

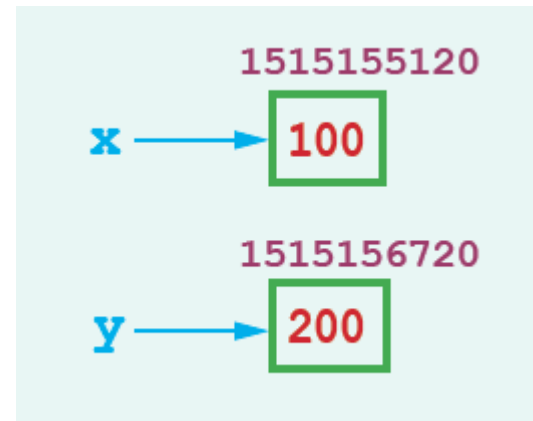
3. 수치 자료형과 연산자

1. 정수 자료형과 연산
2. 실수 자료형과 연산
3. 복소수 자료형과 연산
4. 자료형 변환
5. 다양한 자료형이 섞인 연산식
6. 수치 연산 함수들
7. math 모듈
8. 정리

1. 정수 자료형과 연산

◆ 정수 (int) 표현하기 (..., -3, -2, -1, 0, 1, 2, 3, ...)

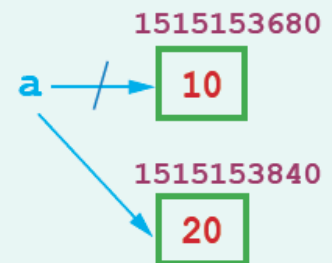
```
>>> x = 100; y = 200
>>> x, y
(100, 200)
>>> id(x), id(y)
(1515155120, 1515156720)
>>> type(x), type(y)
(<class 'int'>, <class 'int'>)
```



1. 정수 자료형과 연산

◆ 정수는 immutable 하다

```
>>> a = 10
>>> id(a)
1515153680
>>> a = 20    # 객체 10은 그냥 두고 새로 객체 20을 만듭니다.
>>> id(a)      # a의 id가 변했습니다.
1515153840
```



1. 정수 자료형과 연산

◆ 정수 자료형은 크기 제한이 없다

```
>>> 2 ** 10      # 2의 10 제곱
1024
>>> 2 ** 20      # 2의 20 제곱
1048576
>>> 2 ** 100     # 2의 100 제곱
1267650600228229401496703205376
>>> 2 ** 1000    # 2의 1000 제곱
10715086071862673209484250490600018105614048117055336074437503883703510511249
36122493198378815695858127594672917553146825187145285692314043598457757469857
48039345677748242309854210746050623711418779541821530464749835819412673987675
59165543946077062914571196477686542167660429831652624386837205668069376
.....
>>> x = 1000000000000000000000000
>>> x + 1000000000000000
1000000000100000000000000
```

1. 정수 자료형과 연산

◆ 산술 연산자

연산자	의미	예	결과
+	더하기	$10 + 5$	15
-	빼기	$20 - 7$	13
*	곱하기	$7 * 8$	56
/	나누기	$30 / 4$	7.5 ←..... 실수
**	제곱	$3 ** 2$	9
//	몫	$50 // 6$	8
%	나머지	$50 \% 6$	2

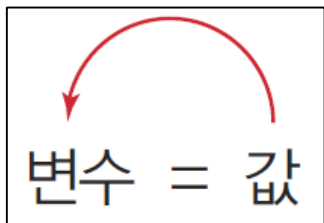
참고 '정수/정수'의 결과는 항상 실수입니다.

```
>>> 100/10
10.0
```

8	← 몫 : $50 // 6$ 결과
6 $\overline{) 50}$	
48	
<hr/>	
2	← 나머지 : $50 \% 6$ 결과

1. 정수 자료형과 연산

◆ 할당 연산자와 산술 연산자



‘=’ 기호는 ‘할당 연산자’ 또는 ‘대입 연산자’라고 함.

