# UNIT 8

# 불과 비교, 논리 연산자 알아보기

# 8 불과 비교, 논리 연산자 알아보기

## » 불과 비교, 논리 연산자 알아보기

- 참(True), 거짓(False)을 나타내는 불(boolean)을 알아보자
- 두 값의 관계를 판단하는 비교 연산자와 두 값의 논릿값을 판단하는 논리 연산자도 함께 알아보자
- 비교, 논리 연산자는 프로그래밍에서 매우 광범위하게 쓰임
- if, while 구문을 작성할 때 비교, 논리 연산자를 자주 사용함

# 8.1 불과 비교 연산자 사용하기

## 불과 비교 연산자 사용하기

- 불은 True, False로 표현하며 1, 3.6, 'Python'처럼 값의 일종

```
>>> True
True
>>> False
False
```

## 비교 연산자의 판단 결과

- 파이썬에서는 비교 연산자와 논리 연산자의 판단 결과로 True, False를 사용
- 비교 결과가 맞으면 True, 아니면 False

```
>>> 3 > 1
True
```

- 부등호 >로 두 숫자를 비교함
- 3이 1보다 크니까 결과는 참이고 True가 나옴

# 8.1 불과 비교 연산자 사용하기

## 숫자가 같은지 다른지 비교하기

- 이제 두 숫자가 같은지 또는 다른지 비교해보자
- 두 숫자가 같은지 비교할 때는 ==(equal), 다른지 비교할 때는 !=(not equal)을 사용

```
>>> 10 == 10    # 10과 10이 같은지 비교
True
>>> 10 != 5     # 10과 5가 다른지 비교
True
```

- 10과 10은 같으므로 True, 10과 5는 다르므로 True가 나옴
- 파이썬에서 두 값이 같은지 비교할 때는 =이 아닌 ==을 사용
- =은 할당 연산자로 이미 사용되고 있기 때문임

# 8.1 불과 비교 연산자 사용하기

## » 문자열이 같은지 다른지 비교하기

- 숫자뿐만 아니라 문자열도 ==와 != 연산자로 비교할 수 있음
- 문자열은 비교할 때 대소문자를 구분
- 단어가 같아도 대소문자가 다르면 다른 문자열로 판단

```
>>> 'Python' == 'Python'
True
>>> 'Python' == 'python'
False
>>> 'Python' != 'python'
True
```

# 8.1 불과 비교 연산자 사용하기

## 부등호 사용하기

- 부등호는 수학 시간에 배운 내용과 같음
- 큰지, 작은지, 크거나 같은지, 작거나 같은지를 판단

```
>>> 10 > 20    # 10이 20보다 큰지 비교
False
>>> 10 < 20    # 10이 20보다 작은지 비교
True
>>> 10 >= 10    # 10이 10보다 크거나 같은지 비교
True
>>> 10 <= 10    # 10이 10보다 작거나 같은지 비교
True
```

- 비교 기준은 첫 번째 값
- 첫 번째 값보다 큰지, 작은지처럼 읽음
- 부등호를 말로 설명할 때 〉은 초과, 〈은 미만, 〉=은 이상, 〈=은 이하라고도 함
- 〉, 〈은 비교할 값과 같으면 무조건 거짓
- 〉=, 〈=은 비교할 값과 같으면 참
- 이상, 이하는 비교할 값도 포함된다는 점이 중요함

# 8.1 불과 비교 연산자 사용하기

## 객체가 같은지 다른지 비교하기

- 이번에는 is와 is not
- 같다는 ==, 다르다는 !=이 이미 있는데 왜 is, is not을 만들었을까?
- is, is not도 같다, 다르다지만 ==, !=는 값 자체를 비교하고, is, is not은 객체(object)를 비교

```
>>> 1 == 1.0
True
>>> 1 is 1.0
False
>>> 1 is not 1.0
True
```

- 1과 1.0은 정수와 실수라는 차이점이 있지만 값은 같음
- ==로 비교해보면 True가 나옴
- 1과 1.0을 is로 비교해보면 False가 나옴
- 1은 정수 객체, 1.0은 실수 객체이므로 두 객체는 서로 다르기 때문임
- 1과 1.0을 is not으로 비교하면 True가 나오겠죠?

