02 기본 문법

content

1. 변수와 자료형

- **변수명 선언**
  - 알파벳, 숫자, 밑줄(_)로 선언할 수 있다.
  - 변수명은 의미 있는 단어로 표기하는 것이 좋다.
  - 변수명은 대소문자가 구분된다.
  - 특별한 의미가 있는 예약어는 사용할 수 없다.

맨앞으로 올수 없다

1. 변수와 자료형

- **기본 자료형**
  - **정수형(integer type)** : 자연수를 포함해 값의 영역이 정수로 한정된 값.
  - **실수형(floating-point type)** : 소수점이 포함된 값.
  - **문자열형(string type)** : 값이 문자로 출력되는 자료형.
  - **불린형(boolean type)** : 논리형으로, 참(True) 또는 거짓(False)을 표현할 때 사용.

유형	자료형	설명	예	선언 형태
수치형	정수형	양수와 정수	1, 2, 3, 100, −9	data = 1
	실수형	소수점이 포함된 실수	10.2, −9.3, 9.0	data = 9.0
문자형	문자형	따옴표에 들어가 있는 문자형	abc, a20abc	data = 'abc'
논리형	불린형	참 또는 거짓	True, False	data = True

1. 변수와 자료형

- **동적 타이핑(dynamic typing)**
  - 변수의 메모리 공간을 확보하는 행위가 실행 시점에서 발생하는 것을 뜻함.
  - C나 자바는 int data=8 과 같이 data라는 변수가 정수 형이라고 사전에 선언
  - 파이썬은 data = 8 형태로 선언, data라는 변수의 자료 형이 정수(integer)인지 실수(float)인지를 프로그래머가 아닌 인터프리터가 실행될 때 스스로 판단
  - 실행 시점에 동적으로 판단하므로 파이썬 언어가 동적으로 자료 형의 결정
  - 다른 언어들과 달리 파이썬은 매우 유연한 언어로, **할당 받는 메모리 공간**도 저장되는 값에 따라 동적으로 할당 받을 수 있다.

2. 기본 자료형

- **숫자형**
  - 숫자형(number)이란?
    - 정수, 실수 와 같이 숫자 형태로 이루어진 자료형

항목	파이썬 사용 예
정수	123, -345, 0
실수	123.45, -1234.5, 3.4e10
8진수	0o34, 0o25
16진수	0x2A, 0xFF

2. 기본 자료형

- **숫자형**
  - 숫자형은 어떻게 만들고 사용할까?

정수형(integer) : 정수를 뜻하는 자료형

```
>>> a = 123    ← 양의 정수 대입
>>> a = -178   ← 음의 정수 대입
>>> a = 0      ← 숫자 0 대입
```

실수형(floating-point) : 소수점이 포함된 숫자

```
>>> a = 1.2
>>> a = -3.45
```

```
>>> a = 4.24E10    ← 4.24×10^10
>>> a = 4.24e-10   ← 4.24×10^-10
```

※ 컴퓨터식 지수 표현 방식

8진수(octal) : 숫자 0 + 알파벳 소문자 o 또는 대문자 O

```
>>> a = 0o177
>>> print(a)
127  ← 1×8^2+7×8+7=127
```

16진수(hexadecimal) : 숫자 0 + 알파벳 소문자 x

```
>>> a = 0x8ff
>>> b = 0xABC
>>> print(b)
2748  ← 10×16^2+11×16+12=2748
```

2. 기본 자료형

간단한 연산

■ 사칙연산(+,-,*,/)

```
>>> 25 + 30
55
>>> 30 - 12
18
>>> 50 * 3
150
>>> 30 / 5 //자동으로 실수로 형변환 됨
6.0
```

■ 제곱승(**)

```
>>> print(3 * 3 * 3 * 3 * 3)      # 3을 다섯 번 곱함
243
>>> print(3 ** 5)                 # 3의 5승
243
```

■ 나눗셈의 정수 몫(//)과 나머지(%) 연산

- 정수 몫을 반환하는 연산자는 2개의 빗금 기호(//), 나머지 연산자는 백분율 기호(%)

```
>>> print(7 // 2)                 # 7 나누기 2의 몫
3
>>> print(7 % 2)                  # 7 나누기 2의 나머지
1
```

