Python

반복문

Spring 2025

AI융합학과
Seongbok Baik
sbbaik@dju.kr

00 Text Book

교재명	으뜸 파이썬
저자	박동규, 강영민
출판사	생능출판사
발행년	2024.06.14

학습목표

- for 반복문을 정의하고 사용하는 방법을 익힌다.
- for in 구문과 리스트에 대해 이해하고 활용할 수 있다.
- 이중 for 루프에 대해 알아보고 활용한다.
- while 반복문을 정의하고 사용하는 방법을 익힌다.
- for 문을 이용하여 작성한 반복문을 while 문으로 변경할 수 있다.
- break와 continue를 이용하여 반복문을 제어하는 방법을 익힌다.

4.2 중첩 for 루프

중첩 for문nested for loop

- for문 안에 for문을 다시 넣음

- 구구단의 구조에 대해 살펴보면, 2~9단까지 있으며 1에서 9를 단마다 곱하여 화면에 출력함

- 이를 구현하기 위해서는 for 문 안에 for 문을 다시 넣는 **이중 for문**이 필요

4.2 중첩 for 루프

코드 4-19: 중첩 for 문을 사용한 구구단 출력

double_for.py

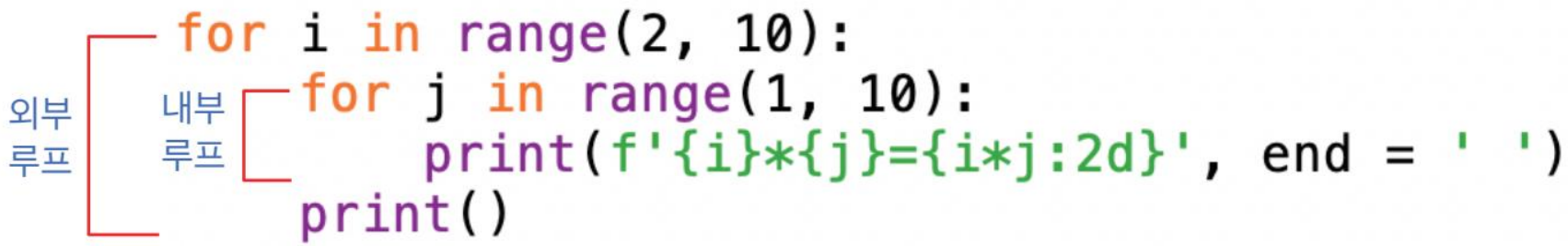
```
for i in range(2, 10): # 외부 for 루프
    for j in range(1, 10): # 내부 for 루프
        print(f'{i}*{j}={i*j:2d}', end = ' ')
    print()    # 내부 루프 수행 후 줄바꿈을 함
```

실행결과

```
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
```

4.2 중첩 for 루프

- double_for.py의 이중 for 루프는 내부 루프와 외부 루프를 가짐



[그림 4-6] 파이썬의 중첩 for 루프의 구조와 내부 루프, 외부 루프

4.2 중첩 for 루프

- 루프의 실행구조를 표로 나타내어 보기

i = 2일 때	i = 3일 때	i = 4일 때	…	i = 9일 때
j = 1 : 2 * 1	j = 1 : 3 * 1	j = 1 : 4 * 1	…	j = 1 : 9 * 1
j = 2 : 2 * 2	j = 2 : 3 * 2	j = 2 : 4 * 2	…	j = 2 : 9 * 2
j = 3 : 2 * 3	j = 3 : 3 * 3	j = 3 : 4 * 3	…	j = 3 : 9 * 3
j = 4 : 2 * 4	j = 4 : 3 * 4	j = 4 : 4 * 4	…	j = 4 : 9 * 4
j = 5 : 2 * 5	j = 5 : 3 * 5	j = 5 : 4 * 5	…	j = 5 : 9 * 5
j = 6 : 2 * 6	j = 6 : 3 * 6	j = 6 : 4 * 6	…	j = 6 : 9 * 6
j = 7 : 2 * 7	j = 7 : 3 * 7	j = 7 : 4 * 7		j = 7 : 9 * 7
j = 8 : 2 * 8	j = 8 : 3 * 8	j = 8 : 4 * 8		j = 8 : 9 * 8
j = 9 : 2 * 9	j = 9 : 3 * 9	j = 9 : 4 * 9		j = 9 : 9 * 9

