

# 응용 프로그래밍

## python

반복문  
- for, while 문 -

# 수업내용

- 멤버십 연산자: in
- for 문
- while 문
- break, continue문

# python

## 멤버십 연산자: in

- in 연산자

- 파이썬에는 여러 개의 자료를 보관하는 시퀀스(sequence) 자료형을 제공하고 있다.
- 멤버십 연산자는 특정 자료가 해당 시퀀스 자료형에 속하는지(in)를 판별하는 연산자이다.
- 시퀀스 자료형에 관한 내용은 뒤에서 상세히 다룰 것이다.

※ 시퀀스 자료형에는 문자열(string), 리스트(list), 튜플(tuple), 딕셔너리(dictionary) 또는 집합(set) 등이 있다.

# Example

```
>>> 1 in [1,2,3]
```

```
True
```

```
>>> 1 not in [1,2,3]
```

```
False
```

```
>>> fruits = ['사과', '배', '복숭아']
```

```
>>> print('사과' in fruits)
```

```
True
```

```
>>> print('수박' in fruits)
```

```
False
```

# python

## for 문

- 반복문의 개념
- for문
- range()
- 중첩 for문

- 반복문(loop)은 정해진 동작(처리 과정)을 반복적으로 수행할 때 사용한다.
- 일상 생활에서 학생 100명의 성적 계산과 출력, 쇼핑몰에서 상품 추천, Word에서 단어 바꾸기 명령 등을 실행한다.
- 예를 들어 Word에서 단어를 바꿀 때 먼저 각 단어가 바꾸고자 하는 단어와 비슷한지 비교하는 과정이 필요하고, 이를 전체 단어에 적용하기 위해 단어를 찾는 과정이 계속 수행되어야 한다.
- 이렇듯 반복문은 모든 프로그램에서 핵심적으로 사용된다.

# 횟수 반복문: for

- 파이썬에서 반복 횟수 제어문으로 for문이 사용된다.
- 문자열 'python'을 반복 출력하면

```
>>> for i in 'python':  
    print(i)
```

p  
y  
t  
h  
o  
n

for 문 끝에 반드시 콜론(:)을 붙여야 한다.

반복되는 문장은 반드시 들여쓰기를 해야 한다.  
※ 백스페이스(backspace)키 또는 엔터키 2번 누르면 반복문을 탈출할 수 있다.

※ 문자열 'python'은 연속된 문자들을 포함하고 있으므로 반복 가능한 시퀀스 자료형이다.  
각 반복에서 각 문자는 반복 변수 i에 순차적으로 할당된다.



# Example

- 구구단 중 9단의 일부를 for문으로 출력해보세요.

```
#ex4-1.py
```

```
for i in [1,2,3,4,5]:
```

```
    print('9 * {} = {}'.format(i, 9*i))
```

