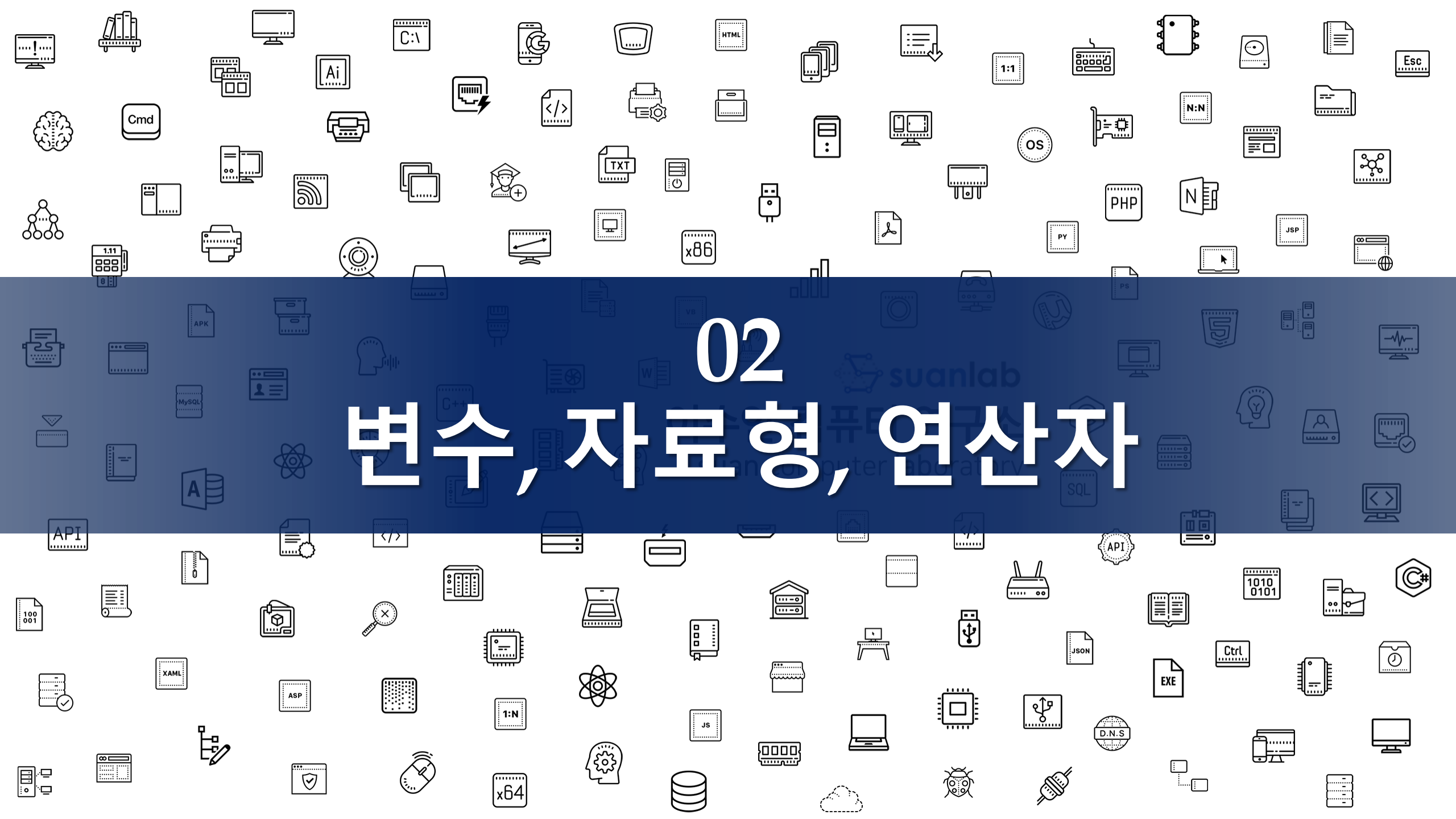
데이터 처리 프로그래밍

Data Processing Programming



02
변수, 자료형, 연산자

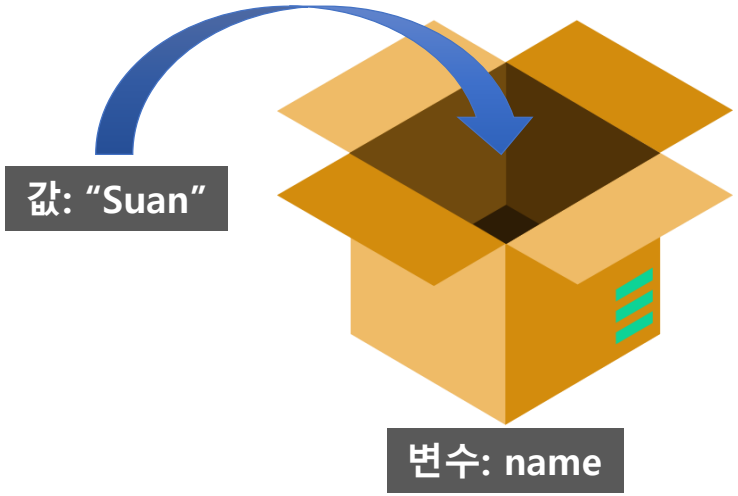
목차

1. 변수
2. 자료형
3. 연산자

1. 변수

변수 개념

- 어떠한 값을 저장하는 공간(메모리)
- 일반적으로 데이터의 저장 위치와 데이터 값으로 구분
- 변수에 값을 넣는 순간 메모리 저장 위치를 할당하고 그 위치는 메모리 주소로 관리



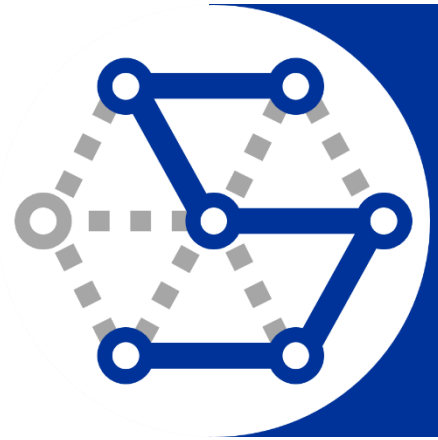
Code

```
>>> name = "Suan"
>>> name
'Suan'
```

변수명 규칙

- 알파벳, 숫자, 밑줄(_)로 선언 가능
- 의미 있는 단어로 표기하는 것이 좋음
- 대소문자가 구분
- 특별한 의미가 있는 예약어는 사용할 수 없음
- 예약어

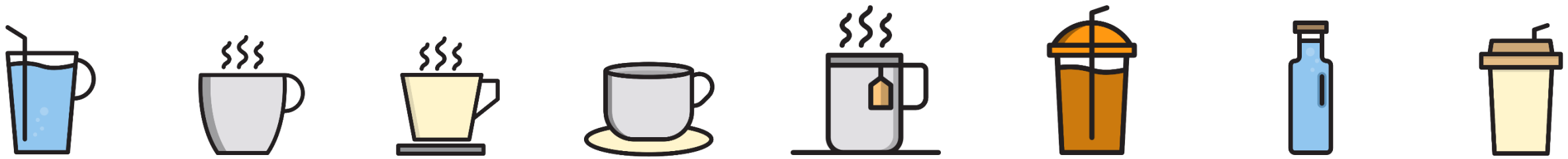
and	exec	not	assert	finally	or
break	for	pass	class	from	print
continue	global	raise	def	if	return
del	import	try	elif	in	while
else	is	with	except	lambda	yield



2. 자료형

자료형Data Types

유형	자료형	설명	선언
논리형	불리언형boolean type	참(True) 또는 거짓(False)을 표현할 때 사용	b=True
수치형	정수형integer type	자연수를 포함해 값의 영역이 정수로 한정된 값	i=10
	8진수 정수형octal type	8진수 자료형(숫자가 0o 또는 0O로 시작)	o=0o134
	16진수 정수형hexadecimal type	16진수 자료형(0x로 시작)	h=0xABC
	실수형floating-point type	소수점이 포함된 값	f=12.34
	복소수형complex type	복소수 사용을 위한 자료형	c=3.14J
문자형	문자열string type	값이 문자로 출력되는 자료형	s="Suan"
구조형	리스트list type	여러 요소를 묶어 하나의 변수로 사용	li=[1, 2, 3]
	튜플tuple type	리스트와 유사하지만 생성, 삭제, 수정 불가	t=(1, 2, 3)
	딕셔너리dictionary type	키key와 값value가 쌍pair으로 들어간 자료형	d={1:'One', 2:'Two'}