2. 기본 자료형

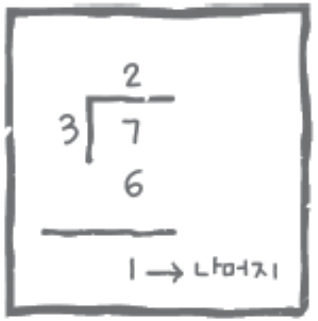
- 숫자형을 활용하기 위한 연산자
  - 사칙 연산

```
>>> a = 3
>>> b = 4
>>> a + b
7
>>> a - b
-1
>>> a * b
12
>>> a / b
0.75
```

```
>>> a = 3
>>> b = 4
>>> a ** b
81
```

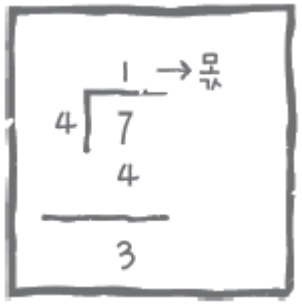
  - 나눗셈 후 나머지를 리턴하는 % 연산자

```
>>> 7 % 3
1
>>> 3 % 7
3
```



  - 나눗셈 후 정수 몫을 리턴하는 // 연산자

```
>>> 7 // 4
1
```



2. 기본 자료형

간단한 연산

■ 증가 연산과 감소 연산 복합 연산자

- 증가 연산자는 +=이고, 감소 연산자는 -=이다.

```
>>> a = 1                    # 변수 a에 1을 할당
>>> a = a + 1                # a에 1를 더한 후 그 값을 a에 할당
>>> print(a)                 # a 출력
2
>>> a += 1                   # a 증가 연산
>>> print(a)                 # a 출력
3
>>> a = a - 1                # a에서 1을 뺀 후 그 값을 a에 할당
>>> a -= 1                   # a 감소 연산
>>> print(a)                 # a 출력
1
```

1. 변수와 자료형

간단한 연산

■ 관계연산

연산자	의미
>	크다
<	작다
>=	크거나 같다
<=	작거나 같다
==	같다
!=	같지 않다

■ 비트 연산

연산자	의미
&	비트 단위 and
|	비트 단위 or
^	비트 단위 배타적 or(xor)
~	비트 단위 not
<<	비트 단위 왼쪽 쉬프트
>>	비트 단위 오른쪽 쉬프트

■ 논리연산

연산자	의미
and	그리고
or	또는
not	부정

1. 변수와 자료형

- **자료형 변환 : 정수형 -> 실수형 변환**
  - float( ) 함수 : 정수를 실수형으로 변환해 주는 함수

```
>>> a = 10                      # a 변수에 정수 데이터 10을 할당
>>> print(a)                    # a가 정수형으로 출력
10
>>> a = float(10)               # a를 실수형으로 변환 / 정수형인 경우 int()
>>> print(a)                    # a를 출력
10.0                            # a가 실수형으로 출력됨

>>> a = 10                       # a에 정수 데이터 10 할당
>>> b = 3                        # b에 정수 데이터 3 할당
>>> print(a / b)                 # 실수형으로 a 나누기 b를 출력
3.3333333333333335               # 실수형 결과값 출력
```

1. 변수와 자료형

- **자료형 변환 : 실수형-> 정수형 변환**
  - int( ) 함수 : 실수형을 정수형으로 변환해 주는 함수.

```
>>> a = int(10.7)
>>> b = int(10.3)

>>> print(a + b)              # 정수형 a와 b의 합을 출력
20
>>> print(a)                  # 정수형 a값 출력
10
>>> print(b)                  # 정수형 b값 출력
10
```

1. 변수와 자료형

- **자동 형 변환이 일어나는 경우**
  - '10 / 3' : 별도의 형 변환을 하지 않아도 자료 형이 실수로 변환
    - 파이썬의 대표적인 특징인 동적 타이핑 때문에 나타나는 현상 중 하나.
  - 값의 크기를 비교
    - 예 : 'I == True'라고 입력하면 결과는 True로 출력,
    - 아무것도 넣지 않은 " " 같은 문자열을 불린형과 비교하면 False로 인식.
  - 모두 파이썬의 특징에 의해 나타나는 현상

