



PYTHON



Python

Programming

- 담당교수 : 최희식
- eMail : choihs3054@seoultech.ac.kr

이번 주 학습

■ 학습목표

- 파이썬 다양한 연산자에 대해 학습한다.
- 복합 대입 연산자, 관계 연산자, 논리 연산자에 대해 학습한다.
- 비트 연산자, 쉬프트 연산자, 삼항 연산자에 대해 학습한다.

■ 학습목차

- 복합 대입 연산자, 관계 연산자, 논리 연산자
- 비트 연산자, 쉬프트 연산자
- 삼항 연산자

1

이번
차시에서는

- 복합 대입 연산자
- 관계 연산자
- 논리 연산자



1. 복합 대입 연산자

■ 복합 대입 연산자 : 산술연산자와 대입연산자를 합쳐 놓은 연산자

복합 대입 연산자	사용 예	수행 결과
<code>+=</code>	<code>a += 5</code>	a에 5를 더한 후, a 변수에 저장
<code>-=</code>	<code>a -= 5</code>	a에서 5를 뺀 후, a 변수에 저장
<code>*=</code>	<code>a *= 5</code>	a에 5를 곱한 후, a 변수에 저장
<code>/=</code>	<code>a /= 5</code>	a를 5로 나눈 후, a 변수에 저장
<code>//=</code>	<code>a //= 5</code>	a를 5로 나눈 후, 몫만 a 변수에 저장
<code>%=</code>	<code>a %= 5</code>	a를 5로 나눈 후, 나머지 값만 a 변수에 저장
<code>**=</code>	<code>a **= 5</code>	a를 5번 거듭제곱 계산 후, a 변수에 저장

1. 복합 대입 연산자

- 복합 대입 연산자 : $a=a+1$ 연산식을 $a+=1$ 로 축약해서 사용할수 있음

```
>>> a=10
>>> a+=5; print(f"a+5를 더한 후 값={a}")
a+5를 더한 후 값=15
>>> a-=5; print(f"a+5를 뺀 후 값={a}")
a+5를 뺀 후 값=10
>>> a*=5; print(f"a+5를 곱한 후 값={a}")
a+5를 곱한 후 값=50
>>> a/=5; print(f"a+5를 나눈 후 값={a}")
a+5를 나눈 후 값=10.0
>>> a//=5; print(f"a+5를 나눈 후 몫 값={a}")
a+5를 나눈 후 몫 값=2.0
>>> a%=5; print(f"a+5를 나눈 후 나머지 값={a}")
a+5를 나눈 후 나머지 값=2.0
>>> a**=5; print(f"a+5를 나눈 후 거듭제곱 값={a}")
a+5를 나눈 후 거듭제곱 값=32.0
```

[실습]

파이썬에서 복합 대입 연산자를 이용하여 문자열 출력도 가능하다.

다음 문자열 변수를 이용하여 [실행 결과]와 같이 출력하시오.

```
str1="Python"  
str2="Programming"  
str3="is"  
str4="a amazing Language."
```

[실행 결과]

Python Programming is a amazing Language.

2. 관계 연산자

■ 관계 연산자 : 관계 연산자는 대소 크기를 비교하는 연산자

관계 연산자	의미	사용 예
>	크다(초과)	a > b
<	작다(미만)	a < b
>=	크거나 같다(이상)	a >= b
<=	작거나 같다(이하)	a <= b
==	같다(동등)	a == b
!=	같지 않다	a != b

2. 관계 연산자

■ 관계 연산자

```
>>> a=10
>>> b=5
>>> print(a>b)
True
>>> print(a<b)
False
>>> print(a>=b)
True
>>> print(a<=b)
False
>>> print(a==b)
False
>>> print(a!=b)
True
```

3. 논리 연산자

■ 논리 연산자 : 연산 결과에 따라 참인지 거짓인지를 판단

논리 연산자	설명	사용 예
and	논리곱으로 두 가지 조건이 모두 참이어야 참	a=15 (a>10) and (a<20)
or	논리합으로 두 조건 중 하나만이라도 참이 되면 참	a=15 (a==10) or (a<20)
not	논리 부정으로 참이면 거짓, 거짓이면 참을 수행	a=15 not(a==10)

3. 논리 연산자

■ 논리 연산자 : and

- and 연산자는 조건 a 와 조건 b가 모두 참이어야만 True를 반환

```
a=100
```

```
b=200
```

```
>>> print(a<b and a==110)
```

```
False
```

```
>>> print(a==100 and a<b)
```

```
True
```

3. 논리 연산자

■ 논리 연산자 : or

- or 연산은 조건 a, b 둘 중에 하나만이라도 참이면 True를 반환

```
a=100
```

```
b=200
```

```
>>> print(a<b or a<10)
```

```
True
```

```
>>> print(a>b or b>300)
```

```
False
```

