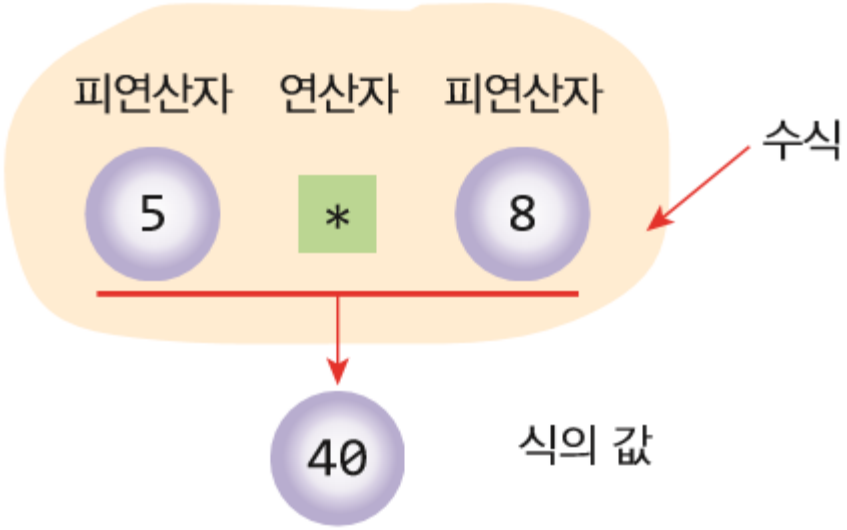
연산자 *(Operator)*

연산자와 피연산자

- 수식(expression) = 피연산자들과 연산자의 조합
  - 연산자(operator): 연산을 나타내는 기호
  - 피연산자(operand): 연산의 대상이 되는 값



산술 연산자

산술 연산자	설명	사용 예	예 설명
=	대입 연산자	a = 3	정수 3을 a에 대입
+	더하기	a = 5 + 3	5와 3을 더한 값을 a에 대입
−	빼기	a = 5 − 3	5에서 3을 뺀 값을 a에 대입
*	곱하기	a = 5 * 3	5와 3을 곱한 값을 a에 대입
/	나누기	a = 5 / 3	5를 3으로 나눈 값을 a에 대입
//	나누기(몫)	a = 5 // 3	5를 3으로 나눈 뒤 소수점을 버리고 a에 대입
%	나머지 값	a = 5 % 3	5를 3으로 나눈 뒤 나머지 값을 a에 대입
**	제곱	a = 5 ** 3	5의 3제곱을 a에 대입

주의) 파이썬 버전 2.X에서는 / 연산자의 결과가 정수

복합 대입 연산자

- 연산과 대입을 동시에 수행

복합 대입 연산자	의미
a += 1	a = a + 1
a -= 1	a = a - 1
a *= 1	a = a * 1
a /= 1	a = a / 1
a //= 1	a = a // 1
a %= 1	a = a % 1

나머지 연산자 예제 (1)

- 짝수와 홀수의 구분

```
>>> number = int(input("정수를 입력하시오: "))
정수를 입력하시오: 28
>>> print(number%2)
0
```

- 초단위의 시간을 받아서 몇 분 몇 초인지를 계산

```
>>> sec = 1000
>>> min = 1000 // 60
>>> remainder = 1000 % 60
>>> print(min, remainder)
16 40
```

나머지 연산자 예제 (2)

- 자동 판매기를 시뮬레이션하는 프로그램
  - 자동 판매기는 사용자로부터 투입한 돈과 물건값을 입력받음
  - 물건값은 100원 단위라고 가정하고 프로그램은 잔돈을 계산하여 출력
  - 자판기는 동전 500원, 100원짜리만 가지고 있다고 가정

```
money = int(input("투입한 돈: "))
price = int(input("물건 값: "))
change = money-price
print("거스름돈: ", change)
coin500s = change // 500          # 500으로 나누어서 몫이 500원짜리의 개수
change = change % 500             # 500으로 나눈 나머지를 계산한다.
coin100s = change // 100          # 100으로 나누어서 몫이 100원짜리의 개수
print("500원 동전의 개수: ", coin500s)
print("100원 동전의 개수: ", coin100s)
```

```
투입한 돈: 5000
물건값: 2600
거스름돈: 2400
500원 동전의 개수: 4
100원 동전의 개수: 4
```

