

# 파이썬 프로그래밍 – 03

## 계산

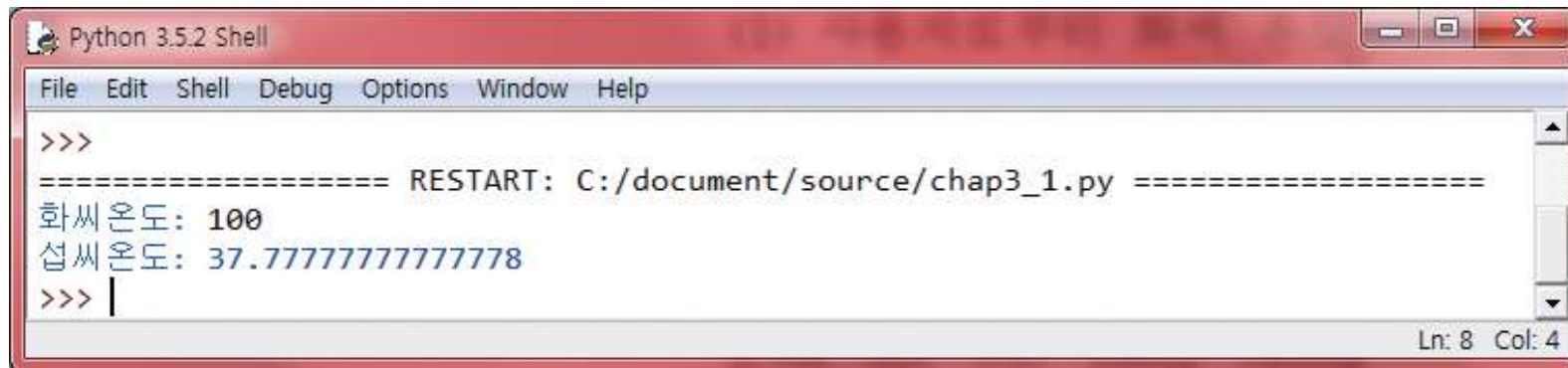
---

컴퓨터소프트웨어공학과  
성낙준



## 이번 장에서 만들 프로그램

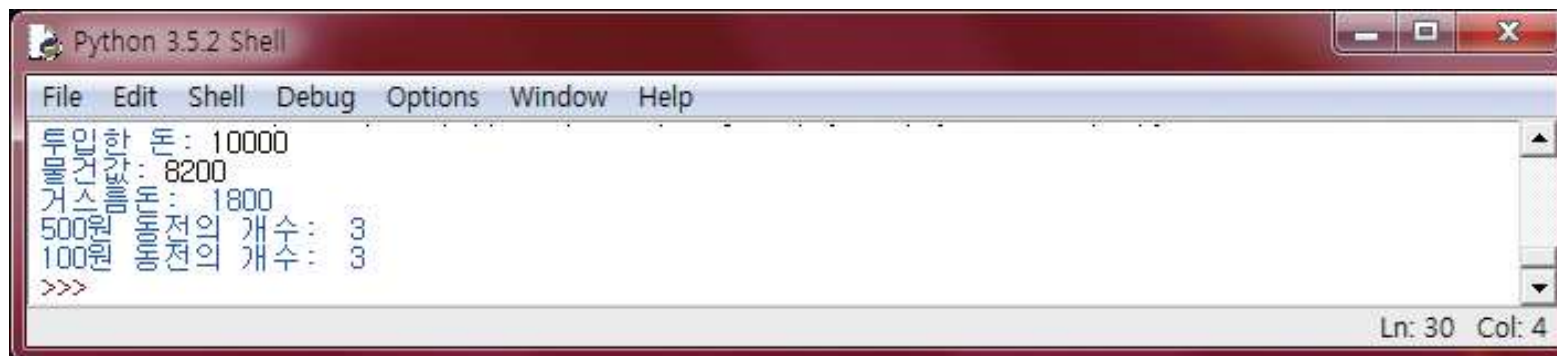
(1) 화씨 온도를 받아서 섭씨 온도로 변환하는 프로그램을 작성해 본다.



```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
>>>
===== RESTART: C:/document/source/chap3_1.py =====
화씨 온도: 100
섭씨 온도: 37.77777777777778
>>> |
```

