않음

```
i = 0 # index용 변수 i를 도입
for x in L :
    L[i] = x * x # L[i] 값이 바뀜
    i += 1
```

```
print(L) # [1,9,36]
```

1.  $i = 0, x = 1, L[0] = 1 * 1$  저장,  $i$  증가
2.  $i = 1, x = 3, L[1] = 3 * 3$  저장,  $i$  증가
3.  $i = 2, x = 6, L[2] = 6 * 6$  저장,  $i$  증가
4. 리스트 L에 더 이상의 데이터가 없으므로 for 문 종료
5. print(L) 실행

# for 반복문(리스트 활용) 사용 예제

```
print("#####1#####")
L = ['bb','aa','gg','dd']
LT = []
for idx, val in enumerate(L):
    LT.append((val,idx))
LT.sort(reverse=True)
print(LT)
print()

print("#####2#####")
L = ['bb','aa','gg','dd']
LT = []
for data in enumerate(L):
    LT.append(data)
LT.sort(reverse=True)
print(LT)
print()

print("#####3#####")
D = {'b':20,'a':10,'c':30,'d':40}
DT = []
for key, val in D.items():
    DT.append((val,key))
DT.sort(reverse=True)
print(DT)
print()
```

#리스트 선언

#enumerate함수의 인덱스와 데이터를 가진 리스트 반환을 이용한 for 활용  
#LT 빈 리스트에 val, idx를 튜플 원소로 append, 튜플에서 val의 인덱스가 0 idx의 인덱스가 1  
#튜플 원소의 인덱스 0 값을 기준으로 내림차순 정렬(알파벳순)  
# 리스트 LT 출력

#동일하게 enumerate함수의 반환 리스트를 활용한 for문 사용  
#enumerate이 반환하는 tuple의 형태는 (index, data) 형태이므로 그 형태 그대로 list에 저장  
#위의 예시 1과 달리 인덱스 값이 튜플의 인덱스 0 저장되므로 숫자의 내림차순 정렬  
#리스트 LT 출력

#사전 D 선언

#key와 value를 tuple 형태로 반환하는 items메소드를 활용하여 for문 사용  
#DT리스트에 저장  
#value를 기준으로 내림차순 정렬  
#리스트 DT 출력

```
#####
[('gg', 2), ('dd', 3), ('bb', 0), ('aa', 1)]

#####
[(3, 'dd'), (2, 'gg'), (1, 'aa'), (0, 'bb')]

#####
[(40, 'd'), (30, 'c'), (20, 'b'), (10, 'a')]
```

# for 명령문과 문자열

- 문자열에 대한 반복

문자열에서 문자 하나씩 순서대로 참조

```
for char in "Sogang" :  
    print(char, end='#')
```

- ① 변수 char에 첫번째 문자열 "S" 할당
- ② 코드 블록의 출력문 실행
- ③ 문자열에 있는 문자들이 차례로 변수 char에 할당되어  
출력문 실행을 반복
- ④ 마지막 문자열 "g"까지 할당하여 반복코드를 수행 후 for문을 종료

출력

```
S#o#g#a#n#g#
```

## for (문자열 사용) 예제

- 문자열을 입력 받아 모음을 전부 없애는 코드

```
s = input('문자열 입력 : ')
vowels = "aeiouAEIOU"
result = ""
for letter in s:
    if letter not in vowels:
        result += letter
print(result)
```

# 입력 받은 문자열에서 모음을 제외한 문자열을 저장  
# letter 가 모음이 아니면  
# result = result + letter (문자열+문자열)

### 출력

```
문 자 열   입 력   : aAbBcC!defDEF&
bBcC!dfDF&
```

```
문 자 열   입 력   : abc!@#def$%^GHI&*(JKL!!!
bc!@#df$%^GH&*(JKL!!!
```



