KG 아이티뱅크

파이썬

PYTHON

반복문 2. for

반복문(For)

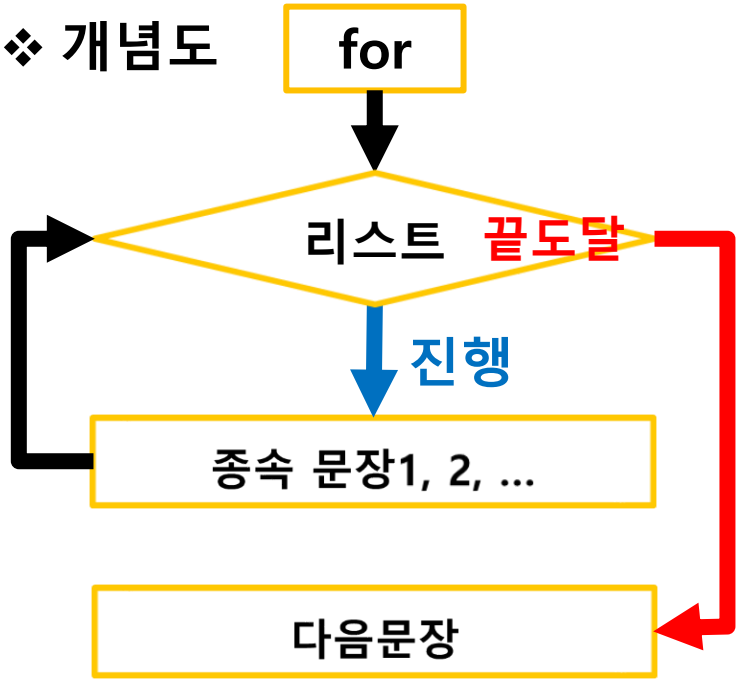
❖ 프로그램의 흐름을 제어하는 제어문 중의 세번째

- **규칙성 있게 반복되는 내용의 특정 횟수 실행**을 처리
- 주로 **리스트나 튜플, 딕셔너리 등으로** 반복횟수를 조절

❖ 코드

```
# 리스트, 튜플 등
lst=[1,2,3,4,5]
# in 연산자와 조합
for i in lst:
    # 리스트내의 값을 i에 저장
    print("%d "%i, end="")
print("반복종료")
```

❖ 개념도



반복문(For)

❖ **for 반복문은 반복횟수가 정해져 있는 반복문**

1) 특정 자료의 처리를 목적으로 사용 : 리스트 등의 값 사용

- 저장되어 있는 값의 처리를 목적으로 반복문을 구성
- 값의 검색 및 가공 등에 사용하기 위한 목적으로 작성

2) 지정횟수의 반복을 목적으로 사용 : range(정수) 를 활용

- 특정범위의 정수로 이루어져 있는 숫자 목록을 생성
- 내부의 **값의 개수만큼 반복**을 처리하는 것은 동일

❖ **코드1. 자료의 처리**

```
# 문자열로 구성된 리스트
lst=['1', '2', '3', '4']
int_lst=[]
for i in lst:
    # 형변환후 새 리스트에 추가
    int_lst.append(int(i))
```

❖ **코드2. 단순 반복**

```
# 특정 횟수만큼의 반복을 처리
for i in range(5):
    print("나는 전설이다!")
```

반복문(For)

< 파일이름 : forEX1.py >

❖ **실습예제1. 아래의 조건을 달성하는 코드를 작성하세요.**

조건

지정한 횟수만큼 아래의 내용을 출력하세요.
“Python is SIMPLE”

결과(입력 횟수 3로 가정)

```
--값의 입력--
횟수 입력 :

--반복 출력—
“Python is SIMPLE”
“Python is SIMPLE”
“Python is SIMPLE”
```

반복문(For)

< 파일이름 : forEX2.py >

❖ **실습예제2. 아래의 조건을 달성하는 코드를 작성하세요.**

조건

구구단을 출력하세요.
단, 출력할 단과 출력할 범위를 지정합니다.

