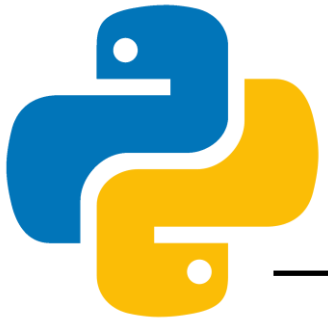


창직 IoT 종합설계입문

파이썬 (2) – 이진수 및 비트 연산

수 체계

십진법과 이진법

십진법에서는 어떤 수를 나타내기 위해 0부터 9까지 열 개의 숫자를 사용합니다.

$$52 = 5 \times 10^1 + 2 \times 10^0$$

이진법에서는 어떤 수를 나타내기 위해 0과 1, 두 개의 숫자를 사용합니다.

$$1001_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

각 숫자의 위치는 가중치를 할당합니다.

2진수

Counting in Binary

이진법에서 사용되는 0 또는 1은 비트라고 합니다.

N 개의 비트로 셀 수 있는 가장 큰 십진수는 $2^N - 1$입니다.

십진수	이진수		
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

2진수 – 10진수 변환

Binary to Decimal Conversion

$$111001_2 = 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 57$$

- 가장 오른쪽에 있는 비트를 LSB (least significant bit), 가장 왼쪽에 있는 비트를 MSB (Most significant bit)라고 합니다.

2진수의 사칙연산

이진수의 사칙 연산

- 덧셈
- 뺄셈
- 곱셈
- 나눗셈

2진수의 덧셈과 뺄셈

이진수의 덧셈과 뺄셈

- **덧셈**

$$0101_2 + 1001_2$$

$$\begin{array}{r} 0101_2 \\ +1001_2 \\ \hline 1110_2 \end{array}$$

- **뺄셈**

$$1001_2 - 0110_2$$

$$\begin{array}{r} 1001_2 \\ -0110_2 \\ \hline 0011_2 \end{array}$$

2진수의 곱셈과 나눗셈

이진수의 곱셈과 나눗셈

- **곱셈**

$$101_2 \times 011_2$$

$$\begin{array}{r} 101_2 \\ \times 011_2 \\ \hline 101_2 \\ 1010_2 \\ \hline 1111_2 \end{array}$$

- **나눗셈**

비트 (1)

비트

- 이진수를 뜻하는 Binary Digit의 약자로 컴퓨터에서 CPU가 처리하는 데이터의 최소 단위 크기를 의미합니다.
- 하나의 비트는 0 이나 1의 값을 가질 수 있습니다.
- 일반적으로 8bit = 1byte.

1	0	0	1	1	0	1	1

- 파이썬에서 ‘0b’를 이용하여 표현.
- bin(x) 함수를 통해 이진수로 표기할 수 있음.

```
print(15 == 0b1111) # 이진수 표기
bin(15)
```

```
True
'0b1111'
```