# for (문자열 사용) 예제

- 문자열을 입력 받아 자음과 모음의 개수를 집계하는 코드

```
string = input('문자열 입력 : ')
vowels = 0
consonants = 0
if len(string) > 0 :
    for char in string :
        if char.isalpha() :
            if char in 'aeiouAEIOU' :
                vowels += 1
            else:
                consonants += 1
print("모음의 개수", vowels)
print("자음의 개수", consonants)
```

# 자음과 모음의 개수를 저장하기 위한 변수

# 빈 문자열이 입력된 경우가 아니면  
# 입력 받은 문자열의 각 문자에 대해  
# 문자가 영어 대소문자인 경우만 체크  
# 모음을 저장한 문자열을 사용한 if문  
#모음일 경우 체크

#모음이 아닐 경우 자음 체크

## 출력

```
문 자 열   입 력   : aAbBcC!defDEF&
모 음 의   개 수   4
자 음 의   개 수   8
```

```
문 자 열   입 력   : abc!@#def$%^GHI&*(JKL!!!
모 음 의   개 수   3
자 음 의   개 수   9
```

실습

# for 문과 range() 함수

- range() 함수

- 객체형이 range인 일련의 정수 sequence를 생성하는 함수  
(함수 단독으로는 사용이 곤란하며, 주로 for loop와 함께 사용 )

```
>>> range(0,5) ←----- 0, 1, 2, 3, 4 정수 sequence 생성
range(0, 5)
>>> range(0, 10, 2) ←----- 0, 2, 4, 6, 8 정수 sequence 생성
range(0, 10, 2)
>>>                                range 형 객체 반환
```

- range(start, end, step)  
: start부터 end-1까지 step씩 증가 하면서 sequence 생성
- 슬라이싱에서의 start, end, step가 같은 의미

```
range(5)           # sequence 0, 1, 2, 3, 4
range(1, 6)        # sequence 1, 2, 3, 4, 5
range(4, 10, 2)     # sequence 4, 6, 8
range(5, -1, -2)    # sequence 5, 3, 1
range(9, 0)         # empty sequence(9 > 0 이므로)
```

## for 문과 range() 함수

```
L = [1, 3, 6] *** L의 원소 값을 제공하는 경우, range()를 사용하여 인덱스로 활용
for i in range(len(L)) : # i = 0,1,2
    L[i] = L[i]**2        # L[i] 값이 바뀜
print(L)                 # [1,9,36]
```

1. i = 0, L[0] = L[0]\*\*2 저장, L[0] 값 변경
2. i = 1, L[1] = L[1]\*\*2 저장, L[1] 값 변경
3. i = 2, L[2] = L[2]\*\*2 저장, L[2] 값 변경

```
for i in range(1,6) :
    print(i, end= " ")    # 1 2 3 4 5
print()
```

```
S = "I love icecream!"
for k in range(0,len(S),2) :
    print(S[k], end="")   # llv ccem 하나 걸러 출력
print()
```

```
for j in range(len(S)-1,-1,-1) : # len(S) = 16
    print(S[j], end="")         # !maerceci evol I 거꾸로 출력
print()
```

## for 문과 range() 함수

- range() 함수가 생성한 sequence를 list, tuple, set 등 다른 데이터 형으로 변환하여 사용 가능
- range() 함수가 생성한 sequence를 반복문의 반복 횟수 제어에도 사용

```
s = range(5)
print("s = {}, type(s) = {}".format(s, type(s)))
print()          # 빈 한줄 출력
L = list(range(4)) # list로 변환
print(L)
print()
# tuple로 변환
T = tuple(range(5, -1, -2))
print(T)
print()
for i in range(5): # 반복횟수로 사용
    print("range() test")
```

출력

```
s = range(0, 5) , type(s) = <class 'range'>
[0, 1, 2, 3]
(5, 3, 1)
range() test
range() test
range() test
range() test
range() test
```