파이썬 자료형 확인하기

- 불리언형 변수 자료형 출력

```
>>> b = True
>>> type(b)
<class 'bool'>
```

- 정수형 변수 자료형 출력

```
>>> i = 10
>>> type(i)
<class 'int'>
>>> i = 1000*1000
>>> i
1000000
>>> type(i)
<class 'int'>
>>> o = 0o123
>>> type(o)
<class 'int'>
>>> h = 0xABC
>>> type(h)
<class 'int'>
>>> print(i, o, h)
1000000 83 2748
```

- 실수형 변수 자료형 출력

```
>>> f = 12.34
>>> print(f)
12.34
>>> type(f)
<class 'float'>
>>> e = 12.34-1E10
>>> print(e)
-9999999987.66
>>> type(e)
<class 'float'>
```

- 복소수형 변수 자료형 출력

```
>>> c = 3.14J
>>> type(c)
<class 'complex'>
>>> c = 3.14e1-5j
>>> print(c)
(31.4-5j)
>>> type(c)
<class 'complex'>
```

파이썬 자료형 확인하기

- 문자열 변수 자료형 출력

```
>>> s = "Suan"
>>> print(s)
Suan
```

- 리스트 변수 자료형 출력

```
>>> l = [1, 2, 3]
>>> print(l)
[1, 2, 3]
>>> type(l)
<class 'list'>
>>> l = [1, 'One', 2, 'Two']
>>> print(l)
[1, 'One', 2, 'Two']
>>> type(l)
<class 'list'>
```

- 튜플 자료형 출력

```
>>> t = (1, 2, 3)
>>> print(t)
(1, 2, 3)
>>> type(t)
<class 'tuple'>
>>> t = (1, 'One', 2, 'Two')
>>> print(t)
(1, 'One', 2, 'Two')
>>> type(t)
<class 'tuple'>
```

- 딕셔너리 자료형 출력

```
>>> d = {1:'One', 2:'Two'}
>>> print(d)
{1: 'One', 2: 'Two'}
>>> type(d)
<class 'dict'>
```

자료형 변환Data Type Conversion

함수	설명
int(x [,base])	x를 정수로 변환 x가 문자열일 경우 base 지정
float(x)	x를 실수형(부동 소수점 숫자)로 변환
complex(real [,imag])	복소수 생성
str(x)	객체 x를 문자열 표현으로 변환
repr(x)	객체 x를 표현식 문자열로 변환
eval(str)	문자열을 평가하고 객체를 반환

함수	설명
tuple(s)	s를 튜플로 변환
list(s)	s를 리스트로 변환
set(s)	s를 집합으로 변환
dict(d)	딕셔너리 생성 d는 (키, 값) 튜플의 시퀀스여야 함
frozenset(s)	s를 고정 세트로 변환
chr(x)	정수를 문자로 변환
ord(x)	단일 문자를 정수 값으로 변환
hex(x)	정수를 16진수 문자열로 변환
oct(x)	정수를 8진수 문자열로 변환

str() 이나 print 는 __str__ 메소드를 호출
__str__ 은 객체의 비공식적인(informal) 문자열을 출력할 때 사용
__str__ 은 사용자가 보기 쉬운 형태로 보여줄 때 사용
repr() 은 __repr__ 메소드를 호출
__repr__ 은 공식적인(official) 문자열을 출력할 때 사용
__repr__ 은 시스템(python interpreter)이 해당 객체를 인식할 수 있는 공식적인 문자열로 나타내 줄 때 사용

자료형 변환하기

```
>>> int(b)
1
>>> int(i)
10
>>> int(o)
83
>>> int(h)
2748
>>> int(f)
12
>>> int(e)
-9999999987
>>> int(c)
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: can't convert complex to int
>>> int("123")
123
>>> int("123", 8)
83
>>> int("123", 16)
291
```

```
>>> float(b)
1.0
>>> float(i)
10.0
>>> float(o)
83.0
>>> float(h)
2748.0
>>> float(f)
12.34
>>> float(e)
-9999999987.66
>>> float(c)
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: can't convert complex to float
>>> float("123")
123.0
```