Ln: 8 Col: 4

(2) 자판기 프로그램을 작성해 본다.



```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
투입한 돈: 10000
물건값: 8200
거스름돈: 1800
500원 동전의 개수: 3
100원 동전의 개수: 3
>>>
```

Ln: 30 Col: 4



# 수식은 어디에나 있다.

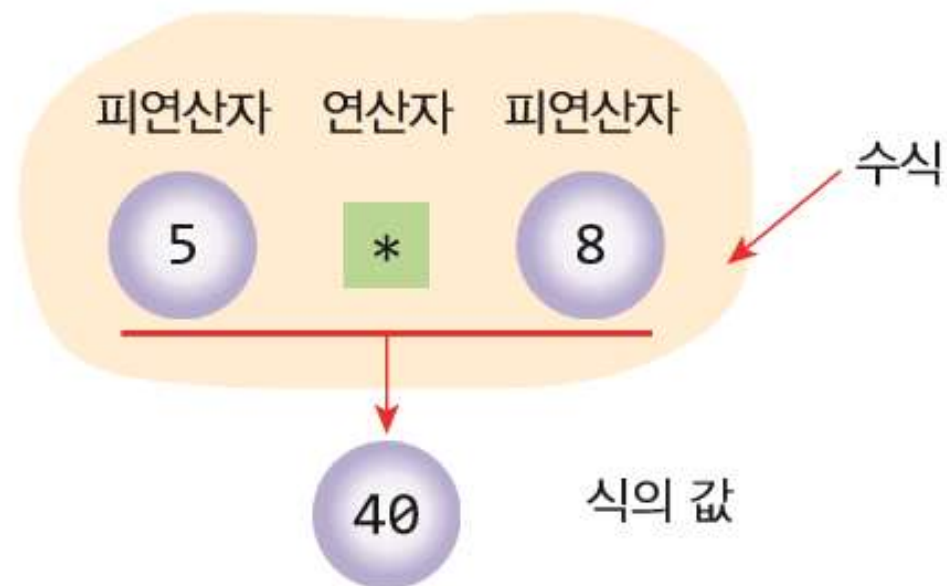
- 우리가 즐겨보는 영화의 컴퓨터 그래픽 장면들이 컴퓨터의 계산 기능을 통하여 이루어진다는 것은 아주 흥미롭다. 예를 들어서 건물들의 폭발 장면은 물리학의 여러 가지 공식들을 이용하여 컴퓨터로 계산한 결과를 화면에 표시하는 것이다.



출처: 영화 어벤저스



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





- 덧셈, 뺄셈, 곱셈, 나눗셈, 나머지 연산

연산자	기호	사용례	결괏값
덧셈	+	$7 + 4$	11
뺄셈	-	$7 - 4$	3
곱셈	*	$7 * 4$	28
나눗셈	//	$7 // 4$	1
나눗셈	/	$7 / 4$	1.75
나머지	%	$7 \% 4$	3



```
>>> 7 / 4  
1.75
```

```
>>> 7 // 4  
1
```

## WARNING

파이썬 버전 2.X에서는 / 연산자의  
결과가 정수가 됩니다. 주의하  
세요!



```
p = int(input("분자를 입력하시오: "))  
q = int(input("분모를 입력하시오: "))  
print("나눗셈의 몫=", p // q)  
print("나눗셈의 나머지=", p % q)
```

분자를 입력하시오: 7  
분모를 입력하시오: 4  
나눗셈의 몫= 1  
나눗셈의 나머지= 3



- 짝수와 홀수의 구분

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

```
정수를 입력하시오: 28  
0
```





- 초단위의 시간을 받아서 몇 분 몇 초인지를 계산하여 보자.

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

16 40



- 우리가 커피 전문점을 내려고 한다. 다음과 같은 커피 메뉴가 있을 때, 얼마나 많은 매출을 올릴 수 있을 지 계산해보고자 한다.



아메리카노 판매 개수: 10  
카페라떼 판매 개수: 20  
카푸치노 판매 개수: 30  
총 매출은 185000 입니다.



```
americano_price = 2000  
cafelatte_price = 3000  
capucino_price = 3500
```

```
americanos = int(input("아메리카노 판매 개수: "))  
cafelattes = int(input("카페라떼 판매 개수: "))  
capucinos = int(input("카푸치노 판매 개수: "))
```

```
sales = americanos*americano_price  
sales = sales + cafelattes*cafelatte_price  
sales = sales + capucinos*capucino_price  
print("총 매출은", sales, "입니다.")
```



## 도전문제

하라.

