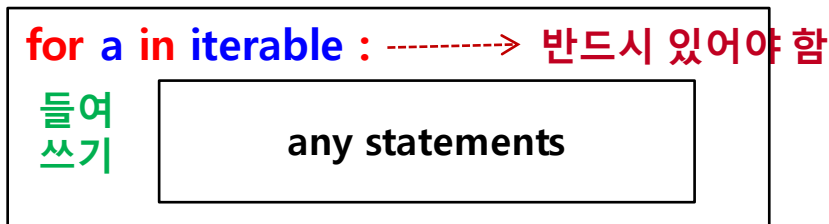


기초 인공지능 프로그래밍

6장 반복문

반복문 (loop)

- 반복(iteration)은 동일한 명령어들을 여러 번 실행하는 구조
- 파이썬에서는 2가지 형태의 반복문 지원
 - for 문 - 정해진 횟수만큼 반복할 때
 - while 문 - 어떤 조건이 만족되는 동안 반복할 때
- for 명령문 형식

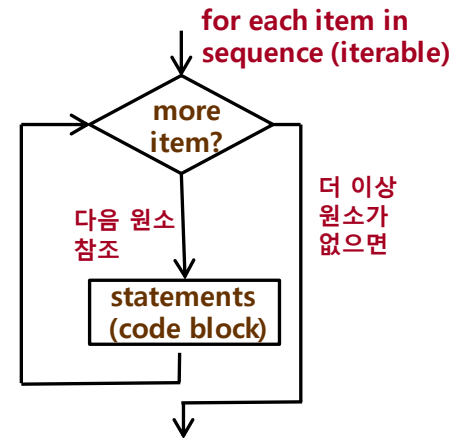


iterable - 문자열, 리스트, 집합, 튜플, 사전, range 객체등이 올 수 있음

a - iterable의 원소 값을 차례로 갖는 **변수**

- 실행 순서

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



for 명령문과 리스트

- 리스트 원소들에 대한 반복

리스트에서 원소를 하나씩 순서대로 참조

```
for name in ["Ann", "Jim", "Yoon", "Dan"] :  
    print("Hi!", end=" ")  
    print(name)
```

출력

```
Hi! Ann  
Hi! Jim  
Hi! Yoon  
Hi! Dan
```

- ① 변수 name에 리스트의 첫번째 원소 값 "Ann" 할당
- ② 코드 블록에 정의되어 있는 두개의 명령어인 출력문 실행
- ③ 리스트에 있는 원소 값이 차례로 변수 name에 할당되며 출력문 실행을 반복
- ④ "Dan"까지 할당하여 출력문 실행 후 리스트에 더 이상 남아있는 항목이 없으므로 for 문 종료 함

for 명령문과 리스트

- 리스트 원소들에 대한 반복

인자 `end=" "`으로 설정해서 출력할 때마다 줄바꿈이 발생하지 않고 빈칸이 추가 출력

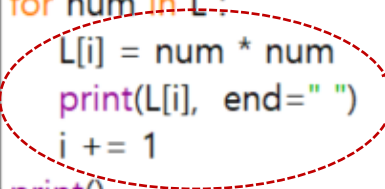
for 반복문 종료 후(마지막 원소 출력 후)에 줄바꿈 발생을 위해

변수 `num`은 리스트의 원소값을 순서대로 할당 받음.
`num`을 제공해서 변수 `result`에 할당하기 때문에 리스트의 원소 값은 바뀌지 않음

for 명령문과 리스트

- 리스트 원소들에 대한 반복

```
test.py - C:\Users\Wadmin\...
File Edit Format Run Options Window Help
L = [10, 20, 30, 40]
i = 0
for num in L:
    L[i] = num * num
    print(L[i], end=" ")
    i += 1
print()
print(L)
```



