

파이썬 프로그래밍

---

# 제어문과 함수 기초



한국기술교육대학교  
온라인평생교육원

## ■ 파이썬 제어문

### 1. 들여쓰기와 제어문

- 파이썬은 들여쓰기를 강제하여 코드의 가독성을 높인다.
- 가장 바깥쪽의 코드는 반드시 1열에 시작한다.
- 블록 내부에 있는 statement들은 동일한 열에 위치해야 한다.
- 블록의 끝은 들여쓰기가 끝나는 부분으로 간주된다.
  - python에는 {, }, begin, end 등의 키워드가 존재하지 않는다.
- 들여쓰기를 할 때에는 탭과 공백을 섞어 쓰지 않는다.

- 들여쓰기를 강제하여 코드의 가독성 높임

```
a = 1 # 성공
  a = 1 # 실패
```

```
File "<ipython-input-160-0453a5e2fe16>", line 2
  a = 1 # 실패 ^
```

IndentationError: unexpected indent

- a와 b가 같은 column(컬럼) 에 존재 X → 에러가 발생
- a = 1과 b = 2는 같은 블록이 존재하는 코드
- 같은 블록이 존재하면 반드시 들여쓰기가 일치해야 함

## ■ 파이썬 제어문

### 1. 들여쓰기와 제어문

```
if a > 1:
    print 'big'
    print 'really?'
```

```
File "<ipython-input-161-96672a766929>", line 3
    print 'really?' ^ I
```

IndentationError: unexpected indent

- 첫번째 라인은 실행하는 함수문의 헤더 역할 수행
- 밑 라인은 함수문의 몸체 역할 수행
- 몸체: 새로운 블록이 시작되는 것으로 새 블록에서 코딩
- 메인 블록과 다르게 새 블록은 반드시 들여쓰기 해야 함
- : 을 넣어 새블록의 시작을 알림
- : 이 아래의 새 블록은 반드시 들여쓰기 필요
- 일반적 메인블록은 반드시 1열에서 시작
- 헤더의 마지막에는 ;(콜론)으로 마무리
- ; 뒤쪽에서 엔터키를 누르면 들여쓰기로 들어감
- Tab키를 누르면 동일한 컬럼에서 제어문의 몸체가 시작
- f=10 까지가 if절의 블록 = if문의 몸체는 한줄
- g=10은 바깥쪽에 있는 if절의 블록
- backspace나 delete 키를 눌러 뒤쪽으로 이동
- a=1, b=2, if a<10;, 과 x=10은 같은 레벨 → statement가 총 4개
- 들여쓰기가 들쭉날쭉하면 파이썬에서는 '문법적 에러'로 간주
- 들여쓰기가 강제화(문법 안에 들여쓰기가 포함됨)
- '들여쓰기'는 개발자가 가지고 가야 할 중요한 원칙
- 들여쓰기를 하면 → 깔끔하고 이해하기 쉬운 코딩 가능
- 들여쓰기르 올바르게 하지 않으면? → syntax 에러 발생

## ■ 파이썬 제어문

### 2. if문

#### •if문의 형식

if 조건식1:

statements

elif 조건식2:

statements

elif 조건식3:

statements

else:

statements

•조건식이나 else 다음에 콜론(:) 표기 필요

•들여쓰기(indentation)를 잘 지켜야 함

- statements: 조건이 만족될 때 수행되어야 하는 문장들
- elif, else문은 있어도 되고 없어도 되는 문법들
- elif: elseif의 약자
- elif: 위 조건식 1이 만족 안되었다면 elif의 조건식2 확인 후 수행
- elif 문은 여러 개 삽입 가능
- 조건식 1,2,3이 모두 불만족 시 else의 statement 수행

## ■ 파이썬 제어문

### 2. if문

```
score = 90
if score >= 90:
    print 'Congratulations!!! '
```

Congratulations!!!

