

기초 인공지능 프로그래밍

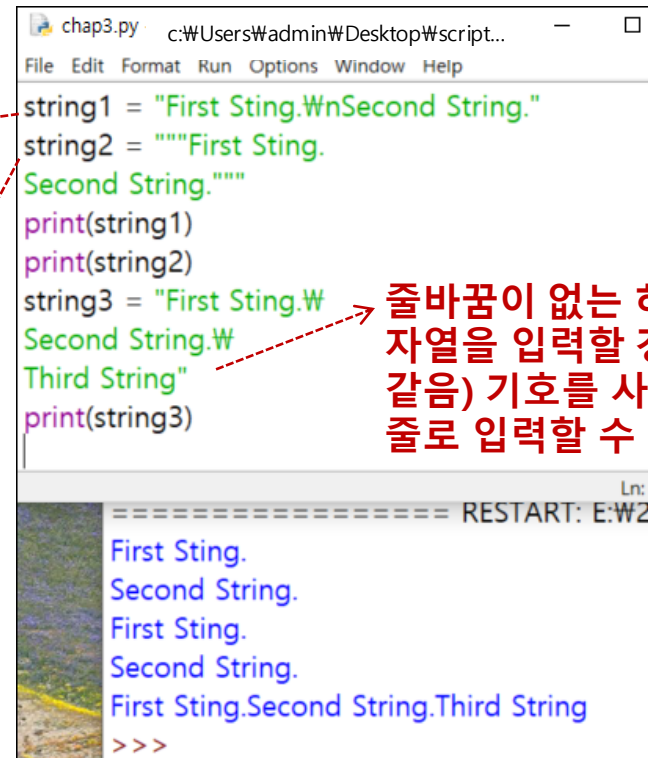
3장 문자열 & 입력/출력 함수

문자열 자료형 (string)

- 문자열은 연속된 문자로 구성된 순서가 있는 자료형
 - 큰따옴표(""), 작은 따옴표(''), 큰따옴표 3개 연속("""), 작은따옴표 3개 연속('') 중에 같은 것으로 양쪽 둘러싸기로 문자열을 정의함 ("String", 'String', """String""", '''String''': 모두 같은 문자열)
 - 문자열은 생성된 후에는 해당 문자열의 구성 문자를 바꿀 수 없음(수정할 수 없는 객체를 immutable 객체라고 함)
 - 파이썬은 문자 하나를 저장하는 문자 자료형을 별개로 지원하지 않고 문자열 자료형으로 다룸

문자열 정의에 작은따옴표 또는 큰따옴표 1개 사용.
줄바꿈 문자가 필요한 경우는 문자열 데이터의 원하는 위치에 줄바꿈 문자(\n) 을 삽입

문자열 정의에 연속된 작은따옴표 3개 또는 큰따옴표 3개 사용.
줄바꿈 문자가 필요한 경우는 줄바꿈 문자 삽입 없이 원하는 위치에 엔터키를 입력.



```
chap3.py c:\Users\Wadmin\Desktop\Wscript...
File Edit Format Run Options Window Help
string1 = "First Sting.\nSecond String."
string2 = """First Sting.
Second String."""
print(string1)
print(string2)
string3 = "First Sting.W
Second String.W
Third String"
print(string3)

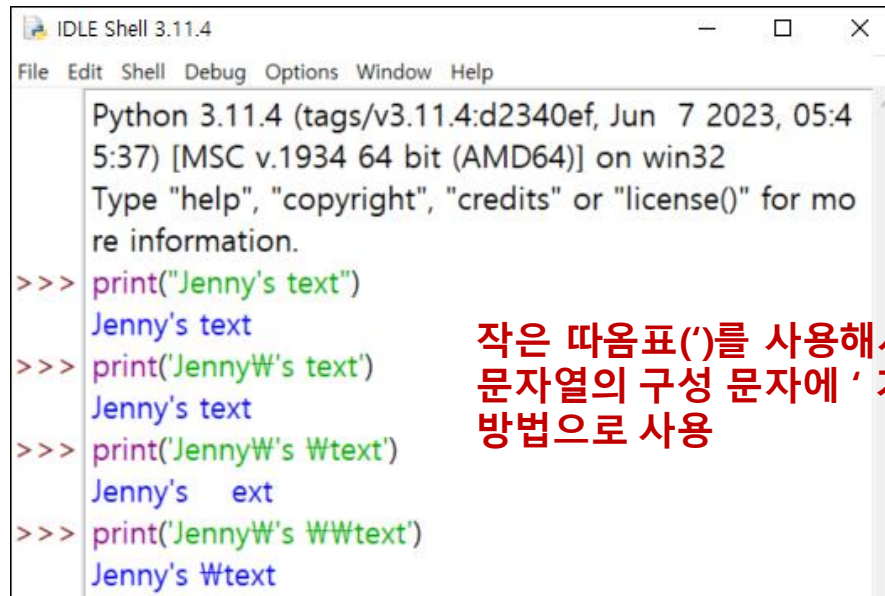
===== RESTART: E:\W2
First Sting.
Second String.
First Sting.
Second String.
First Sting.Second String.Third String
>>>
```