1. 변수와 자료형

- **자료형 변환 : 문자열형 ->숫자형 변환**
  - 실수형 값을 문자형으로 선언하기 위해서는 반드시 따옴표를 붙여 선언해야 한다

```
>>> a = '76.3'                  # a에 문자열 76.3을 할당, 문자열을 의미
>>> b = float(a)                # a를 실수형으로 변환 후 b에 할당
>>> print(a)                    # a값 출력
76.3
>>> print(b)                    # b값 출력
76.3
>>> print(a + b)                # a와 b를 더함 → 문자열과 숫자열의 덧셈이 불가능하여 에러 발생
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (nor "float") to str
```

1. 변수와 자료형

- **자료형 변환 : 더하기 연산시 자료형 통일**
  - 두 변수를 더하기 위해서는 반드시 두 변수의 자료형을 통일해야 한다.

```
>>> a = float(a)          # a를 실수형으로 변환 후 a에 할당
>>> b = a                 # 실수형 a값을 b에 할당
>>> print(a + b)          # 두 실수형을 더한 후 출력
152.6
```

- **자료형 변환 : 숫자형 -> 문자형 변환**
  - str( ) 함수 : 기존의 정수형이나 실수형을 문자열로 변환.
  - 문자형 간의 덧셈은 숫자 연산이 아닌 단순 붙이기(concatenate)가 일어난다.

```
>>> a = str(a)            # 실수형 a값을 문자열로 변환 후 a 할당
>>> b = str(b)            # 실수형 b값을 문자열로 변환 후 b 할당
>>> print(a + b)          # 두 값을 더한 후 출력
76.376.3                  # 문자열 간 덧셈은 문자열 간 단순 연결
```

1. 변수와 자료형

- **멤버 연산자**
  - 집합 연산에 사용하는 연산자
  - 연산자 종류 : in, not in

연산자	연산
in	왼쪽 항목의 값이 오른쪽 집합(객체)안에 포함 되어 있는지 판단
not in	왼쪽 항목의 값이 오른쪽 집합(객체)안에 포함 되어 있지 않는지 판단

```
a=3
b=[1,3,5,8,7]
print(a in b)  // 결과 : true
print( a not in b) // 결과 : false
```

1. 변수와 자료형

- **type( ) 함수 :** 자료형을 확인할 수 있는 함수.

```
>>> a = int(10.3)          # a는 정수형으로 10.3을 할당
>>> b = float(10.3)        # b는 실수형으로 10.3을 할당
>>> c = str(10.3)          # c는 문자형으로 10.3을 할당
>>>
>>> type(a)                # a의 타입을 출력
<class 'int'>
>>> type(b)                # b의 타입을 출력
<class 'float'>
>>> type(c)                # c의 타입을 출력
<class 'str'>
```

2. 문자형 자료형

- **문자열(string)이란?**
  - 문자, 단어 등으로 구성된 문자들의 집합

```
"Life is too short, You need Python"
"a"
"123"
```

2. 문자형 자료형

- 문자열은 어떻게 만들고 사용할까?

1. 큰따옴표(“)

```
"Hello World"
```

2. 작은따옴표(‘)

```
'Python is fun'
```

3. 큰따옴표 3개(“””)

```
"""Life is too short, You need python"""
```

4. 작은따옴표 3개(’’’)

```
'''Life is too short, You need python'''
```

2. 문자형 자료형

- **문자열 안에 작은따옴표나 큰따옴표를 포함시키고 싶을 때**

1. **작은따옴표(')**
   - 큰따옴표(")로 둘러싸기

```
>>> food = "Python's favorite food is perl"
```

2. **큰따옴표(")**
   - 작은따옴표(')로 둘러싸기

```
>>> say = '"Python is very easy." he says.'
```

3. **역슬래시 사용하기**
   - 역슬래시(\) 뒤의 작은따옴표(')나 큰따옴표(")는 문자열을 둘러싸는 기호의 의미가 아니라 문자 ('), (") 그 자체를 의미

```
>>> food = 'Python\'s favorite food is perl'
>>> say = "\"Python is very easy.\" he says."
```