1. $i = 0$, $num = 10$, $L[0] = 10 * 10$ 저장/출력, i 증가
2. $i = 1$, $num = 20$, $L[1] = 20 * 20$ 저장/출력, i 증가
3. $i = 2$, $num = 30$, $L[2] = 30 * 30$ 저장/출력, i 증가
4. $i = 3$, $num = 40$, $L[3] = 40 * 40$ 저장/출력, i 증가
5. L에서 더 이상 참조할 데이터가 없으므로 반복문 종료

L의 원소값이 바뀌어 있음

```
>>> 100 400 900 1600
      [100, 400, 900, 1600]
```

for 명령문과 문자열

- 문자열을 입력 받아 모음을 전부 없애는 코드

```
chap5.py - C:\Users\admin\Desktop\script\chap5.py (3.9.2)
File Edit Format Run Options Window Help
s = input('Enter string : ')
vowels = "aeiouAEIOU"
result = "" # 입력 받은 문자열에서 모음이 아닌 문자를 순서대로 저장할 변수 초기화
for c in s: # 변수 c가 각 문자 할당 받음
    if c not in vowels: # c가 모음이 아니면
        result += c # result = result + c (문자열+문자열)
print(result)
```

Enter string : sogang!!
sgng!!
>>>

Ln: 8 Col: 0

문자열에서 문자 하나씩 순서대로 참조

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

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

출력

for 명령문과 문자열

- 문자열을 입력 받아 자음과 모음의 개수를 집계하는 코드

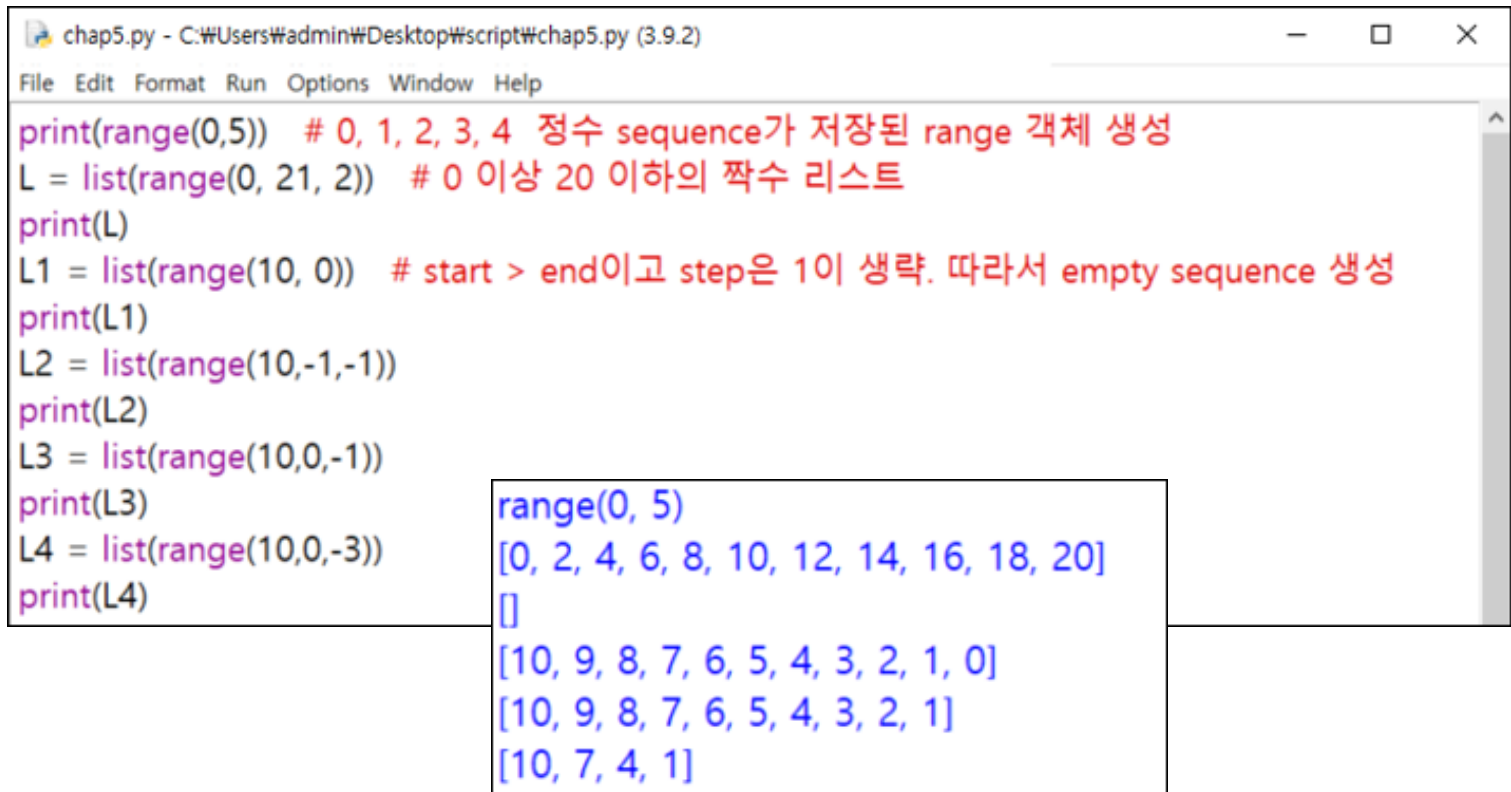
```
chap5.py - C:\Users\admin\Desktop\script\chap5.py (3.9.2)
File Edit Format Run Options Window Help
string = input('Enter string : ')
vowels = 0; consonants = 0 # 자음과 모음의 개수를 저장하기 위한 변수 초기화
for c in string :
    if c.isalpha() : # 문자가 영어 알파벳인 경우
        if c in 'aeiouAEIOU' : #모음인 경우
            vowels += 1
        else: #자음인 경우
            consonants += 1
print("Number of vowels : ", vowels)
print("Number of consonants : ", consonants)
```