줄바꿈이 없는 하나의 긴 문자열을 입력할 경우는 \ (와와 같음) 기호를 사용하여 여러 줄로 입력할 수 있음.

특수 문자 (Escape Sequence)

- 프로그래밍할 때 사용할 수 있도록 미리 정의해 둔 "문자 조합"
 - 주로 출력물을 보기 좋게 정렬하는 용도로 이용
 - backslash(\)와 단일 문자로 구성된 조합

notation	meaning	notation	meaning
\\	backslash (\)	\n	Newline(줄바꿈)
\'	single quote(')	\t	horizontal tab
\"	double quote(")		

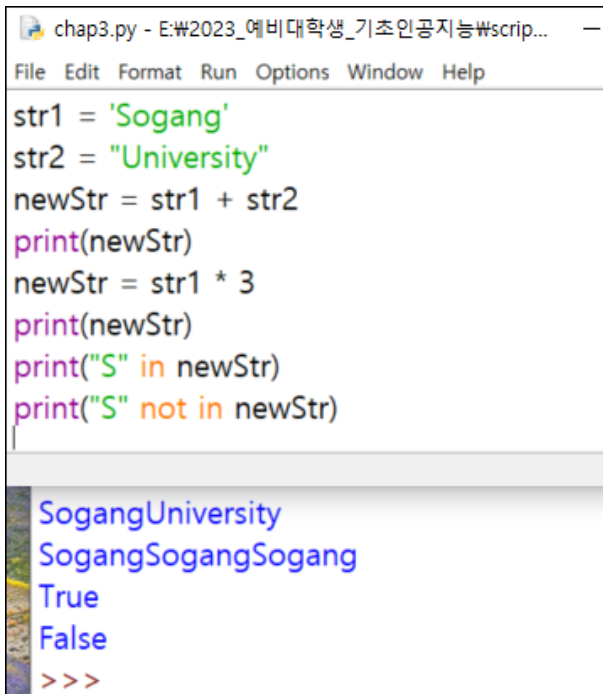


```
IDLE Shell 3.11.4
File Edit Shell Debug Options Window Help
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:4
5:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for mo
re information.
>>> print("Jenny's text")
Jenny's text
>>> print('Jenny's text')
Jenny's text
>>> print('Jenny's Wtext')
Jenny's  ext
>>> print('Jenny's WWtext')
Jenny's Wtext
```

작은 따옴표(')를 사용해서 문자열을 정의할 경우,
문자열의 구성 문자에 ' 가 필요하면 \\' 와 같은
방법으로 사용

문자열 연산

- + 연산자 : 두 문자열을 연결한 새 문자열 생성
- * 연산자 : `str * n` 형태로 문자열 `str`을 `n`번 반복한 새 문자열 생성
- in 연산자 : `str1 in str2` 형태
 - 지정한 문자열(`str1`)이 문자열(`str2`) 안에 존재하는지를 확인
 - 존재하면 `True`, 아니면 `False`를 반환
- not in 연산자 : `in` 연산자와 반대 개념



```
chap3.py - E:\2023_예비대학생_기초인공지능\scrip...
File Edit Format Run Options Window Help

str1 = 'Sogang'
str2 = "University"
newStr = str1 + str2
print(newStr)
newStr = str1 * 3
print(newStr)
print("S" in newStr)
print("S" not in newStr)

SogangUniversity
SogangSogangSogang
True
False
>>>
```

문자열 인덱싱(indexing)

- 인덱스는 문자열의 각 문자마다 번호를 매기는 것

`s1 = "Hi there!"`

0	1	2	3	4	5	6	7	8
H	i		t	h	e	r	e	!