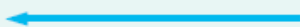
```
>>> a = 10      # 정수 10을 변수 a에 넣습니다.
>>> b = 20      # 정수 20을 변수 b에 넣습니다.
>>> c = a + b    # a와 b를 더해서 변수 c에 넣습니다.
>>> print(a,b,c)
10 20 30
>>> a = a + 50   # a에 50을 더하여 a에 다시 넣습니다. 즉, a를 50만큼 증가시킵니다.
>>> print(a)
60
>>> a = a + b    # a와 b를 더하여 a에 다시 넣습니다. 즉, a를 b만큼 증가시킵니다.
>>> print(a)
80
```

1. 정수 자료형과 연산

◆ 할당 연산자와 산술 연산자

- '='의 실행은 항상 오른쪽에서 왼쪽으로 가야 함.
- 만약에 '='이 여러 개 있으면, 가장 오른쪽부터 왼쪽으로 차례대로 수행됨.

```
>>> a = b = c = d = 10
```



변수 d가 가장 먼저 10을 갖게 되고, 다음으로 c, b, a 순서대로 10을 갖게 되겠죠.

```
>>> x = 10; y = 20
```

```
>>> a = b = c = x + y
```

```
>>> print(a, b, c, x, y)
```

```
30 30 30 10 20
```

x+y를 계산하여 먼저 c에 넣고, c 값을 b에 넣고, b 값을 a에 넣게 됩니다.


1. 정수 자료형과 연산

◆ 산술 연산자 간략히 쓰기

원래 연산식	간략히 쓰기	의미
$a = a + b$	$a += b$	a에 b만큼 더합니다. (a 자체가 b만큼 증가함)
$a = a - b$	$a -= b$	a에서 b만큼 뺍니다. (a 자체가 b만큼 감소함)
$a = a * b$	$a *= b$	a를 b배 합니다. (a 자체가 b배가 됨)
$a = a / b$	$a /= b$	a를 b로 나눈 값을 다시 a에 저장합니다.
$a = a ** b$	$a **= b$	a의 b 제곱 값을 다시 a에 저장합니다.
$a = a // b$	$a //= b$	a에 b로 나눈 몫을 다시 a에 저장합니다.
$a = a \% b$	$a \% = b$	a를 b로 나눈 나머지를 다시 a에 저장합니다.

1. 정수 자료형과 연산

◆ 연산자 우선 순위

우선순위	연산자	결합 순서
<div style="text-align: center;">  높음 </div>	**	←
	*, /, //, %	→
	+, -	→

```
>>> 2 ** 3 ** 2
512
>>> (2 ** 3) ** 2
64
```

- 7가지 연산자가 모두 섞여 있는 경우 연산 예제

```

      3      2      1
>>> 2 + 3 * 4 ** 2
50
      2      3      4      1
>>> 50 // 7 + 10000 - 3 ** 2
9998

```

```

      1      3      4      2
>>> 4 ** 2 * 2 * 3 ** 4
2592
      4      2      5      3      1
>>> 100 - 30 % 4 + 2 * 10 ** 3
2098

```

1. 정수 자료형과 연산

◆ 연산자 사용 시에 주의점

- $3x^2 + 4x + 5$ 수식 쓰기 (곱하기는 반드시 * 기호를 사용해야 함)

$$3 * x ** 2 + 4 * x + 5$$

- 계산의 순서를 바꾸려면, 반드시 괄호를 사용해야 함. (중괄호를 연산식에 사용할 수 없음)

O

X

$$(a + b * (c + a)) / (b - c) \quad \{a + b * (c + a)\} / (b - c)$$

중괄호(집합 기호)는 파이썬에서 집합, 사전을 표현할 때 쓰기 때문에 수식에 사용할 수 없다.

1. 정수 자료형과 연산

CODE 01 하루가 몇 초인지 계산하여 출력하는 코드입니다.

```
sec_in_a_min = 60      # 1분은 60초  
min_in_an_hour = 60   # 1시간은 60분  
hours_in_a_day = 24    # 하루는 24시간  
sec_in_a_day = hours_in_a_day * min_in_an_hour * sec_in_a_min  
print(sec_in_a_day, 'seconds in a day')
```

[결과]

86400 seconds in a day

1. 정수 자료형과 연산

CODE 02 형의 나이는 22살, 나의 나이는 18살, 동생은 16살이에요. 세 사람의 나이 차이를 출력합니다.

```
age_older_brother = 22          # 형 나이
age_me = 18                     # 내 나이
age_younger_brother = 16        # 동생 나이
diff1 = age_older_brother - age_me    # 형과 나의 나이 차이
diff2 = age_older_brother - age_younger_brother    # 형과 동생의 나이 차이
diff3 = age_me - age_younger_brother    # 나와 동생의 나이 차이
print('형은 나보다', diff1, '살 위입니다.')
print('형은 동생보다', diff2, '살 위입니다.')
print('나는 동생보다', diff3, '살 위입니다.')
```

[결과]

형은 나보다 4 살 위입니다.
형은 동생보다 6 살 위입니다.
나는 동생보다 2 살 위입니다.