비트의 연산

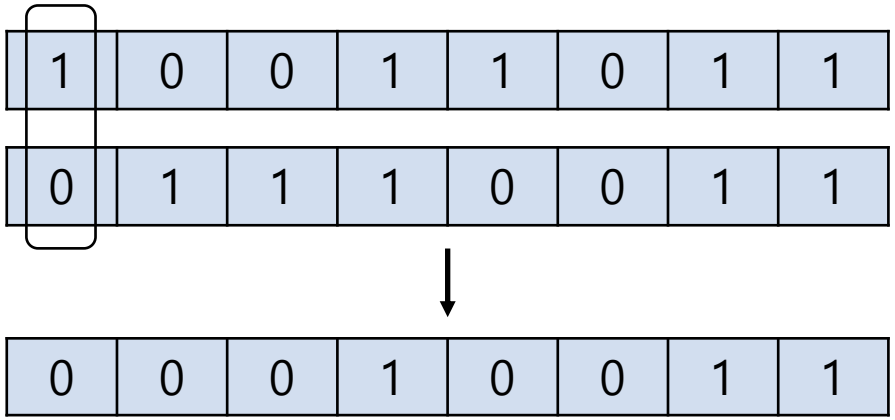
비트의 연산

- AND 연산
- OR 연산
- XOR 연산
- NOT 연산
- Shift 연산

AND 연산

AND 연산

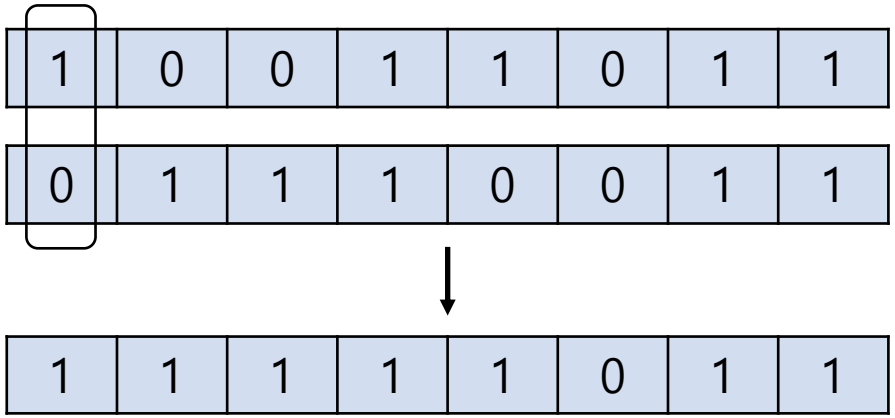
- 두 값이 모두 참이어야 True를 반환.
- 하나라도 False라면 False를 반환.



OR 연산

OR 연산

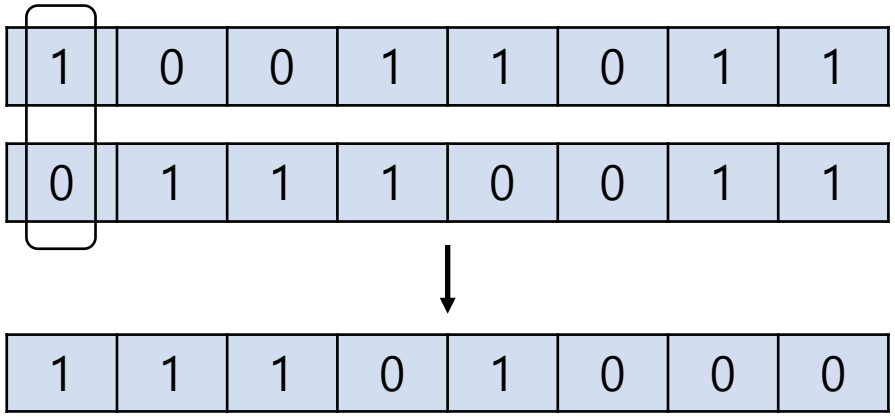
- 두 값이 모두 False이어야 False 를 반환.
- 하나라도 True이면 True를 반환.



XOR 연산

XOR 연산

- 비교하는 두 값이 다를 경우에 True를 반환합니다.
- 비교하는 두 값이 같을 경우에 False를 반환합니다.



NOT 연산

NOT 연산

- 주어진 2진수의 모든 자리의 비트를 반전시킵니다. 이 때, 부호 또한 반전됩니다.

값의 해석은 다음과 같이 2의 보수를 통해 확인합니다.

보수 (1)

부호 크기 체계

- 컴퓨터에서는 부호를 가장 왼쪽 비트(MSB)를 이용하여 나타냅니다.

1	0	0	1	1	0	1	1

- 부호가 있는 이진수를 고려할 때, MSB가 1일 때 음수, 0일 때 양수를 나타냅니다.
- 1byte인 데이터를 고려할 때, 부호가 있는 데이터는 $-2^6 \sim -2^6 - 1$의 범위를 표시할 수 있습니다.
- 부호가 없는 데이터는 $0 \sim -2^7 - 1$의 범위를 표시할 수 있습니다.