총 재료 비용이 100000원이었다고 하자. 이익을 계산해보자. 적자인지, 흑자인지를 표시



## Lab: 화씨 온도를 섭씨 온도로 변환하기

- 화씨 온도를 받아서 섭씨 온도로 바꾸는 프로그램을 작성해보자.

$$C = (F - 32) * \frac{5}{9}$$



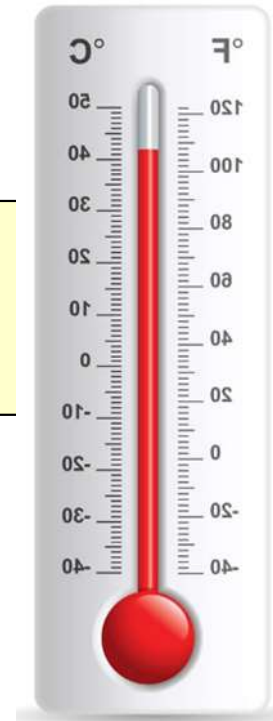
화씨 온도: 100

섭씨 온도: 37.77777777777778



# Solution

```
ftemp = int(input("화씨 온도: "))  
ctemp = (ftemp-32.0)*5.0/9.0  
print("섭씨 온도:", ctemp)
```

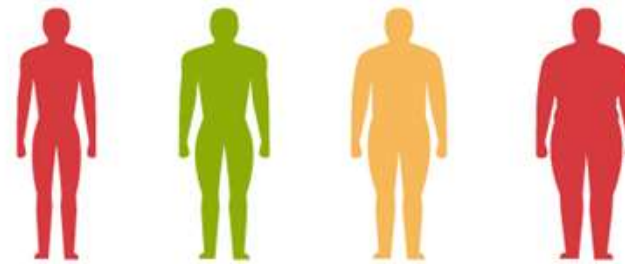


## 도전문제

반대로 섭씨온도를 화씨온도로 변환하는 프로그램도 작성해보자.



- 사용자로부터 신장과 체중을 입력받아서 BMI 값을 출력하는 프로그램을 작성하여 보자.



BMI Chart

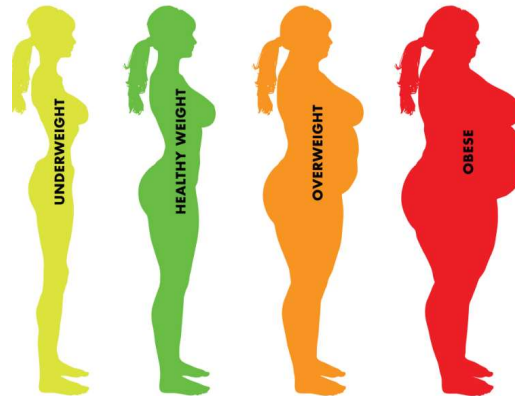
$$\text{BMI} = \frac{(\text{weight in kilograms})}{\text{height in meters}^2}$$

BMI less than 18.50	Underweight
BMI 18.50 - 24.99	Healthy weight
BMI 25.00 - 29.99	Overweight
BMI 30 or more	Obese

몸무게를 kg 단위로 입력하시오: 85.0  
키를 미터 단위로 입력하시오: 1.83  
당신의 BMI= 25.381468541909282



```
weight = float(input("몸무게를 kg 단위로 입력하시오: "))  
height = float(input("키를 미터 단위로 입력하시오: "))  
  
bmi = (weight / (height**2))  
print("당신의 BMI=", bmi)
```





## Lab: 자동 판매기 프로그램 1

- 자동 판매기를 시뮬레이션하는 프로그램을 작성해보자. 자동 판매기는 사용자로부터 투입한 돈과 물건값을 입력 받는다. 물건값은 100원 단위라고 가정한다. 프로그램은 잔돈을 계산하여 출력한다. 자판기는 동전 500원, 100원 짜리만 가지고 있다고 가정하자.