- *Q) 만일 물건값이 10원 단위이고 자판기가 50원짜리 동전과 10원짜리 동전도 거슬러 줄 수 있다면?*

관계 연산자

- 결과는 참(True)이나 거짓(False)
  - *주로 조건문(if)이나 반복문(for, while)에서 사용*

관계 연산자	의미	설명
==	같다	두 값이 동일하면 참
!=	같지 않다	두 값이 다르면 참
〉	크다	왼쪽이 크면 참
〈	작다	왼쪽이 작으면 참
〉=	크거나 같다	왼쪽이 크거나 같으면 참
〈=	작거나 같다	왼쪽이 작거나 같으면 참

논리 연산자

논리 연산자	의미	설명	사용 예
and	~이고, 그리고(AND)	둘 다 참이어야 참	(a 〉 100) and (a 〈 200)
or	~이거나, 또는(OR)	둘 중 하나만 참이어도 참	(a = = 100) or (a = = 200)
not	~아니다, 부정(NOT)	참이면 거짓, 거짓이면 참	not(a 〈 100)

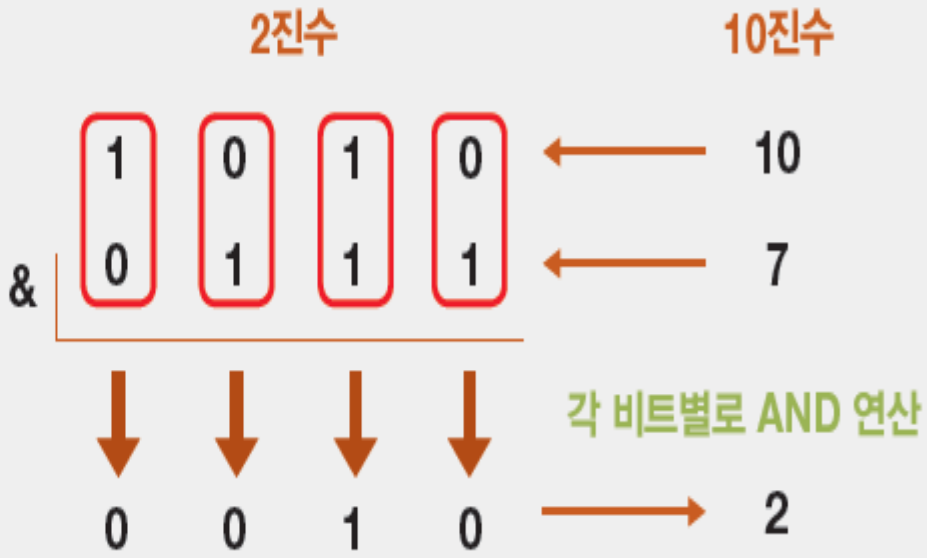
입력값		A 그리고 B	A 또는 B	A가 아니다
A	B	(A and B)	(A or B)	(!A)
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

비트 연산자

비트 연산자	설명	의미
&	비트 논리곱(AND)	둘 다 1이면 1
|	비트 논리합(OR)	둘 중 하나만 1이면 1
^	비트 배타적 논리합(XOR)	둘이 같으면 0, 다르면 1
~	비트 부정	1은 0으로, 0은 1로 변경
〈〈	비트 이동(왼쪽)	비트를 왼쪽으로 시프트(Shift)함
〉〉	비트 이동(오른쪽)	비트를 오른쪽으로 시프트(Shift)함