1. 정수 자료형과 연산

CODE 03

세 명의 성적이 있습니다. 각각의 성적은 90, 85, 88입니다. 세 명의 평균을 구해 보세요.

```
score1 = 90
score2 = 85
score3 = 89
total = score1 + score2      # total은 두 성적의 합을 저장합니다.
total += score3              # total에 score3을 더합니다.
average = total / 3          # 평균을 구합니다.
print('총점:', total)
print('평균:', average)
```

[결과]

총점 : 264

평균 : 88.0

2. 실수 자료형과 연산

◆ 실수 자료형 표현하기

- 숫자에 소수점이 있으면 실수 자료형으로 판단함.
- 아래 a, b, c, d, e는 모두 실수임.

```
>>> a = 3.5; b = 3.0; c = 0.5; d = 3.; e = .5
```

```
>>> a, b, c, d, e
```

```
(3.5, 3.0, 0.5, 3.0, 0.5)
```

```
>>> type(a), type(b), type(c), type(d), type(e)
```

```
(<class 'float'>, <class 'float'>, <class 'float'>, <class 'float'>, <class 'float'>)
```

2. 실수 자료형과 연산

◆ 실수의 과학적 표기법 - 실수를 표현하는 다른 방법

```
>>> x = 3.7e8
```

```
>>> print(x)
```

```
370000000.0
```

```
>>> y = 2.5E-3
```

```
>>> print(y)
```

```
0.0025
```

$3.7e8 \rightarrow 3.7 * 10^8$

$2.5E-3 \rightarrow 2.5 * 10^{-3}$

2. 실수 자료형과 연산

◆ 실수는 immutable하다

◆ 실수의 특징

- $0.2 + 0.1$, $3 * 0.1$ 의 결과가 이상함.
- 이는 컴퓨터가 실수를 저장하는 방식 때문에 생기는 문제임.
- 우리는 소수점 아래 몇 째 자리까지 출력하도록 시키는 것으로 문제를 해결함.

```
>>> 0.1 + 0.1
```

```
0.2
```

```
>>> 2 * 0.2
```

```
0.4
```

```
>>> 0.2 + 0.1
```

```
0.30000000000000004
```

```
>>> 3 * 0.1
```

```
0.30000000000000004
```


2. 실수 자료형과 연산

◆ 실수의 연산

```
>>> 3.5 + 5.7
```

```
9.2
```

```
>>> 9.2 - 5.7
```

```
3.4999999999999999
```

```
>>> 2.1 * 3.0
```

```
6.3000000000000001
```

```
>>> 10.5 / 2.3
```

```
4.565217391304349
```

```
>>> 10.5 / 2.5
```

```
4.2
```

```
>>> 2.5 ** 1.5
```

```
3.952847075210474
```

```
>>> 5.8 // 2.2
```

```
2.0
```

```
>>> 5.8 % 2.2
```

```
1.3999999999999995
```

3. 복소수 자료형과 연산

◆ 복소수 자료형 표현하기

- 실수부와 허수부로 구성된 자료형.
- 허수부에는 j 또는 J라고 붙임.
- 허수부가 1j일 때는 반드시 숫자 1을 붙여야 함.

```
>>> a = 3 + 5j
>>> b = 2 + 7J
>>> print(a, b)
(3+5j) (2+7j)
>>> type(a); type(b)
<class 'complex'>
<class 'complex'>
```

```
>>> c = 2.5 + j          # j라고 적은 부분에서 문제가 발생합니다.
Traceback (most recent call last):
  File "<pyshell#94>", line 1, in <module>
    c = 2.5 + j
NameError: name 'j' is not defined
>>> c = 2.5 + 1J        # 반드시 1을 붙여 주세요.
>>> print(c)
(2.5+1j)
```

3. 복소수 자료형과 연산

◆ 복소수는 immutable 하다

◆ 복소수 연산

- $+$, $-$, $*$, $/$, $**$ 는 사용 가능하지만, $//$ 와 $%$ 는 사용할 수 없음.
- 복소수.real, 복소수.imag라고 하면 실수부와 허수부를 알 수 있음.

```
>>> a = 3 + 5j
>>> b = 2 + 7j
>>> a + b      # 실수부끼리, 허수부끼리 더합니다.
(5+12j)
>>> a - b      # 실수부끼리, 허수부끼리 빼기 연산을 합니다.
(1-2j)
>>> a * b
(-29+31j)
>>> a / b
(0.7735849056603773-0.2075471698113208j)
```

```
>>> a = 3+5j
>>> a.real
3.0
>>> a.imag
5.0
```

4. 자료형 변환

◆ 정수로 변환하기 - int() 함수 사용

- int() 함수의 인수에는 int, float, bool, str 만 넣을 수 있음.