`len(s1) : 9`

`s2 = "안녕?"`

0	1	2
안	녕	?

`len(s2) : 3`

- 인덱스 범위
 - 양수 인덱스 : 0 ~ (len(문자열)-1)
 - `len()` 함수 : 입력 받은 문자열의 길이(문자 개수)를 반환
- 인덱스 표기 방식은 대괄호(`[]`) 안에 인덱스 번호를 기입하여 해당 문자열에서 지정한 문자를 참조하는 것

```
chap3.py - C:\Users\W...
File Edit Format Run Options Window Help
s1 = "Hi there!"
length = len(s1)
c1 = s1[3]
c2 = s1[length - 1]
print(length)
print(c1)
print(c2)

>>>
= RESTART: C:\Users\W...
9
t
!
>>>
```

```
>>> c3 = s1[len(s1)] 인덱스 범위를 벗어나서 에러 발생
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    c3 = s1[len(s1)]
TypeError: 'builtin_function_or_method' object is not subscriptable

>>> s1[1] = "t" 생성한 문자열의 구성 문자를 수정할 수 없음
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    s1[1] = "t"
TypeError: 'str' object does not support item assignment

>>>
```

문자열 슬라이싱(slicing)

- 문자열의 인덱스를 기반으로 문자열의 부분 문자열을 참조하는 방법
- 문자열에서 일부 문자들 추출에 사용
- 문자열 slicing 표기 방식
 - **string[start : end : step]**
 - start : 시작 인덱스 값. default 값은 0
 - end : (end-1) 인덱스까지 추출, default 값은 len(s)
 - step : 추출 간격, default 값은 1

```
chap3.py - C:\Users\admin\Desktop\script\chap3....
File Edit Format Run Options Window Help
s = "My score is 89. That's good!"
score = s[12:14:]
print(score)
s1 = s[:12:]
newS = s1 + str(100) + s[14::]
reverseS = newS[::-1]
print(newS)
print(reverseS)
```

한번 생성된 문자열의 내용은 변경할 수 없으므로
문자열 데이터의 일부분을 변경할 필요가 있는 경우
는 슬라이싱 이용해서 원하는 문자열을 생성

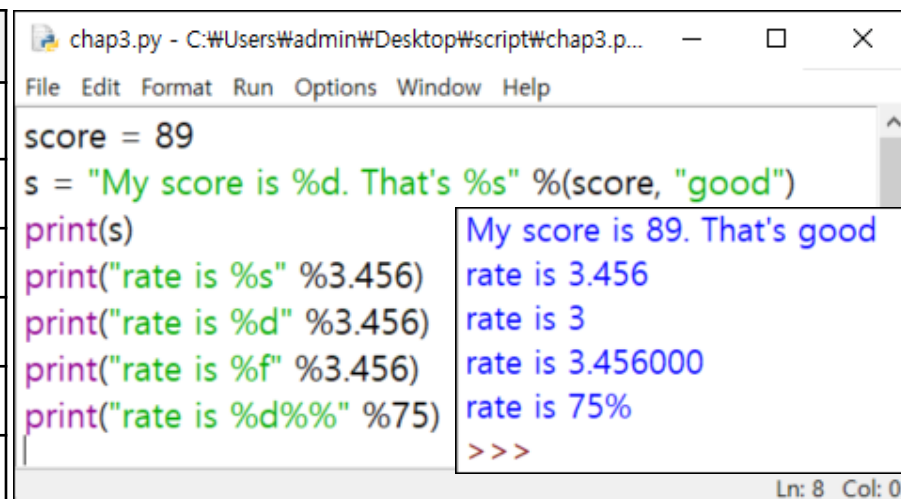
원 문자열을 역순으로
재배열한 새로운 문자열
생성

```
89
My score is 100. That's good!
!doog s'tahT .001 si erocs yM
>>>
```

문자열 포매팅(formatting) - % 사용

- 문자열과 다른 유형의 데이터를 혼합하여 원하는 문자열을 만들 때 사용
 - 여러 형태의 데이터(숫자, 문자열, 변수)를 원하는 위치에 삽입 가능함
 - 값을 삽입하고 싶은 위치에 **%** 기호와 **출력 형식 지정 문자**를 문자열에 추가
 - 문자열 뒤에 **%** 기호와 **삽입할 값**을 지정
- 문자열 포맷 코드(출력 형식 지정 문자)