9 \* 1 = 9

9 \* 2 = 18

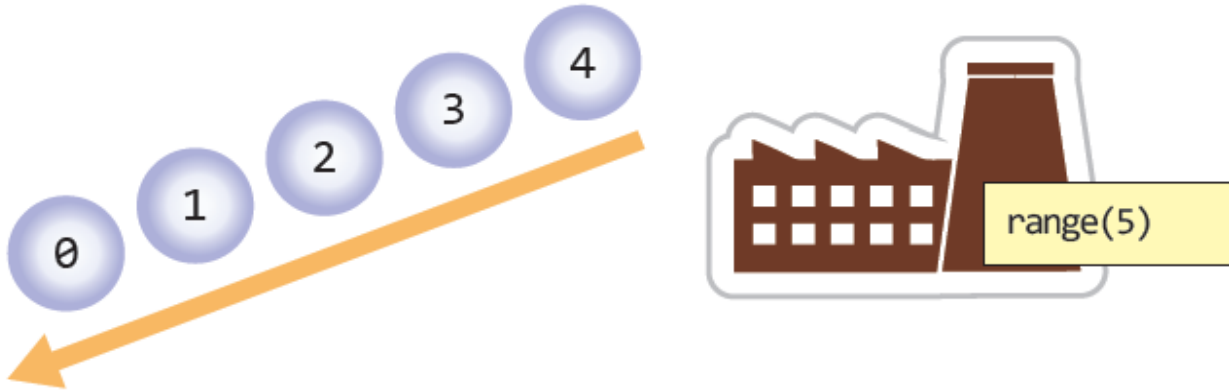
9 \* 3 = 27

9 \* 4 = 36

9 \* 5 = 45

# range( )

- range( )는 숫자들을 만들어내는 공장으로 생각하면 된다.
- range(5)는 5개의 정수를 생성한다. 즉 0, 1, 2, 3, 4가 만들어진다.



```
range([start,] stop[, step])
```

- start: 연속적인 정수 값들 중에서 시작 값
- stop: 연속적인 정수 값들 중에서 끝 값
  - ◆ 끝 값은 포함하지 않고 -1 한 값을 출력한다. (예) range(5), 5-1=4가 됨
- step: 연속적인 정수 값들의 증감 값, 즉 앞의 값과 다음 값의 차이
- **range(5)**라고 하면 시작 값과 증감 값이 생략된 것으로 **range(0, 5, 1)**과 같다.

# Example

- 1부터 5까지 순차적으로 출력하고 싶다면 어떻게 하면 될까요?
  - ◆ 정답: `range(1,6)`을 사용하면 된다.

```
#ex4-2.py
```

```
for i in range(1,6):  
    print(i)
```

1  
2  
3  
4  
5

# Exercise 1

- 10부터 시작하여 1까지 반복 출력하세요.

```
#ex4-3.py
```

```
for i in [ ]:
```

```
[ ]
```

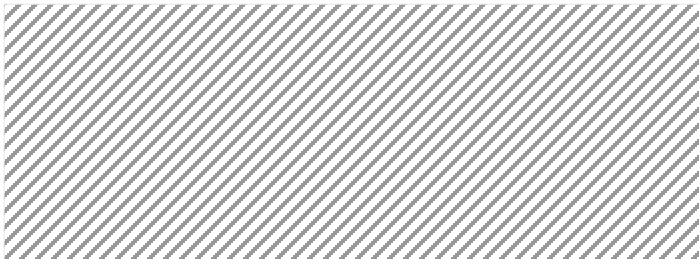
**end=' '**와 같이 지정하면 줄이 바뀌지 않고 한 줄에 전부 출력한다.

10 9 8 7 6 5 4 3 2 1

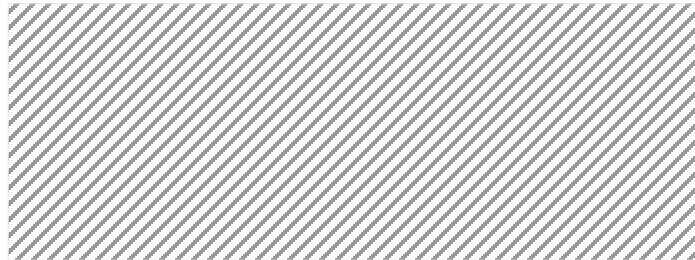
# Quiz

- 다음 코드의 출력 결과는?

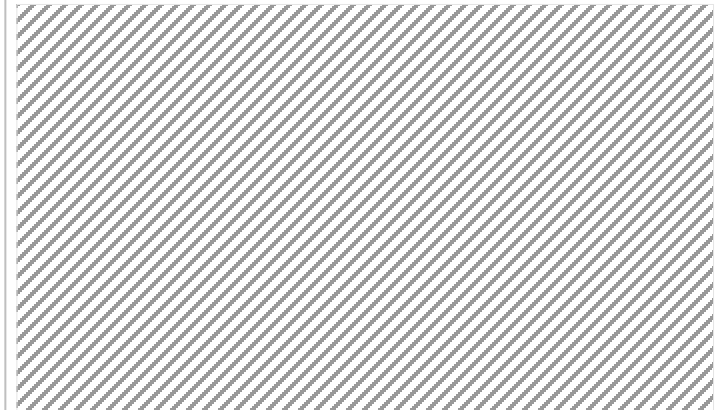
```
>>> for i in range(3,6):  
    print(i)
```



```
>>> for i in range(4,10,2):  
    print(i)
```



```
>>> for i in range(0,-10,-2):  
    print(i)
```



