모두를 위한 파이썬 프로그래밍

7주 반복문

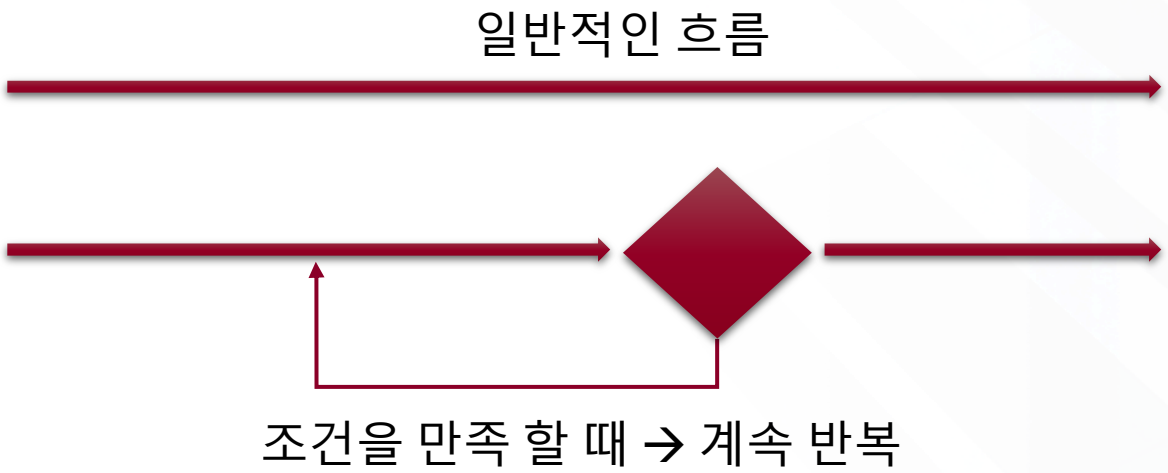
반복문이 필요성

- 지금까지 배운 명령문으로 구구단을 출력하는 프로그램을 작성해 봅시다.

```
print("1 X 1 = ", 1 * 1)
print("1 X 2 = ", 1 * 2)
print("1 X 3 = ", 1 * 3)
print("1 X 4 = ", 1 * 4)
print("1 X 5 = ", 1 * 5)
print("1 X 6 = ", 1 * 6)
print("1 X 7 = ", 1 * 7)
print("1 X 8 = ", 1 * 8)
print("1 X 9 = ", 1 * 9)

print("2 X 1 = ", 1 * 1)
print("2 X 2 = ", 1 * 2)
print("2 X 3 = ", 1 * 3)
print("2 X 4 = ", 1 * 4)
.
.
.
print("9 X 9 = ", 9 * 9)
```

반복문이 필요성

- **동일하거나 비슷한 행위를 반복적으로 수행할 간단한 방법이 필요**
- **컴퓨터를 활용하는 목적은 단순한 계산을 무한하게 반복시킬 때 유용함**
  - ✓ 1부터 100,000,000까지의 누적 합을 구하는 프로그램을 작성해야 한다고 생각해보자
  - ✓ 구구단을 1000단까지 출력하고자 한다면…



반복문의 개요

■ 프로그램 흐름의 반복을 위한 문법

✓ 반복문(loop) : 문장을 반복해 만드는 것으로, 정해진 동작을 반복적으로 수행할 때 내리는 명령어.

■ 반복문의 종류

✓ for 구문

- 특정 코드를 반복적으로 수행하기 위해 가장 많이 사용되는 반복문
- 특정 범위의 자료나 객체에 대해 처음부터 끝까지 하나씩 추출하면서 반복
- 반복할 횟수가 대체로 정해져 있음