2. 문자형 자료형

- 여러 줄인 문자열을 변수에 대입하고 싶을 때

1. 이스케이프 코드 '\n' 삽입

```
>>> multiline = "Life is too short\nYou need python"
```

2. 작은따옴표 3개('')

```
>>> multiline = '''
... Life is too short
... You need python
... '''
```

작은따옴표 3개를 사용한 경우

3. 큰따옴표 3개("")

```
>>> multiline = """
... Life is too short
... You need python
... """
```

큰따옴표 3개를 사용한 경우

2. 문자형 자료형

문자열 연산하기

1. 문자열 더해서 연결하기

```
>>> head = "Python"
>>> tail = " is fun!"
>>> head + tail
'Python is fun!'
```

2. 문자열 곱하기

```
>>> a = "python"
>>> a * 2
'pythonpython'
```

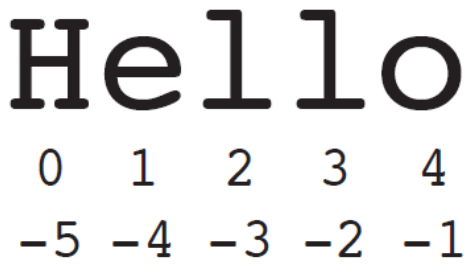
3. 문자열 길이 구하기

- 파이썬 기본 내장 함수 len() 사용

```
>>> a = "Life is too short"
>>> len(a)
17
```

2. 문자형 자료형

- **문자열의 인덱싱**
  - 글자 하나하나가 상대적인 주소(offset)를 가지는데, 이 주소를 사용해 할당된 값을 가져오는 인덱싱을 사용할 수 있다




```
>>> a = "abcde"
>>> print(a[0], a[4])              # a 변수의 0번째, 4번째 주소에 있는 값
a e
>>> print(a[-1], a[-5])            # a 변수의 오른쪽에서 0번째, 4번째 주소에 있는 값
e a
```

2. 문자형 자료형

- **문자열 인덱싱 예:**

L	i	f	e		i	s		t	o	o		s	h	o	r	t	,		Y	o	u		n	e	e	d		P	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

```
>>> a = "Life is too short, You need Python"
>>> a[0]
'L'
>>> a[12]
's'
>>> a[-1]
'n'
```

"파이썬은 0부터 숫자를 센다."

```
a[0]: 'L', a[1]: 'i', a[2]: 'f', a[3]: 'e', a[4]: ' ', …
```

2. 문자형 자료형

- **문자열 인덱싱과 슬라이싱**
  - **문자열 슬라이싱(slicing)**
    - '잘라낸다'는 의미
    - a[start:end:step]
      - 시작 번호부터 끝 번호까지의 문자를 뽑아 냄
      - 끝 번호에 해당하는 것은 포함하지 않음

L	i	f	e		i	s		t	o	o		s	h	o	r	t	,		Y	o	u		n	e	e	d		P	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

```
>>> a = "Life is too short, You need Python"
>>> a[0:4]
'Life'

>>> a = "20230331Rainy"
>>> date = a[:8]
>>> weather = a[8:]
>>> date
'20230331'
>>> weather
'Rainy'
```

2. 문자형 자료형

- 문자열 슬라이싱(slicing) 예

```
>>> a = "TEAMLAB MOOC, AWESOME Python"
>>> print(a[0:6], " AND ", a[-9:])          # a 변수의 0부터 5까지, -9부터 끝까지
TEAMLA  AND  ME Pyhon
>>> print(a[:])                             # a 변수의 처음부터 끝까지
TEAMLAB MOOC, AWESOME Python
>>> print(a[-50:50])                        # 범위를 넘어갈 경우 자동으로 최대 범위를 지정
TEAMLAB MOOC, AWESOME Python
>>> print(a[::2], " AND ", a[::-1])
TALBMO,AEOEPto  AND  nohtyP EMOSEWA ,COOM BALMAET
```

2. 문자형 자료형

문자열 포매팅이란?