# 리스트 내포(List comprehension)

- 리스트 안에 for 문을 포함

```
a = [1, 2, 3, 4]
result = [ ]
for num in a:
    result.append(num*3)
print(result)      # [3, 6, 9, 12]
```

동일한 코드.  
리스트 안에 for문을  
포함하는 문장으로 변환

```
a = [1, 2, 3, 4]
result = [ num * 3 for num in a]
print(result)      # [3, 6, 9, 12]
```

```
a = [1, 2, 3, 4]
result = [ num * 3 for num in a if num % 2 == 0]
print(result)      # [6, 12]
```

리스트 원소 중 2의 배  
수인 원소만 3배로 하여  
다른 리스트에 저장

```
L = ["apple", "banana", "pear"]
L1 = [ s.upper() for s in L ]
print(L1)          # ['APPLE', 'BANANA', 'PEAR']
```

리스트 문자열 원소 들을  
대문자로 변환하여 다른  
리스트에 저장

## for (리스트 내포형) 예제

- range 함수를 이용하여 1 부터 100 까지의 숫자 중에서 짝수들을 원소로 하는 리스트 L 을 생성한 후 , 그 중 8 의 배수를 출력하는 프로그램

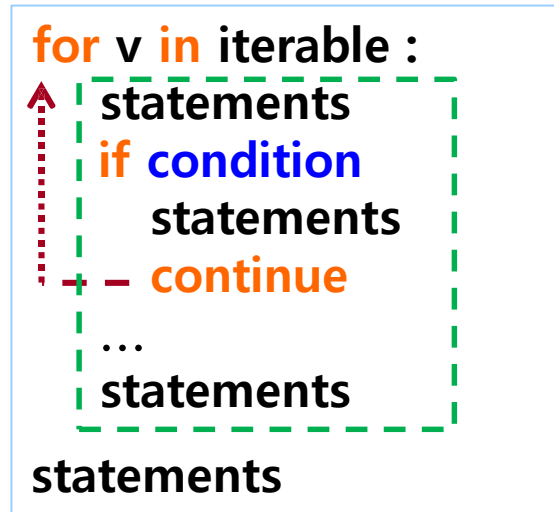
```
L= [ x for x in range(1,101) if x % 2 == 0 ]    #1부터 100까지의 짝수 리스트 생성
for i in L:                                     #8의 배수이면 출력
    if i % 8 == 0 :
        print(i, end = ' ')
print()
```

**출력**

```
8 16 24 32 40 48 56 64 72 80 88 96
```

# continue 명령문

- **continue** 명령어는 현재 loop에서 continue문 다음 명령어들(코드 블록 내에서)을 실행하지 않고 loop의 시작 위치로 가서 반복을 계속 실행
- 프로그램 흐름
  - if 문을 이용해서 continue 문을 실행할 조건 체크
  - 조건이 True이면 continue 문이 실행되어 반복문의 처음으로 돌아가고, False 이면 코드 블록 내의 나머지 명령어들을 실행
  - 조건이 True일때 continue 문 전에 실행할 명령어가 있다면 추가 가능





# continue 명령문

- 1부터 20까지의 수 중에서 3이나 4로 나누어지는 수를 제외한 숫자들을 차례로 출력하는 코드

## continue를 사용한 코드

```
for x in range(1, 21):  
    if x % 3 == 0 or x % 4 == 0:  
        continue  
    print(x, end=" ")  
print()
```

## continue를 사용하지 않은 코드

```
for x in range(1, 21):  
    if not(x % 3 == 0 or x % 4 == 0):  
        print(x, end=" ")  
print()
```

