

2. 문자열과 출력

목차

- 문자열
- 출력
- INDEX
- 주식

문자열

문자열 : 문자들을 차례대로 나열하여 붙인 데이터

- 쉽게 말해서, 우리가 쓰는 모든 글자, 문장들! 예를 들어, "hello, erica!"

파이썬에서 문자열은 “내용” 또는 ‘내용’으로 나타냅니다

#파이썬 인터프리터 (IDLE 쉘) 실행결과!

```
>>> "String"  
'String'  
>>> 'Also, string'  
'Also, string'
```

문자열

문자열 연산

더하기 (+) : 두 문자열을 붙일 수 있어요!

```
>>> "string. " + 'Also, string'  
'string. Also, string'
```

곱하기 (*) : 한 문자열을 여러 번 반복해서 붙일 수 있어요!

```
>>> "string" * 3  
'stringstringstring'
```

출력

Q. 출력이라면, 아까 IDLE 쉘에서 파란 글씨로 나온 게 출력된 거 맞나요?

A. 음, 쉘에서는 출력된 게 맞지만, 코드 편집기에서도 출력이 되는지 확인해봅시다!

IDLE 코드 편집기 실행결과!



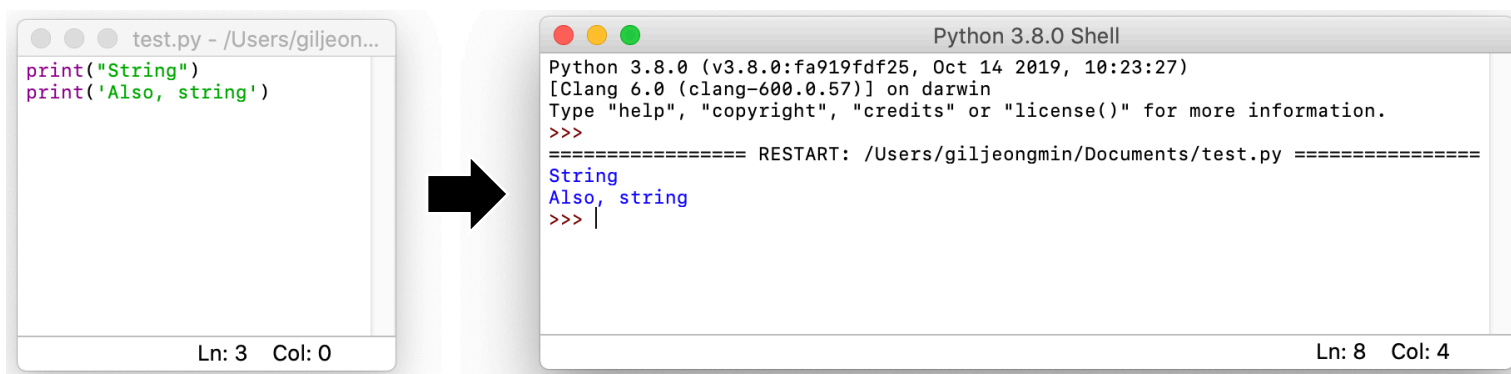
보시다시피, 아까와 같은 코드를 적었음에도 그 무엇도 출력되지 않아요!

대화형 쉘에서는 print라는 함수 없이도 출력을 해주지만, 편집기는 그렇지 않기 때문이에요.

그러므로, 이제 print()에 대해 알아보시다!

출력

print(문자열) : 문자열을 출력합니다!