투입한 돈: 5000

물건값: 2600

거스름돈: 2400

500원 동전의 개수: 4

100원 동전의 개수: 4







```
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)
```



## 도전문제

자판기가 만약 50원짜리 동전과 10원짜리 동전도 거슬러 줄 수 있다면 위의 코드를 어떻게 수정하여야 하는가?



- 지수(power)를 계산하려면 \*\* 연산자를 사용한다.

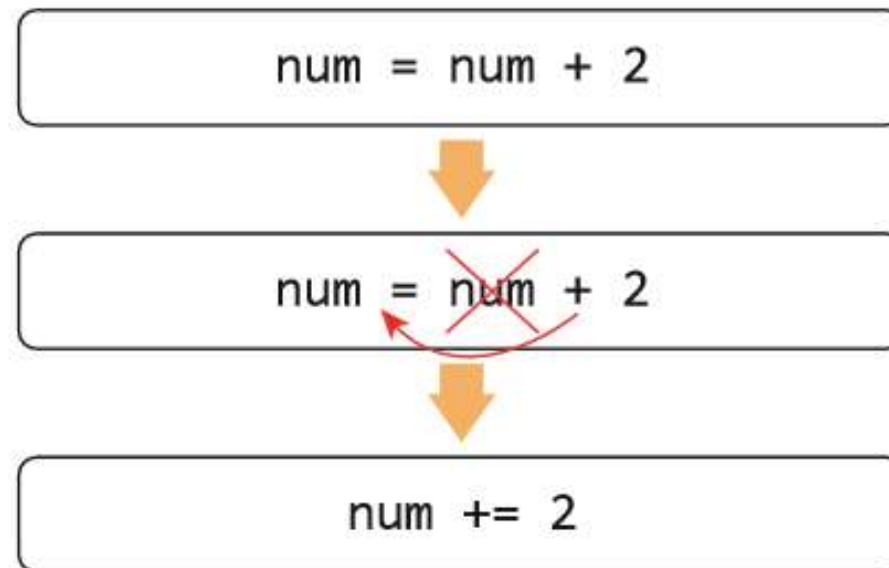
```
>>> 2 ** 7  
128
```

- 원리금 계산: 원리금 = 원금\*(1+이자율)<sup>년</sup>

```
>>> a=1000  
>>> r=0.05  
>>> n=10  
>>> a*(1+r)**n  
1628.894626777442
```



- 복합 연산자(compound operator)
  - +=처럼 대입 연산자와 다른 연산자를 합쳐 놓은 연산자이다.





복합 연산자	의미
$x += y$	$x = x + y$
$x -= y$	$x = x - y$
$x *= y$	$x = x * y$
$x /= y$	$x = x / y$
$x \% = y$	$x = x \% y$



```
x = 1000  
print("초기값 x=", x)  
x += 2  
print("x += 2 후의 x=", x)  
x -= 2  
print("x -= 2 후의 x=", x)
```

초기값 x= 1000  
x += 2 후의 x= 1002  
x -= 2 후의 x= 1000



## ■ 주석(comment)

- 소스 코드에 붙이는 설명글과 같은 것이다. 주석은 프로그램이 하는 일을 설명한다. 주석은 프로그램의 실행 결과에 영향을 끼치지 않는다.

```
# 사용자로부터 화씨온도를 입력받는다.
```

```
ftemp = int(input("화씨 온도: "))
```

```
ctemp = (ftemp-32.0)*5.0/9.0
```

```
print("섭씨 온도:", ctemp)
```

```
# 화씨 온도->섭씨 온도
```

```
# 섭씨 온도를 화면에 출력한다
```