4.2 중첩 for 루프

- 이중 for 루프나 삼중 for 루프의 경우 코드를 이해하는 것이 어려워지기 때문에 중첩 루프는 삼중 루프 이상의 구조를 잘 사용하지 않음

4.2 중첩 for 루프

4.2.1 중첩 for 루프를 이용한 패턴 만들기

- 다음과 같은 패턴 만들어 보기

```
#
 #
  #
   #
    #
     #
      #
       #
```

4.2 중첩 for 루프

- 4.2.1 중첩 for 루프를 이용한 패턴 만들기

코드 4-20: 이중 for 루프를 사용한 패턴 생성하기
double_for_pattern.py

```
n = 7
# 외부 for 루프는 n번 수행, i는 0에서 6까지 증가
for i in range(n):
    st = ''
    for j in range(i): # 내부 for 루프는 i번 수행
        st = st + ' ' # 공백을 i개 추가함
    print(st + '#') # 공백 추가 후 '#' 출력
```

- 이중 for 루프를 사용하여 외부 루프에서는 i 값이 0에서 1씩 증가하도록 하고 내부 루프에서는 i의 개수만큼의 공백을 '#' 표시 앞에 추가

코드 4-21: for 루프와 *를 사용한 패턴 생성하기
for_pattern.py

```
n = 7
# 외부 for 루프는 n번 수행, i는 0, 1, 2, 3, 4, 5, 6까지 증가
for i in range(n):
    print(' ' * i + '#') # 공백을 i번 추가한 후 '#' 출력
```

- 이중 for 루프를 사용하지 않고 다음과 같이 print문 내에 공백의 출력횟수를 지정하는 방식(' ' * i)
- for_pattern.py의 방식이 더 이해하기 쉽고 간단함

4.2 중첩 for 루프

- 4.2.1 중첩 for 루프를 이용한 패턴 만들기

LAB 4-4: 패턴 출력 응용

1. 이중 for 루프를 이용하여 다음과 같은 패턴을 출력하여라.

```
      #
     #
    #
   #
  #
 #
#
```

4.2 중첩 for 루프

- 4.2.2 상향식 문제풀이 기법
- 이중 for 루프를 사용하여 다음과 같은 패턴 만들어보기

```
    +
   +++
  +++++
 +++++++
+++++++++
```

4.2 중첩 for 루프

- 4.2.2 상향식 문제풀이 기법
- 4, 3, 2, 1, 0을 순서대로 출력하는 프로그램

코드 4-22: 4, 3, 2, 1, 0을 순서대로 출력하는 프로그램

pattern_test1.py

```
n = 5
for i in range(n): # i는 0, 1, 2, 3, 4까지 증가
    # n이 5이므로 n-(i+1)은 4, 3, 2, 1, 0이 됨
    print(n - (i + 1), end = ' ')
```

실행결과

```
4 3 2 1 0
```

- 4.2.2 상향식 문제풀이 기법
  - '+' 문자의 증가 패턴인 1, 3, 5, 7, 9를 출력

코드 4-23: 1, 3, 5, 7, 9를 순서대로 출력하는 프로그램

pattern_test2.py

```
n = 5
for i in range(n): # i는 0, 1, 2, 3, 4까지 증가
    print(2 * i + 1, end = ' ') # 2 * i + 1은 1, 3, 5, 7, 9가 됨
```

실행결과

```
1 3 5 7 9
```

4.2.2 상향식 문제풀이 기법

- 앞서 완성한 두 가지 기능을 바탕으로 삼각형 패턴을 출력해보기
- 패턴을 출력하기 위해 우선 공백을 for j in range(n - i - 1):를 이용하여 출력한 다음, '+' 패턴을 for j in range(2 * i + 1):를 이용하여 출력

