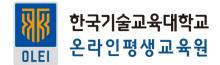


# 파이썬의 반복문



#### 학습목표

- 불 제어문 중 for 반복문에 대해 알고, 기본적인 코딩을 할 수 있다.
- while 반복문에 대해 알고, 조건에 따라 프로그램의 실행을 반복할 수 있다.
- 리스트와 반복문을 함께 사용하는 <mark>리스트 내포에</mark> 대해 알고, 코딩을 할 수 있다.

#### 학습내용

- for 반복문
- while 반복문
- 리스트 내포

# 1) for 반<del>복문</del>의 정의

#### 파이썬의 제어문 중 하나로 <u>프로그램의 실행을 반복할</u> 수 있음

- 조건문과 마찬가지로 들여쓰기와 콜론이 매우 중요!
- 반복 가능한 객체를 순회하며 반복문 안의 코드를 한번씩 실행



for 아이템 in 반복 가능한 객체: 실행 코드

- \*
- ① 반복 객체에서 순서대로 하나씩 값을 가져옴
- ② 아이템에 가져온 값을 담음
- ③ 실행 코드 수행
- ④ 반복 객체가 끝날 때까지 순차적으로 반복

# 1) for 반<del>복문</del>의 정의



리스트, 튜플 등 집합 자료형을 사용하여 요소를 하나씩 가져와 반복 가능



반복 가능 객체인 문자열, 사전, 집합 자료형으로도 반복문 사용 가능

```
for i in {1,2,3}:
    print(i)

1
2
3
```

```
for i in [1,2,3]:
    print(i)

1
2
3
```

```
for i in "Hello":
    print(i)

H
e
l
l
o
```

- 2 반복 객체 및 range 함수
  - 1 반복 객체 함수

#### Iterable 객체란 뜻으로 객체의 멤버를 하나씩 반화할 수 있음

- 대표적으로 시퀀스 데이터 타입인 리스트(List), 튜플(Tuple), 스트링(String) 등이 있음
- collections.lterable에 속한 인스턴스(Instance)인지 아닌지로 반복 가능한 객체를 판별할 수 있음

```
import collections
obj = [1, 2, 3, 4]
isinstance(obj, collections.Iterable)
```

```
import collections
obj = "Hello, World!"
isinstance(obj, collections.Iterable)
```

True

True

- 2 반복 객체 및 range 함수
  - 2 range() 함수

#### 반복 가능한 객체를 반환해주는 함수

- range(시작(이상), 끝(미만), 스텝)로 정의
- 반복문에 많이 활용
- 순차적으로 값을 반환하기 때문에 리스트 등의 다른 자료형과 함께 사용할 수 있음

```
print(range(10))
```

range(0, 10)

```
1 = list(range(10))
print(1)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
import collections
obj = range(10)
isinstance(obj, collections.Iterable)
True
```

# 3 반복문 활용



반복 객체의 값을 바로 가져와서 사용하거나, range(), len() 함수를 활용해 인덱스로 접근 또한 가능

```
l = ['a',9,[1,2,3],('a','b')]
for i in l :
    print(i)

a
9
[1, 2, 3]
('a', 'b')
```

```
l = ['a',9,[1,2,3],('a','b')]
for i in range(len(l)):
    print(l[i])

a
9
[1, 2, 3]
('a', 'b')
```

# 3 반복문 활용



#### 제어문을 여러 개 사용할 경우, 들여쓰기 주의!

```
for i in [1,2,3]:
    for j in (10,20):
        print(j)
    print(i)
10
20
1
10
20
2
10
20
2
10
3
```

```
for i in [1,2,3]:
    for j in (10,20):
         print(j)
         print(i)
10
1
20
1
10
2
20
2
10
3
20
3
```





요소 값을 아이템 변수로 할당하여 응용 가능

```
l = [[1,2],[3,4],[5,6]]
for i in 1:
    print(i)

[1, 2]
[3, 4]
[5, 6]
```

```
l = [[1,2],[3,4],[5,6]]
for i,j in 1:
    print(i+j)
3
7
11
```





조건문과 함께 사용해 복잡한 프로그램 작성 가능



continue를 활용해 한 차례 반복을 건너뛸 수 있음

```
s = "Hello"
cnt = 0
for i in s:
    if(i == 'e'):
        print("i is e")
    cnt +=1
print(cnt)

i is e
5
```

```
s = "Hello"
cnt = 0
for i in s:
    if(i == 'e'):
        print("i is e")
        continue
    cnt +=1
print(cnt)

i is e
4
```

# while 반복문

# 1) while 반<del>복문</del>의 정의

#### 파이썬의 제어문 중 하나로 프로그램의 실행을 반복할 수 있음

• while (조건):