## Exercise 2

- 사용자로부터 정수를 입력 받아서 팩토리얼(factorial number)을 계산해 보세요.
  - ◆  $4!$  은  $1*2*3*4 = 24$ 이다.
  - ◆ 팩토리얼(fact) 변수의 초기값은 반드시 1로 초기화 시켜야 한다.

정수를 입력하세요:4  
 $4!$ 은 24이다.

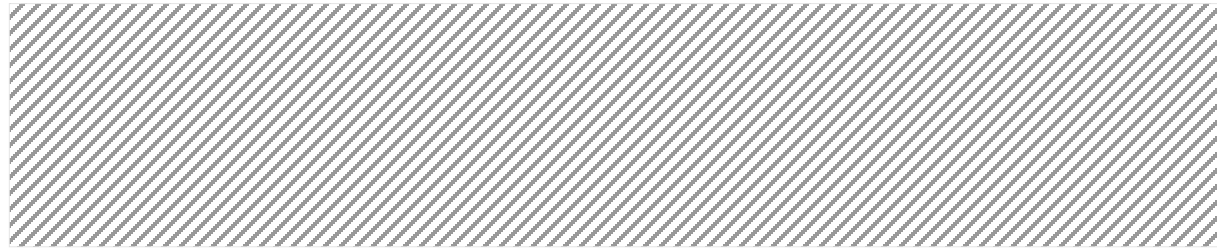
정수를 입력하세요: 10  
 $10!$ 은 3628800이다.

## Exercise 2: Solution

#ex4-4.py

```
number = int(input('정수를 입력하세요: '))
```

```
fact = 1
```



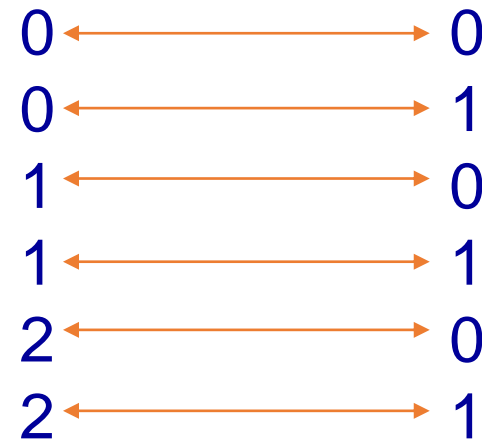
```
print('{}!은 {}이다'.format(number, fact))
```



- 중첩 for문은 for문 안에 또 다른 for문이 있는 중첩구조이다.
- 안쪽 for 문은 바깥쪽 for 문이 한번 반복할 때마다 새롭게 실행된다.
- 중첩 for문에서는 반복 변수가 달라야 한다.
  - ◆ 예를 들어 바깥쪽 for문의 반복 변수가 i라면 안쪽 for문의 반복 변수는 j로 서로 다르다.
  - ◆ 만약 같은 변수가 사용되면 오류가 발생할 가능성이 높다.

```
for i in range(3):  
    for j in range(2):  
        print('*')
```

바깥쪽 for문의  
반복 변수 i      안쪽 for문의  
반복 변수 j






# Example

- 중첩 for문을 사용하여 구구단을 출력하세요.

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

# Example




- 중첩 for문을 사용하여 구구단을 출력하세요.

```
#ex4-5.py  
  
for i in     for j in         print('{}*{}={}' , end=' ')  
    print("")
```

# Exercise 3

- \*(star) 기호를 사용해서 다음과 같은 피라미드 모양을 출력하세요.