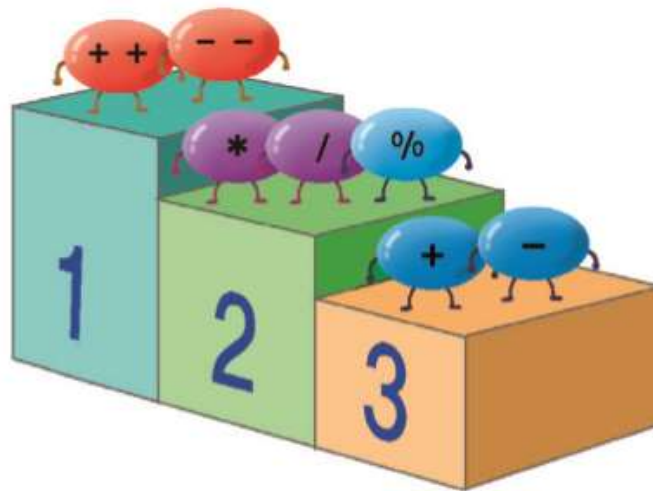
# 연산자의 우선 순위

$$x + \underbrace{y * z}_{\textcircled{1}}$$

$\underbrace{\hspace{10em}}_{\textcircled{2}}$

$$\underbrace{(x + y) * z}_{\textcircled{1}}$$

$\underbrace{\hspace{10em}}_{\textcircled{2}}$





```
>>> 10 + 20 / 2
```

```
20.0
```

```
>>> (10 + 20) / 2
```

```
15.0
```





# 우선 순위표

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



- 평균을 구하고자 한다. 잘못된 부분은 어디일까?

```
x = int(input("첫 번째 수를 입력하시오: "))  
y = int(input("두 번째 수를 입력하시오: "))  
z = int(input("세 번째 수를 입력하시오: "))  
avg = x + y + z / 3  
print("평균 =", avg)
```

```
첫 번째 수를 입력하시오: 10  
두 번째 수를 입력하시오: 20  
세 번째 수를 입력하시오: 30  
평균 = 40.0
```



```
x = int(input("첫 번째 수를 입력하시오: "))  
y = int(input("두 번째 수를 입력하시오: "))  
z = int(input("세 번째 수를 입력하시오: "))  
avg = (x + y + z) / 3  
print("평균 =", avg)
```

첫 번째 수를 입력하시오: 10  
두 번째 수를 입력하시오: 20  
세 번째 수를 입력하시오: 30  
평균 = 20.0



## ■ 관계 연산자

- 어떤 것이 큰지, 작은지, 같은지를 비교하는 것, 결과는 참(True)이나 거짓(False)
- 주로 조건문(if )이나 반복문(for, while)에서 사용

$a < b = \begin{cases} \text{참} : \text{True} \\ \text{거짓} : \text{False} \end{cases}$

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



- 관계 연산자 예

```
a,b = 100,200  
print(a == b, a != b, a > b, a < b, a >= b, a <= b)
```

출력 결과

```
False True False True False True
```

- a와 b를 비교하기 위한 관계 연산자 ==를 사용시 =을 하나만 쓰는 경우 → 오류발생

a = b는 b의 값을 a에 대입하라는 의미이지 관계 연산자가 아님

```
print(a = b)
```



## ■ 논리 연산자

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

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



- 논리 연산자 예

```
a = 99  
(a > 100) and (a < 200)  
(a > 100) or (a < 200)  
not(a == 100)
```

출력 결과

```
False True True
```



## ■ 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

- 10 & 7



- 10 & 7의 결과는 2
- 10진수를 2진수로 변환한 다음 각 비트마다 AND 연산을 수행하기 때문에 그 결과 2진수  $0010_2$ 이 되고 10진수로는 2가 됨



