Lecture 3. 연산자와 명제 논리

BAE SEKANG(배세강)

Contents

Operators

♣ 산술 연산자(Arithmetic Operators)

연산자	내용	예시
+	덧셈	4 + 2
-	뺄셈	10 - 5
*	곱셈	2 * 10
/	나누기	5 / 3
//	몫	5 // 3
%	나머지	5 % 3
**	제곱	3 ** 2

Operators

♣ 산술 연산자(Arithmetic Operators)

/ 연산의 결과는 float임에 주의

```
# /의 결과
print(5/2)

# //의 결과
print(5//2)

# %의 결과
print(5%2)
```

```
>> 2.5

>> 2

>> 1
```

Operators

♣ 비교 연산자(Comparison Operators)

연산자	내용	예시
==	같다	5 == 5
!=	다르다	3 != 1
>	크다	5 > 2
<	작다	5 < 2
>=	크거나 같다	5 >= 2
<=	작거나 같다	5 <= 2
is	같다	a is 100

Operators

♣ 비교 연산자(Comparison Operators)

비교 연산의 결과는 True 또는 False

```
print("가"=="나")
print(10 != 20)
print(5 + 3 < 10)
print("a" < "b")
print(100 <= 100)
print(False == True)
a = 5
print(a == 3)
```

```
>> False
>> True
>> True
>> True
>> True
>> False

>> False
```

Operators

♣ Integer Interning

작은 정수 영역(-5 ~ 256)을 메모리에 미리 딱 하나만 만들어두고 재사용

```
a = 100
b = 100

print(a is b)

# (PEP 8) None을 비교할 때는 ==보다 is를 쓰는 것을 권장
```

>> True

Operators

♣ 논리 연산자(Logical Operators)

연산자	내용	예시
or	둘 중 하나만 참이면 참	x == 1 or y > 5
and	둘 다 참이어야 참	2 < 4 and True
not	논리 부정	not False

Operators

♣ 논리 연산자(Logical Operators)

(기본적으로)논리 연산의 결과는 True 또는 False

```
p = 5 < 7
q = 2 ** 5 == 32
r = True
print(p)
print(q)
print(p or q)
print(p and q)
print(not r)
print("python" and 3)
```

```
>> True
>> True
>> True
>> True
>> False
>> 3
```

Operators

♣ Truthy / Falsy

- Truthy: 값이 들어 있는 웬만한 모든 것들
- Falsy: “”, 0, [], None, …

```
print(int(“”))

print(“python” and 10)
```

>> 10

- and: 결과를 최종적으로 확정 지은 마지막 값을 그대로 return (Short-circuit Evaluation)

Operators

♣ 비트와 진법

- 비트(Bit): 정보를 표현하는 최소 단위. 즉 0 또는 1

- n진법: 수를 표현할 때, n개의 문자를 활용하는 방법
예) 10진법은 0~9로 모든 수를 표현.

- 10진수(decimal): 0~9를 활용하는 수 체계
- 2진수(binary): 0과 1 두 개의 숫자만 활용하는 수 체계

Operators

♣ 자릿값 표현

- b진법일 때,

$$(d_n d_{n-1} \dots d_1 d_0)_b = d_n \cdot b^n + d_{n-1} \cdot b^{n-1} + \cdots + d_1 \cdot b^1 + d_0 \cdot b^0$$

e.g.) $(135)_{10} = 1 \times 10^2 + 3 \times 10^1 + 5 \times 10^0$

e.g.) $(101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

Operators

♣ 비트 연산자

- 정수(int)에 대한 비트 단위 연산

연산자	이름	내용	예시
|	OR	두 비트 중 하나가 1이면 1	3 | 1
&	AND	두 비트가 1이면 1	6 & 2
~	NOT	모든 비트를 반전함	~5
^	XOR	두 비트가 다르면 1, 같으면 0	4 ^ 3
<<	Left Shift	왼쪽으로 비트를 shift함	5 << 2
>>	Right Shift	오른쪽으로 비트를 shift함	5 >> 2

Operators

♣ 비트 연산자

- OR 연산

```
# 0010 | 0110 -> 0110
print(2 | 6)

# 1100 | 0011 -> 1111
print(12 | 3)
```

>> 6

>> 15

Operators

♣ 비트 연산자

- AND 연산

```
# 0010 & 0110 -> 0010
print(2 & 6)

# 1100 & 0011 -> 0000
print(12 & 3)
```

>> 2

>> 0

Operators

♣ 비트 연산자

- NOT 연산

```
# ~(0010) -> 1101
print(~2)
```

>> -3

Q. 왜 음수로 나올까?

→ 2의 보수 체계를 활용하기 때문

Operators

♣ Infinite Precision Integer

- Python에서는 메모리가 허용하는 한 숫자를 무한대로 늘릴 수 있다.
  ➔ 부호 비트의 개념이 물리적으로 정해져 있지 않고, 내부적으로 음수 기호를 따로 관리 한다.
- 2의 보수 형태를 보려면 비트 연산자를 활용

```
# 이진수 형태로 출력하는 bin()
print(bin(5))

# -가 그대로 붙어서 출력
print(bin(-5))
```

>> 0b101

>> -0b101

Operators

♣ 2의 보수(2's complement)

- 어떤 숫자의 2의 보수를 구하는 방법

STEP1. 해당 수를 이진수로 나타낸다.
STEP2. 비트를 반전한다.
STEP3. 1을 더한다.