3. 논리 연산자

■ 논리 연산자 : not

- not 연산자는 a가 거짓이면 True를 반환하고, a가 참이면 False를 반환

```
>>> a=100
```

```
>>> print(not a)
```

```
False
```

```
>>> print(not False)
```

```
True
```

[실습]

각 과목 점수는 다음과 같다. 각 과목 모두 80점 이상인지 판별하여 결과 값을 출력하시오.

```
kor=90, eng=80, math=100
```

[실행 결과]

```
>>> _____
```

True

[실습]

사용자로 부터 국어, 영어, 수학 점수를 입력받아, 관계연산자, 논리 연산자를 이용하여 각 과목 40점 이상, 평균이 60점 이상인 경우, [실행 결과]와 같이 조건 결과를 True/False로 반환하여 출력하는 프로그램을 작성하시오.

[실행 결과]

국어 점수? 70

영어 점수? 35

수학 점수? 80

국어 : 70점, 영어 : 35점, 수학 : 80점, 평균 : 61.67점 결과 : False

감사합니다.



2

이번
차시에서는

- 비트 연산자
- 쉬프트 연산자



4. 비트 연산자

■비트 연산자

- 비트 연산자는 0과 1로만 처리되는 연산자로 비트 연산자 종류로는 &, |, ~, ^, <<, >> 등이 있다.
- 비트 연산을 수행하기 위해 2진수로 먼저 변환 후, 각 비트와 비트끼리 연산을 수행한다.

연산자	의미	설명
&	비트 논리곱(and)	두 비트 중 모두 1이어야만 1
	비트 논리합(or)	두 비트 중 하나만이라도 1이면 1
~	비트 부정(not)	1은 0으로, 0은 1로 변환
^	비트 배타적 합(xor)	같은 비트는 0, 다른 비트는 1
<<	비트 좌시프트	비트를 왼쪽으로 시프트
>>	비트 우시프트	비트를 오른쪽으로 시프트

4. 비트 연산자

■비트 논리곱(&) : 4 & 6에 대한 비트 논리곱 계산

```
>>> print(4&6)
```

```
4
```

```
>>> print(bin(4&6))
```

```
0b100
```

■비트 논리곱(&) 연산으로 수행

0100

0110

0100이 된다. 이것을 십진수로 바꾸게 되면

$0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$ 이 된다.

$= 0 + 4 + 0 + 0$

$= 4$ 가 된다.

4. 비트 연산자

■비트 논리합 : (|) : 4 | 6에 대한 비트 논리합 계산

```
>>> print(4|6)
```

```
6
```

```
>>> print(bin(4|6))
```

```
0b110
```

■비트 논리합(|) 연산으로 수행

0100

0110

0110이 된다. 이것을 십진수로 바꾸게 되면

$0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$ 이 된다.

$= 0 + 4 + 2 + 0$

$= 6$ 이 된다.

4. 비트 연산자

■비트 배타적 합 : (^) : 4 ^ 6에 대한 비트 배타적 합 계산

```
>>> print(4^6)
```

```
2
```

```
>>> print(bin(4^6))
```

```
0b10
```

■비트 xor 연산으로 수행

0100

0110

0010이 된다. 이것을 십진수로 바꾸게 되면

$0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$ 이 된다.

$= 0 + 0 + 2 + 0$

$= 2$ 가 된다.

5. 쉬프트 연산자

■ 쉬프트 연산자 : 쉬프트 연산은 왼쪽 쉬프트와 오른쪽 쉬프트가 있다.

연산자	의미	설명
<<	왼쪽 쉬프트 연산자	<ul style="list-style-type: none">어떤 값을 왼쪽으로 지정된 비트 수만큼 이동왼쪽으로 1비트 이동할 때마다 2배씩 늘어난다.왼쪽 쉬프트는 2^n 곱한 것과 같은 효과와 같다. 즉, 값 << 2 ➡ 값 * 2^n
>>	오른쪽 쉬프트 연산자	<ul style="list-style-type: none">어떤 값을 오른쪽으로 지정된 비트 수만큼 이동오른쪽으로 1비트 이동할 때마다 1/2씩 줄어든다.2^n으로 나눈 것과 같은 효과와 같다. 즉, 값 >> 2 ➡ 값 / 2^n

5. 쉬프트 연산자

■왼쪽 쉬프트 연산자 : 12 << 2

```
>>> print(12<<2)
```

```
48
```

```
>>> print(bin(12<<2))
```

```
0b110000
```

우선 10진수 12를 2진수로 변환하면 1100이 된다.

0000 1100이 된다. 이 때 << 2 만큼 좌쉬프트를 이동하기 위해서 맨 좌측에 있는 2비트가 왼쪽으로 소멸된다.

① 00 1100이 된다.

② 위 숫자를 재정비하면 0011 00이 된다.