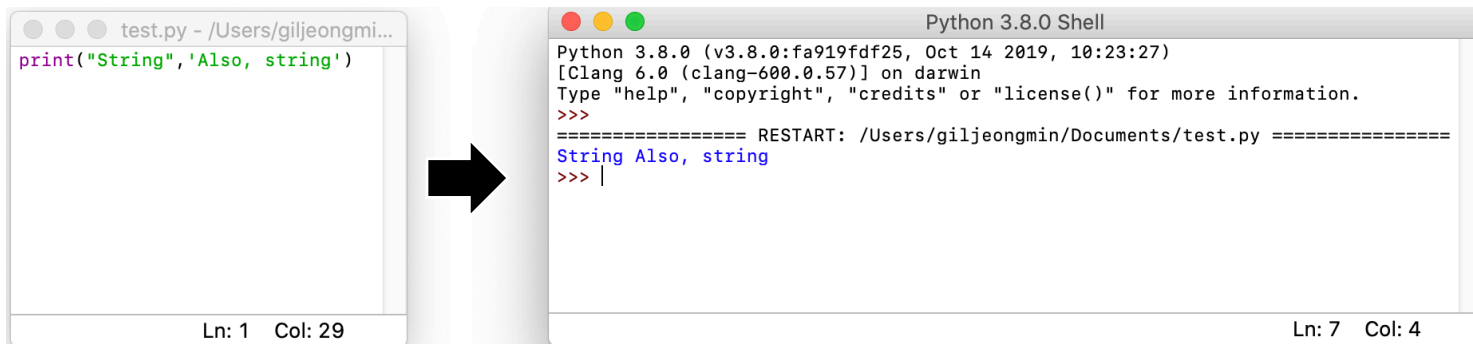


```
test.py - /Users/giljeon...
print("String")
print('Also, string')
Ln: 3 Col: 0
```

Python 3.8.0 Shell

```
Python 3.8.0 (v3.8.0:fa919fdf25, Oct 14 2019, 10:23:27)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /Users/giljeongmin/Documents/test.py =====
String
Also, string
>>> |
Ln: 8 Col: 4
```

문자열을 콤마로 연결해서 여러 개 넣어도 괜찮아요!



```
test.py - /Users/giljeongmi...
print("String", 'Also, string')
Ln: 1 Col: 29
```

Python 3.8.0 Shell

```
Python 3.8.0 (v3.8.0:fa919fdf25, Oct 14 2019, 10:23:27)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /Users/giljeongmin/Documents/test.py =====
String Also, string
>>> |
Ln: 7 Col: 4
```

출력

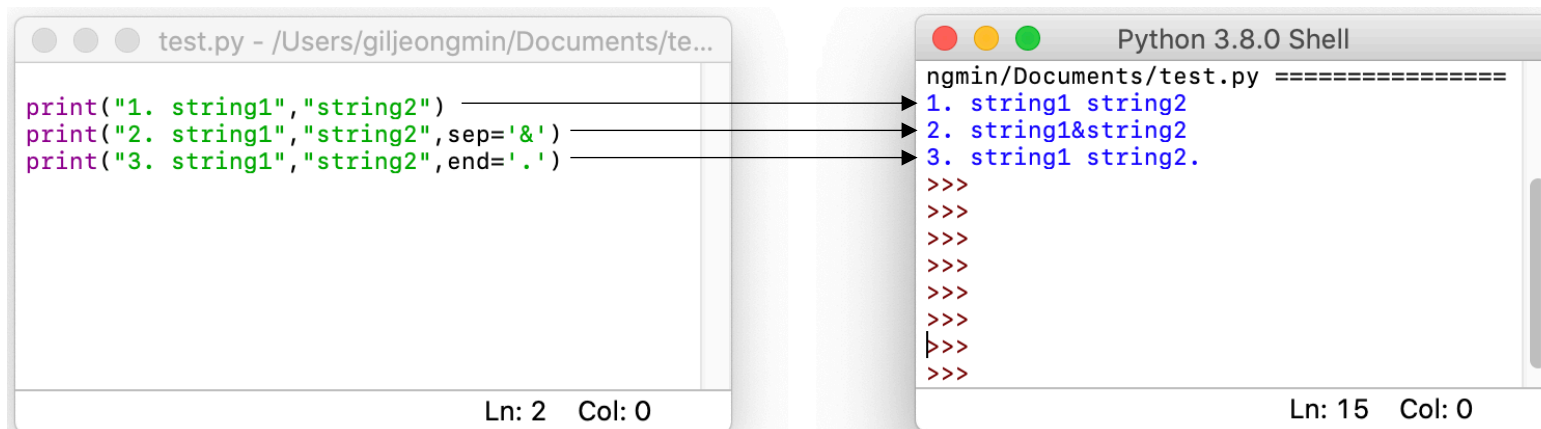
Q. 어라.. 그런데 출력 형태가 조금 다른데요..??
위에는 분명 2줄인데, 아래는 1줄이에요!!!