코드	설명
%s	문자열 (String). 어떤 데이터 형의 값이든 문자열로 변환
%c	문자 1개(character)
%d	정수 (Integer)
%f	부동소수 (floating-point)
%o	8진수
%x	16진수
%%	문자 % 자체



```
chap3.py - C:\Users\wadmin\Desktop\script\chap3.p...
File Edit Format Run Options Window Help

score = 89
s = "My score is %d. That's %s" %(score, "good")
print(s)
print("rate is %s" %3.456)
print("rate is %d" %3.456)
print("rate is %f" %3.456)
print("rate is %d%%" %75)

My score is 89. That's good
rate is 3.456
rate is 3
rate is 3.456000
rate is 75%
>>>
```

3.456은 실수 데이터.
출력 형식 지정 문자(%s, %d, %f)에 따라서
삽입되는 문자열 형태가 다름

문자열 포매팅(formatting) - % 사용

- 특정 서식에 맞춰 숫자를 출력하는 것이 필요할 때 사용
 - 실수 서식 지정

```
print("average = %10.2f " % 57.467657)
print("average = %10.2f" % 12345678.923)
print("average = %10.2f" % 57.4)
print("average = %10.2f" % 57)
```

10

57.47
12345678.92
57.40
57.00

전체 10 자리 중, 소수점 이하 2 자리

```
a = 1/3
print("1/3 =", a, "(too many digits).")
print("a = %.3f" % a)
```

출력

전체 자리 지정은 없이,
소수점 이하 3 자리

1/3 = 0.3333333333333333 (too many digits).
a = 0.333

- 정수 서식 지정

```
print("average = %10d" % 59832)
```

10

59832

숫자 데이터는 오른쪽 정렬이 기본

문자열 포매팅(formatting) - % 사용

- 추가 예시
 - `%nd` : `n` 자리 정수로 출력

<pre>a = -123 print("a = %d" %a) print("a = %4d" %a) print("a = %1d" %a) print("a = %6d" %a)</pre>	<pre># 자릿수 지정 안 함 # 정확한 자릿수 지정(부호 포함) # 자릿수 모자라도 문제 없음 # 자릿수 남으면 빈칸으로 채움(좌측)</pre>
--	--

출력

<pre>a = -123 a = -123 a = -123 a = -123</pre>

문자열 포매팅(formatting) - % 사용

- 추가 예시
 - `%n.mf` : 전체 `n` 자리에서 소수점 이하 `m` 자리로 출력

```
a = -12345.1234567890123456789
print("a = %f" %a)           # 자릿수 지정 안 함
print("a = %.19f" %a)        # 소수점 이하 19자리(오차 보임)
print("a = %.3f" %a)         # 소수점 이하 3자리
print("a = %10.3f" %a)       # 정확한 자릿수 지정
print("a = %5.3f" %a)        # 자릿수 모자라도 문제 없음
print("a = %12.3f" %a)       # 남는 자릿수 빈칸으로 채움(좌측)
```

출력

```
a = -12345.123457
a = -12345.1234567890114703914
a = -12345.123
a = -12345.123
a = -12345.123
a = -12345.123
```

문자열 메소드

- 메소드
 - 파이썬의 모든 데이터는 객체(클래스)이며 객체지향 프로그래밍이 가능함
 - 각 클래스에 정의되어 함수를 메소드(method)라고 하며 문자열의 메소드를 사용하기 위해서는 `str.methodName()`과 같은 형태로 사용해야 함(. 앞에는 문자열 데이터를 갖는 변수나 문자열 상수가 와야 함)
 - 메소드 사용시, 메소드 이름 뒤의 괄호 안에 사용하는 데이터를 인자라고 하며 인자는 메소드에 따라 다양한 개수와 형태를 가짐
 - 객체와 클래스의 자세한 개념은 추후 다른 과정에서 배움

