More on Type Conversion

```
>>> int(4/3)          # 4/3 = 1.33333…
1
>>> int(5/3)          # 5/3 = 1.66666…
1
```

반올림 아님, 무조건 내림

```
>>> int("two")
Traceback (most recent call last):
  File "<pyshell#77>", line 1, in <module>
    int("two")
ValueError: invalid literal for int() with base 10: 'two'
```

에러

```
>>> int("3.3")
Traceback (most recent call last):
  File "<pyshell#78>", line 1, in <module>
    int("3.3")
ValueError: invalid literal for int() with base 10: '3.3'
```

에러

오늘의 강의 목표

- 연산자(Operator)에 대한 이해
- 연산자 우선순위(Precedence)에 대한 이해

Expression, Value, and Statement

```
>>> 1
1      # the value of 1          is 1
>>> 1 + 1
2      # the value of 1 + 1      is 2
>>> 1 + 1 + 1
3      # the value of 1 + 1 + 1  is 3
```

Expressions **Values**

```
>>> a = 1 + 1 + 1   # save the value of 1 + 1 + 1 to a
>>> b = a + 1       # save the value of a + 1     to b
>>> print(b)        # print the value of b
4
```

Statements **Expressions**

Expression

- Expression
  - A piece of syntax which can be evaluated to some value (https://docs.python.org/3/glossary.html)

Expression 인가? 그렇다면 그 Value는?

(A)

```
>>> a = 1
>>>
```

(B)

```
>>> a + 1
2
>>>
```

(C)

```
>>> print(a + 1)
2
>>>
```

Statement

- Statement
  - A unit of code that can be executed

Statement 인가?

(A)

```
>>> a = 1
>>>
```

(B)

```
>>> a + 1
2
>>>
```

(C)

```
>>> print(a + 1)
2
>>>
```

Operators and Operands

assignment operator

```
a = 1
```

operands

assignment operator add operator

```
b = a + 1
```

operands

assignment operator minus operator

```
a = -1
```

operands

assignment operator concatenate operator

```
b = "Good" + "Boy"
```

operands

Arithmetic Operators

Operator	Description	Example
+	더하기	a + b
-	빼기	a – b
*	곱하기	a * b
/	나누기	a / b
%	나머지	a % b
**	지수	a ** b
//	몫	a // b

Arithmetic Operators

```
a = 31
b = 17
c = 0

c = a + b
print(a, " + ", b, " = ", c)
c = a - b
print(a, " - ", b, " = ", c)
c = a * b
print(a, " + ", b, " = ", c)
c = a / b
print(a, " + ", b, " = ", c)
c = a % b
print(a, " + ", b, " = ", c)
c = a ** b
print(a, " ** ", b, " = ", c)
c = a // b
print(a, " // ", b, " = ", c)
```

스크립트 파일로 만들어서
실행해 보세요

Arithmetic Error

```
>>> 3 / 0
Traceback (most recent call last):
  File "<pyshell#89>", line 1, in <module>
    3/0
ZeroDivisionError: division by zero
```

Assignment Operators

Operator	Description	Example
=	대입	a = b
+=	더하여 대입	a += b
-=	빼서 대입	a -= b
*=	곱해서 대입	a *= b
/=	나눠서 대입	a /= b
%=	나머지 대입	a %= b
**=	지수 대입	a **= b
//=	몫 대입	a //= b

Assignment Operators

```
>>> a = 10
>>> a += 10
>>> a -= 3
>>> a *= 4
>>> a /= 2
>>> a %= 5
>>> a **= 2
>>> a //= 3
```

a의 최종 값은?

Comparison Operators

- Boolean Type

```
>>> 20 > 10
True
>>> type(20 > 10)
<class 'bool'>
>>> 20 < 10
False
>>> a = 20 < 10
>>> print(a)
False
```

✓ True or False 값을 가지는 Data Type
✓ Comparison operator의 결과값은 Boolean Type

Comparison Operators

Operator	Description	Example
==	같은가?	a == b
!=	다른가?	a != b
<>	다른가?	a <> b
>	(좌변이) 큰가?	a > b
<	(좌변이) 작은가?	a < b
>=	(좌변이) 크거나 같은가?	a >= b
<=	(좌변이) 작거나 같은가?	a <= b

= (assignment)와 == (equal) 혼동하기 쉬움

Comparison Operators

```
>>> a = 10
>>> b = 10
>>> a == b
True
>>> print(a == b)
True
>>> if a == b:
      print("equal")


equal
```

colon

엔터 두번

Comparison Operators

```
>>> a = "A"
>>> b = "B"
>>> a > b
False
>>> a < b
True
>>> "Emphasis" > "Empty"
False
>>> "Code0" > " Code1"
True
```

Dictionary Order
for Strings Comparison

Logical Operators

Operator	Description	Example
and	Logical AND (논리곱)	a == b and c == d
or	Logical OR (논리합)	a == b or c == d
not	Logical NOT (부정)	not a == b

x	y	x AND y
T	T	T
F	T	F
T	F	F
F	F	F

x	y	x OR y
T	T	T
F	T	T
T	F	T
F	F	F

x	NOT x
T	F
F	T

Logical Operators

```
>>> bool(1)
True
>>> bool(-1)
True
>>> bool(0)
False
```

Zero is recognized as False,
non-zero is recognized as True

```
>>> bool("a")
True
>>> bool("")
False
```