# 8.1 불과 비교 연산자 사용하기

## » 참고 | 정수 객체와 실수 객체가 서로 다른 것은 어떻게 확인하나요?

- 정수 객체와 실수 객체가 서로 다른지 확인하려면 id 함수를 사용하면 됨
- id는 객체의 고유한 값(메모리 주소)을 구함(이 값은 파이썬을 실행하는 동안에는 계속 유지되며 다시 실행하면 달라짐)

```
>>> id(1)
1714767504
>>> id(1.0)
55320032
```

- 두 객체의 고유한 값이 다르므로 서로 다른 객체임
- 1과 1.0을 is로 비교하면 False가 나옴
- is, is not은 클래스로 객체를 만든 뒤에 객체가 서로 같은지 비교할 때 주로 사용
- 여기에 나오는 객체의 고유한 값(메모리 주소)에 대해서는 신경 쓸 필요 없음
- ==, !=와 is, is not의 동작 방식이 다르다는 정도만 알아 두면 됨

# 8.1 불과 비교 연산자 사용하기

## » 참고 | 값 비교에 is를 쓰지 않기

- 값을 비교할 때는 is를 사용하면 안 됨
- 변수 a에 -5를 할당한 뒤 a is -5를 실행하면 True가 나오지만 다시 -6을 할당한 뒤 a is -6을 실행하면 False가 나옴

```
>>> a = -5
>>> a is -5
True
>>> a = -6
>>> a is -6
False
```

- 변수 a가 있는 상태에서 다른 값을 할당하면 메모리 주소가 달라질 수 있기 때문임
- 다른 객체가 되므로 값이 같더라도 is로 비교하면 False가 나옴
- 값(숫자)를 비교할 때는 is가 아닌 비교 연산자를 사용

# 8.2 논리 연산자 사용하기

## 논리 연산자 사용하기

- 논리 연산자를 사용해보자
- 논리 연산자는 and, or, not이 있는데 먼저 and임

- a and b

```
>>> True and True
True
>>> True and False
False
>>> False and True
False
>>> False and False
False
>>>
```

# 8.2 논리 연산자 사용하기

## 논리 연산자 사용하기

- and는 두 값이 모두 True라야 True임
- 하나라도 False이면 False가 나옴
- 이번에는 or임

- **a or b**

```
>>> True or True
True
>>> True or False
True
>>> False or True
True
>>> False or False
False
```

# 8.2 논리 연산자 사용하기

## 논리 연산자 사용하기

- or는 두 값 중 하나라도 True이면 True임
- 두 값이 모두 False라야 False가 됨
- 마지막으로 not임

- not x

```
>>> not True
False
>>> not False
True
```

- not은 논릿값을 뒤집음
- not True는 False가 되고, not False는 True가 됨
- 여기서 and, or, not 논리 연산자가 식 하나에 들어있으면 not, and, or 순으로 판단함

```
>>> not True and False or not False
True
```

# 8.2 논리 연산자 사용하기

## 논리 연산자 사용하기

- not True와 not False를 판단하여 False and False or True가 됨
- False and False를 판단하여 False가 나와서 False or True가 되므로 최종 결과는 True가 됨

```
not True and False or not False
False and False or True
False or True
True
```

- 이 식을 괄호로 표현하면 다음과 같은 모양이 됨

```
>>> ((not True) and False) or (not False)
True
```

- 순서가 헷갈릴 때는 괄호로 판단 순서를 명확히 나타내 주는 것이 좋음

# 8.2 논리 연산자 사용하기

## 논리 연산자와 비교 연산자를 함께 사용하기

- 조금 응용해서 논리 연산자와 비교 연산자를 함께 사용해보자

```
>>> 10 == 10 and 10 != 5    # True and True
True
>>> 10 > 5 or 10 < 3        # True or False
True
>>> not 10 > 5              # not True
False
>>> not 1 is 1.0            # not False
True
```