문자열 메소드

- `str.split()` / `str.split('구분자')`
 - `split('구분자')`의 인자로 받은 구분자(문자열 형태)를 기준으로 문자열을 분리하여 분리된 문자열들을 원소로 하는 리스트 자료형을 반환
 - `split()`와 같이 인자를 지정하지 않은 경우는 default 구분자로 공백(space) 문자가 사용됨(`split(" ")`와 `split()`는 다른 기능을 함)
 - 문자열에서 구분자를 기준으로 분리되는 각각은 문자열 자료형임
 - 분리된 문자열은 원하는 자료형으로 변환해서 사용할 수 있음

```
chap3.py - C:\Users\admin\Desktop\script\chap3.p...
File Edit Format Run Options Window Help
data = "David 20 2023"
L = data.split()
print(L, type(L))
L1 = data.split("/")
print(L1, type(L1))
name, age, year = data.split()
print(name, type(name))
print(age, type(age))
print(year, type(year))
age = int(age)
print(age, type(age))
```

구분자를 기준으로 문자열을 분리할 것이 없으면 결과는 단일 문자열을 원소로 하는 리스트를 반환

```
['David', '20', '2023'] <class 'list'>
['David 20 2023'] <class 'list'>
David <class 'str'>
20 <class 'str'>
2023 <class 'str'>
20 <class 'int'>
>>>
```

문자열 메소드

- `str.split()` / `str.split('구분자')`
 - `split()`의 결과를 각 변수에 저장할 경우는 분리되는 개수를 정확하게 파악해서 할당(대입)연산자의 왼쪽에 개수에 맞는 변수들을 지정해야 함
 - 분리된 각각은 문자열 자료형

```
>>> data = "David 20 2023"
>>> name, age = data.split()
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    name, age = data.split()
ValueError: too many values to unpack (expected 2)
>>> name, age, year, data = data.split()
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    name, age, year, data = data.split()
ValueError: not enough values to unpack (expected 4, got 3)
>>> |
```

문자열 메소드

- `str.join(iter)` **반복 가능한 객체**
 - 인자로 지정된 iterable 객체(리스트, 튜플, 집합, 문자열)의 원소들이 문자열일 때, 문자열 `str`을 각 원소들 사이에 삽입해서 새로운 문자열 한개를 생성 반환

```
chap3.py - C:\Users\Wad...
File Edit Format Run Options Window Help
s="ABCDEFGF"
print(", ".join(s)) 문자열 s의 원소인 문자(문자열) 사이에 comma 삽입해서 새로운 문자열 생성
print(" ".join(s)) 문자열 s의 원소인 문자(문자열) 사이에 space 삽입해서 새로운 문자열 생성
print("_".join(s)) 문자열 s의 원소인 문자(문자열) 사이에 underbar 삽입해서 새로운 문자열 생성
L = ["David", "Lyn", "Ann"]
sep = ","
s1 = sep.join(L)
print(s1, type(s1))
Ln: 9 Col: 0
```

```
A,B,C,D,E,F,G
A B C D E F G
A_B_C_D_E_F_G
David,Lyn,Ann <class 'str'>
>>>
```

문자열 메소드

- `str.join(iter)`
 - iterable 객체(리스트, 튜플, 집합)의 원소들로 원하는 형태의 문자열을 생성할 필요가 있을 때 사용

```
>>> L = ["a", 10, "b", "c"]
>>> s = "".join(L)  리스트 L의 원소 중 문자열이 아닌 원소가 있기 때문에 에러 발생
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    s = "".join(L)
TypeError: sequence item 1: expected str instance, int found
>>>
```

```
>>> L = ["A", "b", "C", "d"]
>>> s = "".join(L)  리스트 L의 원소 사이에 어떤 문자도 삽입하지 않고 새로운 문자열 생성
>>> print(s)
AbCd
>>>
```

문자열 메소드

- `str.find()`, `str.rfind()`
 - **`str.find(subStr)`**: 문자열 `subStr`이 발견되는 첫번째 인덱스를 반환, 존재하지 않으면 `-1`을 반환.
 - **`str.rfind(subStr)`**: 문자열 `subStr`이 발견되는 가장 큰 인덱스를 반환, 존재하지 않으면 `-1`을 반환.

```
chap3.py - C:\Users\Wadmin\Desktop\Wsc...
File Edit Format Run Options Window Help
s = "it is good. it is bad. it is poor"
find = "it"
idx1 = s.find(find)
print(idx1)
idx2 = s.find(find, idx1+len(find))
print(idx2)
idx3 = s.find(find, idx2+len(find))
print(idx3)
idx = s.rfind(find)
print(idx)
idx = s.find("Good")
print(idx)
>>>
```