A. 음, 여러가지 방법이 있지만 먼저 print()의 '옵션' 두 가지를 알려드릴게요!

옵션이란? : 간단히 말해서, 부가 기능. 예시 : print(문자열, 옵션1, 옵션2, ...)

1. sep = 문자열 : 콤마(,)를 만날 때 출력 할 문자열을 지정한다. 기본값은 ' ' # 공백 문자예요! (space bar)

2. end = 문자열 : print문이 끝날 때 출력 할 문자열을 지정한다. 기본값은 '\n' # 개행 문자예요!

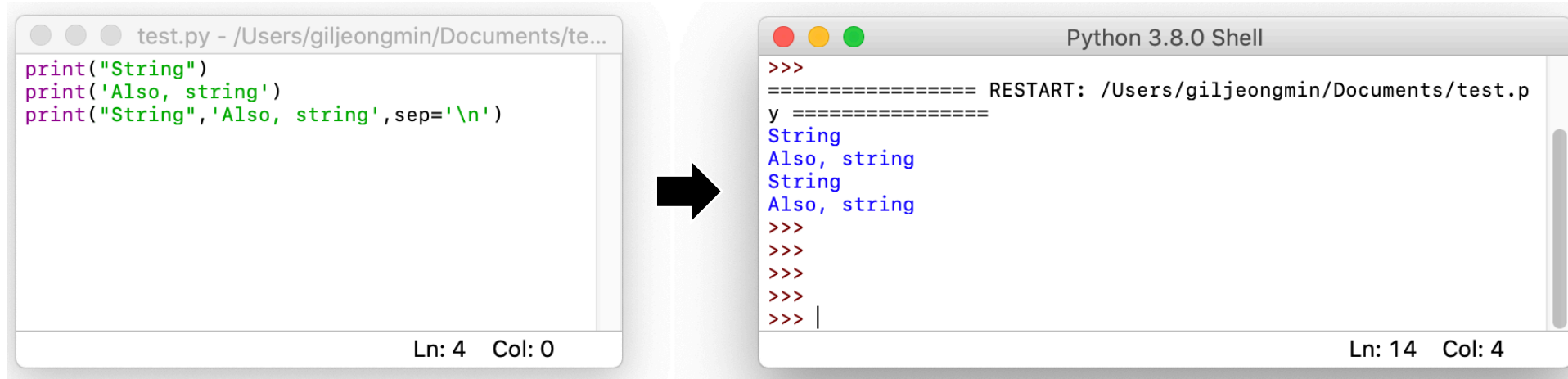


```
test.py - /Users/giljeongmin/Documents/te...
print("1. string1", "string2")
print("2. string1", "string2", sep='&')
print("3. string1", "string2", end='.')
```

```
Python 3.8.0 Shell
ngmin/Documents/test.py =====
1. string1 string2
2. string1&string2
3. string1 string2.
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
```

출력

즉, sep 옵션을 기본값인 ' '가 아닌 개행 문자 '\n'로 바꾸면 됩니다.



```
test.py - /Users/giljeongmin/Documents/te...
print("String")
print('Also, string')
print("String", 'Also, string', sep='\n')
Ln: 4 Col: 0
```

Python 3.8.0 Shell

```
>>>
===== RESTART: /Users/giljeongmin/Documents/test.p
y =====
String
Also, string
String
Also, string
>>>
>>>
>>>
>>>
>>>
>>> |
Ln: 14 Col: 4
```

Q. 개행 문자..? '\n' ...??? 그게 뭐예요?

이스케이프 문자

이스케이프 문자('\'')

- 다음 문자가 특수 문자임을 알리는 문자예요!

개행 문자('\n') : 줄바꿈 문자, 다음 줄로 내려준다.

```
>>> print("string\nstring")
string
string
```

단일 인용 부호('\''') : “”(‘ ’)로 구분된 문자열에서, “”(‘ ’)를 문자로 사용할 수 있도록 해준다.

```
>>> print("\\"다음표!\")
"다음표!"
```

백슬래시('\\') : 백슬래시를 문자로 사용할 수 있도록 해준다. # 특수 문자는 이 밖에도 많습니다

```
>>> print("\\n은 개행문자입니다.")
\n은 개행문자입니다.
```

INDEX

string index (문자열 위치번호)

문자열의 각 문자는 차례로 인덱스가 매겨집니다.