- 비트 논리곱 예

10 & 7

123 & 456

0xFFFF & 0x0000

출력 결과

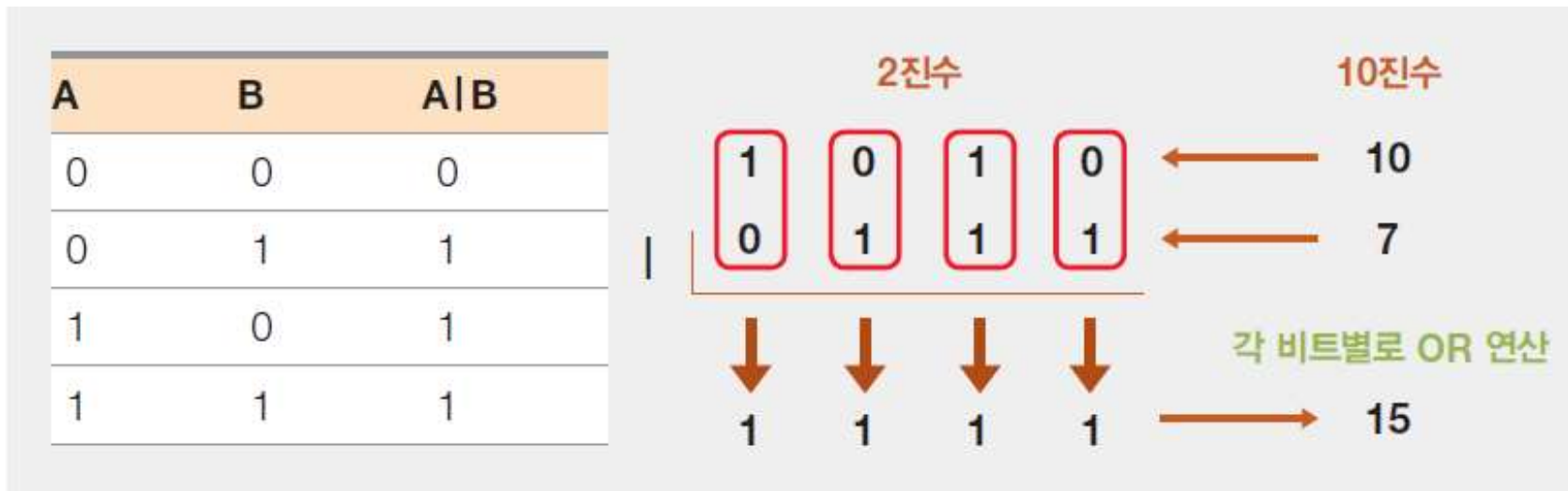
2 72 0

- 123 & 456은 123의 2진수인  $1111011_2$ 과 456의 2진수인  $111001000_2$ 의 비트 논리곱(&) 결과는  $1001000_2$ 이므로 10진수로 72가 나옴
- 0xFFFF & 0x0000은 16진수 FFFF(2진수는 1111 1111 1111 1111)와 0000(2진수는 0000 0000 0000 0000)의 비트 논리곱(&) 결과인 0이 출력. 0과 비트 논리곱을 수행하면 어떤 수든 무조건 0이 나옴