③ 그리고 맨 뒤 2비트만큼은 0으로 채워진다.

④ 이를 재정비하면 0011 0000이 된다.

5. 쉬프트 연산자

■오른쪽 쉬프트 연산자 : 12 >> 2

```
>>> print(12>>2)
```

```
3
```

```
>>> print(bin(12>>2))
```

```
0b11
```

우선 10진수 12를 2진수로 변환하면 1100이 된다.

0000 1100이 된다. 이 때 >> 2 만큼 오른쪽 쉬프트를 이동하기 위해서 맨 우측에 있는 2비트가 오른쪽으로 소멸된다.

① 0000 11이 된다.

② 위 숫자를 재정비하면 00 0011이 된다.

③ 이 때 상위 비트가 양수일 때는 0, 음수일 때는 1로 소멸된 비트수 만큼 채우게 되는데
상위비트가 양수이므로 맨 앞 2비트만큼은 0으로 채워지게 된다.

④ 이를 재정비하면 0000 0011이 된다.

6. 파이썬 진법 계산

■파이썬에서 진법 계산

- 대부분의 숫자 데이터는 10진으로 처리되지만 컴퓨터 처리 언어인 숫자로 표현하기 위해 진법(2진수, 8진수, 16진수) 변환 처리가 가능하다.
- 2진수 : bin()
- 8진수 : oct()
- 16진수 : hex()

```
>>> print(bin(12))
```

```
0b1100
```

```
>>> print(oct(12))
```

```
0o14
```

```
print(hex(12))
```

```
0xc
```

6. 파이썬 진법 계산

■format() 함수를 이용한 진법 변환

- 출력문에서 format()를 이용하여 진법을 쉽게 변환하여 출력할 수 있다.

```
>>> num=12
```

```
>>> print("%o" %num)
```

```
14
```

```
>>> print(format(num, "o"))
```

```
0o14
```

6. 파이썬 진법 계산

■format() 함수를 이용한 진법 변환

- 출력문에서 format()를 이용하여 진법을 쉽게 변환하여 출력할 수 있다.

```
>>> num=12
```

```
>>> print("%x" %num)
```

```
c
```

```
>>> print(format(num, "#x"))
```

```
0xc
```

[실습]

- 사용자로부터 어떤 수를 각각 입력 받아 앞의 수는 데이터 값으로 사용하고, 뒤의 숫자는 지수 값으로 사용하여 왼쪽 쉬프트 값을 출력하는 프로그램을 작성하시오.

[실행 결과]

어떤 값 : 1

왼쪽 쉬프트 : 3

왼쪽 쉬프트 $1 \ll 3 = 8$

감사합니다.



3

이번
차시에서는

- 삼항 연산자



7. 삼항 연산자

■삼항 연산자

- 복잡한 if~else 조건에 대한 조건문을 삼항 연산자로 사용하게 되면 간단하고 빠르게 결과 값을 처리할 수 있는 장점이 있다.
- 조건 절을 기준으로 비교 판단한 후, [True]에는 조건이 참인 경우 값 또는 수식을 입력, [False]에는 조건이 거짓인 경우 값 또는 수식을 입력하여 간단하게 조건 처리를 수행한다.

변수 = [True 값] if [조건 절] else [False 값]

7. 삼항 연산자

■삼항 연산자 : 3개의 변수에 삼항 연산자로 가장 큰 값 비교

```
>>> num1=125
```

```
>>> num2=100
```

```
>>> num3=120
```

```
>> max=num1 if num1 > num2 else num2
```

```
>> max=num3 if num3 > max else max
```

```
>>> print("가장 큰 값",max)
```

```
가장 큰 값 125
```


[실습]

사용자로부터 어떤 숫자 하나를 입력 받아, 입력한 숫자가 3의 배수인지를 판별하는 프로그램을 삼항 연산자를 이용하여 작성하시오.

[실행 결과]

어떤 숫자 :24

당신이 입력한 숫자 24는(은) 3의 배수 입니다.

어떤 숫자 :16

당신이 입력한 숫자 16는(은) 3의 배수가 아닙니다.

[실습]

사용자로부터 물건 가격을 입력받아, 물건 가격이 6만원 이상인 경우 할인 대상에 포함될 수 있는지를 판별하는 프로그램을 삼항 연산자를 이용하여 작성하시오.

[실행 결과]

상품 가격 ? 75000

할인 대상 : True

상품 가격 ? 42000

할인 대상 : False

[실습]

사용자로부터 나이를 입력 받아, 영화를 볼 수 있는 등급인지를 판별하는 프로그램을 삼항 연산자를 이용하여 작성하시오.

[실행 결과]

나이? 17

이 등급의 영화를 볼 수 있습니다.

나이 :15

이 등급의 영화를 볼 수 없습니다.

감사합니다.