- 이와 같은 기법을 **상향식**Bottom up 문제풀이 기법이라고 함
  - 작은 기능을 구현하여 이 기능을 바탕으로 전체 기능을 구현하는 것

4.2 중첩 for 루프

- 삼각형 패턴을 출력하는 기능

코드 4-24: 삼각형 패턴을 출력하는 기능

triangle_pattern1.py

```
n = 5
for i in range(n):
    for j in range(n - (i + 1)): # 공백을 출력함
        print(' ', end = '')
    for j in range(2 * i + 1):    # '+'를 출력함
        print('+', end = '')
    print()
```

4.2 중첩 for 루프

NOTE: Think big but start small.

문제 해결을 위한 프로그래밍 작성 시에는 항상 `pattern_test1.py`, `pattern_test2.py`와 같은 작은 기능을 수행하는 프로그램을 만들어 보고 이 기능이 제대로 잘 동작하는가를 충분히 테스트하여야 한다. **작은 기능조차 제대로 동작하지 않는다면 큰 문제는 절대로 해결되지 않는다**. 처음에 프로그램을 시작하는 개발자들은 자신이 만들 목표만 크게 생각하고 작은 기능들을 어떻게 구현해야 하는가를 모르는 경우가 많다. 이렇게 막연히 목표만 보는 것이 아니라 자신이 구현하고자 하는 기능을 작은 단위로 쪼개고 구현하여 구현이 가능한지 검토하는 방법이 바람직하다.

4.2 중첩 for 루프

- 삼각형 패턴을 출력하는 기능을 가진 짧은 코드

코드 4-25: 삼각형 패턴을 출력하는 기능을 가진 짧은 코드

triangle_pattern2.py

```
n = 5
for i in range(n):
    print(' ' * (n - (i + 1)), end = '')
    print('+' * (2 * i + 1))
```

4.2.3 소수 구하기

- 이중 for문을 활용해 소수를 구하는 프로그램을 구현
- 소수란 1과 자기 자신 이외의 약수를 가지지 않는 수
- 다음과 같은 코드로 구현할 수 있다

```
n = int(input('수를 입력하세요 :'))
is_prime = True
for num in range(2, n):   # 2부터 (n-1) 사이의 수 num에 대하여
    if n % num == 0:      # 이 수 중에서 n의 약수가 있으면
        is_prime = False  # 소수가 아님
print(n,'is prime :', is_prime)
```

4.2 중첩 for 루프

- 2부터 100까지의 소수 구하기

코드 4-26: 2부터 100까지의 소수 구하기

get_primes.py

```
# 소수를 담을 리스트 초기화
primes = []

for n in range(2, 101):
    # 일단 n을 소수라고 두자.
    is_prime = True
    for num in range(2, n):    # 2~(n-1) 사이의 수 num에 대하여
        if n % num == 0:       # 이 수 중에서 n의 약수가 있으면
            is_prime = False   # 소수가 아님

    if is_prime: # 소수일 경우 primes라는 리스트에 추가한다
        primes.append(n) # append() 메소드는 리스트에 n을 추가함

print(primes)
```

실행결과

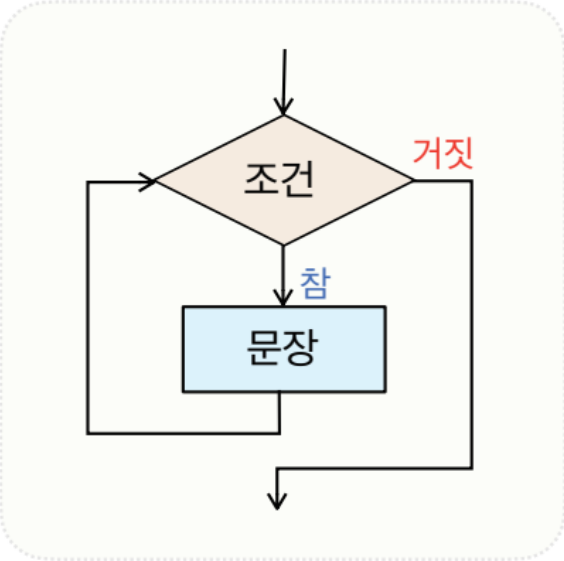
```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89, 97]
```