STEP1 ~ STEP3을 거쳐 나온 비트가 처음 수의 2의 보수에 해당한다.

e.g.) 6 -> 0110 -> 1001 -> 1010

Operators

♣ 2의 보수(2's complement)

- 2의 보수로 음수를 표현할 수 있다.
- 6을 2진수로 나타내면 0110이고, 2의 보수는 1010이다. 즉, 1010은 -6을 나타내는 이진수 표현이다.
- 6 + (-6)은 0인 것처럼, 0110과 1010을 더하면 0000을 얻을 수 있다.

```
  6: ···0000000000110
 -6: ···1111111111010
-----------------------
  0: ···0000000000000
```

Operators

♣ 2의 보수(2's complement)

- Python은 정수를 개념적인 무한 길이의 2의 보수로 취급한다.
- ~x의 결과는 -(x+1)의 결과와 같다.

Operators

♣ 비트 연산자

- XOR 연산

```
# 0010 ^ 0110 -> 0100
print(2 ^ 6)

# 1100 ^ 0011 -> 1111
print(12 ^ 3)
```

>> 4

>> 15

Operators

♣ 비트 연산자

- Left shift 연산: x << k는 x의 비트를 왼쪽으로 k칸 밀고, 오른쪽은 0으로 채운다.
  e.g.) 15 << 1의 결과는 30, -3 << 2의 결과는 -12

- Right shift 연산: x >> k는 x의 비트를 오른쪽으로 k칸 밀고, 왼쪽은 부호 비트로 채운다.
  e.g.) 15 >> 1의 결과는 7, -3 >> 2의 결과는 -1

Operators

♣ 대입 연산자

연산자	예시
=	x=5이면, x에 5를 대입한다.
+=	x +=5이면, x+5의 결과를 x에 대입한다.
-=	x -=5이면, x-5의 결과를 x에 대입한다.
…	+=, -=외에도 /=, %=, **=, &=, >>= 등 존재한다.
:=	Walrus Operator (다음 페이지)

Operators

♣ 대입 연산자

- :=는 Walrus Operator로, 다음과 같이 수행된다.

  name := expr일 때, expr을 먼저 평가하여 값 v를 얻는다.
  이후 name = v를 수행 후 (name := expr) 자체도 v로 바꾼다.

```
print(x:=3+5, x)

print(x:=5*2, y:=x*5, x+y)
```

>> 8 8

>> 10 50 60

Operators

♣ 멤버 연산자(Membership Operators)

연산자	내용	예시
in	x in y에서, y에 x이 있으면 참	“a” in “apple”
not in	x not in y에서, y에 x이 없으면 참	3 not in [1, 2, 3]

Operator Precedence

♣ 연산자 우선순위

- 연산자가 여러 개 있을 땐, 우선 순위에 따라 순서대로 계산된다.

```
x = 5 + 10 * 2
y = 1 * (-3 * -1) & 2
print(x)
print(y)
```

```
>> 25
>> 2
```

Operator Precedence

♣ 연산자 우선순위

- 우선순위는 오른쪽 표와 같으며, 위쪽에 있을수록 연산의 우선순위가 높다.

- 오른쪽 표는 Python의 모든 연산자의 우선순위를 나타내지 않았으므로, 아래 공식 문서를 참고한다.

https://docs.python.org/3/reference/expressions.html#operator-precedence

연산자
()
**
+x, -x, ~x
*, //, /, %
+, -
>>, <<
&
^
...

Propositional Logic

♣ 명제와 논리연산

- 명제: 참과 거짓이 확실히 결정되는 문장

  예) 한국의 수도는 서울이다 – True, 2는 홀수이다 – False

- 명제 p, q에 대하여 아래와 같은 논리 연산을 수행할 수 있다.

  부정(NOT): $\neg p$
  논리합(OR): $p \vee q$
  논리곱(AND): $p \wedge q$

Propositional Logic

♣ 진리표(Truth Table)

- 논리식의 참과 거짓을 판정하기 위한 표

예)

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

Propositional Logic

♣ 동치(Equivalence)

- 두 논리식 A, B가 같은 진리값을 가질 때, A와 B는 동치이다.
- $A \equiv B$로 표기한다.

예) $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$ … De Morgan's Law

Propositional Logic

♣ 조건문

- 조건문 $p \rightarrow q$: p이면 q이다.

- $p \rightarrow q$의 진리표

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Propositional Logic

♣ 쌍조건문

- 조건문 $p \leftrightarrow q$: p이면 q이고, q이면 p이다. (필요충분조건)

- $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$와 같다.

p	q	$p \rightarrow q$	$q \rightarrow p$	$p \leftrightarrow q$
T	T	T	T	T
T	F	F	T	F
F	T	T	F	F
F	F	T	T	T

QUIZ

Problem 1.

print(5 * 2 ** 3)의 출력 결과는?

Problem 2.

정수를 입력 받았을 때, 홀수이면 True, 짝수이면 False를 출력하는 프로그램을 작성한다.

Problem 3.

$p \oplus q$ (xor 연산)에 대한 진리표를 작성한다.

THANK YOU!

Any Question?