Enter string : abcd!!ABCD****
Number of vowels : 2
Number of consonants : 6
>>>

Ln: 7 Col: 25

for 명령문과 range() 함수

- range() 함수 (for 반복문에 많이 사용)
 - 일련의 정수 sequence를 생성하여 range 객체에 저장/반환하는 함수
 - range(start, end, step) 형식 : 슬라이싱의 start, end, step과 같은 의미



The screenshot shows a Python IDE window titled 'chap5.py - C:\Users\admin\Desktop\script\chap5.py (3.9.2)'. The code in the editor is as follows:

```
print(range(0,5)) # 0, 1, 2, 3, 4 정수 sequence가 저장된 range 객체 생성
L = list(range(0, 21, 2)) # 0 이상 20 이하의 짝수 리스트
print(L)
L1 = list(range(10, 0)) # start > end이고 step은 1이 생략. 따라서 empty sequence 생성
print(L1)
L2 = list(range(10,-1,-1))
print(L2)
L3 = list(range(10,0,-1))
print(L3)
L4 = list(range(10,0,-3))
print(L4)
```

The output of the code is displayed in a separate box:

```
range(0, 5)
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
[]
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
[10, 7, 4, 1]
```


for 명령문과 range() 함수

- range() 함수 (for 반복문에 많이 사용)

→ 강의자료 p5에 있는 예제 코드와 비교

```
L = [1, 3, 6]
for i in range(len(L)): # i = 0,1,2
    L[i] = L[i]**2      # L[i] 값이 바뀜
print(L)
```

L의 원소 값을 제공하는 경우,
range()를 사용하여 인덱스로
활용

```
print("#####")
S = "abcdef"
for j in range(len(S)-1,-1,-1): # j = 5, 4, 3, 2, 1, 0
    print(S[j], end="")        # 거꾸로 출력
print("#####")
for i in range(3): # 단순히 반복문의 반복횟수 제어로 사용
    print("range() test")
```

```
[1, 9, 36]
#####
fedcba#####
range() test
range() test
range() test
>>>
```

리스트 내포 (list comprehension)