4.3 while 반복문

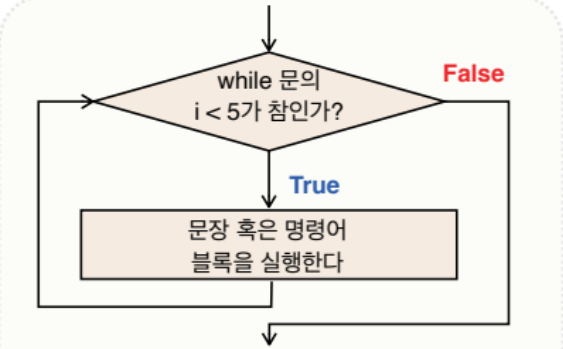
4.3.1 while 반복문의 구조

- 조건이 참인 경우에 계속 실행하는 반복문

while 문의 제어 흐름



while 문과 그 예시



예시

```
i = 0
while i < 5:
    print('환영합니다')
    i += 1
```

4.3 while 반복문

while 문의 문법

- if 문과 매우 유사
- 조건식이 참이라면 계속 반복하여 해당 코드를 실행

[표 4-3] while 문의 형식과 예시

형식	예시

초기 값 지정
while 조건식 :
　　실행할 코드 블록
	`i = 0 # 초기 값 지정`
	`while i < 5 : # 조건식`
	`    print('Welcome to everyone!!')`
	`    i += 1`

4.3 while 반복문

while 문의 문법

코드 4-27: for 문을 이용한 'Welcome to everyone!!'의 반복 출력 기능
print_welcome_with_for.py

```
for i in range(5):
    print('Welcome to everyone!!')
```

코드 4-28: while 문을 이용한 'Welcome to everyone!!'의 반복 출력 기능
print_welcome_with_while.py

```
i = 0          # 초기 값
while i < 5:   # 루프의 조건식이 참이면 내부 블록이 실행됨
    print('Welcome to everyone!!')
    i += 1     # 조건 값의 변경
```

실행결과

```
Welcome to everyone!!
Welcome to everyone!!
Welcome to everyone!!
Welcome to everyone!!
Welcome to everyone!!
```

- i를 0부터 1씩 증가시키며 5보다 작을 때까지 while 내부의 코드를 반복함
- 다음과 같이 새로 작성 가능

4.3 while 반복문

while 문의 문법

코드 4-29: 지정된 수까지의 누적 합을 구하는 기능

while_sum_input.py

```
n = int(input('합계를 구할 수를 입력하세요 : '))
s = 0
i = 1
while i <= n:
    s = s + i
    i += 1
print('1부터 {}까지의 합은 {}'.format(n, s))
```

실행결과

```
합계를 구할 수를 입력하세요 : 100
1부터 100까지의 합은 5050
```

- 반복실행의 횟수가 명확한 경우는 while 문의 코드가 길어지기 때문에 for 문을 사용하는 것이 더 나은 방법

4.3 while 반복문

while 문과 for 문 비교

- while 문은 수행횟수를 정확히 모르지만 수행의 조건이 명확한 경우에 더 적합
- 반복 횟수가 명확한 경우 for 문이 적합

1에서 n까지의 합을 구하는 코드 비교	
while 문	for 문

`s = 0`
`i = 1`
`while i <= n:`
`    s = s + i`
`    i += 1`
	`s = 0`
	`for i in range(1, n+1) :`
	`    s = s + i`