0
12
23
23
-1
>>>

```
>>> idx = s.find("it", 15, 25)
>>> print(idx)
23
>>> idx = s.rfind("it", 0, 15)
>>> print(idx)
12
>>>
```

Ln: 4 Col: 0

문자열 메소드

메소드	설명
lower()	문자열 데이터를 모두 소문자로 바꾼 문자열 을 반환
upper()	문자열 데이터를 모두 대문자로 바꾼 문자열 을 반환
islower()	문자열 내의 모든 문자들이 소문자이면, True 를 반환
isupper()	문자열 내의 모든 문자들이 대문자이면, True 를 반환
isalpha()	문자열이 알파벳(영문, 한글등)으로만 구성되어 있으면 True 를 반환
isnumeric() isdigit()	문자열이 숫자로만 구성되어 있으면 True 반환
isalnum()	문자열이 알파벳과 숫자로만 구성되어 있으면 True 반환
swapcase()	문자열 데이터의 대소문자를 상호 변환한 문자열 반환
title()	각 단어의 제일 앞 글자만 대문자로 변환한 문자열 반환

문자열 메소드

메소드	설명
replace()	문자열 내에서 지정한 문자열을 새로운 문자열로 바꾼 전체 문자열을 반환
startswith()	문자열이 매개변수로 입력한 문자열로 시작하면 True 반환
endswith()	문자열이 매개변수로 입력한 문자열로 끝나면 True 반환
find()	문자열 내에서 매개변수로 입력한 문자열이 시작하는 인덱스를 반환 (존재하지 않으면 -1을 반환). 여러 번 존재하면 첫번째 발견하는 인덱스 반환
rfind()	문자열 내에서 매개변수로 입력한 문자열을 뒤에서부터 찾아서 인덱스 반환 (존재하지 않으면 -1을 반환)
count()	문자열 내에서 매개변수로 입력한 문자열이 몇 번 있는지 그 개수를 반환
strip()	문자열 양쪽에 있는 공백을 제거한 문자열 반환 (lstrip() / rstrip(): 왼쪽/오른쪽에서 공백제거)
center(width)	주어진 폭의 가운데 중심으로 정렬된 문자열을 반환 (ljust(width) / rjust(width): 왼쪽/오른쪽 중심으로 정렬)

문자열 메소드 사용 예시

```
chap3.py - C:\Users\admin\Desktop\script\chap3.py (3.9.2)
File Edit Format Run Options Window Help
s1 = "aba"; s2 = "Aba!"; s3 = "123abababa"; s4 = " "; s5 = "123"
print(s1.islower())    # 소문자로만 구성?
print(s2.isalpha())    # 영문자로만 구성?
print(s3.isalnum())    # 영문자와 숫자만으로 구성?
print(s4.isspace())    # 빈칸으로만 구성?
print(s5.isnumeric())  # 숫자(0~9)로만 구성?
print(s3.count(s1))    # s3에서 s1의 반복 횟수
print(s3.find(s1))     # s3에 s1이 존재하는 경우, 첫번째로 발견되는 인덱스
s = s1.upper()         # s1의 문자들을 대문자로 바꾼 문자열 반환, s1은 바뀌지 않음
print(s)
s = s2.lower()         # s2의 문자들을 소문자로 바꾼 문자열 반환, s2은 바뀌지 않음
print(s)
s = "Hi everybody!"
s1 = s.replace("Hi", "Hello") # Hi를 Hello로 바꾼 문자열 반환
s2 = s.replace("!", "")      # !를 제거한 문자열 반환
print(s1)
print(s2)
```

```
True
False
True
True
True
2
3
ABA
aba!
Hello everybody!
Hi everybody
>>>
```

Ln: 18 Col: 0

입력 함수 input()

- 키보드로부터 문자열을 입력 받는 함수

variable_name = input("Prompt string")

