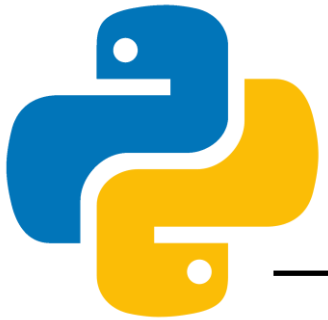

창직 IoT 종합설계입문

파이썬 (2)

연산자

연산자의 종류

파이썬 연산자는 크게 다음과 같이 분류할 수 있습니다.

- 산술 연산자 (Arithmetic Operator)
- 할당 연산자 (Assignment Operator)
- 비교 연산자 (Comparison Operator)
- 논리 연산자 (Logical Operator)
- 비트 연산자 (Bitwise Operator)
- 멤버 연산자 (Membership Operator)
- 식별 연산자 (Identity Operator)

산술 연산자

산술 연산자 (Arithmetic Operator)

- 연산자의 우선 순위는 일반적인 사칙 연산과 같습니다.

+	-	*	**	/	//	%

- 계산 결과는 아래와 같습니다.

```
print( "2 + 3 = ",  2 + 3)  # 덧셈
print( "2 - 3 = ",  2 - 3)  # 뺄셈
print( "2 * 3 = ",  2 * 3)  # 곱셈
print("2 ** 3 = ", 2 ** 3)  # 거듭제곱
print( "2 / 3 = ",  2 / 3)  # 나눗셈
print("2 // 3 = ", 1 // 3)  # 몫
print( "2 % 3 = ",  1 % 3)  # 나머지
```

```
2 + 3 =  5
2 - 3 =  -1
2 * 3 =  6
2 ** 3 =  8
2 / 3 =  0.6666666666666666
2 // 3 =  0
2 % 3 =  1
```

할당 연산자

할당 연산자 (Assignment Operator)

=	+=	-=	*=	/=	//=	%=

- 계산 결과는 아래와 같습니다.

```
a = 13; b = 5
a +=  b; print(a, b) # a = (a +  b)
a -=  b; print(a, b) # a = (a -  b)
a *=  b; print(a, b) # a = (a *  b)
a /=  b; print(a, b) # a = (a /  b)
a //= b; print(a, b) # a = (a // b)
a %=  b; print(a, b) # a = (a %  b)
```

```
18 5
13 5
65 5
13.0 5
2.0 5
2.0 5
```

비교 연산자

비교 연산자 (Comparison Operator)

==	!=	〉	〈	〉=	〈=

- 계산 결과는 아래와 같습니다.

```
a = 7
b = 3
print("a == b", a == b) # a와 b가 같은지?
print("a != b", a != b) # a와 b가 다른지?
print("a > b ",  a > b) # a가 b보다 큰 지?
print("a < b ",  a < b) # a가 b보다 작은지?
print("a >= b", a >= b) # a가 b보다 크거나 같은지?
print("a <= b", a <= b) # a가 b보다 작거나 같은지?
```

```
a == b False
a ≠ b True
a > b  True
a < b  False
a ⩾ b True
a ⩽ b False
```

논리 연산자

논리 연산자 (Logical Operator)

and	or	not

- 계산 결과는 아래와 같습니다.

```
print(True and True)
print(True and False)
print(False and True)
print(False and False)

print(True or True)
print(True or False)
print(False or True)
print(False or False)

print(not True)
print(not False)
```

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

비트 연산자

비트 연산자 (Bitwise Operator)

&	|	^	~	〈〈	〉〉

- 계산 결과는 아래와 같습니다.

```
a = 7; b = 13; # a = 0111(2) b = 1101(2)
print("a & b  = ", a & b)  #  AND 연산
print("a | b  = ", a | b)  #   OR 연산
print("a ^ b  = ", a ^ b)  #  XOR 연산
print("~a     = ", ~a)     # 보수 연산
print("a << 1 = ", a << 1) # 왼쪽으로 1 비트 수만큼 이동
print("b >> 1 = ", b >> 1) # 오른쪽으로 1 비트 수만큼 이동
```

```
a & b  =  5
a | b  =  15
a ^ b  =  10
~a     =  -8
a << 1 =  14
b >> 1 =  6
```