- score가 90과 같으니까 'True'로 Congratulations!!! 출력

```
a = 10
if a > 5:
    print 'Big'
else:
    print 'Small'
```

Big

- 만약 if문 조건이 만족되지 않으면 else문 statement 수행
- elif문이 없어도 바로 else문 삽입 가능

•statement가 1개인 경우 조건식과 한 줄에 위치 가능 (추천하지는 않음)

```
a = 10
if a > 5: print 'Big'
else: print 'Small'
```

Big

## ■ 파이썬 제어문

### 2. if문

```
n = -2
if n > 0:
    print 'Positive'
elif n < 0:
    print 'Negative'
else:
    print 'Zero'
```

Negative

- 조건이 2가지로 각각의 경우에 따라 프린트 내용이 다르도록 코딩
- 두 조건 불만족 시 else문의 statement 수행

```
order = 'spagetti'

if order == 'spam':
    price = 500
elif order == 'ham':
    price = 700
elif order == 'egg':
    price = 300
elif order == 'spagetti':
    price = 900

print price
```

900

## ■ 파이썬 제어문

### 2. if문

• 위 코드와 동일한 dictionary 자료구조를 사용한 코드

```
order = 'spagetti'
menu = {'spam':500, 'ham':700, 'egg':300, 'spagetti':900}
price = menu[order]
print price
```

900

- 메뉴가 사전으로 구성되어 있음
- spam, ham, egg, spaghetti → key
- 500, 700, 300, 900 → 각 key에 대응하는 value
- order가 spaghetti니까 spaghetti key에 value값이 price에 할당됨

## ■ 파이썬 제어문

### 3. for문

#### •for문의 형식

```
for <타겟> in <컨테이너 객체>:
```

```
    statements
```

```
else:
```

```
    statements
```

- 컨테이너 객체에서 원소를 꺼내 타겟에 삽입
- statement는 타겟의 value를 활용하여 코딩

```
a = ['cat', 'cow', 'tiger']
```

```
for x in a:
```

```
    print len(x), x
```

```
3 cat
3 cow
5 tiger
```

- a는 리스트 → 리스트, 문자, 튜플, 사전은 컨테이너 객체
- len(x)의 x자리에 a 리스트의 첫번째 원소가 들어감
- len: 문자열의 길이를 구하는 내장함수

```
for x in [1,2,3]:
```

```
    print x,
```

```
1 2 3
```



## ■ 파이썬 제어문

### 3. for문

```
print range(10)
```

```
for x in range(10):  
    print x,
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
0 1 2 3 4 5 6 7 8 9
```

- range 내장함수가 반환해주는 것은 list
- range(10) = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

```
sum = 0  
for x in range(1, 11):  
    sum = sum + x  
  
print sum
```

```
55
```

- range(1, 11): 1은 start값, 11은 end 값
- range(1, 11) = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

```
prod = 1  
for x in range(1, 11):  
    prod = prod * x  
  
print prod
```

```
3628800
```

- prod \* x : prod라는 변수에 곱하기의 누적
- 1 \* 2=결과 \* 3 = 결과 \* 4=.....= 결과 \* 10

## ■ 파이썬 제어문

### 3. for문

• enumerate() 내장 함수: 컨테이너 객체가 지닌 각 요소값뿐만 아니라 인덱스 값도 함께 반환한다.

```
l = ['cat', 'dog', 'bird', 'pig']  
for k, animal in enumerate(l):  
    print k, animal
```

```
0 cat  
1 dog  
2 bird  
3 pig
```

- k → enumerate하고 있는 컨테이너 객체의 index
- animal → 실제 값 들어옴
- 앞에 있는 0, 1, 2, 3이 그대로 각각의 문자열에 대한 index 역할 수행
- enumerate: key와 실제 그 값을 같이 가져옴
- key: 컨테이너 객체가 지니고 있는 index값

```
t = ('cat', 'dog', 'bird', 'pig')  
for k, animal in enumerate(t):  
    print k, animal
```

```
0 cat  
1 dog  
2 bird  
3 pig
```

## ▣ 파이썬 제어문

### 3. for문

```
d = {'c':'cat', 'd':'dog', 'b':'bird', 'p':'pig'}  
for k, key in enumerate(d):  
    print k, key, d[key]
```

```
0 p pig  
1 c cat  
2 b bird  
3 d dog
```