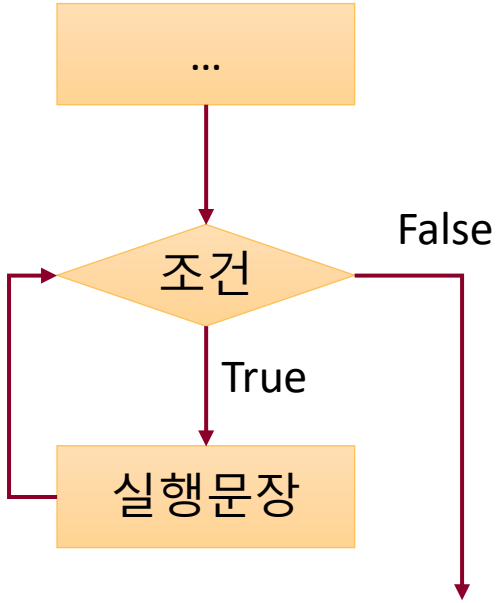
✓ while 구문

- 논리적인 조건을 만족할 때 계속 반복
- 반복할 횟수가 정해지지 않은 경우에 활용

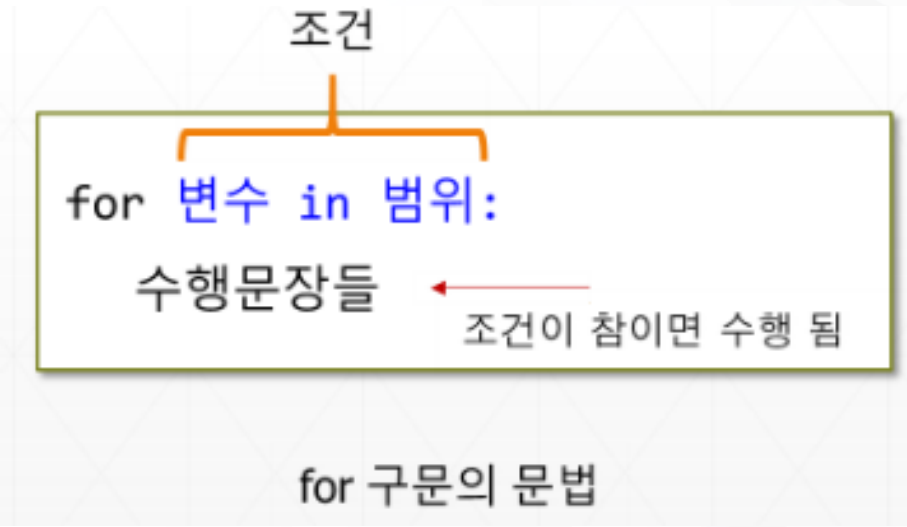
반복문: for

■ 가장 기본적인 반복문

✓ 조건식이 참이면 특정 구간의 코드를 반복해서 수행하는 반복문



기본형 for구문의 흐름도



for 구문의 문법

반복문: for

for 구문의 조건

- ✓ 값의 범위를 설정
- ✓ 현재 변수의 값이 특정 범위내에 속하는 지 평가

for 문의 범위

- ✓ 문자열
- ✓ 리스트 또는 튜플
- ✓ range()
- ✓ Dictionary
- ✓ 기타 반복 가능한 객체들

```
for 변수 in 범위:
    반복 수행할 문장 . . .
```

반복문: for

■ 문자열 기반의 반복문 사용

✓ for 변수 in 문자열:

    수행할 문장1

    수행할 문장2

    . . .

```
1    str = 'Korea'
2    for i in str:
3        print(i)
```

```
1    for i in 'Korea':
2        print(i)
```

```
K
o
r
e
a
```

■ 의미

✓ (좌) str이라는 문자열 변수에 저장되어 있는 문자의 수만큼 반복해서, 문자를 출력하라

✓ (우) 문자열의 개수만큼 반복해서, 문자를 출력하라

반복문: for

리스트 기반의 반복문 사용

✓ for 변수 in 리스트:

    수행할 문장1

    수행할 문장2

    . . .

```
1  for i in [2,1,3,4,5]: # 리스트: 순서가 있는 값들의 모임
2      print(i)
```

```
2
1
3
4
5
```

의미

- 리스트에 있는 요소의 개수만큼 반복됨
- 리스트의 값만큼 변수에 대입되어 출력된다.