## ■ 비트 논리합(OR) 연산자

- 10 | 7



- 각 비트의 논리합 결과는  $1111_2$ 이며 이는 10진수 15가 됨



10 | 7

123 | 456

0xFFFF | 0x0000

출력 결과

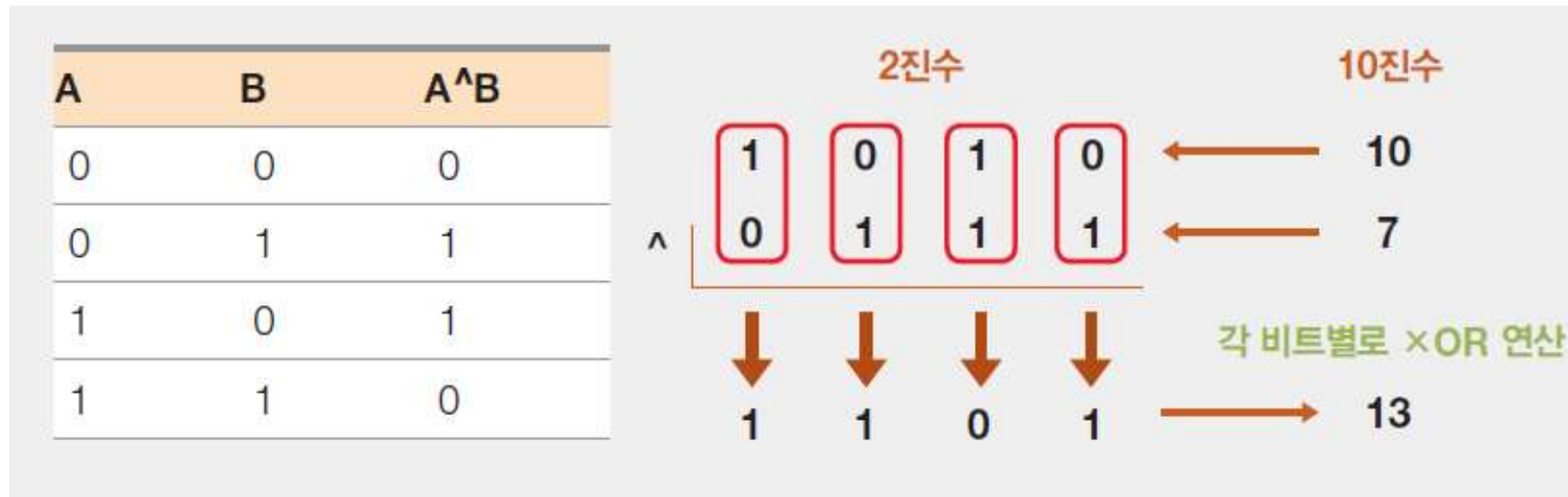
15 507 65535

- 123 & 456은 123의 2진수인  $001111011_2$ 과 456의 2진수인  $111001000_2$ 의 비트 논리합( $\wedge$ ) 결과는  $111111011_2$ 이므로 10진수로 507가 나옴
- 0xFFFF|0x0000을 보면 0xFFFF와 0000의 비트 논리합은 0xFFFF 임. 그러므로 16진수 FFFF16는 10진수 65535가 됨



## ■ 비트 배타적 논리합(^) 연산자

- 두 값이 다르면 1, 같으면 0. 즉 참(1)^참(1)이나 거짓(0)^거짓(0)이면 결과는 거짓(0)이고, 참(1)^거짓(0)이나 거짓(0)^참(1)이면 그 결과는 참(1)



- 10과 7의 각 비트 배타적 논리합 결과는  $1101_2$ 이며 이는 10진수로 13임



$10 \wedge 7$

$123 \wedge 456$

$0xFFFF \wedge 0x0000$

출력 결과

13 435 65535

- $123 \wedge 456$ 은 123의 2진수인  $001111011_2$ 과 456의 2진수인  $111001000_2$ 의 비트 배타적 논리합( $\wedge$ ) 결과는  $110110011_2$ 이므로 10진수로 435가 나옴
- $0xFFFF|0x0000$ 을 보면  $0xFFFF$ 와  $0000$ 의 비트 배타적 논리합은  $0xFFFF$  임. 그러므로 16진수 FFFF16는 10진수 65535가 됨



## ■ 비트 부정(~) 연산자

- 두 수에 대해 연산하는 것이 아니라, 각 비트를 반대로 만드는 연산자
- 즉 각 비트에 대해 0은 1로, 1은 0으로 바꿈. 예로 0000을 비트 부정 연산하면 1111이 되고, 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