- dictionary 자체가 index를 관리하지 않아 index 바로 나오진 않음
- k에는 반드시 0부터 시작되는 index가 들어감
- $k \rightarrow 0, 1, 2, 3$
- key  $\rightarrow$  하나의 문자로 이루어진 문자열(c, d, b, p)이 하나씩 들어감
- d[key]: 키워드 index
- d[c]  $\rightarrow$  cat, d[d]  $\rightarrow$  dog

## ■ 파이썬 제어문

### 3. for문

•break: 루프를 빠져나간다.

```
for x in range(10):  
    if x > 3:  
        break  
    print x  
  
print 'done'
```

```
0  
1  
2  
3  
done
```

- break, continue
- range (10)→ [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
- break: 현재 수행중인 for문 또는 while문 반복 하지 않음
- x가 4가 되어 3보다 커지는 순간 for문 수행 중지 (break)

## ■ 파이썬 제어문

### 3. for문

- continue: 루프 블록 내의 continue 이후 부분은 수행하지 않고 루프의 시작부분으로 이동한다.

```
for x in range(10):  
    if x < 8:  
        continue  
    print x  
  
print 'done'
```

```
8  
9  
done
```

- continue: 특정 조건이 만족되었을 때 continue하면서 하위 코드 무시
- x자리에 0~7까지는 continue가 수행되어 print X 무시

- else: 루프가 break에 의한 중단 없이 정상적으로 모두 수행되면 else 블록이 수행된다.

```
for x in range(10):  
    print x,      # 콤마(,) 때문에 줄이 바뀌지 않는다.  
else:  
    print 'else block'  
  
print 'done'
```

```
0 1 2 3 4 5 6 7 8 9 else block  
done
```

- ,(콤마)가 들어가 있어서 원소들이 바로 옆에 프린트 됨
- else block에는 콤마가 없어서 done은 아래 블록에 프린트

## ■ 파이썬 제어문

### 3. for문

- break에 의하여 루프를 빠져나가면 else 블록도 수행되지 않는다.

```
for x in range(10):  
    break  
    print x,  
else:  
    print 'else block'  
  
print 'done'
```

done

- 아무 조건 없는 break → 0부터 break 걸려 print X, else문 무시
- else 블록은 for문이 break 걸리지 않고 정상적 수행 시 수행 가능

## ■ 파이썬 제어문

### 3. for문

- for 루프의 중첩

```
for x in range(2, 4):  
    for y in range(2, 10):  
        print x, '*', y, '=', x*y  
    print
```

```
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 * 2 = 6  
3 * 3 = 9  
3 * 4 = 12  
3 * 5 = 15  
3 * 6 = 18  
3 * 7 = 21  
3 * 8 = 24  
3 * 9 = 27
```

- for 루프를 여러 개 중첩하여 사용 가능
- x가 2일 때 y가 2, 3, 4, 5, 6, 7, 8, 9 수행
- x가 3일 때 y가 역시 2, 3, 4, 5, 6, 7, 8, 9 수행

## ▣ 파이썬 제어문

### 4. While문

- while 조건식이 만족하는 동안 while 블록내의 statements 들을 반복 수행한다.

```
count = 1
while count < 11:
    print count
    count = count + 1
```

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

```
sum = 0
a = 0
while a < 10:
    a = a + 1
    sum = sum + a
print sum
```

```
55
```

- a가 10이 되어 조건 불만족 시 중단되고 현재 누적 값 출력



## ■ 파이썬 제어문

### 4. While문

```
x = 0
while x < 10:
    print x,      # 콤마(,) 때문에 줄이 바뀌지 않는다.
    x = x + 1
else:
    print 'else block'

print 'done'
```

```
0 1 2 3 4 5 6 7 8 9 else block
done
```

- 콤마가 있으므로 옆으로 값 프린트
- while문 끝내기 직전에 else문 수행
- else문은 while문에서도 쓰일 수 있고, for문에서도 쓰일 수 있음