Operators and Expressions

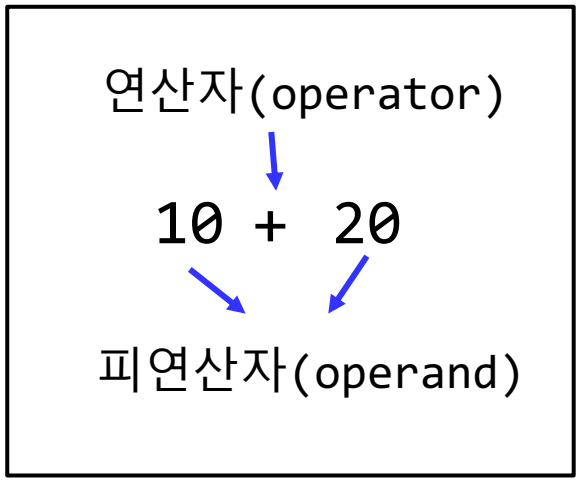
Contents

- Operator
- Arithmetic Operator
- Relational Operator
- Logical Operator

연산자의 개념

- 연산자
  - 연산자(operator)는 산술 연산자 +, -, * 기호와 같이, 이미 정의된 연산을 수행하는 기호나 키워드를 의미
  - 문제를 해결하는 방법에서 도구(장비)와 같은 역할

- 피연산자
  - 연산(operation)에 참여하는 변수나 값을 피연산자(operand)



산술 연산자(Arithmetic Operator)

- + : 덧셈을 수행
- - : 뺄셈을 수행
- * : 곱셈을 수행
- / : 나눗셈을 수행하고, 실수 결과를 반환
- // : 나눗셈을 수행하고, 몫을 반환
- % : 나머지 연산을 수행
- ** : 거듭제곱을 수행

산술 연산자(Arithmetic Operator) 예시

```
a = 10
b = 3
# 덧셈
sum_result = a + b
print("덧셈 결과:", sum_result)
# 뺄셈
sub_result = a - b
print("뺄셈 결과:", sub_result)
# 곱셈
mul_result = a * b
print("곱셈 결과:", mul_result)
# 나눗셈 (실수 결과)
div_result = a / b
print("나눗셈 결과:", div_result)
# 나눗셈 (정수 결과)
floor_div_result = a // b
print("나눗셈 (몫) 결과:", floor_div_result)
# 나머지 연산
mod_result = a % b
print("나머지 연산 결과:", mod_result)
# 거듭제곱
exp_result = a ** b
print("거듭제곱 결과:", exp_result)
```

```
덧셈 결과: 13
뺄셈 결과: 7
곱셈 결과: 30
나눗셈 결과: 3.3333333333333335
나눗셈 (몫) 결과: 3
나머지 연산 결과: 1
거듭제곱 결과: 1000
```

관계 연산자(Relational Operator)

- == : 두 값이 같은지를 확인
- != : 두 값이 다른지를 확인
- > : 왼쪽 값이 오른쪽 값보다 큰지를 확인
- < : 왼쪽 값이 오른쪽 값보다 작은지를 확인
- >= : 왼쪽 값이 오른쪽 값보다 크거나 같은지를 확인
- <= : 왼쪽 값이 오른쪽 값보다 작거나 같은지를 확인

관계 연산자(Relational Operator) 예시

```
a = 10
b = 5

# 같은지 비교
equal_result = a == b
print("a와 b가 같은지:", equal_result)
# 다른지 비교
not_equal_result = a != b
print("a와 b가 다른지:", not_equal_result)
# 큰지 비교
greater_result = a > b
print("a가 b보다 큰지:", greater_result)
# 작은지 비교
less_result = a < b
print("a가 b보다 작은지:", less_result)
# 크거나 같은지 비교
greater_equal_result = a >= b
print("a가 b보다 크거나 같은지:",
greater_equal_result)
# 작거나 같은지 비교
less_equal_result = a <= b
print("a가 b보다 작거나 같은지:",
less_equal_result)
```

```
a와 b가 같은지: False
a와 b가 다른지: True
a가 b보다 큰지: True
a가 b보다 작은지: False
a가 b보다 크거나 같은지: True
a가 b보다 작거나 같은지: False
```

논리 연산자(Logical Operator)