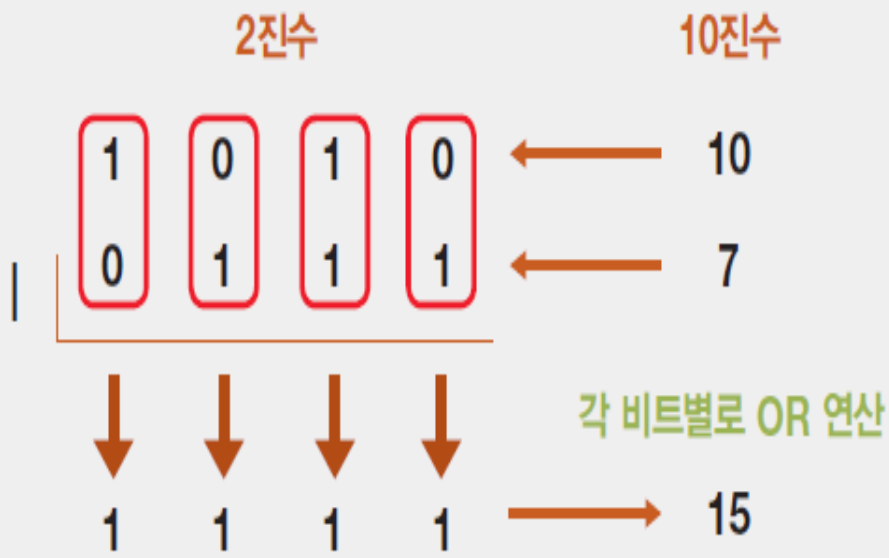
비트 논리곱(&) 연산자

A	B	A&B
0	0	0
0	1	0
1	0	0
1	1	1



비트 논리합(|) 연산자

A	B	A|B
0	0	0
0	1	1
1	0	1
1	1	1



비트 배타적 논리합(^) 연산자

- 두 값이 다르면 1 같으면 0

A	B	A^B
0	0	0
0	1	1
1	0	1
1	1	0

비트 부정(~) 연산자

- 각 비트에 대해 0은 1로, 1은 0으로 바꿈
  - 예) 0000을 비트 부정 연산하면 1111, 0101을 비트 부정 연산 하면 1010
  - 이렇게 반전된 값을 '1의 보수'라 함
  - 비트 부정 연산자는 어떤 값의 반대 부호의 값을 찾고자 할 때 활용

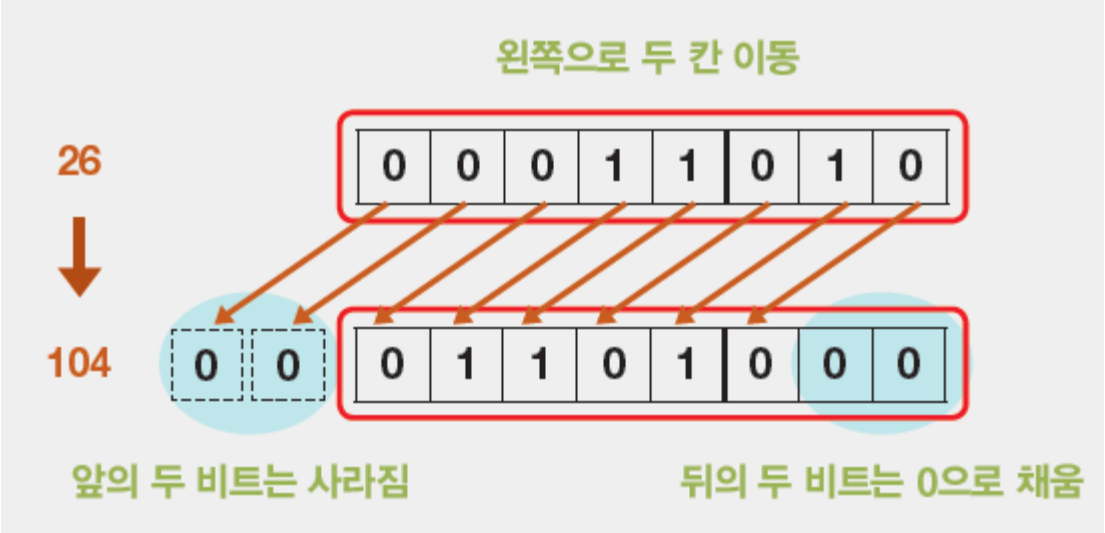
```
a = 12345
~a + 1
```

출력 결과

```
-12345
```

왼쪽 시프트(<<) 연산자

- 왼쪽으로 1회 시프트 할 때는 2^1을, 2회에는 2^2을, 3회에는 2^3을 곱한 효과




```
a = 10
a << 1 ; a << 2 ; a << 3 ; a << 4
```

출력 결과

```
20 40 80 160
```

오른쪽 시프트(>>) 연산자

- 오른쪽으로 1회 시프트 할 때는 2^1, 2회에는 2^2, 3회에는 2^3으로 나누는 효과
  - 오른쪽의 두 비트는 떨어져나가고 왼쪽의 두 비트에는 부호 비트(양수는 0이, 음수는 1)가 채워짐
  - 또한 시프트 연산은 정수만 연산하므로 몫만 남음(26 /2^2 =6)


```
a = 10
a >> 1 ; a >> 2 ; a >> 3 ; a >> 4
```

출력 결과

```
5 2 1 0
```

연산자 우선 순위

우선순위	연산자	설명
1	( ) [ ] { }	괄호, 리스트, 딕셔너리, 세트 등
2	**	지수
3	+ − ~	단항 연산자
4	* / % //	산술 연산자
5	+ −	산술 연산자
6	<< >>	비트 시프트 연산자
7	&	비트 논리곱
8	^	비트 배타적 논리합
9	|	비트 논리합
10	< > >= <=	관계 연산자
11	== !=	동등 연산자
12	= %= /= //= −= += *= **=	대입 연산자
13	not	논리 연산자
14	and	논리 연산자
15	or	논리 연산자
16	if ~ else	비교식