4.3.2 while 반복문과 입력조건

- 사용자로부터 '가위', '바위', '보'를 입력으로 받아서 이 값을 출력하는 게임 프로그램 만들기

4.3.2 while 반복문과 입력조건

코드 4-30: while 반복문을 이용한 가위, 바위, 보 선택하기
rsp_input.py

```
selected = None
while selected not in ['가위', '바위', '보']:
    selected = input('가위, 바위, 보 중에서 선택하세요> ')
print('선택한 값은:', selected)
```

실행결과

```
가위, 바위, 보 중에서 선택하세요> 묵
가위, 바위, 보 중에서 선택하세요> 찌
가위, 바위, 보 중에서 선택하세요> 빠
가위, 바위, 보 중에서 선택하세요> 가위
선택한 값은: 가위
```

- 원하는 값 입력 시 실행되도록 하는 반복문에 적합함

4.3.2 while 반복문과 입력조건

코드 4-31: 양수 n을 입력 받아 1부터 n까지의 합을 구하는 코드
for_sum_input2.py

```
n = int(input('합계를 구할 양의 정수를 입력하세요 : '))
s = 0
for i in range(1, n+1) :
    s = s + i
print('1부터 {}까지의 합은 {}'.format(n, s))
```

실행결과

```
합계를 구할 양의 정수를 입력하세요 : -10
1부터 -10까지의 합은 0
```

- 입력 값의 범위를 양의 자연수로 한정하려고 할 경우에도 while 문을 사용하는 것이 적합

4.3.2 while 반복문과 입력조건

코드 4-32: 1부터 n까지의 합을 구하는 코드로 while 문 내에서 입력문 사용
for_sum_input2_1.py

```
n = -1
while n <= 0: # 양수가 입력될 때까지 input() 문을 반복 수행함
    n = int(input('합계를 구할 양의 정수를 입력하세요 : '))
s = 0
for i in range(1, n+1):
    s = s + i
print('1부터 {}까지의 합은 {}'.format(n, s))
```

실행결과

```
합계를 구할 양의 정수를 입력하세요 : -10
합계를 구할 양의 정수를 입력하세요 : 0
합계를 구할 양의 정수를 입력하세요 : 10
1부터 10까지의 합은 55
```

- n <= 0이라는 조건을 만족할 경우 재입력을 받는 부분이 핵심

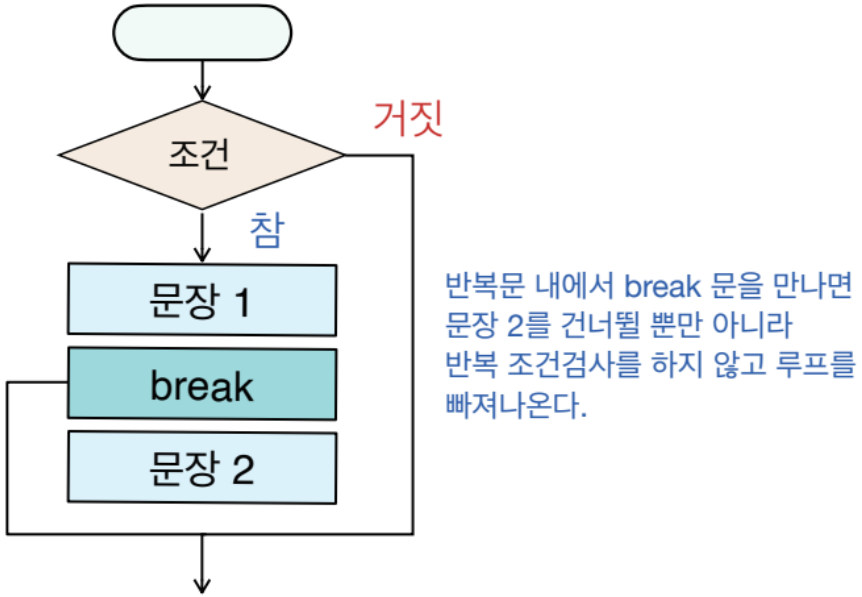
4.4 break와 continue

- 반복문을 제어하는 키워드
  - 반복 실행을 종료 -> break
  - 반복문 루프 내의 나머지 실행부를 건너뛰고 계속해서 반복 루프를 실행 -> continue
  - continue는 반복 실행을 종료하지 않음