왼쪽에서부터는 0에서부터 늘어나고, 오른쪽에서부터는 -1에서부터 줄어듭니다.

'ERICA'라는 문자열이 있을 때,

0	-5	1	-4	2	-3	3	-2	4	-1
E		R		I		C		A	

형식은 문자열[인덱스] 입니다.

```
>>> print("ERICA")
ERICA
>>> print("ERICA"[0] + "ERICA"[1] + "ERICA"[2] + "ERICA"[3] + "ERICA"[4])
ERICA
>>> print("ERICA"[-5] + "ERICA"[-4] + "ERICA"[-3] + "ERICA"[-2] + "ERICA"[-1])
ERICA
```

INDEX

Q. 위치 번호로 하나의 문자가 아닌 특정 범위의 문자열 또한 가져올 수 있나요?

A. 네 가능합니다. 파이썬에서는 **문자열 슬라이싱**이라는 기능을 제공합니다.

형식은 **문자열[시작 인덱스 : 끝 인덱스]** 입니다.

자르는 범위는 시작 인덱스에서부터, 끝 인덱스-1 까지 자릅니다. # "ERICA"[2:5] 면, "ICA"

만약 시작 인덱스가 비어 있으면 0부터, 끝 인덱스가 비어 있으면 끝까지 잘라옵니다.

```
>>> print("ERICA"[1:3])
RI
>>> print("ERICA"[:3])
ERI
>>> print("ERICA"[1:])
RICA
>>> print("ERICA"[:])
ERICA
```

0	-5	1	-4	2	-3	3	-2	4	-1
E		R		I		C		A	

문제! print("ERICA"[-2:]) 의 출력 값은 뭔가요?

주석

주석이란?

- 소스 코드에 영향을 주지 않는, 메모와 같은 기능입니다.
- # 뒤에 오는 모든 문자는 주석 처리가 됩니다.

```
test.py - /Users/giljeongmin/Documents...
#hello, world! 를 출력합니다.
print("hello, world!")

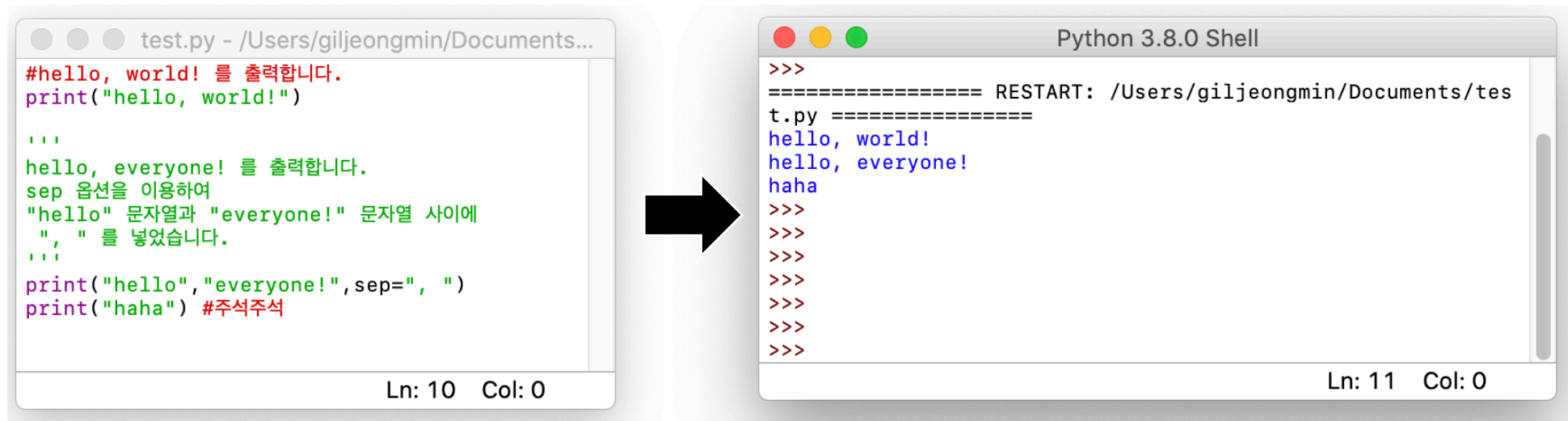
#hello, everyone! 를 출력합니다.
#sep 옵션을 이용하여
#"hello" 문자열과 "everyone!" 문자열 사이에
# ", " 를 넣었습니다.
print("hello", "everyone!", sep=", ")
print("haha") #주석주석

Ln: 9 Col: 19
```

```
Python 3.8.0 Shell
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /Users/giljeongmin/Documents/test.py =====
hello, world!
hello, everyone!
haha
>>>
>>>

Ln: 10 Col: 0
```