실행 코드

- 콜론, 들여쓰기 주의!
- 무한루프에 빠지지 않도록 조건 설정에 주의!

```
In [*]: while(True):
    print(1)

1
    1
    1
    1
    1
    1
    1
    1
    1
    1
```

```
num = 5
while (num > 0):
    print(num)
    num -= 1

5
4
3
2
1
```

## while 반복문

# 2 while 반복문의 활용



조건문과 break 보조 제어문을 활용해 특별한 조건에서 반복문을 완전히 빠져나올 수 있음



break를 만나는 순간 제어문을 빠져나가기 때문에 코드 작성에 유의

```
num = 10
while (num > 0):
    if(num == 6):
        print("--end--")
        break
    print(num)
    num -= 1
10
9
8
7
--end--
```

## while 반복문

# 2 while 반<del>복문</del>의 활용



조건문과 continue 보조 제어문을 활용해 특별한 조건에 해당 반복을 건너뛸 수 있음



continue를 만나는 순간 해당 반복을 건너뛰기 때문에 코드 작성에 유의

```
num = 10
while (num > 0):
    print(num, end =', ')
    num -= 1
    if(num == 6):
        continue

10, 9, 8, 7, 6, 5, 4, 3, 2, 1,
```

## 리스트 내포



#### 리스트를 만드는 반복문을 짧고 간결하게 만들어 줌

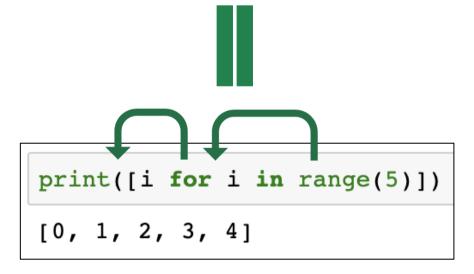
• [식 for 변수 in 반복 가능한 객체]로 정의

```
l = []

for i in range(5):
#파이썬의 반복문

l.append(i)
print(l)

[0, 1, 2, 3, 4]
```



## 리스트 내포





#### 조건문을 함께 사용할 수 있음

■ [식 for 변수 in 반복 가능한 객체 if 조건]로 정의

```
1 = []
for i in range(5):
    if(i%2 == 0):
        l.append(i)
print(1)

[0, 2, 4]
```



```
print([i for i in range(5) if i%2 == 0])
[0, 2, 4]
```

## 리스트 내포





반복문을 여러 개 사용할 수 있음(중첩)

 두 개 이상의 제어문을 사용할 경우 코드의 가독성이 떨어져 오히려 불편할 수 있음

```
l = []
for i in range(3):
    for j in range(3):
        l.append(i+j)
print(1)

[0, 1, 2, 1, 2, 3, 2, 3, 4]
```



```
print([i+j for i in range(3) for j in range(3)])
[0, 1, 2, 1, 2, 3, 2, 3, 4]
```

## Run! 프로그래밍

# Mission 1

#### 반복문 활용

num = input("출력할 단 수를 입력해주세요.")

#### 정답

```
for i in range(1,10):

print("{} X {} = {}".format(num,i,int(num)*i))
```

## Mission 2

#### 리스트 내포 활용

#### 정답

print([i\*j for i in range(2,10) for j in range(1,10)
if i !=4])

# 학습정리

# 1. for 반복문

for 반복문의 정의	• 반복 가능한 객체를 순회하며 프로그램 실행을 반복하는 제어문
반복 객체 및 range 함수	<ul> <li>반복 가능한 객체는 리스트, 튜플, 문자열 등이 있음</li> <li>반복이 가능하지 않은 객체는 반복문에 활용할 수 없음</li> </ul>
반복문 활용	<ul> <li>조건문 등과 함께 사용할 수 있음</li> <li>두 개 이상의 반복문을 함께 사용해 복잡한 프로그래밍을 할 수 있음</li> </ul>

# 2. while 반복문

while 반복문의 정의	<ul> <li>조건을 만족할 경우 프로그램을 반복 실행하는 제어문</li> <li>조건을 만족하지 않을 때까지 반복되며, 무한 루프에 빠질 수 있음</li> </ul>
while 반복문의 활용	<ul> <li>break문으로 반복문 전체를 완전히 빠져나올 수 있음</li> <li>continue문으로 해당 반복을 건너뛸 수 있음</li> </ul>

# 학습정리

## 3. 리스트 내포

2	스	트
	포	
7	성으	-1

• 리스트를 만들어 주는 반복문을 짧고 간결하게 만들어 줌

#### 리스트 내포의 활용

• 여러 개의 for문을 사용할 수 있고, 조건문 또한 추가할 수 있음