- and : 두 조건이 모두 참인지를 확인
- or : 두 조건 중 하나 이상이 참인지를 확인
- not : 조건을 부정

논리 연산자(Logical Operator) 예시

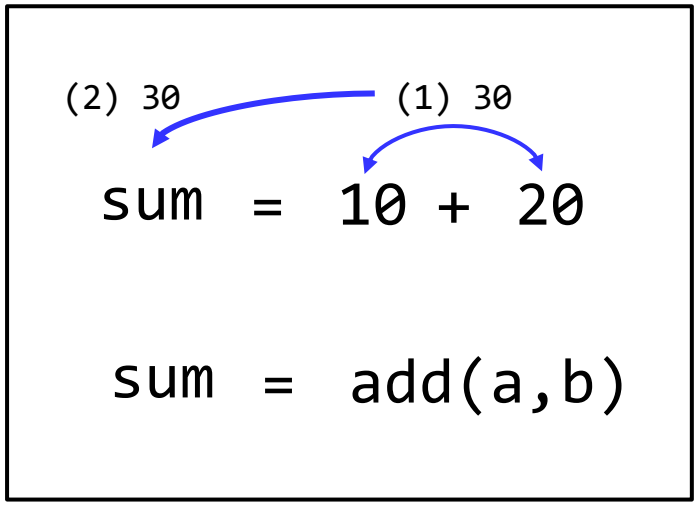
```
a = True
b = False

# 논리 AND
and_result = a and b
print("a와 b의 논리 AND 결과:", and_result)

# 논리 OR
or_result = a or b
print("a와 b의 논리 OR 결과:", or_result)

# 논리 NOT
not_result_a = not a
not_result_b = not b
print("a의 논리 NOT 결과:", not_result_a)
print("b의 논리 NOT 결과:", not_result_b)
```

```
a와 b의 논리 AND 결과: False
a와 b의 논리 OR 결과: True
a의 논리 NOT 결과: False
b의 논리 NOT 결과: True
```

대입 연산자(assignment operator)

- 변수의 저장 값을 대입하는 = 기호가 대입(할당) 연산자(assignment operator)
- = 연산자 오른쪽 수식을 먼저 계산하고 결과 값을 왼쪽 변수에 대입하는 기능
- 대입 연산자의 왼쪽 부분에는 반드시 변수만이 가능, 대입 연산자 왼쪽에 온 변수는 값이 변경
- 수식의 결과 값을 변수에 대입하지 않으면 프로그램에는 영향이 없음

```
10 + 20
```



단축대입 연산자

- 단축 대입 연산자는 산술 연산자와 대입 연산자를 결합하여 코드를 간결하게 작성하는 데 사용

- +=: 덧셈 후 대입 연산자 a += b는 a = a + b와 동일
- -=: 뺄셈 후 대입 연산자 a -= b는 a = a - b와 동일
- *=: 곱셈 후 대입 연산자 a *= b는 a = a * b와 동일
- /=: 나눗셈 후 대입 연산자 a /= b는 a = a / b와 동일
- %=: 나머지 연산 후 대입 연산자 a %= b는 a = a % b와 동일
- //=: 몫 연산 후 대입 연산자 a //= b는 a = a // b와 동일
- **=: 거듭제곱 후 대입 연산자 a **= b는 a = a ** b와 동일

연산자 예제 : 동전 교환 프로그램

사용자로부터 금액을 입력 받아 해당 금액을 가능한한 적은 동전 개수로 교환하는 프로그램을 구현

```
교환할 금액을 입력하세요: 1770
500원 짜리 동전 3개
100원 짜리 동전 2개
50원 짜리 동전 1개
10원 짜리 동전 2개
```

연산자 예제 : 동전 교환 프로그램

```
def coin_change(amount):
    coins = [500, 100, 50, 10]  # 동전의 종류

    change = []  # 동전 교환 결과를 담을 리스트

    for coin in coins:
        count = amount // coin  # 해당 동전으로 교환할 개수 계산
        amount %= coin  # 남은 금액 갱신
        change.append((coin, count))  # 동전 종류와 개수를 리스트에 추가

    return change

# 사용자로부터 금액 입력 받기
amount = int(input("교환할 금액을 입력하세요: "))

# 동전 교환 실행
result = coin_change(amount)

# 결과 출력
for coin, count in result:
    print(f"{coin}원 짜리 동전 {count}개")
```

연산자 예제 : 동전 교환 프로그램

1. coin_change 함수는 amount라는 인자를 받음
2. coins 변수에는 동전의 종류인 [500, 100, 50, 10]이 리스트로 저장
3. change 변수는 동전 교환 결과를 담을 빈 리스트
4. for 루프를 통해 coins 리스트의 각 동전에 대해 반복
5. count 변수에는 amount를 현재 동전으로 나눈 몫이 저장하여 해당 동전으로 교환할 개수를 계산
6. amount에는 amount를 현재 동전으로 나눈 나머지가 저장하여 남은 금액을 갱신
7. change.append((coin, count))를 통해 동전의 종류와 개수를 (coin, count) 형태로 change 리스트에 추가
8. change 리스트가 완성되면, return change를 통해 교환 결과를 반환
9. 사용자로부터 교환할 금액을 입력받고, amount 변수에 저장
10. coin_change(amount)를 호출하여 동전 교환을 실행하고, 결과를 result 변수에 저장
11. for 루프를 통해 result 리스트의 각 요소를 반복하면서 동전의 종류와 개수를 출력

