02.파이썬 기본 문법-2

2 자료형과 연산자

2.1 기본자료형

파이썬이 처리하는 기본 데이터 유형

유 형			설명	예시	선언형태
수치 자료형	정수형	**int**	양/음의 정수	1, 21, 100, -9	data = 23
	실수형	**float**	소수점이 포함된 실수	10.2, -9.7	data = 78.2
	복소수형	**complex**	실수부+허수부	4 + 5j	data = 4+5j
문자 자료형		**str**	' 또는 " 에 들어가있는 문자형	'abc', "123"	data = 'abc'
논리/불린 자료형		**bool**	참 또는 거짓	True, False	data = True

2.2 bool 자료형

참이나 거짓을 나타내는 True, Flase 두 상수를 갖는다.

[예제 bool.py]

```
a = 1
b = a < 10
print( b, type(b) )
```

```
print(True + True)
print(1 == True)
print(0 == False)
```

```
b1 = True
b2 = False

print(b1 + 10)
print(b2 + 10)
print(True + True)
```

2.3 수치 자료형

1) 정수형

[예제 int.py]

10진, 2진, 8진, 16진 정수를 자료형으로 사용한다

```
a = 101
print(type(a))
print(isinstance(a, int))

b = 0b101  # 2진수
c = 0o101  # 8진수
d = 0x101  # 16진수
```

파이썬 3.x 에서는 long형이 없어지고 int 타입으로 처리된다. 따라서 표현범위는 무한대이다.

```
e = 2**1024
print(type(e))
print(e)
```

2) 실수형

[예제 float.py]

소수점을 포함하거나 e나 E로 지수로 표현할 수 있다.

```
a = 1.2
print(type(a))
print(isinstance(a, float))
print(a.is_integer())

b = 3e3
c = -0.2e-4

print(a, b, c)
```

3) 복소수형

[예제 complex.py]

복소수 타입의 객체는 실수부+허수부로 나뉘며 허수부에는 j 또는 J를 숫자 뒤에 붙인다.

```
a = 4 + 5j
print(type(a))
print(isinstance(a, complex))
```

복소수 연산이 가능하며 실수부와 허수부 값만 따로 참조할 수 있다.

```
a = 4 + 5j
b = 7 - 2j
print(a + b)
print(b.real, b.imag)
```

complex 함수를 이용하여 복소수 타입의 객체를 만들 수 있다.

```
a = 2.0
print(type(a))
print(complex(a))
```

4) 수치 자료형의 연산자

4-1) 산술 연산자

[예제 arithmetic_operator.py]

```
print(8 * 3)
print(8 + 3)
print(8 - 3)
print(8 / 3)
```

나눗셈(/)은 파이썬 3.x 부터는 혼란을 피하기 위해 실제 나눗셈을 한다.

```
print(8 / 3)
print(8.0 / 3)
print(8 / 3.0)
print(8.0 / 3.0)
```

//(몫 연산자), **(지수승), %(나머지 연산)

```
print(2 // 3)
print(2 ** 3)
print(2 % 3)
print(divmod(14, 3))
```

4-2) 산술 연산자 우선순위

- + , - (단항 연산자) ←
- ** ←
- *, /, %, // →
- +, - (더하기, 빼기) →

```
print(2 + 3 * 4)
print(-2 + 3 * 4)
print(-(2+3) * 4)

print(4 / 2 * 2)
print(4 / (2 * 2))

print(2 ** 3 ** 4)
print((2 ** 3) ** 4)
print(2 ** (3 ** 4))
```

4-3) 확장 연산자

연산자	의미		
+=	수를 더한 후 다시 대입	x = x+5	x +=5
-=	수를 뺀 후 다시 대입	x = x-3	x -=3
*=	수를 곱한 후 다시 대입	x = x*5	x *=5
/=	수를 나눈 후 다시 대입	x = x/2	x /=2
**=	수를 거듭 제곱한 후 다시 대입	x = x**3	x **=3
//=	수를 나눈 후 몫을 대입	x = x//4	x //=3
%=	수를 나눈 후 나머지를 대입	x = x%3	x %=3

4-4) 관계 연산자

[예제 relational_operator.py]

객체의 대소 비교하는 연산

```
print(1 > 3)
print(2 < 4)
print(4 <= 5)
print(4 >= 5)
print(6 == 9)
print(6 != 9)
```

객체의 대소 비교하는 연산

```
a = 6
print(0 < a < 10)
print(0 < a and a < 10)
```

== 는 객체의 값을 비교한다.(동질성)

같은 객체(동일성)을 비교할 때는 is 연산자를 사용한다. (주소값 비교)

```
a = 10
b = 20
c = a
print(a == b)
print(a is b)
print(a is c)
```

4-5) 논리연산자

[예제 logical_operator.py]

- **and 연산자**

표현식	결과
True and True	True
True and False	False
False and True	False
False and False	False

- **or 연산자**

표현식	결과
True or True	True
True or False	True
False or True	False
False or False	False

- **not 연산자**

표현식	결과
not True	False
not False	True

4-6) 내장 수치 함수

[예제 arthmetic_func.py]

```
print(abs(-3))          # 절대값으로
print(int(3.1415))      # 정수형으로
print(float(3))         # 실수형으로
print(complex(3))       # 복소수형으로
print(pow(2, 10))       # 승수 계산
```

2.3 문자 자료형

' 또는 " 로 묶인 문자들의 모임이다.