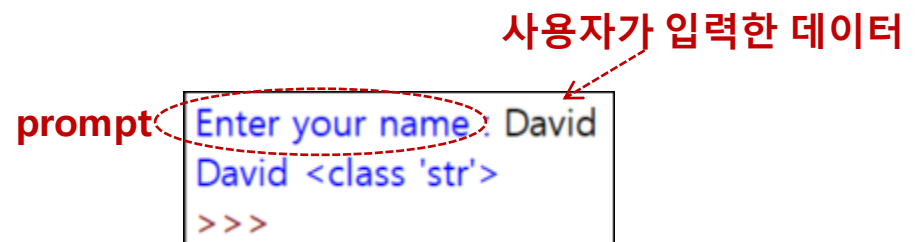
- ① input() 함수의 인자로 주어진 문자열(prompt)이 출력됨
- ② 그 문자열 뒤에 커서가 깜박이며 사용자가 데이터를 입력하길 기다림
- ③ 사용자가 키보드로 데이터를 입력하고 Enter 키를 누르면
- ④ 입력한 데이터들이 문자열 형태로 input() 함수의 결과값(반환 값)이 됨
- ⑤ 반환된 값은 variable_name에 할당되어 사용함



```
chap3.py - C:\Users\Wadmin\Des...  
File Edit Format Run Options Window Help  
name = input("Enter your name : ")  
print(name, type(name))  
Ln: 3 Col: 0
```

사용자가 입력한 데이터

prompt



```
Enter your name : David  
David <class 'str'>  
>>>
```

input() 함수의 반환값은 문자열 자료형!!!

문자열 형태가 아닌 다른 자료형의 데이터가 필요하면 입력 받은 문자열을 다른 자료형으로 변환하여야 함

입력 함수 input()

- 여러 개의 데이터들을 한번에 입력 받는 경우
 - input() 함수로 입력 받은 문자열을 문자열 메소드 split()을 사용하여 분리함
 - 입력 받는 데이터의 개수를 정확히 알고 있는 경우에는 아래와 같은 방법으로 처리함 그렇지 못한 경우는 리스트 데이터형으로 해결 가능(추후에 설명)

입력 받는 데이터
개수가 두개인 경우,
할당 연산자 왼쪽에
변수 2개

```
chap3.py - C:\Users\wadmin\Desktop\scr...
File Edit Format Run Options Window Help

s = input("Enter your name and score : ")
print(s, type(s))
name, score = s.split()
print(name, type(name))
print(score, type(score))
```

Enter your name and score : Lee 89
Lee 89 <class 'str'>
Lee <class 'str'>
89 <class 'str'>

```
chap3.py - C:\Users\wadmin\Desktop\script\chap3.py (3.9.2)
File Edit Format Run Options Window Help

name, score, avg = input("Enter your name, score and average : ").split()
score = int(score)
avg = float(avg)
print(name, type(name))
print(score, type(score))
print(avg, type(avg))
```

Enter your name, score and average : Lee 89 61.2
Lee <class 'str'>
89 <class 'int'>
61.2 <class 'float'>

입력 함수 input()

- input() 함수 실행 예시

```
>>> name = input("What is your name? ")
What is your name? ParkJiWoo
>>> age = input("How old are you? ")
How old are you? 22
>>> print("Your name is %s. After 10 years, your age is %d." %(name, int(age) + 10))
Your name is ParkJiWoo. After 10 years, your age is 32.
>>>
```

input() 함수로 입력 받은 데이터는 문자열이기 때문에 정수 연산을 위해서 int 자료형으로 변환



출력 함수 print()

- 모니터 화면에 결과를 출력하는 함수
 - print() 함수는 인자를 모두 문자열로 바꾸어 출력함
 - 인자가 여러 개인 경우는 인자와 인자 사이에 디폴트로 빈칸이 추가되며 마지막에 줄바꿈 문자도 디폴트로 추가되어 출력시 줄바꿈이 발생.
 - 콤마로 구분되는 인자를 출력시 추가 출력되는 빈칸을 다른 것으로 변경하려면, print() 함수의 **sep 인자** 설정을 변경하면 됨
 - 또한 print() 함수 실행 후 줄바꿈이 필요 없는 경우는 **end 인자** 설정을 변경하면 됨

```
chap3.py - C:\Users\admin\Desktop\Ws...
File Edit Format Run Options Window Help
print("Sogang", "Uniiversity")
print("Sogang", "Uniiversity", sep="***")
print("Sogang", "Uniiversity", sep="")
print("First printing.", end="_")
print("Second printing.", end="")
print("Last printing.")
print("End.")
```

인자 간에 다른 문자 추가 없이 출력

```
Sogang Uniiversity
Sogang***Uniiversity
SogangUniiversity
First printing._Second printing.Last printing.
End.
>>>
```

줄바꿈뿐만 아니라, 출력 문자열의 마지막에 어떤 문자도 추가하지 않을 때