Empty string is recognized as False,
everything else is True

```
>>> if 1:    # 1 is recognized as True
      print("Will this be printed?")


Will this be printed?
```

Logical Operators

```
>>> a = 10
>>> b = 10
>>> c = 20
>>> if a == b and b == c:
      print("Triplets")


>>> if a == b or b == c:
      print("Not Triplets")


Not Triplets
>>> if not b == c:
      print("Not Equal")


Not Equal
```

Bitwise Operators

Operator	Description	Example
&	이진수 AND	a & b
|	이진수 OR	a | b
^	이진수 XOR	a ^ b
~	이진수 일의 보수	~a
<<	좌측 Shift	a << 2
>>	우측 Shift	a >> 2

```
    0101 (십진수 5)
AND 0011 (십진수 3)
  = 0001 (십진수 1)
```

```
    0101 (십진수 5)
 OR 0011 (십진수 3)
  = 0111 (십진수 7)
```

```
    0101 (십진수 5)
XOR 0011 (십진수 3)
  = 0110 (십진수 6)
```

```
NOT 0011 (십진수 3 )
  = 1100 (십진수 12)
```

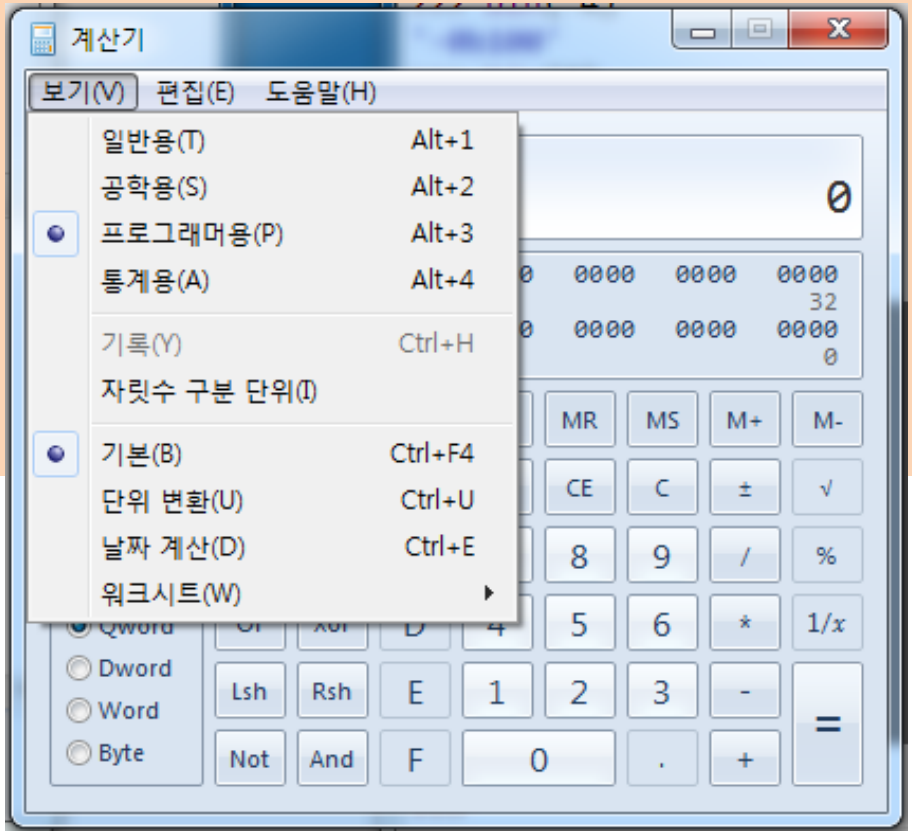
```
      2
    <=
    0011 (십진수 3 )
  = 1100 (십진수 12)
```

```
      2
    =>
    1100 (십진수 12)
  = 0011 (십진수 3)
```

Bitwise Operators

```
>>> 5 & 3
1
>>> 5 | 3
7
>>> 5 ^ 3
6
>>> ~3
-4
>>> 3 << 2
12
>>> 12 >> 2
3
```

why ~3 is -4?
Let's use programmer's calculator for it.



Operator Precedence

Low

Operator	Description
lambda	Lambda expression
if – else	Conditional expression
or	Boolean OR
and	Boolean AND
not x	Boolean NOT
in, not in, is, is not, <, <=, >, >=, !=, ==	Comparisons, including membership tests and identity tests
|	Bitwise OR
^	Bitwise XOR
&	Bitwise AND
<<, >>	Shifts
+, -	Addition and subtraction
*, /, //, %	Multiplication, division, remainder
+x, -x, ~x	Positive, negative, bitwise NOT
**	Exponentiation
x[index], x[index:index], x(arguments...), x.attribute	Subscription, slicing, call, attribute reference
(expressions...), [expressions...], {key: value...}, {expressions...}	Binding or tuple display, list display, dictionary display, set display

High

https://docs.python.org/3.4/reference/expressions.html

Operator Precedence

```
>>> 3 + 5 >> 1
4
>>> (3 + 5) >> 1
4
>>> 3 + (5 >> 1)
5
```

```
>>> 3 + 4 and 3 - 3
0
>>> 3 + (4 and 3) - 3
3
>>> (3 + 4) and (3 - 3)
0
```

Questions