반복문: for

■ 리스트 기반의 반복문 사용

✓ for 변수 in 리스트:

    수행할 문장1

    수행할 문장2

    . . .

```
1    for i in ['a', 'b', 1, 2, 'korea']:
2        print(i)
```

```
a
b
1
2
korea
```

■ 의미

✓ 문자열로 이루어진 리스트의 값들도 사용 가능

✓ 정수, 문자열로 이루어진 혼합된 리스트의 값들도 사용 가능

반복문: for

■ 리스트 기반의 반복문 사용

✓ for 변수 in 리스트:

수행할 문장1

수행할 문장2

. . .

```
1    str = 'Korea'
2    for i in [1,2,3,4,5]: # 리스트: 순서가 있는 값들의 모임
3        print(str)
```

```
Korea
Korea
Korea
Korea
Korea
```

■ 의미: [값1, 값2, …, 값n]

- [1, 2, 3, 4, 5]라는 리스트에 있는 각각의 값을 하나씩 가져와서 i라는 변수에 할당하고 print함수를 리스트에 있는 요소의 개수만큼 반복실행한다.
- 최종적으로 [1, 2, 3, 4, 5]에서, 총 다섯 번의 반복이 일어나 'Korea'가 다섯 번 출력된다.

반복문: for

■ 리스트 기반의 반복문 사용

✓ for 변수 in 리스트:

    수행할 문장1

    수행할 문장2

    . . .

```
1    str = '*'
2    for i in [1, 2, 3, 4, 5]:
3        print(str * i)
```

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

■ 의미: [값1, 값2, …, 값n]

✓ 요소의 개수 = 5개 → 5번 반복

✓ '*' * i

반복문: for

- 만약 100번 반복해야 한다면… 리스트에 들어갈 요소는 100개가 되고, 코딩하기 불편하다
- range()기반의 for구문 활용1
  - ✓for 변수 in range(반복횟수):
    수행할 문장 1
    수행할 문장 2
    . . .

```
1    for i in range(100):
2        print(i)
```

- 의미
  - ✓ 반복횟수는 'range'라는 키워드를 사용한다.
  - ✓ range(100)은 0 ~ 99까지 100번 반복하게 된다.
  - ✓ print 함수에 의해 0 ~ 99까지 출력됨

반복문: for

range()기반의 for구문 활용2

for 변수 in range(시작 값, 마지막 값, 증가값):
    수행할 문장 1
    수행할 문장 2
    . . .

✓시작 값: 범위의 시작 값

✓마지막 값: 범위의 종료 값 -1(종료 값 자체는 포함되지 않음)

✓증가값: 시작 번호에서 마지막 번호로 이동하는 증가 단위

의미

✓시작 값부터 마지막 값까지 증가치 만큼 증가하며 수행하라

✓ range는 마지막 번호의 마지막 숫자 바로 앞까지 리스트를 생성
- 예) range(1, 5) → [1, 2, 3, 4]의 리스트를 생성
    range(0, 5) → [0, 1, 2, 3, 4]의 리스트를 생성

✓앞의 시작 번호와 증가값은 생략할 수 있으며, 생략했을 경우 시작값은 0을,증가값은 1로 간주됨

실습: 반복문

■ range()기반의 for 구문 사용

✓ 1부터 100까지 수 중에서 홀수의 합을 구하기

```
1    sum = 0
2    for i in range(1, 101, 2):
3        sum += i
4    print(sum)
```

✓ 구구단 2단을 출력하기

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

실습: 반복문

■ range()기반의 for 구문 사용

✓9단을 거꾸로 출력하기

```
1    for i in range(9,0,-1):
2        print('9 * %d = %d' % (i, 9 * i))
```

✓2단 ~ 9단까지 출력하기

```
1    for i in range(2, 10):
2        for j in range(1, 10):
3            print('%d * %d = %2d' % (i, j, i * j))
```

실습: 반복문

■ range()기반의 for 구문 사용

✓증가값을 생략할 수 있다. (default: 증가값 =1)

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

✓시작 값과 증가값을 생략할 수 있다. (default: 시작 값= 0, 증가값 =1)

```
1    for i in range(10):
2        print(i)
```