$a = 12345$

$\sim a + 1$

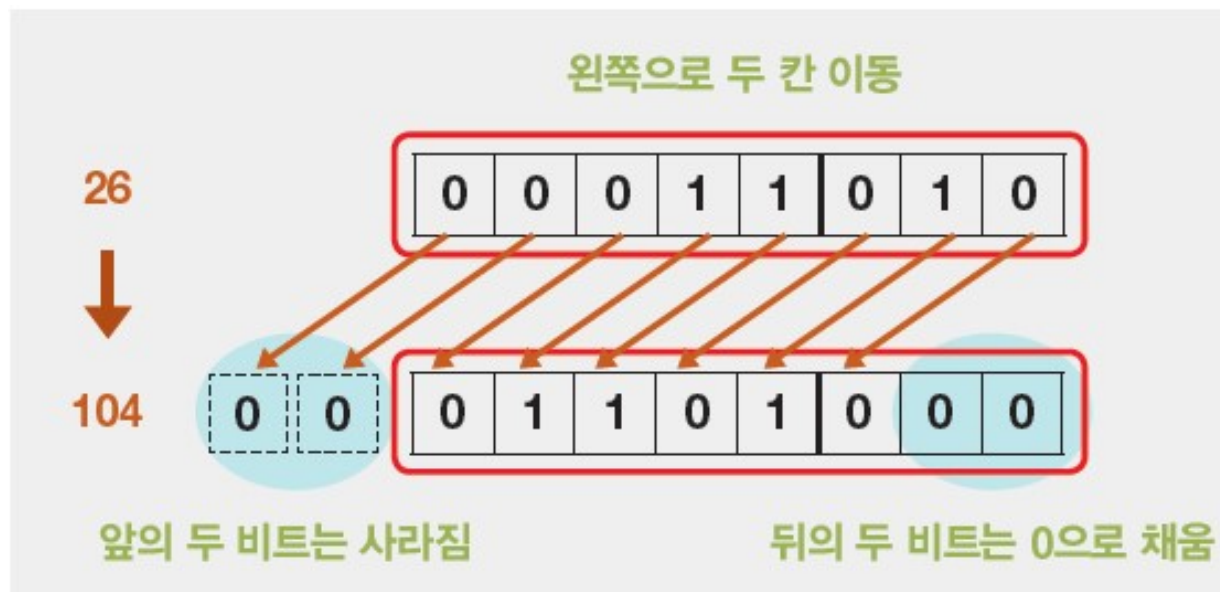
출력 결과

-12345



## ■ 왼쪽 시프트(<<) 연산자

- 나열된 비트를 왼쪽으로 시프트(Shift)해주는 연산자



- 앞의 두 00은 떨어져나가고, 뒤에 빈 두 칸에는 00이 채움. 결과는 26에서 104로 바뀌었는데, 이는 왼쪽으로 시프트 할 때마다  $2^n$ 을 곱한 효과가 나기 때문임 ( $26 \times 2^2 = 104$ ). 즉 왼쪽으로 1회 시프트 할 때는  $2^1$ 을, 2회에는  $2^2$ 을, 3회에는  $2^3$ 을 곱한 효과가 남





- 왼쪽 시프트(<<) 연산자 예

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

출력 결과

```
20 40 80 160
```

- 시프트 할 때마다  $10 \times 2^1 = 20$ ,  $10 \times 2^2 = 40$ ,  $10 \times 2^3 = 80$ ,  $10 \times 2^4 = 160$ 의 결과가 나옴



## ■ 오른쪽 시프트(>>) 연산자

- 나열된 비트를 오른쪽으로 시프트(Shift)해주는 연산자



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



- 오른쪽 시프트(>>) 연산자 예

```
a = 10
```

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

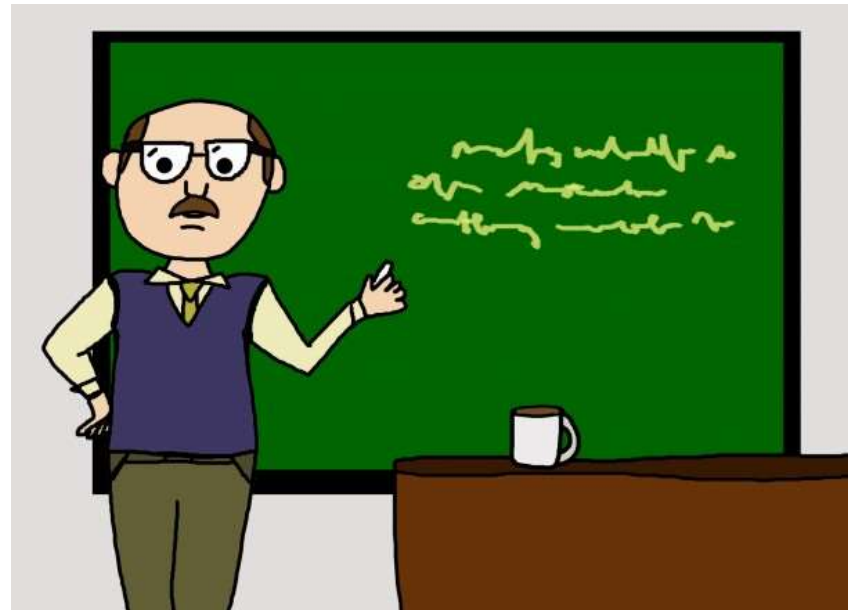
출력 결과

```
5 2 1 0
```

- 시프트 할 때마다  $10/2^1=5$ ,  $10/2^2=2$ ,  $10/2^3=1$ ,  $10/2^4=0$ 의 결과가 나옴



- 수식은 피연산자와 연산자로 이루어진다.
- 덧셈, 뺄셈, 곱셈, 나눗셈을 위하여  $+$ ,  $-$ ,  $*$ ,  $/$  기호를 사용한다.
- 지수 연산자는  $**$ 이다.
- 나눗셈에서 몫을 계산하려면  $//$  연산자를 사용한다.
- 나눗셈에서 나머지를 계산하려면  $%$  연산자를 사용한다.
- $*$ 와  $/$ 가  $+$ 와  $-$ 보다 우선순위가 높다.
- 연산자의 우선 순서를 변경하려면 괄호를 사용한다.
- 관계 연산자
- 논리 연산자
- 2진수 값으로 처리되는 비트 연산자



**THANK**  

---

**YOU**