- 리스트 안에 for 반복문 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 명령어로 변환

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

```
a = list(range(1, 21, 2)) # [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
result = [ num * 10 for num in a if num % 3 == 0]
print(result)           # [30, 90, 150]
```

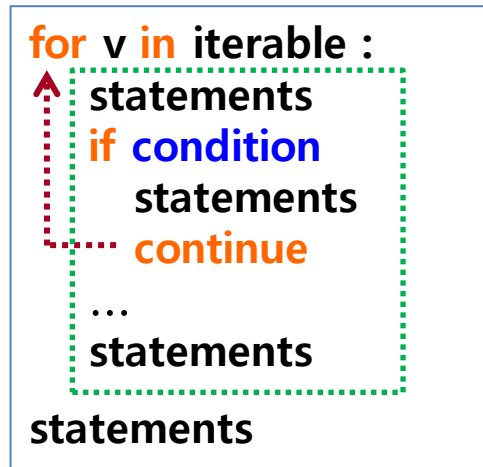
리스트 원소 중 3의 배수인 원소만
10배로 하여 다른 리스트에 저장

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

리스트 문자열 원소들을 대문자로
변환하여 다른 리스트에 저장

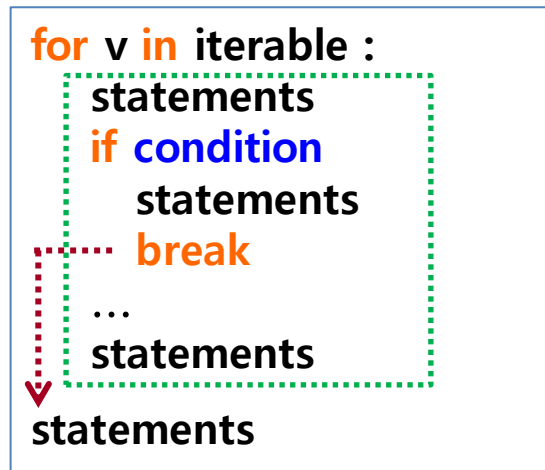
반복문 제어 명령어 (continue 명령문)

- continue 명령어는 반복문의 코드 블록 안에서 continue문 다음 명령어들을 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 시작 위치로 돌아가고, False 이면 코드 블록 내의 나머지 명령어들을 계속 실행
 - 조건이 True일 때 continue 문 실행 전에 실행할 명령어가 있다면 추가 가능



반복문 제어 명령어 (break 명령문)

- break 명령어는 반복문에서 반복을 종료하며 반복문을 빠져나옴
- 프로그램 흐름
 - if 문을 이용해서 break 문을 실행할 조건 체크
 - 조건이 True이면 break 문이 실행되어 반복문을 종료하고 빠져나오며, False 이면 코드 블록 내의 나머지 명령어들을 계속 실행
 - 조건이 True일 때 break 문 실행 전에 실행할 명령어가 있다면 추가 가능



반복문 제어 명령어 사용 예시

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

동일한 결과를 내는 코드

```
test.py - C:\Users\wadmin\Desktop...
File Edit Format Run Options Window Help
for x in range(1, 21):
    if x % 3 == 0 or x % 4 == 0:
        continue
    print(x, end=" ")
print()
Ln: 6 Col: 0
```

```
test.py - C:\Users\wadmin\Desktop...
File Edit Format Run Options Window Help
for x in range(1, 21):
    if x % 3 != 0 and x % 4 != 0:
        print(x, end=" ")
print()
Ln: 5 Col: 0
```

x가 3의 배수이거나 4의배수이면
print(x)를 실행하지 않고 반복문의
시작 위치로 제어가 돌아가서 다음
x 값으로 실행

```
>>> 1 2 5 7 10 11 13 14 17 19
```

반복문 제어 명령어 사용 예시

- 자연수 N이 소수(a prime number)인지 판단하는 코드

```
test.py - C:\Users\Wadmi...
File Edit Format Run Options Window Help
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")
Ln: 10 Col: 0
```