변환	예제
정수 ← 실수 (int ← float)	소수점 뒤의 숫자들을 버리면서 정수로 변환함.
	<pre>>>> a = 5.2; b = 3.5; c = 1.9 >>> x = int(a); y = int(b); z = int(c) >>> x, y, z (5, 3, 1)</pre>
정수 ← 부울 (int ← bool)	True는 1로, False는 0으로 변환함.
	<pre>>>> int(True) 1 >>> int(False) 0</pre>

4. 자료형 변환

◆ 정수로 변환하기 - int() 함수 사용

변환	예제
정수 ← 문자열 (int ← str)	정수로 된 문자열만 정수로 변환 가능함.
	<pre>>>> a = '10'; b = '3.5' >>> int(a) # '10'은 정수 문자열 10 >>> int(b) # '3.5'는 실수 문자열이라서 에러 발생 Traceback (most recent call last): File "<pyshell#31>", line 1, in <module> int(b) ValueError: invalid literal for int() with base 10: '3.5'</pre>

4. 자료형 변환

◆ int() 함수가 필요한 경우

```
>>> a = '25'
>>> b = a + 5      # 문자열 + 정수는 자료형이 맞지 않음 (TypeError)
Traceback (most recent call last):
  File "<pyshell#183>", line 1, in <module>
    b = a + 5
TypeError: must be str, not int
>>> b = int(a) + 5  # 문자열 '25'를 정수로 변환한 후에 5와 더함.
>>> print(b)
30
```

4. 자료형 변환

◆ 실수로 변환하기 - float() 함수 사용

- float() 함수의 인수에는 int, float, bool, str 만 넣을 수 있음.

변환	예제
실수 ← 정수 (float ← int)	정수에 .0을 붙여서 실수로 변환함.
	<pre>>>> a = 5; b = 0; c = -7 >>> x = float(a); y = float(b); z = float(c) >>> x, y, z (5.0, 0.0, -7.0)</pre>
실수 ← 부울 (float ← bool)	True는 1.0으로, False는 0.0으로 변환함.
	<pre>>>> v = True; w = False >>> g = float(v); h = float(w) >>> g, h (1.0, 0.0)</pre>

4. 자료형 변환

◆ 정수로 변환하기 - float() 함수 사용

변환	예제
실수 ← 문자열 (float ← str)	정수 또는 실수로 된 문자열만 실수로 변환 가능함.
	<pre>>>> a = '5'; b = '5.7' >>> float(a) # 정수로 된 문자열 5.0 >>> float(b) # 실수로 된 문자열 5.7</pre>

4. 자료형 변환

◆ float() 함수가 필요한 경우

```
>>> x = '2.5'
```

```
>>> y = 3.0
```

```
>>> z = x + y          # 문자열 + 실수는 계산 불가능함.
```

Traceback (most recent call last):

File "<pyshell#57>", line 1, in <module>

z = x + y

TypeError: must be str, not float

```
>>> z = float(x) + y    # 문자열을 실수로 변환한 후에 계산함.
```

```
>>> z
```

```
5.5
```

4. 자료형 변환

◆ 복소수로 변환하기 - complex() 함수 사용

- complex() 함수의 인수에는 int, float, bool, str 만 넣을 수 있음.

```
>>> complex(3)
```

```
(3+0j)
```

```
>>> complex(2.3)
```

```
(2.3+0j)
```

```
>>> complex('3')
```

```
(3+0j)
```

```
>>> complex('2.3')
```

```
(2.3+0j)
```

```
>>> complex(True)
```

```
(1+0j)
```

```
>>> complex(False)
```

```
0j
```

```
>>> complex('2+3j')
```

```
(2+3j)
```

5. 다양한 자료형이 섞인 연산식

- ◆ 기본적으로 수치 자료형끼리는 섞어서 계산할 수 있음.
- ◆ bool 자료형도 True는 1, False는 0으로 간주하므로 숫자들과 섞어서 계산 가능함.

```
>>> i = 10; f = 2.5; b = True; c = 5 + 7j    # 정수, 실수, 부울, 복소수
>>> i + f, i + b, i + c                    # 정수 + 실수, 정수 + 부울, 정수 + 복소수
(12.5, 11, (15+7j))
>>> f + b, f + c                          # 실수 + 부울, 실수 + 복소수
(3.5, (7.5+7j))
>>> b + c                                # 부울 + 복소수
(6+7j)
```