문자열 포매팅(formatting)

- 문자열 안에 어떤 값을 삽입하는 방법

1. 숫자 바로 대입
   - 문자열 포맷 코드 **%d**

```
>>> "I eat %d apples." % 3
'I eat 3 apples.'
```

2. 숫자 값을 나타내는 변수로 대입

```
>>> number = 3
>>> "I eat %d apples." % number
'I eat 3 apples.'
```

3. 문자열 바로 대입
   - 문자열 포맷 코드 **%s**

```
>>> "I eat %s apples." % "five"
'I eat five apples.'
```

4. 2개 이상의 값 넣기

```
>>> number = 10
>>> day = "three"
>>> "I ate %d apples. so I was sick for %s days." % (number, day)
'I ate 10 apples. so I was sick for three days.'
```

2. 문자열 자료형

- **문자열 포맷 코드**
  - 문자열 포맷 코드의 종류

코드	설명
%s	문자열(string)
%c	문자 1개(character)
%d	정수(integer)
%f	부동소수(floating-point)
%o	8진수
%x	16진수
%%	Literal %(문자 % 자체)

```
>>> "I have %s apples" % 3
'I have 3 apples'
>>> "rate is %s" % 3.234
'rate is 3.234'
```

2. 문자열 자료형

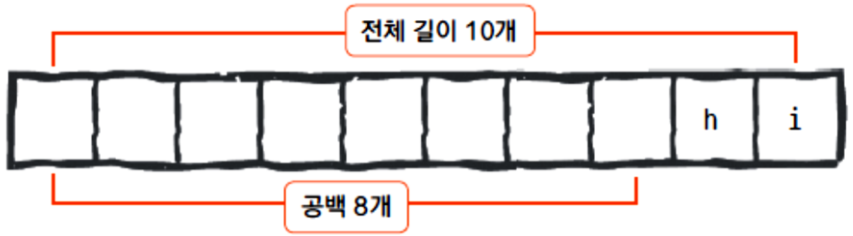
포맷 코드와 숫자 함께 사용하기

I. 정렬과 공백

- %s를 숫자와 함께 사용하면, 공백과 정렬 표현 가능

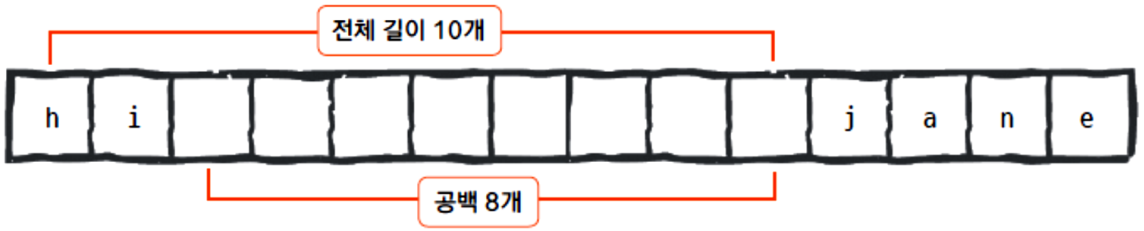
```
>>> "%10s" % "hi"
'        hi'
```

← hi가 오른쪽 정렬됨.




```
>>> "%-10sjane." % 'hi'
'hi        jane.'
```

← hi가 왼쪽 정렬됨.



2. 문자열 자료형

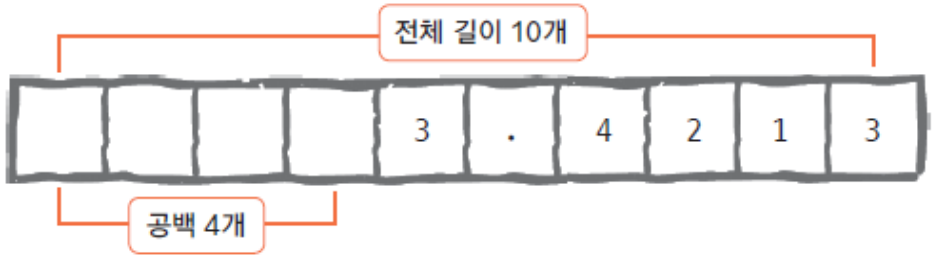
포맷 코드와 숫자 함께 사용하기

2. 소수점 표현하기

- %f를 숫자와 함께 사용하면, 소수점 뒤에 나올 숫자의 개수 조절 및 정렬 가능

```
>>> "%0.4f" % 3.42134234
'3.4213'
```

```
>>> "%10.4f" % 3.42134234
'    3.4213'
```