N이 어떤 수로 나누어 떨어지더라도 N-1 까지 실행

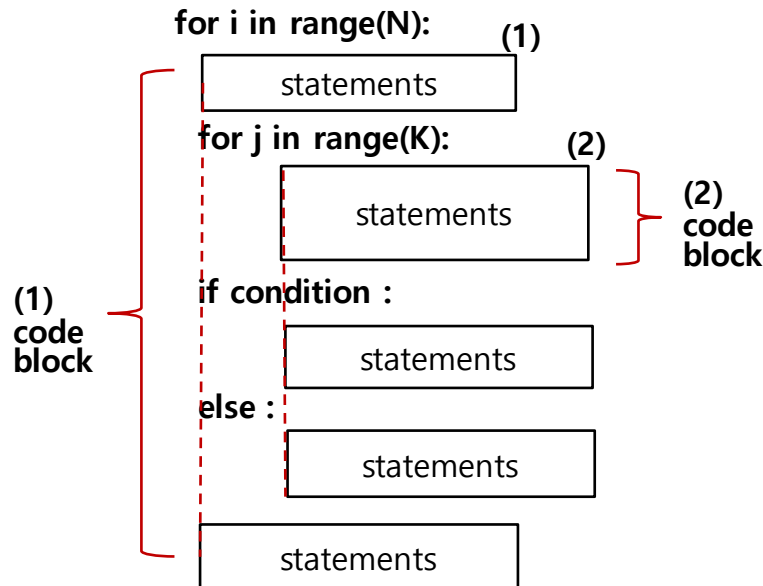
```
*test.py - C:\Users\Wad...
File Edit Format Run Options Window Help
N = int(input("N(> 1)? "))
primeChk = True
for k in range(2, N):
    if N % k == 0:
        primeChk = False
        break
if primeChk == True:
    print("prime")
else:
    print("not prime")
Ln: 11 Col: 0
```

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

```
N(> 1)? 13
prime
>>>
```

Nested for 반복문

- for 반복문의 코드 블록에 또 다른 for 반복문이 있는 형태
- 각 코드 블록은 확실하게 들여쓰기를 해서 블록 구분을 해야함



Nested for 반복문

- * 기호로 삼각형 모양을 출력하는 프로그램

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

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

출력

1. 변수 x가 1일 때
변수 y는 range(1)에 의해 가질 수 있는 값이 0. 즉 print(" * ", end=" ") 명령 한번 실행.
마지막 줄바꿈 실행.
2. 변수 x가 2일 때
변수 y는 range(2)에 의해 가질 수 있는 값이 0과 1. 즉 print("*", end="") 명령 두번 실행.
마지막 줄바꿈 실행.
3. 변수 x가 3, 4, 5 일 때 동일한 패턴으로 실행함

Nested for 반복문

- * 기호로 역삼각형 모양을 출력하는 프로그램
 - $N(\geq 1)$ 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 호출하면 됨

```
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
```

Nested for 반복문

- 구구단 출력하는 프로그램

```
test.py - C:\Users\admin\Desktop\script\test.py (3.1...
File Edit Format Run Options Window Help
for x in range(1, 10):
    for y in range(1, 10):
        print('%2d*%2d = %2d' % (x, y, x*y), end=' ')
    print("")
Ln: 5 Col: 0
```

```
>>> 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 반복문에서 제어 명령어

- Nested 반복문에서의 break 명령어는 자신을 포함하는 반복문의 코드 블록만 빠져 나감