5. 다양한 자료형이 섞인 연산식

- ◆ 정수끼리 계산하면 나눗셈(/) 결과는 실수이다.

```
>>> a = 50; b = 6; c = 5
```

```
>>> a + b, a - b, a * b
```

```
(56, 44, 300)
```

```
>>> a / b, a / c
```

나누기 연산의 결과는 항상 실수임.

```
(8.333333333333334, 10.0)
```

```
>>> a ** b, a // b, a % b
```

```
(15625000000, 8, 2)
```

5. 다양한 자료형이 섞인 연산식

- ◆ 정수와 실수를 섞어서 계산하면 결과는 항상 실수임.

```
>>> x = 100; y = 2.5  
>>> x + y, x - y, x * y, x / y  
(102.5, 97.5, 250.0, 40.0)  
>>> x ** y, x // y, x % y  
(100000.0, 40.0, 0.0)
```

6. 수치 연산 함수들

◆ 수치 연산 관련 내장함수

함수	설명
<code>abs(x)</code>	x의 절대값을 반환한다
<code>divmod(x,y)</code>	$(x//y, x\%y)$ 쌍을 반환한다
<code>pow(x,y)</code>	x^y 을 반환한다

6. 수치 연산 함수들

◆ 수치 연산 관련 내장함수

함수명	사용법
abs()	<pre>>>> x = -7 >>> y = abs(x) >>> y 7</pre> <p># x에 넣은 수의 절대값을 반환함.</p>
divmod()	<pre>>>> a = 100; b = 8 >>> p, q = divmod(a, b) >>> p, q (12, 4)</pre> <p># a//b 값은 p에, a%b 값은 q에 반환함.</p>
pow()	<pre>>>> a = 5; b = 3 >>> y = pow(a,b) >>> y 125</pre> <p># a^b을 반환합니다.</p>

7. math 모듈

- ◆ 모듈 - 연관된 함수들을 모아서 모듈로 관리함
- ◆ math 모듈 - 수학 관련 함수들을 모아 놓은 모듈
- ◆ 모듈을 import 한 후에 모듈에 있는 함수를 사용할 수 있다.

```
>>> import math          # math 모듈을 가져와야 그 안에 함수를 사용 가능함.  
>>> math.sqrt(100)       # 반드시 math를 붙여야 합니다.  
10.0
```


7. math 모듈

함수명	의미	예제
<code>math.ceil(x)</code>	x : 정수 또는 실수 x 이상의 수 중에서 가장 작은 정수를 반환함.	<pre>>>> math.ceil(5.1) 6 >>> math.ceil(5.9999) 6 >>> math.ceil(5) 5</pre>
<code>math.floor(x)</code>	x : 정수 또는 실수 x 이하의 수 중에서 가장 큰 정수를 반환함.	<pre>>>> math.floor(3.5) 3 >>> math.floor(3.9999) 3 >>> math.floor(3) 3</pre>
<code>math.fabs(x)</code>	x의 절대값을 실수로 반환함. 내장 함수 <code>abs()</code> 와 하는 일을 같은데, <code>abs()</code> 함수는 정수 결과를 반환함.	<pre>>>> math.fabs(-5) 5.0 >>> math.fabs(5) 5.0</pre>

7. math 모듈

함수명	의미	예제
<code>math.pow(x,y)</code>	x^y 결과를 실수로 반환합니다. 내장 함수 <code>pow()</code> 와 하는 일이 같은데, <code>pow()</code> 함수는 정수 결과를 반환해 줍니다.	<pre>>>> math.pow(2,3) 8.0</pre>
<code>math.pi</code>	원주율 π 값을 알려 줍니다. 함수가 아니라서 괄호를 붙이지 않습니다.	<pre>>>> math.pi 3.141592653589793</pre>
<code>math.sqrt(x)</code>	x 의 루트값을 반환합니다.	<pre>>>> math.sqrt(100) 10.0 >>> math.sqrt(1000) 31.622776601683793</pre>
<code>math.trunc(x)</code>	x 의 소수점 이하를 버립니다.	<pre>>>> math.trunc(5.5) 5 >>> math.trunc(5.1) 5</pre>

8. 정리

- ◆ 수치 자료형의 연산에 대해 학습함.
- ◆ 서로 다른 자료형 간의 변환에 대해 학습함.
- ◆ 다양한 자료형이 섞인 연산식 계산에 대해 학습함.
- ◆ 수학 연산 관련 함수들에 대해 학습함.