2. 문자열 자료형

- **format 함수를 사용한 포매팅**
  - 숫자 바로 대입하기

```
>>> "I eat {0} apples".format(3)
'I eat 3 apples'
```

  - 숫자 값을 가진 변수로 대입하기

```
>>> number = 3
>>> "I eat {0} apples".format(number)
'I eat 3 apples'
```

  - 문자열 바로 대입하기

```
>>> "I eat {0} apples".format("five")
'I eat five apples'
```

2. 문자열 자료형

- **format 함수를 사용한 포매팅**
  - 2개 이상의 값 넣기

```
>>> number = 10
>>> day = "three"
>>> "I ate {0} apples. so I was sick for {1} days.".format(number, day)
'I ate 10 apples. so I was sick for three days.'
```

  - 이름으로 넣기

```
>>> "I ate {number} apples. so I was sick for {day} days.".format(number=10, day=3)
'I ate 10 apples. so I was sick for 3 days.'
```

2. 문자열 자료형

- **format 함수를 사용한 포매팅**
  - 인덱스와 이름을 혼용해서 넣기

```
>>> "I ate {0} apples. so I was sick for {day} days.".format(10, day=3)
'I ate 10 apples. so I was sick for 3 days.'
```

  - 왼쪽 정렬

```
>>> "{0:<10}".format("hi")
'hi        '
```

  - 왼쪽 정렬

```
>>> "{0:>10}".format("hi")
'        hi'
```

  - 왼쪽 정렬

```
>>> "{0:^10}".format("hi")
'    hi    '
```

2. 문자열 자료형

- **format 함수를 사용한 포매팅**
  - 공백 채우기

```
>>> "{0:=^10}".format("hi")
'====hi===='
>>> "{0:!<10}".format("hi")
'hi!!!!!!!!'
```

  - 소수점 표현하기

```
>>> y = 3.42134234
>>> "{0:0.4f}".format(y)
'3.4213'
```

```
>>> "{0:10.4f}".format(y)
'    3.4213'
```

  - {또는} 문자 표현하기

```
>>> "{{ and }}".format()
'{ and }'
```

2. 문자열 자료형

- **f 문자열 포매팅**
  - 파이썬 3.6 버전부터 f 문자열 포매팅 기능 제공
  - 문자열 앞에 f 접두사를 붙이면 f 문자열 포매팅 기능 사용 가능

```
>>> name = '홍길동'
>>> age = 30
>>> f'나의 이름은 {name}입니다. 나이는 {age}입니다.'
'나의 이름은 홍길동입니다. 나이는 30입니다.'
```

```
>>> age = 30
>>> f'나는 내년이면 {age + 1}살이 된다.'
'나는 내년이면 31살이 된다.'
```

2. 문자열 자료형

- **f 문자열 포매팅**
  - 딕셔너리에서 사용

```
>>> d = {'name':'홍길동', 'age':30}
>>> f'나의 이름은 {d["name"]}입니다. 나이는 {d["age"]}입니다.'
'나의 이름은 홍길동입니다. 나이는 30입니다.'
```

  - 정렬

```
>>> f'{"hi":<10}' ← 왼쪽 정렬
'hi        '
>>> f'{"hi":>10}' ← 오른쪽 정렬
'        hi'
>>> f'{"hi":^10}' ← 가운데 정렬
'    hi    '
```

  - {} 문자 그대로 표시

```
>>> f'{{ and }}'
'{ and }'
```

2. 문자열 자료형

- **f 문자열 포매팅**
  - 공백 채우기

```
>>> f'{"hi":=^10}'  ← 가운데 정렬하고 '='로 공백 채우기
'====hi===='
>>> f'{"hi":!<10}'  ← 왼쪽 정렬하고 '!'로 공백 채우기
'hi!!!!!!!!'
```