자료형 변환하기


```
>>> complex(b)
(1+0j)
>>> complex(i)
(10+0j)
>>> complex(o)
(83+0j)
>>> complex(h)
(2748+0j)
>>> complex(f)
(12.34+0j)
>>> complex(e)
(-9999999987.66+0j)
>>> complex(c)
3.14j
>>> complex("123")
(123+0j)
```

```
>>> str(b)
'True'
>>> str(i)
'10'
>>> str(o)
'83'
>>> str(h)
'2748'
>>> str(f)
'12.34'
>>> str(e)
'-9999999987.66'
>>> str(c)
'3.14j'
>>> str(s)
'Suan'
>>> str(l)
'[1, 2, 3]'
>>> str(t)
'(1, 2, 3)'
>>> str(d)
"{1: 'One', 2: 'Two'}"
```

자료형 변환하기


```
>>> repr(b)
'True'
>>> repr(i)
'10'
>>> repr(o)
'83'
>>> repr(h)
'2748'
>>> repr(f)
'12.34'
>>> repr(e)
'-9999999987.66'
>>> repr(c)
'3.14j'
>>> repr(s)
"'Suan'"
>>> repr(l)
'[1, 2, 3]'
>>> repr(t)
'(1, 2, 3)'
>>> repr(d)
"{1: 'One', 2: 'Two'}"
```

```
>>> eval('1+2')
3
>>> eval('50+50+50')
150
>>> eval("'hi'+' yo~'")
'hi yo~'
```

자료형 변환하기


```
>>> tuple(s)
('S', 'u', 'a', 'n')
>>> tuple(l)
(1, 2, 3)
>>> tuple(t)
(1, 2, 3)
```

```
>>> list(s)
['S', 'u', 'a', 'n']
>>> list(l)
[1, 2, 3]
>>> list(t)
[1, 2, 3]
```

```
>>> set(s)
{'S', 'n', 'a', 'u'}
>>> set(l)
{1, 2, 3}
>>> set(t)
{1, 2, 3}
>>> set(d)
{1, 2}
```

```
>>> frozenset(s)
frozenset({'S', 'n', 'a', 'u'})
```

```
>>> chr(97)
'a'
>>> chr(65)
'A'
>>> chr(90)
'Z'
```

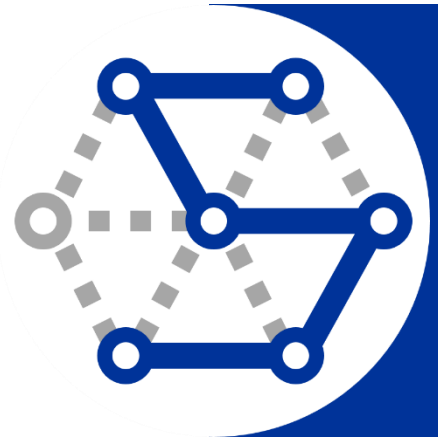
```
>>> ord('a')
97
>>> ord('A')
65
>>> ord('Z')
90
```

```
>>> hex(b)
'0x1'
>>> hex(i)
'0xa'
>>> hex(o)
'0x53'
>>> hex(h)
'0xabc'
```

```
>>> oct(b)
'0o1'
>>> oct(i)
'0o12'
>>> oct(o)
'0o123'
>>> oct(h)
'0o5274'
```

동적 타이핑dynamic typing

- 변수의 메모리 공간 확보가 실행 시점에서 발생
- C나 자바는 int data = 4와 같이 data 변수를 정수형으로 사전 선언
- 파이썬은 data = 4 형태로 선언하여 data라는 변수의 자료형이 정수integer인지 실수float인지를 프로그래머가 아닌 인터프리터가 판단
- 파이썬 언어가 실행 시점에 동적으로 자료형 결정
- 다른 언어들과 달리 파이썬은 매우 유연한 언어로, 할당받는 메모리 공간도 저장되는 값의 크기에 따라 동적으로 다르게 할당받을 수 있음



3. 연산자

연산자 종류

- 산술 연산자Arithmetic Operators
- 비교(관계) 연산자 Comparison(Relational) Operators
- 할당 연산자Assignment Operators
- 비트 연산자Bitwise Operators
- 논리 연산자Logical Operators
- 멤버 연산자Membership Operators
- 식별 연산자Identity Operators