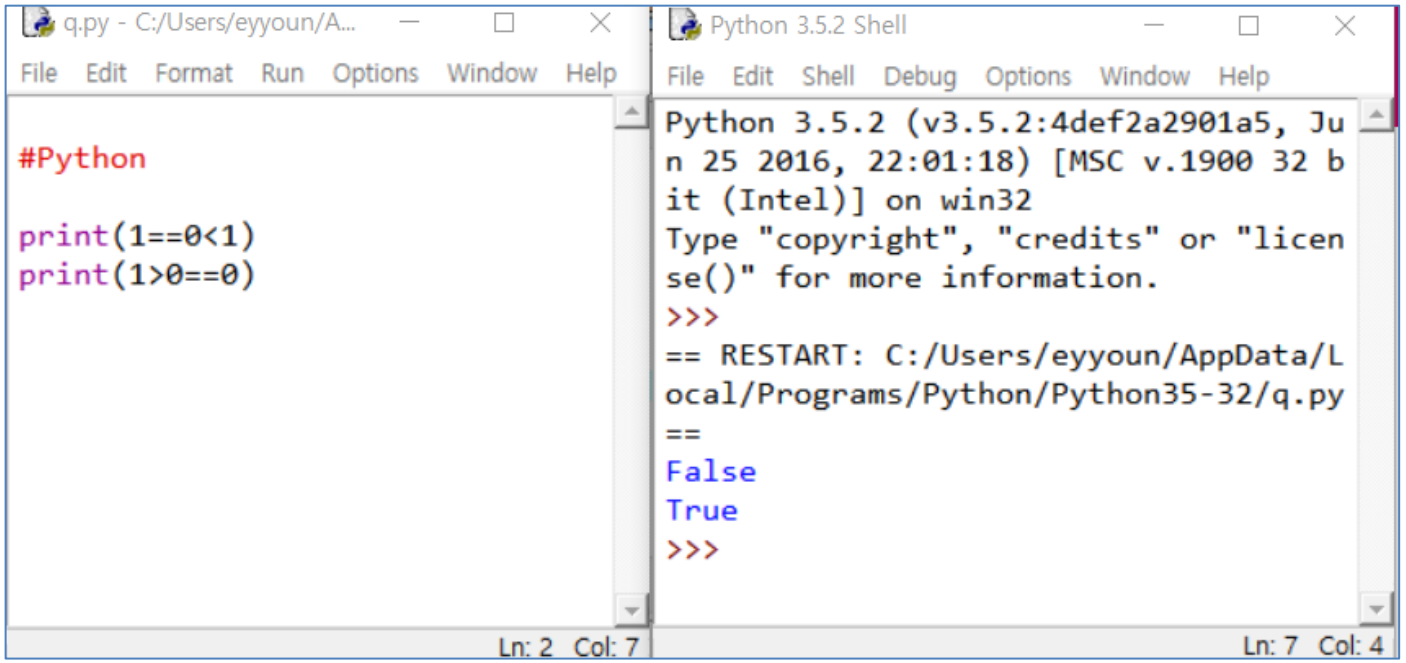
관계연산자 우선순위(참고자료)

Python은 <, <=, >, >=, ==, != 등에 해당하는 비교 및 동등 연산자가 동일한 우선 순위

a < b < c < d 와 같은 표현은 a < b and b < c and c < d 실행됨

1 == 0 < 1 의 경우 1 == 0 and 0 < 1 // False

1 > 0 == 0 의 경우 1 > 0 and 0 == 0 // True



관계연산자 우선순위(참고자료)

C 는 <, <=, >, >= 비교연산자 보다 ==, != 동등 연산자가 우선 순위가 낮음

1 == 0 < 1 의 경우 1 == (0 < 1) -> 1 == 1 // 1 (T)

1 > 0 == 0 의 경우 (1 > 0) == 0 -> 1 == 0 // 0 (F)

```c
// C

#include<stdio.h>

int main(void)
{

    printf("%d \n",1==0<1);
    printf("%d \n",1>0==0);
    return 0;

}
```

```
1
0

--------------------------------
Process exited after 0.4979 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```