  - 소수점 표현

```
>>> y = 3.42134234
>>> f'{y:0.4f}'     ← 소수점 4자리까지만 표현
'3.4213'
>>> f'{y:10.4f}'    ← 소수점 4자리까지 표현하고 총 자릿수를 '10'으로 맞춤.
'    3.4213'
```

2. 문자열 자료형

문자열 함수

함수명	기능
len()	문자열의 문자 개수를 반환
upper()	대문자로 변환
lower()	소문자로 변환
title()	각 단어의 앞글자만 대문자로 변환
capitalize()	첫 문자를 대문자로 변환
count('찾을 문자열')	'찾을 문자열'이 몇 개 들어 있는지 개수 반환
find('찾을 문자열')	'찾을 문자열'이 왼쪽 끝부터 시작하여 몇 번째에 있는지 반환
rfind('찾을 문자열')	find() 함수와 반대로 '찾을 문자열'이 오른쪽 끝부터 시작하여 몇 번째에 있는지 반환
startswith('찾을 문자열')	'찾을 문자열'로 시작하는지 여부 반환
endswith('찾을 문자열')	'찾을 문자열'로 끝나는지 여부 반환

2. 문자열 자료형

문자열 함수

함수명	기능
strip()	좌우 공백 삭제
rstrip()	오른쪽 공백 삭제
lstrip()	왼쪽 공백 삭제
split()	문자열을 공백이나 다른 문자로 나누어 리스트로 반환
isdigit()	문자열이 숫자인지 여부 반환
islower()	문자열이 소문자인지 여부 반환
isupper()	문자열이 대문자인지 여부 반환

2. 문자열 자료형

- **문자열 관련 함수들**
  - 문자열 자료형이 가진 내장 함수
  - 문자 개수 세기 - **count**
    - 특정 문자의 개수를 셈
  - 문자열 삽입 - **join**

```
>>> a = "hobby"
>>> a.count('b')
2
```

```
>>> ",".join('abcd')
'a,b,c,d'
```

2. 문자열 자료형

- **문자열 관련 함수들**
  - 위치 알려 주기 I – **find**
    - 찾는 문자열이 처음 나온 위치 반환
    - 없으면 -I 반환

```
>>> a = "Python is the best choice"
>>> a.find('b')
14    <- 문자열에서 b가 처음 나온 위치
>>> a.find('k')
-1
```

  - 위치 알려 주기 I – **find**
    - find와 마찬가지로, 찾는 문자열이 처음 나온 위치 반환
    - 단, 찾는 문자열이 없으면 오류 발생

```
>>> a = "Life is too short"
>>> a.index('t')
8
>>> a.index('k')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: substring not found
```

k가 없으므로 오류 발생

2. 문자열 자료형

- **문자열 관련 함수들**
  - 소문자를 대문자로 바꾸기 – **upper**

```
>>> a = "hi"
>>> a.upper()
'HI'
```

  - 대문자를 소문자로 바꾸기 – **upper**

```
>>> a = "HI"
>>> a.lower()
'hi'
```

  - 왼쪽 공백 지우기 - **lstrip**

```
>>> a = " hi "
>>> a.lstrip()
'hi '
```

  - 오른쪽 공백 지우기 - **lstrip**

```
>>> a= " hi "
>>> a.rstrip()
' hi'
```

2. 문자열 자료형

- **문자열 관련 함수들**
  - 양쪽 공백 지우기 – **strip**

```
>>> a = " hi "
>>> a.strip()
'hi'
```

  - 문자열 바꾸기 – **replace**
    - replace(바뀔_문자열, 바꿀_문자열)

```
>>> a = "Life is too short"
>>> a.replace("Life", "Your leg")
'Your leg is too short'
```

  - 문자열 나누기– **split**
    - 공백 또는 특정 문자열을 구분자로 해서 문자열 분리
    - 분리된 문자열은 리스트로 반환됨

```
>>> a = "Life is too short"
>>> a.split()     ← 공백을 기준으로 문자열 나눔.
['Life', 'is', 'too', 'short']
>>> b = "a:b:c:d"
>>> b.split(':') ← :를 기준으로 문자열 나눔.
['a', 'b', 'c', 'd']
```