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

```
for i in   
    for j in   
        print('*', end='')  

```

# Homework 1

- Exercise 3을 응용하여 다음과 같은 역 피라미드 모양을 출력하세요.

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

# Homework 1: Solution

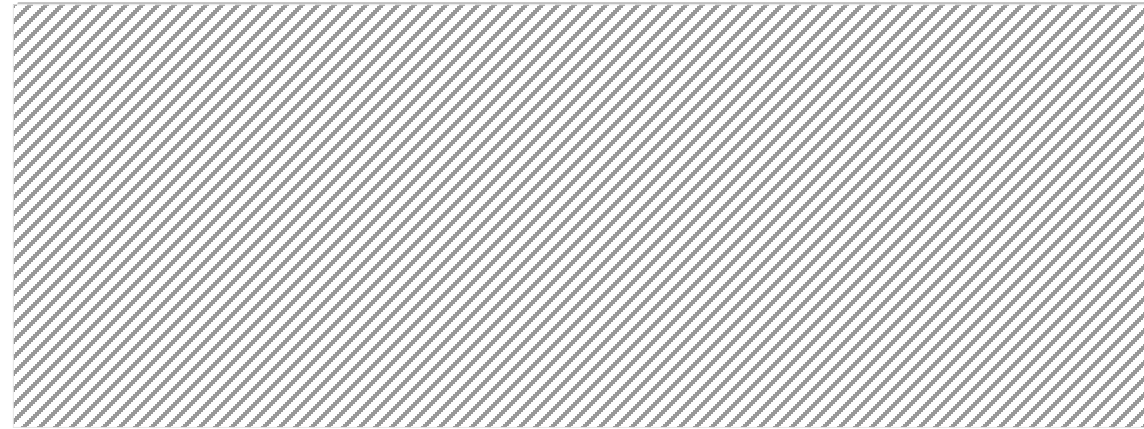
\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

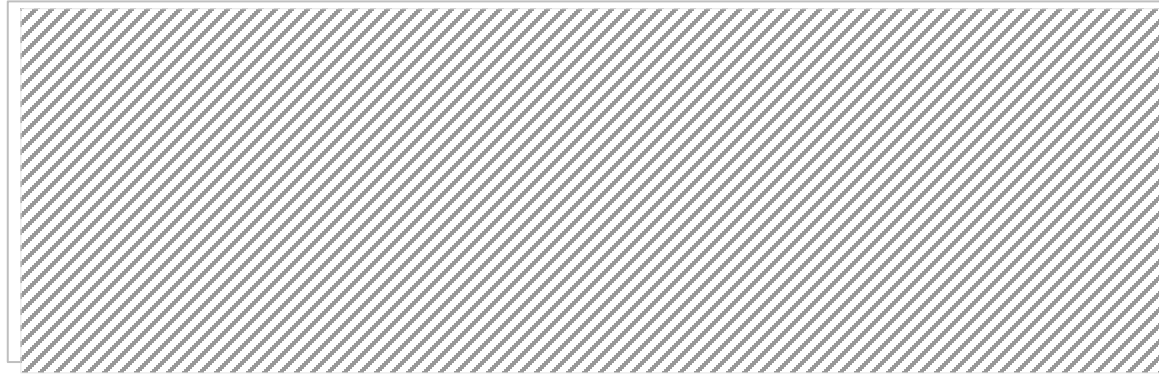


# Homework 2

- for문과 if을 함께 사용하여 1에서 10 사이의 수 중에서 3의 배수에 해당하는 3, 6, 9를 제외한 나머지 수를 모두 출력하세요.

1  
2  
4  
5  
7  
8  
10

# Homework 2: Solution





# python

## while 문

- 조건 제어 반복문

# 조건 제어 반복문 : While

- while문은 조건식의 값이 참인 동안 주어진 문장을 반복 실행한다.
- 조건식의 값이 거짓이 되면 반복을 중단한다.
- While문은 반복의 횟수는 모르지만, 반복의 조건은 알고 있는 경우에 사용하는 반복 구조이다.