동일한 조건문

```
if x % 3 != 0 and x % 4 != 0:
```

# break 명령문

- **break** 명령어는 현재 loop에서 반복문을 종료하며 loop을 빠져나옴
- 프로그램 흐름
  - if 문을 이용해서 break 문을 실행할 조건 체크
  - 조건이 True이면 break 문이 실행되어 loop을 빠져 나가고, False이면 코드 블록 내의 나머지 명령어들을 실행
  - 조건이 True일때 break 문 전에 실행할 명령어가 있다면 추가 가능

```
for v in iterable :  
    statements  
    if condition  
        statements  
    - - break  
    ...  
    statements  
↓  
statements
```

# break 명령문

- 자연수 N이 소수(a prime number)인지 판단하는 코드

```
N = int(input("N(> 1)? "))
```

```
primeChk = True
```

```
for k in range(2, N) :
```

```
    if N % k == 0 :
```

```
        primeChk = False
```

```
if primeChk == True :
```

```
    print("prime")
```

```
else :
```

```
    print("not prime")
```

N이 어떤 수로 나누어지면 소수가 아닌 것을 loop 반복  
중간에 알아도, 이 코드블록들은 정확히 N-2 번 실행

N이 소수가 아닌 경우, loop 에서  
빠져 나오는 break 문을 실행

```
N = int(input("N(> 1)? "))
```

```
primeChk = True
```

```
for k in range(2, N) :
```

```
    if N % k == 0 :
```

```
        primeChk = False
```

```
        break
```

```
if primeChk == True :
```

```
    print("prime")
```

```
else :
```

```
    print("not prime")
```

# for (range(), break 사용) 예제

- 입력 받은 정수의 양수, 음수, 홀수, 짝수 여부를 판단하는 프로그램

```
for i in range(10):  
    num= int(input("Enter a number : "))  
    if num == 0:  
        print("입력 받은 수가 0 입니다")  
        break  
    elif num > 0:  
        if num % 2 :  
            print("{} : 양수, 홀수".format(num))  
        else :  
            print("{} : 양수, 짝수".format(num))  
    else :  
        if num % 2 :  
            print("{} : 음수, 홀수".format(num))  
        else :  
            print("{} : 음수, 짝수".format(num))  
  
print("프로그램을 종료합니다")
```

#range(10)을 사용하여 10번 입력받는다.  
#숫자가 0이면 반복문 종료  
#break 사용하여 for문 종료  
#양수인지 확인  
#홀짝 확인

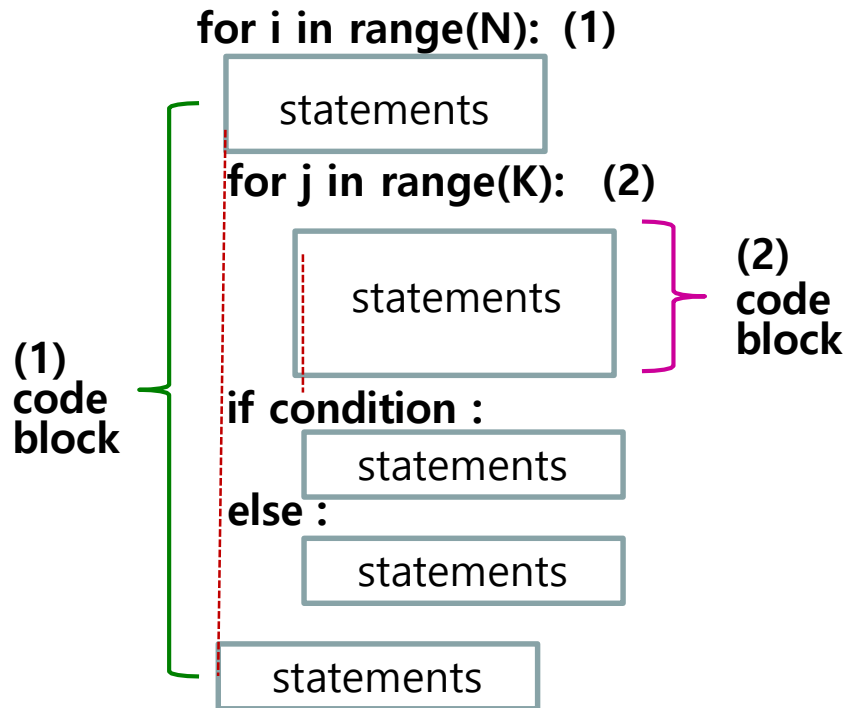
## 출력

```
Enter a number : -45  
-45 : 음수 , 홀수  
Enter a number : 12  
12 : 양수 , 짝수  
Enter a number : -2  
-2 : 음수 , 짝수  
Enter a number : 51  
51 : 양수 , 홀수  
Enter a number : 0  
입력 받은 수가 0 입니다  
프로그램을 종료합니다
```