멤버 연산자

멤버 연산자 (Membership Operator)

in	Not in

- 계산 결과는 아래와 같습니다.

```
aList = [1, 2, 3, 4, 5]; a = 3
print("a in aList :", a in aList)
# a가 aList에 존재하는지?

print("a not in aList : ", a not in aList)
# a가 aList에 존재하지 않는지?
```

```
a in aList : True
a not in aList :  False
```

식별 연산자

식별 연산자 (Identity Operator)

is	is not

- 계산 결과는 아래와 같습니다.

```
a = 7; b = 3; c = 3;
print("a is c", a is c)
# a와 c가 값 또는 메모리 위치가 같은가?

print("b is c", b is c)
# b와 c가 값 또는 메모리 위치가 같은가?
```

```
a is c False
b is c True
```

연산자 우선순위

연산자 우선순위

1	**			
2	+	-	~	
3	*	/	//	%
4	+	-		
5	<<	>>		
6	&	^	|	
7	비교 연산자			
8	할당 연산자			
9	식별 연산자			
10	멤버 연산자			
11	논리 연산자			

변수

변수 (Variable)

- 값을 저장하기 위한 메모리
- 명시적으로 메모리 공간을 예약 선언할 필요가 없음.
- 할당 연산자 (=)를 사용하여 값을 할당
- 숫자형 (Numbers), 문자열 (String)

```
a = 1
# 변수 a에 정수 값 1을 할당
print(a)

a = b = c = 1
# 변수 a, b, c에 정수 값 1을 할당
print(a, b, c)

a, b, c = 1, 2, 3
# 변수 a, b, c에 각각 정수 값 1, 2, 3을 할당
print(a, b, c)
```

```
1
1 1 1
1 2 3
```

숫자형

숫자형 (Numbers)

- int
- float
- complex

```
# int 형 변수 선언
a, b = 1, -2
print(type(a), type(b))

# float 형 변수 선언
c, d = 1.0, -2.0
print(type(c), type(d))

# complex 형 변수 선언
e, f = complex(1, 3), 1 - 3j
print(type(e), type(f))
```

```
<class 'int'> <class 'int'>
<class 'float'> <class 'float'>
<class 'complex'> <class 'complex'>
```

숫자형 – 내장 함수 (1)

내장함수

- 파이썬 인터프리터에는 항상 사용할 수 있는 많은 함수와 형이 내장.
- 숫자형을 지원하는 연산에 대해 소개.

- abs(x)
- int(x), float(x), complex(x)
- .conjugate()
- divmod(x, y)
- pow(x, y)

숫자형 – 내장 함수 (2)

abs(x)

- 숫자형의 절대값을 반환.

```
a, b, c = 5, -4.73, 1 + 5.6j
print("abs(a) = ", abs(a))
print("abs(b) = ", abs(b))
print("abs(c) = ", abs(c))
```

```
abs(a) =  5
abs(b) =  4.73
abs(c) =  5.688585061331157
```

x.conjugate()

- 복소수 x의 켤레 복소수를 반환.

```
x = 1 + 3j

print(x)
print(x.conjugate())
```

```
(1+3j)
(1-3j)
```

숫자형 – 내장 함수 (3)

int(x), float(x), complex(x)

- 정수, 실수 또는 복소수로 변환된 값을 반환.

```
print(int(1.2))
print(type(int(1.2)))

print(float(5))
print(type(float(5)))

print(complex(3))
print(type(complex(3)))
```

```
1
<class 'int'>
5.0
<class 'float'>
(3+0j)
<class 'complex'>
```

숫자형 – 내장 함수 (4)

divmod(x, y)

- x를 y로 나눈 몫과 나머지를 반환.

```
x, y = 7, 3.7
divmod(x, y)
```

```
(1.0, 3.3)
```

pow(x, y)

- x의 y 거듭제곱을 반환

```
x = 1 + 3j

print(x)
print(x.conjugate())
```

```
(1+3j)
(1-3j)
```

Math 모듈

import math

- 다양한 수학에 관련된 메서드 제공.
- 복소수에 대한 지원 X

```
# n! 구하기
# 1. Loop 문
returnValue = 1.0;
for i in range(1, 5):
  returnValue *= i

print(returnValue)

# 2. 재귀 함수
def nFactorial(n):
  if n == 1:
    return 1
  else :
    return n * nFactorial(n-1)

print(nFactorial(5))

# 3. Math 모듈 이용
import math

print(math.factorial(6))
```

```
24.0
120
720
```