```
for i in range(1,10):  
    for j in range (1,20):  
        if j == 3 :  
            break  
        print(i*j)  
    if i == 3 :  
        break  
print('The end')
```

출력

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

i * j

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

- Nested 반복문에서의 continue 명령어는 자신을 포함하는 반복문의 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

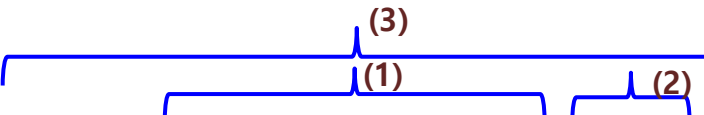
```
1 * 1  
1 * 3  
2 * 1  
2 * 3  
4 * 1  
4 * 3
```

Nested 반복문 예시 코드

- 여러 개의 숫자 데이터들 입력 받는 파이썬 코드

```
S = input("Numbers? ") # 정수로 구성된 문자열 입력.  
L = S.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


L=list(int(x) for x in input("Numbers? ").split())

동일
명령어

- 10 -20 40 -50 를 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.

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

while 명령문

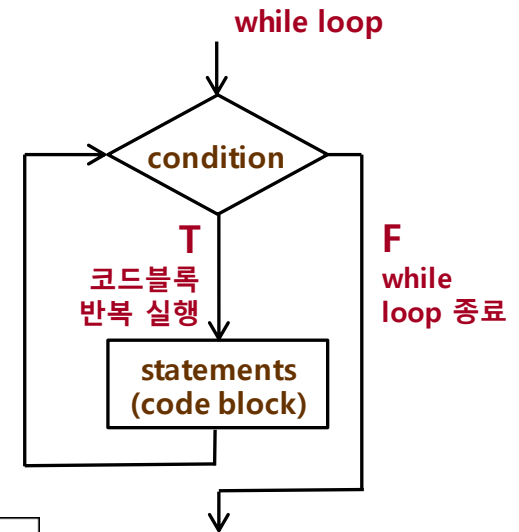
- while 반복문은 어떤 조건이 만족하는 동안 반복하는 구조
- while 명령문 형식

while condition : ----> 반드시 있어야 함

들여
쓰기

any statements

while 문: condition의 결과가 True이면 코드 블록 내의 명령어들을 반복 실행

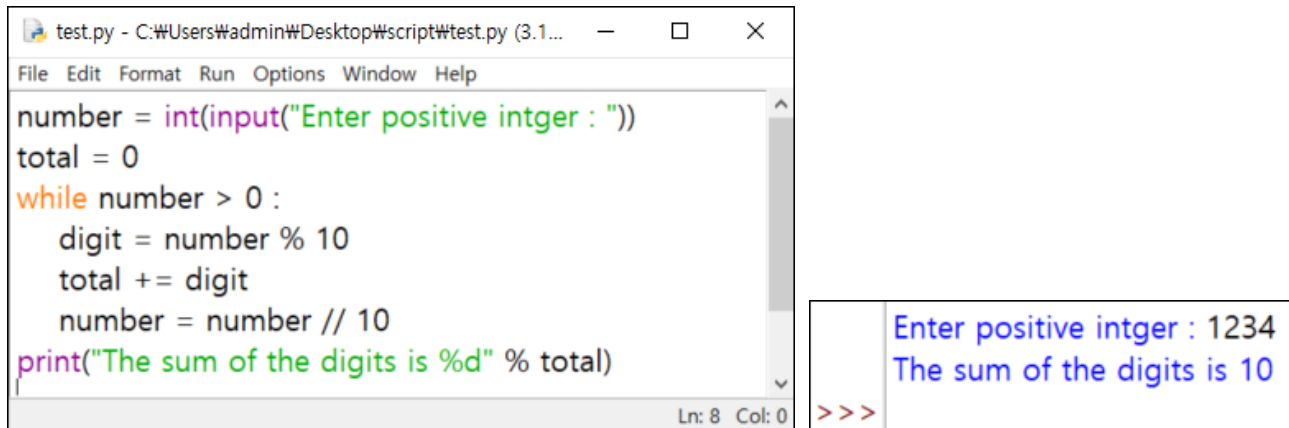


```
chap5.py - C:\Users\wadmin\Desktop\script\chap5.py (3.9.2)
File Edit Format Run Options Window Help
N = 100
s = 0      # 합계를 저장할 변수 s를 0으로 초기화
number = 1 # range() 대신, 변수 number를 사용
while number <= N :
    if number % 3 == 0 : # 3으로 나눠 떨어지면 3의 배수
        s += number
    number = number + 1
print("The sum of all 3 multiples between 1 and 100 is %d." % s)

The sum of all 3 multiples between 1 and 100 is 1683.
>>>
```

while 명령문 예시 코드

- 입력 받은 정수의 각 자리수의 합을 계산하는 프로그램(1234 경우, $1+2+3+4$ 를 계산하는 것)