실습

# Nested for loop

- for loop 안에 또 다른 for loop가 포함
- loop 안에 if 조건문, while 문등 어떤 것도 올 수 있음
- 각 코드 블록은 확실하게 들여쓰기를 해서 블록을 구별해야 함



# Nested for loop

- \* 기호를 삼각형 모양으로 출력하는 프로그램

```
for x in range(1, 6):  
    for y in range(x):  
        print("*", end="")  
    print("")          # 내부 반복문이 종료될 때마다 줄바꿈 실행
```

출력

```
*  
**  
***  
****  
*****
```

1.  $x = 1$   
y는 range(1)에 의해 가질 수 있는 값이 0. print("\*", end="") 명령 한번 실행.  
줄바꿈 실행.
2.  $x = 2$   
y는 range(2)에 의해 가질 수 있는 값이 0, 1. print("\*", end="") 명령 두번 실행.  
줄바꿈 실행.
3.  $x = 3$   
y는 range(3)에 의해 가질 수 있는 값이 0, 1, 2. print("\*", end="") 명령 세번 실행.  
줄바꿈 실행.
4.  $x = 4, x = 5$  일 때 동일한 패턴으로 실행함

# Nested for loop

- 구구단 출력하는 프로그램

```
for x in range(1,10) :  
    for y in range(1,10) :  
        print('%d*%d = %2d' % (x, y, x*y), end='  ')  
    print()
```

1. x = 1 에 대해 y 값 1,2,.....9 일때 실행
2. x = 2 에 대해 y 값 1,2,.....9 일때 실행
3. x = 3, x = 4,....., x= 9 일 때 까지 실행함