'''주석 내용''' 으로, 여러 줄을 한 번에 주석 처리할 수 있습니다.



```
test.py - /Users/giljeongmin/Documents...
#hello, world! 를 출력합니다.
print("hello, world!")

'''
hello, everyone! 를 출력합니다.
sep 옵션을 이용하여
"hello" 문자열과 "everyone!" 문자열 사이에
", " 를 넣었습니다.
'''
print("hello", "everyone!", sep=", ")
print("haha") #주석주석

Ln: 10 Col: 0
```

```
Python 3.8.0 Shell
>>>
===== RESTART: /Users/giljeongmin/Documents/tes
t.py =====
hello, world!
hello, everyone!
haha
>>>
>>>
>>>
>>>
>>>
>>>
>>>

Ln: 11 Col: 0
```

Q. 주석이 왜 필요한가요?

A. 코딩을 하다 보면, 자신의 코드를 자신은 물론이고, 팀플, 실무 등에서 남들이 볼 때가 많아요.

이 때, 제대로 된 주석이 있는 코드와 없는 코드의 분석 난이도는 비교가 불가능할 정도로 차이가 나요!

```
121 size_t i, o, s, doff;  
122 unsigned char *t;  
123 size_t ts;  
124  
125 if (!n || !buf || !buf_size) return;  
126  
127 /*  
128  * Allocate enough memory for all entries and the number  
129  * of entries.  
130  */  
131 *buf_size = 2 + n->count * 12 + 4;  
132 *buf = exif_mem_alloc (ne->mem, sizeof (char) * *buf_size);  
133 if (!*buf) {  
134     EXIF_LOG_NO_MEMORY(ne->log, "ExifMnoteCanon", *buf_size);  
135     return;  
136 }  
137  
138 /* Save the number of entries */  
139 exif_set_short (*buf, n->order, (ExifShort) n->count);  
140  
141 /* Save each entry */  
142 for (i = 0; i < n->count; i++) {  
143     o = 2 + i * 12;  
144     exif_set_short (*buf + o + 0, n->order, (ExifShort) n->entries[i].tag);  
145     exif_set_short (*buf + o + 2, n->order, (ExifShort) n->entries[i].format);  
146     exif_set_long (*buf + o + 4, n->order,  
147                  n->entries[i].components);  
148     o += 8;  
149     s = exif_format_get_size (n->entries[i].format) *  
150         n->entries[i].components;  
151     if (s > 65536) {  
152         /* Corrupt data: EXIF data size is limited to the  
153          * maximum size of a JPEG segment (64 kb).  
154          */  
155         continue;  
156     }  
157     if (s > 4) {  
158         ts = *buf_size + s;  
159  
160         /* Ensure even offsets. Set padding bytes to 0. */  
161         if (s & 1) ts += 1;
```

왼쪽 사진은 삼성 카메라에 실제로 쓰이는 코드 중 일부입니다.

그 중 빨간색 괄호는 전부 주석이에요!

주석을 통해 소스 코드를 전부 읽지 않아도 의도를 파악할 수 있습니다.

이렇게 거창한 코드가 아니더라도 주석을 작성 해두시면

나중에 자신의 코드를 다시 볼 때 이해하기 쉬우므로

주석을 작성하는 습관을 가지시는 것이 좋습니다!

삼성 카메라 오픈소스

<https://opensource.samsung.com>

- 끝 -