산술 연산자Arithmetic Operators

▪ 산술 연산자 종류

연산자	설명	예제
+	덧셈,	c = a + b
-	뺄셈	c = a – b
*	곱셈	c = a * b
/	나눗셈	c = a / b
%	나머지	c = a % b
**	제곱	c = a ** b
//	몫	c = a // b

▪ 산술 연산자 예제

```
>>> a = 20
>>> b = 12
>>> c = a + b
>>> print(c)
32
>>> c = a - b
>>> print(c)
8
>>> c = a * b
>>> print(c)
240
>>> c = a / b
>>> print(c)
1.6666666666666667
>>> c = a % b
>>> print(c)
8
>>> c = a ** b
>>> print(c)
4096000000000000
>>> c = a // b
>>> print(c)
1
```

비교(관계) 연산자Comparison(Relational) Operators

▪ 비교(관계) 연산자 종류

연산자	설명	예제
==	두 피연산자의 값이 동일하면 True	a == b
!=	두 피연산자의 값이 다르면 True	a != b
>	왼쪽 피연산자가 오른쪽 피연산자 보다 크면 True	a > b
<	왼쪽 피연산자가 오른쪽 피연산자 보다 작으면 True	a < b
>=	왼쪽 피연산자가 오른쪽 피연산자 보다 크거나 같으면 True	a >= b
<=	왼쪽 피연산자가 오른쪽 피연산자 보다 작거나 같으면 True	a <= b

▪ 비교(관계) 연산자 예제

```
>>> a = 20
>>> b = 12
>>> a == b
False
>>> a != b
True
>>> a > b
True
>>> a < b
False
>>> a >= b
True
>>> a <= b
False
```

할당 연산자Assignment Operators

▪ 할당 연산자 종류

연산자	설명	예제
=	오른쪽 피연산자의 값을 왼쪽 피연산자에 할당	a = 20 b = 12
+=	왼쪽 피연산자의 값을 오른쪽 피연산자에 더하고 그 결과를 왼쪽 피연산자에 대입	a += b
-=	왼쪽 피연산자의 값을 오른쪽 피연산자에 더하고 그 결과를 왼쪽 피연산자에 대입	a -= b
*=	왼쪽 피연산자의 값을 오른쪽 피연산자에 곱하고 그 결과를 왼쪽 피연산자에 대입	a *= b
/=	왼쪽 피연산자의 값을 오른쪽 피연산자에 나누고 그 결과를 왼쪽 피연산자에 대입	a /= b
%=	왼쪽 피연산자의 값을 오른쪽 피연산자에 나눈 나머지 값을 왼쪽 피연산자에 대입	a %= b
**=	왼쪽 피연산자의 값을 오른쪽 피연산자에 제곱한 값을 왼쪽 피연산자에 대입	a **= b
//=	왼쪽 피연산자의 값을 오른쪽 피연산자에 나눈 몫을 왼쪽 피연산자에 대입	a //= b

▪ 할당 연산자 예제

```
>>> a, b = 20, 12
>>> a += b
>>> print(a)
32
>>> a, b = 20, 12
>>> a -= b
>>> print(a)
8
>>> a, b = 20, 12
>>> a *= b
>>> print(a)
240
>>> a, b = 20, 12
>>> a /= b
>>> print(a)
1.6666666666666667
```

```
>>> a, b = 20, 12
>>> a %= b
>>> print(a)
8
>>> a, b = 20, 12
>>> a **= b
>>> print(a)
4096000000000000
>>> a, b = 20, 12
>>> a //= b
>>> print(a)
1
```

비트 연산자Bitwise Operators

▪ 비트 연산자 종류

연산자	설명	예제
&	두 피연산자의 비트를 AND 연산	a & b
|	두 피연산자의 비트를 OR 연산	a | b
^	두 피연산자의 비트를 XOR 연산	a ^ b
~	피연산자의 비트를 NOT 연산	~a
<<	왼쪽 피연산자의 비트를 오른쪽 피연산자의 비트만큼 left shift 연산	a << b
>>	왼쪽 피연산자의 비트를 오른쪽 피연산자의 비트만큼 right shift 연산	a >> b