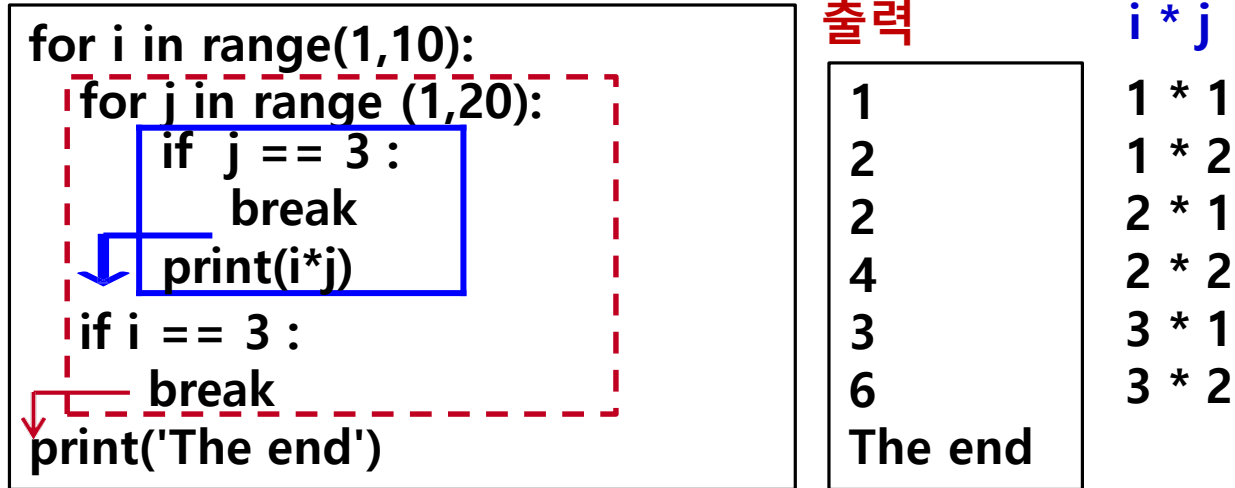
## 출력

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



# Nested for Loop

- Nested Loop (for 또는 while) 에서의 break
  - Nested Loop에서의 break 명령어는 자신을 포함하는 loop의 코드 블록을 빠져 나감

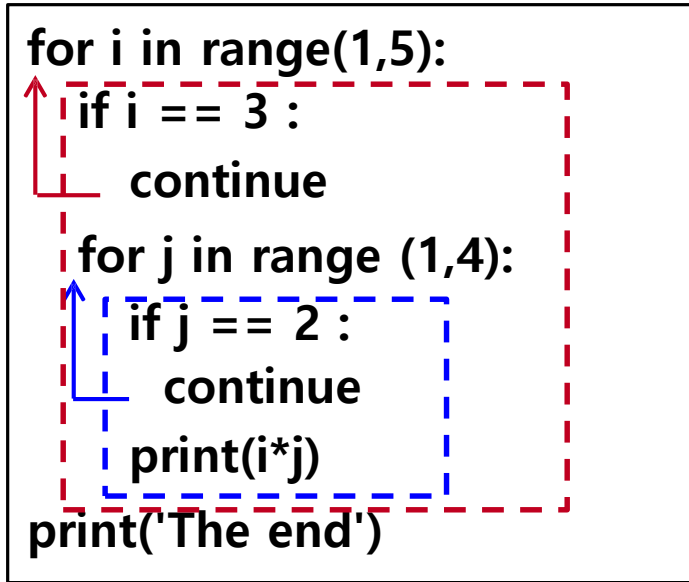


1.  $i = 1, i = 2$   
 $j$ 는  $\text{range}(1,20)$ 에 의해 1부터 19까지. 내부 파란색 코드 블록을 19번 실행.  
하지만,  $j$  값이 3일때는  $\text{break}$ 에 의해  $\text{print}(i*j)$  실행하지 않고  $j$  루프 종료.  
즉,  $i$ 가 1일때는  $j$ 가 1,2인 경우만  $\text{print}(i*j)$  실행,  $i$ 가 2일 때도 동일.  
 $i$ 가 3이 아니기 때문에  $i$  루프 계속 실행
2.  $i = 3$   
 $j$ 는  $\text{range}(1,20)$ 에 의해 1부터 19까지. 내부 파란색 코드 블록을 19번 실행.  
하지만,  $j$  값이 3일때는  $\text{break}$ 에 의해  $\text{print}(i*j)$  실행하지 않고  $j$  루프 종료.  
즉,  $i$ 가 3일때는  $j$ 가 1,2인 경우만  $\text{print}(i*j)$  실행  
 $j$  루프 종료후, 빨간 코드 블록에서  $i$ 가 3이기 때문에  $\text{break}$ 에 의해  $i$  루프를 중단.

# Nested for Loop

- Nested Loop (for 또는 while)에서의 continue
  - Nested Loop에서의 continue 명령어는 자신을 포함하는 loop의 시작 위치로 감

```
for i in range(1,5):  
    if i == 3 :  
        continue  
    for j in range (1,4):  
        if j == 2 :  
            continue  
        print(i*j)  
print('The end')
```