결과(3 6을 입력했을 경우)

```
-- 값의 입력 --
단수, 범위 설정(띄어쓰기로 구분) :
구구단 3단
3 x 1 =  3
3 x 2 =  6
3 x 3 =  9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
```

반복문(For)

< 파일이름 : forEX3.py >

❖ **실습예제3. 아래의 조건을 달성하는 코드를 작성하세요.**

조건

지정한 횟수만큼 입력을 받아 리스트를 만드세요.
모두 문자열로 저장합니다.

결과(입력한 정수가 3일 경우)

```
입력횟수 지정 :
값1 입력 :
값2 입력 :
값3 입력 :
최종결과 : [(값1),(값2),(값3)]
```

반복문(For)

< 파일이름 : forEX4.py >

❖ **실습예제4. 아래의 조건을 달성하는 코드를 작성하세요.**

조건

3번문제에서 생성되는 리스트를 이용합니다.
동일한 값의 개수를 가지는 딕셔너리를 생성합니다.
단, 값은 새로 입력을 받아 생성합니다.

결과(입력한 정수가 3일 경우)

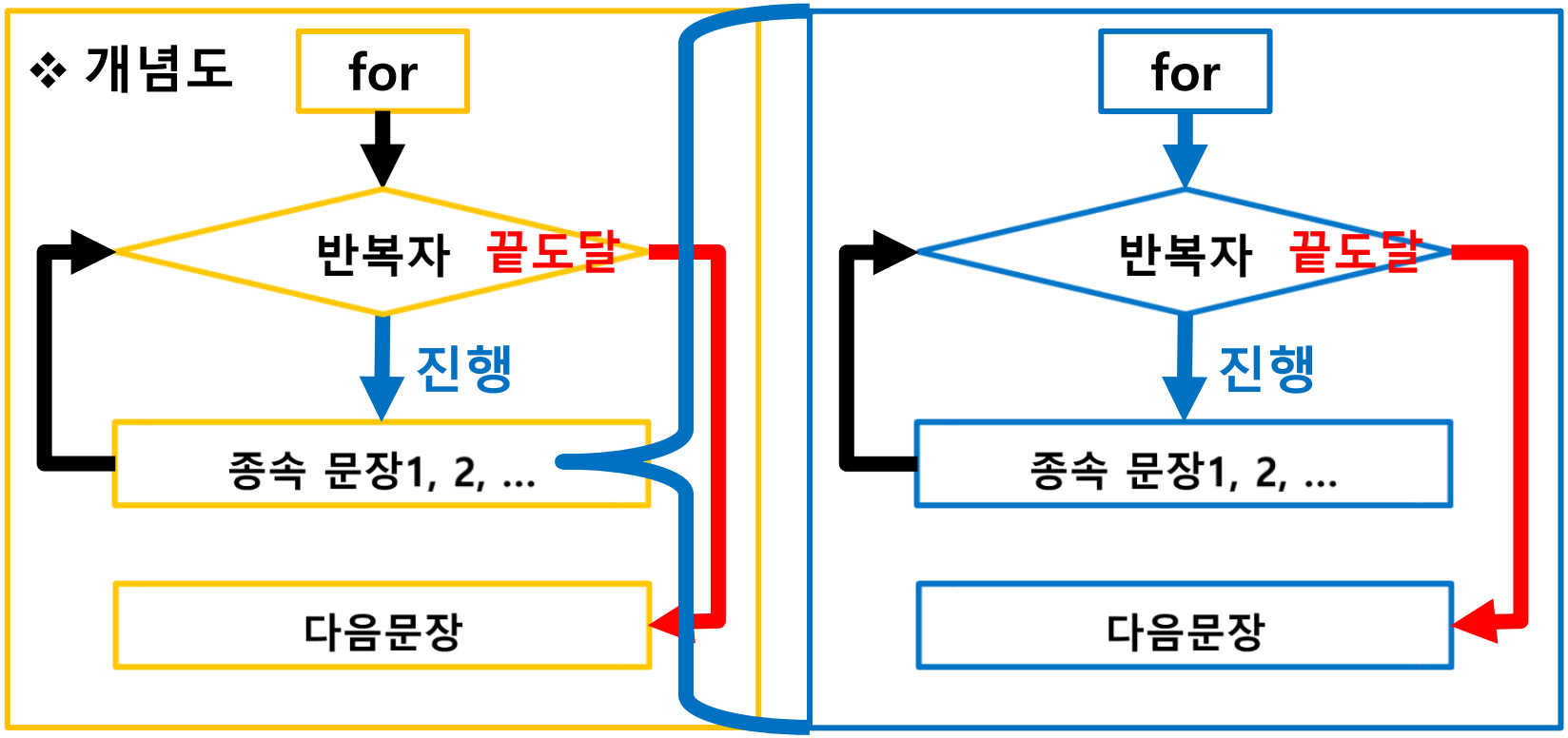
```
새로운 값1 입력 :
새로운 값2 입력 :
새로운 값3 입력 :
최종결과 : {(키1):(값1),(키2):(값2),(키3):(값3)}
```