▪ 비트 연산자 예제

```
>>> a, b = 20, 12
>>> bin(a)
'0b10100'
>>> bin(b)
'0b1100'
>>> print(a & b, bin(a & b))
4 0b100
>>> print(a | b, bin(a | b))
28 0b11100
>>> print(a ^ b, bin(a ^ b))
24 0b11000
>>> print(~a, bin(~a))
-21 -0b10101
>>> print(a << b, bin(a << b))
81920 0b10100000000000000
>>> print(a >> b, bin(a >> b))
0 0b0
```

논리 연산자Logical Operators

▪ 논리 연산자 종류

연산자	설명	예제
and	두 피연산자가 모두 True이면 True	True and True
or	두 피연산자 중 하나라도 True이면 True	True or False
not	피연산자가 True이면 False, False이면 True	not False

▪ 논리 연산자 예제

```
>>> True and True
True
>>> True and False
False
>>> True or False
True
>>> not True
False
>>> not False
True
>>> not (True and False)
True
```

멤버 연산자Membership Operators

▪ 멤버 연산자 종류

연산자	설명	예제
in	멤버로 포함되어 있으면 True, 포함되어 있지 않으면 False	a in l b in l
not in	멤버로 포함되어 있지 않으면 True, 포함되어 있으면 False	a not in l b not in l

▪ 멤버 연산자 예제

```
>>> a, b = 20, 12
>>> l = [10, 20, 30]
>>> a in l
True
>>> b in l
False
>>> a not in l
False
>>> b not in l
True
```

식별 연산자Identity Operators

▪ 식별 연산자 종류

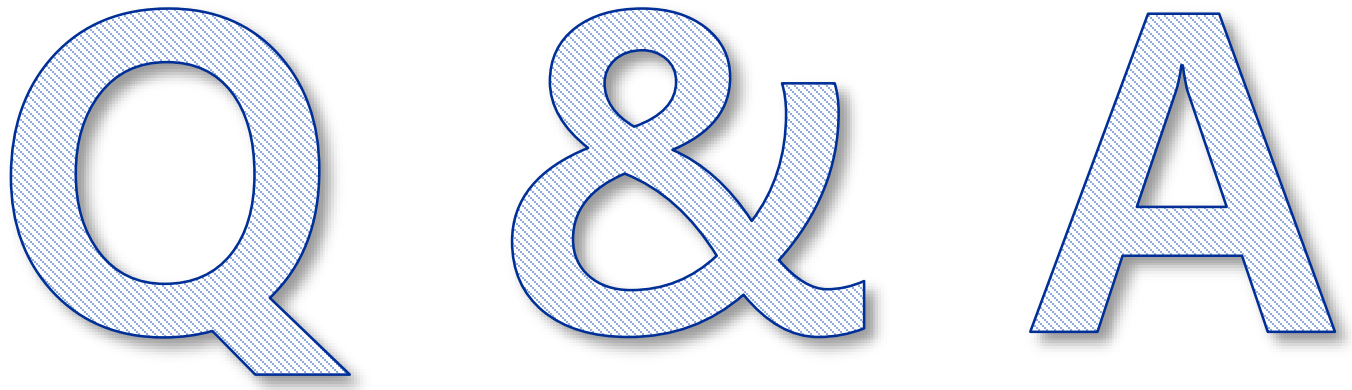
연산자	설명	예제
is	피연산자가 동일한 객체를 가리키면 True, 동일한 객체를 가리키고 있지 않으면 False	a is l b is l
is not	피연산자가 동일한 객체를 가리키고 있지 않으면 True, 동일한 객체를 가리키면 False	a is not l b is not l

▪ 식별 연산자 예제

```
>>> a, b = 20, 12
>>> a is b
False
>>> a is not b
True
>>> a, b = 20, 20
>>> a is b
True
>>> a is not b
False
```

연산자 우선순위Operators Precedence

연산자	설명
()	괄호
**	지수(승수)
~ + -	보수, 단항 플러스와 마이너스
* / % //	곱셈, 나눗셈, 나머지, 몫
+ -	덧셈과 뺄셈
>> <<	좌우 비트 시프트
&	비트 'AND'
^ |	비트 전용 'OR'와 정기적 인 'OR'
<= < > >=	비교 연산자
== !=	평등 연산자
= %= /= //= -= += *= **=	할당 연산자
is is not	식별 연산자
in not in	맴버 연산자
not or and	논리 연산자


Q & A