실습: 반복문

■ 사용자로 부터 1부터 99사이의 숫자를 입력 받고, 입력 받은 숫자에 해당하는 구구단을 출력하는 프로그램을 작성하시오

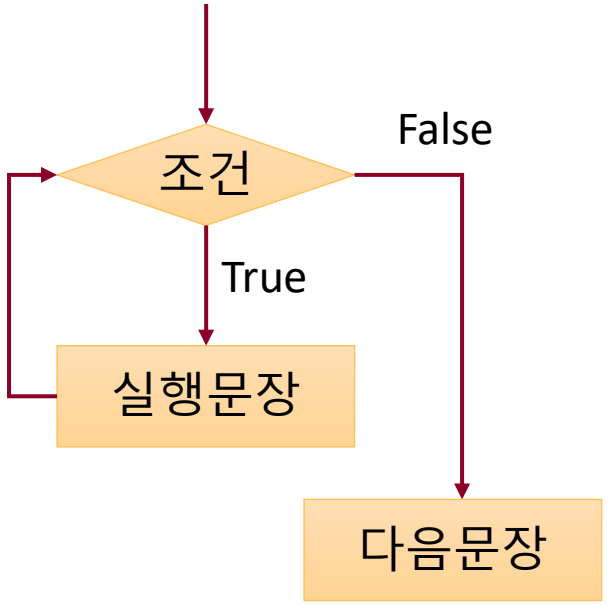
✓ for 구문의 리스트를 활용

```
출력 할 단을 입력하시오 : 88
88 * 1 =  88  88 * 9 = 792
88 * 2 = 176  88 * 8 = 704
88 * 3 = 264  88 * 7 = 616
88 * 4 = 352  88 * 6 = 528
88 * 5 = 440  88 * 5 = 440
88 * 6 = 528  88 * 4 = 352
88 * 7 = 616  88 * 3 = 264
88 * 8 = 704  88 * 2 = 176
88 * 9 = 792  88 * 1 =  88
```

반복문: while

■ 조건식이 참이면 특정 구간의 코드를 반복해서 수행하는 반복문

✓ 언제 조건식이 거짓이 되는가? → 참일 때만 반복, 참이 아닐때까지 반복



while문의 흐름도

```
while 조건:
    수행문장들
```

조건이 참이면 수행 됨

while문 구문

반복문: while

■ while 문 예제

- ✓ 반복 구간을 탈출 할 수 있는 로직이 별도로 존재해야 함
- ✓ 탈출 조건을 만족하는 식이 존재하지 않을 경우 무한 반복됨
- ✓ 0 ~ 9까지 순서대로 출력하기 위한 프로그램 예제

```
1    cnt = 0
2    while cnt < 10:
3        print(cnt)
4        cnt += 1
```

- ✓ 위 코드에서 4번 라인의 코드(cnt 변수의 변화)가 없다면...

반복문: while

- for, while 문의 비교

	For 구문	While 구문
조건식	범위(리스트, 문자열, range 등) 형태	직접 설정
종료조건	주어진 범위를 벗어난 경우	직접 설정
조건 값 갱신	자동 갱신	직접 설정
성격	계수 (counting)	논리 조건 비교

```
1  for i in range(10):
2      print(i)
```

```
1  cnt = 0
2  while cnt < 10:
3      print(cnt)
4      cnt += 1
```

반복문: while

for문과 while문 상호 변환 가능

✓for문과 while문은 차이점
- For문은 일반적으로 반복 횟수를 정확하게 알고 있고, 반복 횟수가 변하지 않을 때 사용 → 10번 반복
- while문은 반복 실행 횟수가 명확하지 않고 어떤 조건을 만족하면 프로그램을 종료하고자 할 때 사용
→ 특정 조건일 때만 반복

예) 인원이 고정된 학생들의 성적처리 → for
백화점 영업시간 중 방문한 고객처리 → while
가위바위보 게임에서 '이기면 종료하라.'하는 조건 → while문

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

반복 실행할 횟수가 명확한 경우

```
1    import random
2    i = num = 0
3    while num != 7:
4        num = random.randint(0, 100)
5        i += 1
6        print(i, num)
```

반복 실행 횟수가 유동적인 경우

반복문: while

- 예제) 1부터 10까지의 합을 계산하는 프로그램

```
1  sum = 0
2  num = 1
3  while num <= 10:
4      sum += num
5      num += 1
6  print('Sum : ', sum)
```