4.4 break와 continue

break를 표현한 흐름도

- while이나 반복문은 조건이 참이면 블록내의 문장을 수행
- 도중에서 break를 만나면 그 즉시 반복 실행을 종료하고 루프를 빠져나옴



[그림 4-8] break 문의 흐름도

break를 표현한 흐름도

코드 4-33: break를 사용하여 모음이 나타나면 즉시 반복문을 종료하는 기능
skip_vowel_break.py

```
st = 'Programming'     # 자음이 나타나는 동안만 출력하는 기능
for ch in st:
    if ch in ['a','e','i','o','u']:
        break          # 모음일 경우 반복문을 종료한다.
    print(ch)
print('The end')
```

실행결과

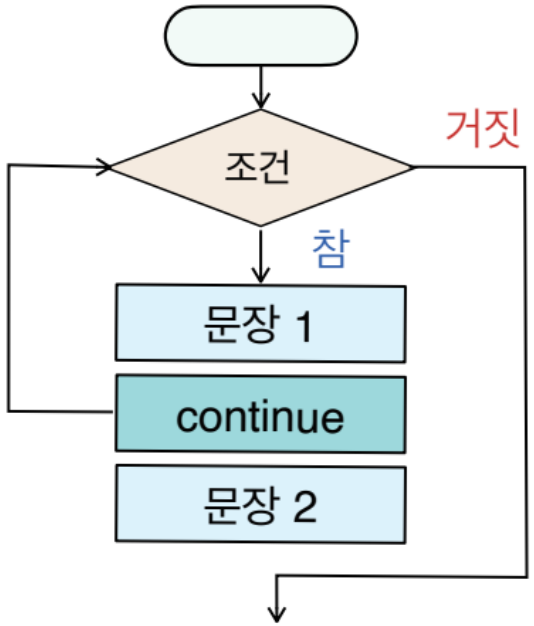
```
P
r
The end
```

- 모음에 해당하면 break
- 그렇지 않으면 print를 이용하여 출력
- break를 작동시키면 반복문의 나머지 부분을 실행하지 않고 루프를 중지시킴

4.4 break와 continue

continue를 표현한 흐름도

- 루프를 빠져나오지 않고 continue 아래의 문장만을 건너뛰는 역할
- 반복문이 종료되는 것은 조건이 거짓일 때에만 해당



반복문 내에서 continue 문을 만나면
문장 2를 건너뛰어
반복 조건 검사를 한다
조건이 참이면 루프를 실행하고
조건이 거짓일때 루프를 빠져나온다.

[그림 4-9] continue 문의 흐름도

4.4 break와 continue

continue를 표현한 흐름도

코드 4-34: continue를 사용하여 모음일 경우 출력을 건너뛰는 기능

skip_vowel_continue.py

```
st = 'Programming' # 자음이 나타날 때만 출력하는 기능
for ch in st:
    if ch in ['a','e','i','o','u']:
        continue # 모음일 경우 아래 출력을 건너뛴다.
    print(ch)

print('The end')
```

실행결과

```
P
r
g
r
m
m
n
g
The end
```

- continue를 넣게 되면 아래에 있는 아래의 나머지 부분,
  즉 print를 실행하지 않고 반복문의 처음으로 돌아가는 기능을 함

4.4 break와 continue

주의 - break와 continue 흐름 제어의 위험성

break와 continue는 프로그램의 제어를 효율적으로 하는 데 편리하게 사용할 수 있다. 하지만 break와 continue 문이 너무 많이 사용되는 경우 제어의 흐름에 일관성이 없어 프로그램을 이해하는 것이 어려워진다. 따라서 continue와 break는 필요한 경우에만 제한적으로 사용하는 것이 좋다.

Leistung ist nicht alles / Keinen Studierenden zurücklassen

DAEJEON UNIVERSITY
SINCE 1980
대전대학교