- 비교 연산자로 비교한 결과를 논리 연산자로 다시 판단함
- 비교 연산자(is, is not, ==, !=, 〈, 〉, 〈=, 〉=)를 먼저 판단하고 논리 연산자(not, and, or)를 판단하게 됨
- 파이썬은 영어 문장과 흡사한 구조를 가지고 있어서 코드가 읽기 쉬운 것이 장점임

# 8.2 논리 연산자 사용하기

## 참고 | 정수, 실수, 문자열을 불로 만들기

- 정수, 실수, 문자열을 불로 만들 때는 bool을 사용하면 됨
- 정수 1은 True, 0은 False임
- 만약 문자열의 내용이 'False'라도 불로 만들면 True임
- 문자열의 내용 자체는 판단하지 않으며 값이 있으면 True임

**bool(값)**

```
>>> bool(1)
True
>>> bool(0)
False
>>> bool(1.5)
True
>>> bool('False')
True
```

- 즉, 정수 0, 실수 0.0이외의 모든 숫자는 True임
- 빈 문자열 '', ""를 제외한 모든 문자열은 True임

# 8.2 논리 연산자 사용하기

## » 참고 | 단락 평가

- 논리 연산에서 중요한 부분이 단락 평가(short-circuit evalution)임
- 단락 평가는 첫 번째 값만으로 결과가 확실할 때 두 번째 값은 확인(평가)하지 않는 방법을 말함
- and 연산자는 두 값이 모두 참이라야 참이므로 첫 번째 값이 거짓이면 두 번째 값은 확인하지 않고 바로 거짓으로 결정함

```
# 첫 번째 값이 거짓이므로 두 번째 값은 확인하지 않고 거짓으로 결정
print(False and True)     # False
print(False and False)    # False
```

- or 연산자는 두 값 중 하나만 참이라도 참이므로 첫 번째 값이 참이면 두 번째 값은 확인하지 않고 바로 참으로 결정함

```
# 첫 번째 값이 참이므로 두 번째 값은 확인하지 않고 참으로 결정
print(True or True)     # True
print(True or False)    # True
```

# 8.2 논리 연산자 사용하기

## » 참고 | 단락 평가

- 파이썬에서 논리 연산자는 이 단락 평가에 따라 반환하는 값이 결정됨
- True, False를 논리 연산자로 확인하면 True, False가 나왔는데, True and 'Python'의 결과는 무엇이 나올까?

```
>>> True and 'Python'
'Python'
```

- 문자열 'Python'도 불로 따지면 True라서 True and True가 되어 True가 나올 것 같지만 'Python'이 나옴
- 파이썬에서 논리 연산자는 마지막으로 단락 평가를 실시한 값을 그대로 반환하기 때문임
- 논리 연산자는 무조건 불을 반환하지 않음
- 다음과 같이 마지막으로 단락 평가를 실시한 값이 불이면 불을 반환하게 됨

```
>>> 'Python' and True
True
>>> 'Python' and False
False
```

# 8.2 논리 연산자 사용하기

## » 참고 | 단락 평가

- 문자열 'Python'을 True로 쳐서 and 연산자가 두 번째 값까지 확인하므로 두 번째 값이 반환됨
- and 연산자 앞에 False나 False로 치는 값이 와서 첫 번째 값 만으로 결과가 결정나는 경우에는 첫 번째 값이 반환됨

```
>>> False and 'Python'
False
>>> 0 and 'Python'    # 0은 False이므로 and 연산자는 두 번째 값을 평가하지 않음
0
```

- or 연산자도 마찬가지로 마지막으로 단락 평가를 실시한 값이 반환됨
- or 연산자에서 첫 번째 값만으로 결과가 결정되므로 첫 번째 값이 반환됨

```
>>> True or 'Python'
True
>>> 'Python' or True
'Python'
```

# 8.2 논리 연산자 사용하기

## 참고 | 단락 평가

- 두 번째 값까지 판단해야 한다면 두 번째 값이 반환됨

```
>>> False or 'Python'
'Python'
>>> 0 or False
False
```