보수 (2)

보수 연산 종류

- 1의 보수
- 2의 보수 (컴퓨터에서 사용)

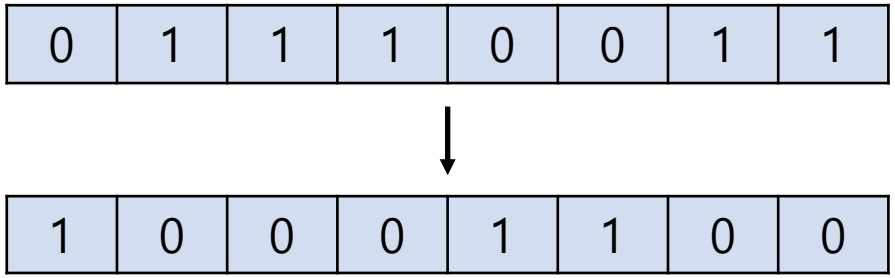
컴퓨터에서 '2의 보수' 사용 이유

- 음수에 대한 표현과 보다 싶게 연산을 하기 위해서 입니다.
- 컴퓨터는 뺄셈 회로가 없습니다.
- 보수를 취하고 더하면 뺄셈이 되므로, 굳이 뺄셈 회로가 있을 필요가 없습니다.

1의 보수

1의 보수

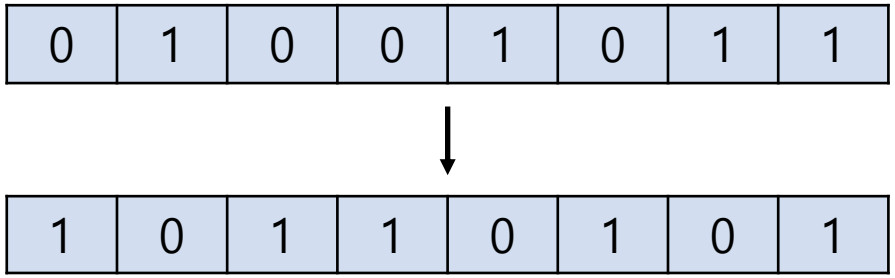
- 1의 보수란 어떤 수를 커다란 2의 거듭 제곱수−1에서 빼서 얻은 이진수입니다.
- 또는, 주어진 이진수의 모든 자리의 비트를 반전시키면 1의 보수를 얻을 수 있습니다.



2의 보수

2의 보수

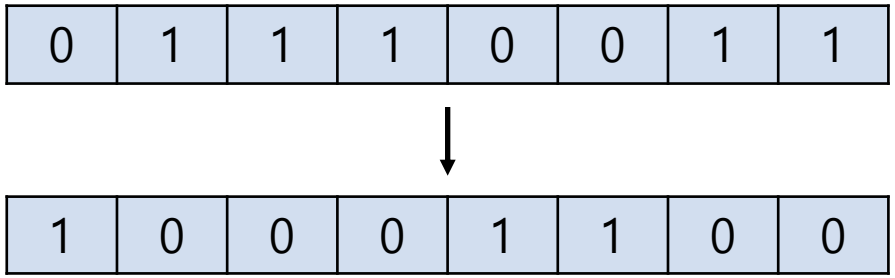
- 어떤 수를 커다란 2의 제곱수에서 빼서 얻은 2진수입니다.
- 주어진 이진수의 모든 자리의 비트를 반전시킨 뒤 1을 더하면 2의 보수를 얻을 수 있습니다.



Shift 연산

Shift 연산

- 산술 시프트
- 논리 시프트
- 산술 Shift와 논리 Shift의 차이점은 부호 비트가 보존되지 않는다는 것입니다.

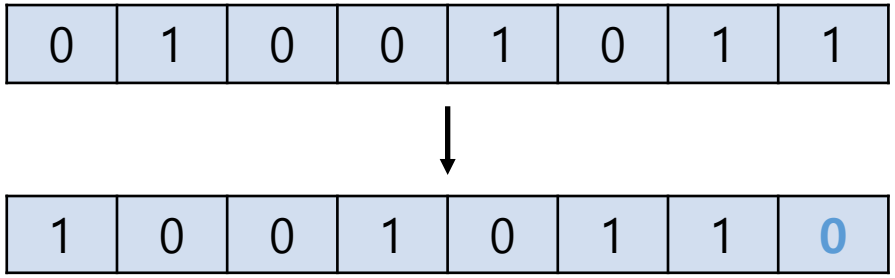


산술 Shift

산술 Shift

- $a \ll n$는 a를 좌측으로 n비트만큼 이동.
- $a \gg n$는 a를 우측으로 n비트만큼 이동.

- 좌측 시프트



- 우측 Shift