반복문: 중첩

- **if 문의 중첩과 같이, 반복문도 중첩이 가능함**
  - ✓ 반복문이 다른 반복문을 내포하는 구조
- **99단 출력하기**
  - ✓ 중첩된 for 문을 이용한 프로그램 작성
  - ✓2 ~ 9단은 첫번째 반복문으로, 1 ~ 9 는 중첩으로 구성
  - ✓print(‘’) # 한줄 띄우기

```
for i in range(2,10):
    for j in range(1,10):
        print('%d * %2d = %2d' % (i, j, i*j))
    print('')
```

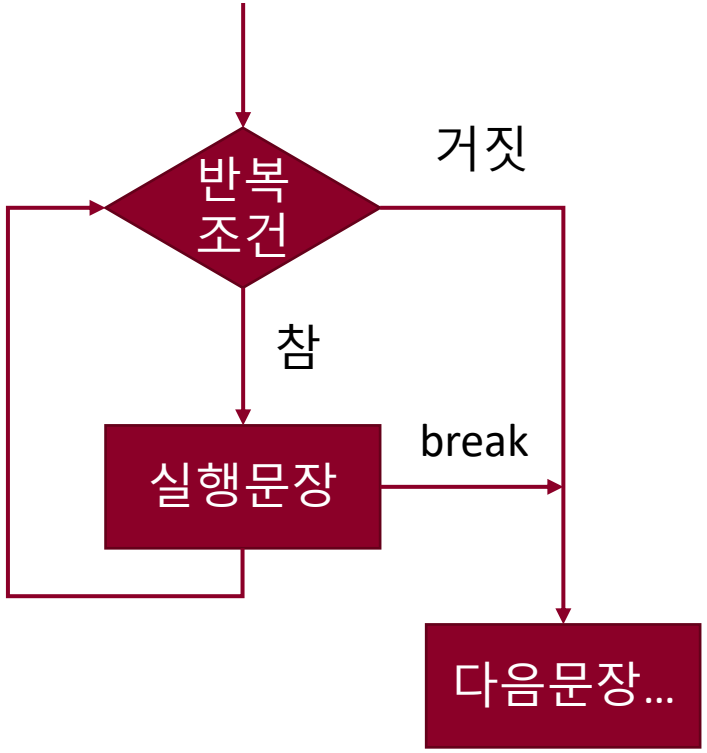
```
2 *  1 =  2
2 *  2 =  4
2 *  3 =  6
2 *  4 =  8
2 *  5 = 10
2 *  6 = 12
2 *  7 = 14
2 *  8 = 16
2 *  9 = 18

3 *  1 =  3
3 *  2 =  6
```

반복 제어: Break

break문

✓ 반복문에서 논리적으로 반복을 종료하는 방법으로 반복문의 흐름을 강제로 벗어날 수 있게(탈출) 해 주는 제어문

✓ 현재 소속되어 있는 반복문에 대해서만 적용됨




```
1  for i in [1,2,3,4,5]:
2      if i == 5: break
3      print(i)
4  i = 1
5  while i <= 5:
6      if i == 5: break
7      print(i)
8      i += 1
```

반복 제어: else문

else 문

✓ 반복 조건이 중단되지 않고 수행되었을 때 실행되는 문장(확인 역할)

```
1  for i in range(5):
2      print(i)
3  else:
4      print('반복문이 완료되었군요')
```

```
0
1
2
3
4
반복문이 완료되었군요
```

```
for i in range(5):
    if i >= 4:
        break
    else:
        print(i)
else:
    print('반복문이 완료되었군요')
```

```
0
1
2
3
```

정리

■ 반복문

- ✓ for 구문: 반복횟수 성격이 큰 반복문 → 몇번 수행하면 끝낼까?
- ✓ while 구문: 논리 조건 성격이 큰 반복문 → 어떠한 상황에 종료할까?

■ 중첩

- ✓ 중첩된 for/while문을 통하여 보다 복잡한 반복구문 수행 가능

■ 제어

- ✓ 현재 소속된 반복 루프에 대해서만 적용됨
- ✓ break: 반복문 탈출
- ✓ 제어구문의 사용은 예측하지 못한 동작이 일어날 수 있으므로 되도록 사용하지 않도록 한다.

Thank you