# Example

- 1부터 10까지 정수의 합을 출력하세요.

```
#ex4-6.py
```

```
total = 0
```

```
i = 1
```

```
while i <= 10:
```

```
    total += i
```

```
    i += 1
```

```
print('1~10의 합은 {}이다.'.format(total))
```

1~10의 합은 55이다.

while 문 끝에 반드시 콜론(:)을 붙여야 한다.

반복되는 문장은 반드시 들여쓰기를 해야 한다.

## Exercise 4

- 사용자로부터 정수를 입력 받아서 1부터 입력한 정수까지의 합을 출력하세요.

정수를 입력하세요:4  
1부터 4까지의 합은 10이다.

정수를 입력하세요:18  
1부터 18까지의 합은 171이다.

## Exercise 4: Solution

```
#ex4-7.py
```

```
number = int(input('정수를 입력하세요:'))
```

```
total = 0
```

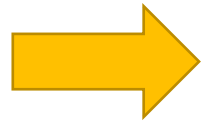
```
i = 1
```

```
print('1부터 {}까지의 합은 {}이다.'.format(number, total))
```

- while문 블록 내부에서 반복문을 제어하기 위한 값을 증가하거나 감소시키지 않으면, while문은 무한 반복문이 된다.

```
x = 1  
  
while x < 10:  
    print('while loop= ', x)
```

```
while loop= 1  
while loop= 1  
while loop= 1  
while loop= 1  
while loop= 1  
while loop= 1  
...
```



무한 반복 while문의 실행을 종료하려면  
***Ctrl+C*** 키를 누르면 된다.

# 무한 반복 while문

- while문의 조건식 값이 항상 참인 경우에도 무한 반복문이 된다.

```
x = 1  
  
while True:  
    print(x)  
    x += 1
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
...

# python

## break, continue문

- 반복을 종료하는 break
- 조건의 명령을 건너뛰고 반복하는 Continue



# 반복을 종료하는 제어문: break

- 반복문에서 논리적으로 반복을 종료하는 방법이 있는데, 바로 break이다.
- Break가 반복문에 있으면, 이를 종료할 수 있다.

# Example

- 무한 반복 while문에서 1에서 10까지만 출력하세요.

```
x = 1  
  
while True:  
    print(x)  
    if x == 10:  
        break  
    x += 1
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

## Exercise 5

- for문을 사용하여 1에서 1000사이의 자연수 중 1에서 10까지만 출력하세요.

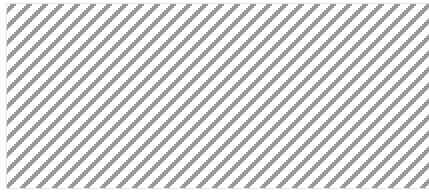
1  
2  
3  
4  
5  
6  
7  
8  
9  
10

# Exercise 5: Solution

```
#ex4-8.py
```

```
for i in :
```

```
    print(i)
```

```

```

# 조건의 명령을 건너뛰고 반복하는 제어문: continue

- Continue는 break와 달리 특정 조건에서 남은 명령을 건너뛰고 다음 반복문을 실행한다.
- 반복문은 종료되지 않고 다음 반복문으로 계속된다.

# Example

- while문을 사용하여 1에서 10까지의 자연수 중 2의 배수를 제외하고 출력하세요.

```
#ex4-9.py
i = 0
while i < 10:
    i += 1
    if i % 2 == 0:
        continue
    print(i)
```

1  
3  
5  
7  
9

# Homework 3

- 무한 while문으로 작성된 소스 코드를 완성하여 0과 73사이의 숫자 중에서 3으로 끝나는 숫자만 출력되도록 하세요.

```
i=0
```

```
while True:
```

```
     ①
```

```
     ②
```

```
     ③
```

```
     ④
```

```
     ⑤
```

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

```
    i += 1
```

<출력 결과>

3 13 23 33 43 53 63 73

Q&A