1) 문자열의 정의

[예제 str.py]

```
s = ''
str1 = 'hello world'
str2 = "hello world"
print(type(s), type(str1), type(str2))
print(isinstance(str2, str))
```

여러 줄 문자열도 정의할 수 있다.

```
str3 = """ABCDEFG
abcdef
가나다라마
123456789"""
print(str3)
```

```
str3 = '''ABCDEFG
abcdef
가나다라마
123456789'''
print(str3)
```

이스케이프 문자표

코드	설명
₩n	문자열 안에서 줄을 바꿀 때 사용
₩t	문자열 사이에 탭 간격을 줄 때 사용
₩₩	역슬래쉬 ₩ 을 표현할 때 사용
₩'	작은따옴표(') 을 표현할 때 사용
₩"	큰 따옴표(")을 그대로 사용할 때 사용
₩r	현재 커서를 가장 앞으로 이동시킬대 사용
₩b	백스페이스

```
print('Hello₩nWorld₩nI₩'d Say "hello World"')
print("Hello₩nWorld₩nI'd Say ₩"hello World₩"")
print("Hello₩rWorld")
print("Hello₩bWorld")
print("Hello₩tWorld")
```

2) 문자열의 연산

기본적으로 시퀀스형이므로 시퀀스의 연산(인덱싱, 슬라이싱, 연결(+), 반복(*), len(), in, not in) 등의 연산이 가능하다.

F	i	r	s	t	S	t	r	i	n	g	
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	→
[-11]	[-10]	[-9]	[-8]	[-7]	[-6]	[-5]	[-4]	[-3]	[-2]	[-1]	←

```
str1 = "FirstString"
str2 = "SecondString"

print(str1 + str2)
print(str1*3)
print(str1[2])
print(str2[2:5])
print(len(str2))
```

2 자료형과 연산자

문자열 객체의 내용은 변경 할 수 없는 변경 불가(immutable) 자료형이다.

```
str1[0] = 'Q'
```

X

문자열 객체와 정수형 객체는 + 연산을 할 수 없다.

```
name = "길동"
age = 40
print( name + age )
```

X

3) 문자열 포멧팅

3-1) 문자열 포멧 코드를 이용한 포멧팅

```
age = 33
str1 = "현재 온도는 %d 도 입니다." % 23
str2 = "오늘은 %s년 %d 월 %d 일 입니다." % ("2017", 5, 11)
area = 76.483

print(str1)
print(str2)
print("제나이는 %d 살 입니다." % age )
print("원의 넓이는 %d 입니다." % area )

print("제나이는 %10d 살 입니다." % age )
print("제나이는 %-10d 살 입니다." % age )
print("원의 넓이는 %7.1f 입니다." % area )
print("원의 넓이는 %-20.1f 입니다." % area )
```

코드	설명
%s	문자열
%d	정수
%f	부동소수
%%	% 자체
%c	문자 1개
%o	8진수
%x	16진수

3-2) format() 함수를 이용한 포멧팅

```
age =35
print("제나이는 {0}살 입니다.".format(43) )
print("제나이는 {0}살 입니다.".format("스무") )
print("제나이는 {0}살 입니다.".format(age) )
print("오늘은 {0}년 {1} 월 {2} 일 입니다.".format("2017", 5, 11))
print("오늘은 {year}년 {month} 월 {day} 일 입니다.".format(year="2017", month=5, day=11))



print("오늘은 {0:<10}년 {1:>10} 월 {2:^10} 일 입니다.".format("2017", 5, 11))
```

4) 문자열 메소드

문자열 객체는 다양한 메소드를 제공한다.

[예제 str_func.py]

4-1) 대소문자 관련 메소드

```
s = 'i like Python'

print(s.upper())
print(s.lower())
print(s.swapcase())
print(s.capitalize())
print(s.title())
```

4-2) 검색관련 메소드

```
s = 'I Like Phyton. I Like Java Also'
print(s.count('Like'))

print(s.find('Like'))
print(s.find('Like', 5))
print(s.find('JS'))
print(s.rfind('a'))
# print(s.lfind('Like'))    X



print(s.index('to'))
# print(s.index('JS'))
print(s.rindex('Like'))

print(s.startswith('I Like'))
print(s.startswith('Like', 2))
print(s.endswith('Also'))
print(s.endswith('Java', 0, 26))
```

3-3) 편집과 치환에 관한 메소드

```
s = '   spam and ham    '
print(s.strip())
print(s.rstrip())
print(s.lstrip())

s = '<><abc><><defg><><>'
print(s.strip('<>'))

s = 'Hello Java Java java'
print(s.replace( 'Java', 'Python'))
```

3-4) 정렬에 관한 메소드

```
s = 'king and queen'

print(s.center(60))
print(s.center(60, '-'))
print(s.ljust(60, '-'))
print(s.rjust(60, '-'))
```

3-5) 판별에 관한 메소드

```
print('1234'.isdigit())
print('abcd'.isalpha())

print('1234'.isalpha())
print('abcd'.isdigit())

print('abcd'.islower())
print('ABCD'.isupper())

print('₩n₩n'.isspace())
print('    '.isspace())
print(''.isspace())
```

3-6) 0 으로 채우기

```
print('20'.zfill(10))
print('1234'.zfill(10))
```