The screenshot shows a Python IDE window titled 'test.py - C:\Users\admin\Desktop\script\test.py (3.1...'. The code in the editor is as follows:

```
number = int(input("Enter positive integer : "))
total = 0
while number > 0 :
    digit = number % 10
    total += digit
    number = number // 10
print("The sum of the digits is %d" % total)
```

Below the code editor, the output of the program is displayed in a separate window:

```
>>> Enter positive integer : 1234
The sum of the digits is 10
```

while 반복문 코드 블록 안에서 (number가 1234인 경우)

1. $\text{digit} = 4$, $\text{total} = 0 + 4$, $\text{number} = 123$
2. $\text{digit} = 3$, $\text{total} = 4 + 3$, $\text{number} = 12$
3. $\text{digit} = 2$, $\text{total} = 7 + 2$, $\text{number} = 1$
4. $\text{digit} = 1$, $\text{total} = 9 + 1$, $\text{number} = 0$
5. $\text{number} > 0$ 이 False이기 때문에 while 문 종료
6. $\text{print}(\text{total})$ 실행

while 명령문 예시 코드

- 숫자 맞추기 게임

`random.random()` : 0.0 이상 1.0 미만 실수를 반환
`random.random() + 1.0` : 1.0 이상 2.0 미만 실수를 반환
(난수의 범위를 설정할 수 있음)

`random.randint(n, m)` : n 이상 m 이하 임의의 정수 반환

```
test.py - C:\Users\admin\Desktop\script\test.py (3.11.4)
File Edit Format Run Options Window Help
import random

tries = 0
number = random.randint(1, 100)
print("Match game a number between 1 and 100")
while tries < 10:
    guess = int(input("Enter a number : "))
    tries = tries + 1
    if guess < number:
        print("No! Larger number!")
    elif guess > number:
        print("No! Smaller number!")
    else:
        break
if guess == number:
    print("Succeeded with %d attempts" % tries)
else:
    print("Failed. number is %d" % number)
```

```
Match game a number between 1 and 100
Enter a number : 50
No! Smaller number!
Enter a number : 30
No! Larger number!
Enter a number : 40
No! Larger number!
Enter a number : 45
No! Smaller number!
Enter a number : 43
No! Larger number!
Enter a number : 44
Succeeded with 6 attempts
```

>>>

Ln: 19 Col: 0

while 명령문의 break, continue 문

- break, continue 명령어의 기능은 for 반복문과 같음
- 자연수 N이 소수(a prime number)인지 판단하는 while 문 코드

```
test.py - C:\Users\Wadmin\...
File Edit Format Run Options Window Help
N = int(input("N(> 1)? "))
k = 2
primeChk = True
while k < N :
    if N % k == 0 :
        primeChk = False
        break
    k = k + 1
if primeChk == True :
    print("prime")
else :
    print("not prime")
Ln: 13 Col: 0
```

▪ 무한 루프

```
test.py - C:\Users\Wadmin\...
File Edit Format Run Options Window Help
while(True) :
    n = int(input("Enter the number : "))
    if n == 0 :      # 0이 입력되면 반복문 종료
        break
    print(n)
Ln: 6 Col: 0
```

조건이 언제나 True(무한 루프)
: 코드 블록에는 어떤 조건일 때 반복이
종료하는 명령어가 있어야함

```
Enter the number : 10
10
Enter the number : -4
-4
Enter the number : 0
>>>
```