다중 반복문

다중 반복문

❖ for 반복문의 종속문장으로 for 반복문이 들어간 코드

- **while로도 가능하나 작성난이도가 높음**
- 내부의 for 반복문의 반복이 다 끝나야 다음 코드를 진행
  - 동시반복, 병렬반복이 아닌 별개의 반복으로 처리됨



다중 반복문

❖ lst1=[['A', 'B'],['C', 'D'],['E', 'F']]일 때

```
for i in lst1:
    for j in i:
        print(j)
```

```
[ ['A', 'B'],
  ['C', 'D'],
  ['E', 'F'] ]
```

for i in lst1:

```
i = ['A', 'B']
i = ['C', 'D']
i = ['E', 'F']
```

for j in i:

```
j='A',
j='B'
j='C',
j='D'
j='E',
j='F'
```

print(j)

```
A
B
C
D
E
F
```

반복문(For)

❖ 다중 반복문도 목적에 따라 크게 두가지로 구분되어 사용

1) 내부의 자료를 다시 처리하기 위해 사용 : 2차원 리스트 등
   - ➢ 리스트 내부에 리스트, 딕셔너리 내부에 리스트 등
   - ➢ 내부에 포함된 반복가능객체의 수만큼 반복문을 구성

2) 내부에서 반복되는 내용의 반복 : range(정수) 를 활용
   - ➢ 반복되는 내용중 다시 반복되는 내용을 반복하는 것
   - ➢ 코드의 전체길이를 줄이며, 코드이 가독성도 향상됨

❖ 코드1. 자료의 처리

```
# 2차원 리스트
lst=[[1,2,3],[4,5,6]]
# [1,2,3]을 통째로 받아옴
for i in lst:
    # 내부의 정수값을 받음
    for j in i:
        print("%d "%j, end="")
```

❖ 코드2. 단순 반복

```
# 5회 반복을 지정
for i in range(5):
    # 1회당 5회 반복을 지정
    for j in range(5):
        print("ㅁ",end="")
    print()
```

반복문(For)

< 파일이름 : dforEX1.py >

❖ **실습예제1. 아래의 조건을 달성하는 코드를 작성하세요.**

조건

크기와 문자를 입력을 받아 큐브를 만드세요.

결과(크기 3, 문자 ㅁ으로 가정)

크기 입력 :
문자 지정 :

ㅁㅁㅁ
ㅁㅁㅁ
ㅁㅁㅁ

반복문(For)

< 파일이름 : dforEX2.py >

❖ **실습예제2. 아래의 조건을 달성하는 코드를 작성하세요.**

조건

지정한 단수의 계단을 만드세요.

결과(4를 입력했을 경우)

```
계단 단수 :
□
□ □
□ □ □
□ □ □ □
```

반복문(For)

< 파일이름 : dforEX3.py >

❖ **실습예제3. 아래의 조건을 달성하는 코드를 작성하세요.**

조건

구구단을 출력해보세요.

결과

```
1 x 1 =  1
…
3 x 6 = 18
…
9 x 9 = 81
```

반복문(For)

< 파일이름 : dforEX4.py >

❖ **실습예제4. 아래의 조건을 달성하는 코드를 작성하세요.**

조건

1부터 25까지의 숫자를 아래와 같이 출력합니다.

결과

```
 1  6 11 16 21
 2  7 12 17 22
 3  8 13 18 23
 4  9 14 19 24
 5 10 15 20 25
```