출력

```
1  
3  
2  
6  
4  
12  
The end
```

i \* j

```
1 * 1  
1 * 3  
2 * 1  
2 * 3  
4 * 1  
4 * 3
```

- 내부 파란색 loop 에서 j 값이 2이면 continue에 의하여 print(i\*j) 실행 없이 내부 loop의 시작 위치로 감
- 외부 빨간색 loop에서는 i 값이 3이면 외부 loop의 나머지 명령어들을 실행하지 않고 시작 위치로 감
- 결국, 이 코드는 1\*1, 1\*3, 2\*1, 2\*3, 4\*1, 4\*3을 출력

# Nested for Loop

- 역삼각형 그리기

- $N(\geq 1)$ 이 주어졌을 때, 첫 번째 줄에는 별  $N$ 개, 두 번째 줄에는 별  $N-1$ 개.....매 줄마다 별을 하나씩 줄여 출력
- 총  $N$  줄을 출력해야 함 : for loop이 필요
- 각 줄에 출력해야 할 별의 수가 다르기 때문에 또 다른 loop에서 별을 하나씩 출력하는 `print()` 함수를  $N, N-1, \dots, 1$  번 호출 : 내부 loop 필요
- $N = 4$ 일 때

외부 for loop는 총 4 번  
수행, `range()`를 사용하여  
 $i = 4, 3, 2, 1$ 이 되도록  
설정

4	****
3	***
2	**
1	*

내부 for loop에서는 `print()`를  
각각 4, 3, 2, 1 번 호출. 즉, 각  
줄을 출력할 때 `print()`를  $i$  번 호  
출하면 됨

# Nested for Loop

- 역삼각형 출력 코드

```
N = int(input("Enter # of lines : "))
for i in range(N,0,-1): # N 번 반복 ( i = N, N-1,...,1 )
    for j in range( i ) :    # i 번 반복 (역삼각형)
        print( "*", end="" ) # 각 줄에 i번 출력 (N,N-1,...,1개씩 출력)
    print()                 # 한 줄 출력하고 줄 바꿈
```

**출력**

```
Enter # of lines : 1
*
```

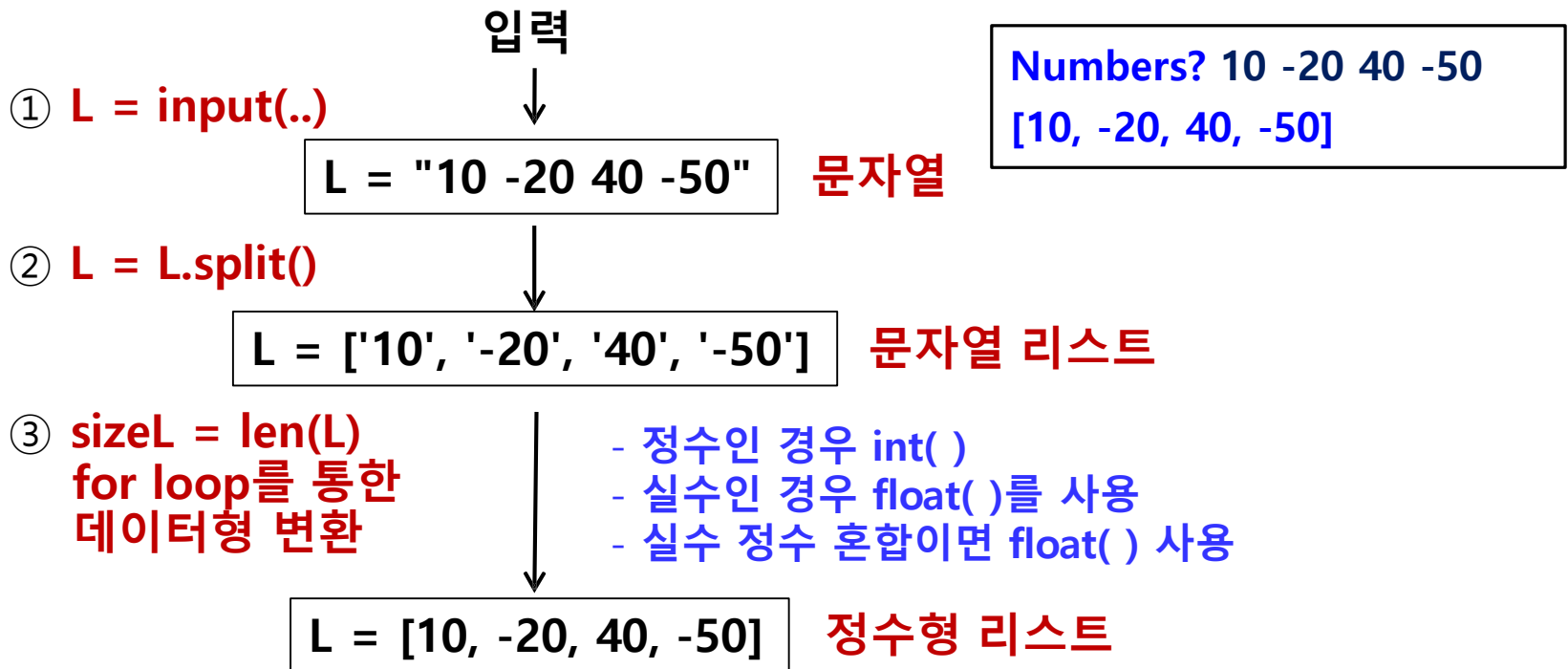
```
Enter # of lines : 4
****
***
**
*
```

```
Enter # of lines : 7
*****
*****
*****
****
***
**
*
```

# Nested for Loop

- 개수를 모르는 데이터 입력 받기

- ① 일련의 숫자들이 포함된 문자열 L을 입력 받음
- ② `str.split()` 메소드로 L을 분리하여 리스트 L에 저장  
: 리스트 L의 원소는 문자열이 됨
- ③ 리스트 L의 각 원소(숫자형태 문자열)를 정수로 변환  
: for loop를 사용하여 변환, 리스트 L의 크기는 `len(L)`



# Nested for Loop

- 개수를 모르는 데이터 입력 받는 파이썬 코드

```
L = input("Numbers? ") # 정수로 구성된 문자열 입력.  
L = L.split()          # 정수 형태 문자열로 각각 분리.  
for i in range(len(L)) : # 반복문에서 리스트 원소인 정수 형태 문자열을  
    L[i] = int(L[i])    # 정수로 변환.  
print(L)               # 변환된 정수 리스트 출력.
```

한 줄 명령문으로 가능

(3)  
(1) (2)  
**L=list( int(x) for x in input("Numbers? ").split())**

- 10 -20 40 -50를 입력하면 문자열 "10 -20 40 -50"을 얻음 (1).
- split()에 의하여 "10" "-20" "40" "-50" 으로 분할 (2).
- 이들은 for loop의 변수 x로 하나씩 참조 (3).
- 참조된 x 값은, int( )를 통하여 정수 10 -20 40 -50으로 변환.
- 변환된 값들은 list() 변환에 의하여 list로 생성.

동일  
코드

**L =[ int(x) for x in input("Numbers? ").split() ]**

# for (Nested for loop, range() 사용) 예제

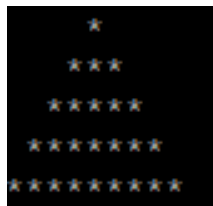
- 아래의 그림처럼 \*로 이루어진 모양을 nested for loop 형태와 range를 사용하여 출력하는 프로그램 작성



#스페이스 반복과 별 반복 모두 i에 좌우된다.

```
for i in range(1, 6):  
    for j in range(1, 6-i):  
        print(" ", end = "")  
    for j in range(1, 2*i):  
        print("*", end = "")  
    print()  
#스페이스 출력  
#줄넘김 없이 공백 출력  
#별출력  
#줄넘김 없이 * 출력
```

## 출력



실습