Print, 출력문

파이썬에서 출력은 내장함수 `print()`를 사용합니다.

함수에 대한 개념은 추후 다루도록 하고,

먼저 파이썬이 `Hello, Python!`이라는 언어를 출력할 수 있게 해보겠습니다.

```
print('Hello, Python!')
```

```
> Hello, Python!
```

C, Java와 같은 언어에서는 `'(작은 따옴표)`와 `"(큰 따옴표)`를 구별합니다.

작은 따옴표는 (단일)문자(=Char형식)을 큰 따옴표는 문자열을 나타내는데 사용됩니다.

그러나, 파이썬에서는 문자열을 출력할 때 작은 따옴표나 큰 따옴표 다 사용할 수 있습니다.

우리는 파이썬을 배울 것이기에 열고 닫는 짝만 맞춰주는 것을 제외하곤, 신경쓰지 않아도 됩니다.

```
print("Hello, Python")
```

```
> Hello, Python
```

위와 같이 `'`을 사용한 방법과 `"`의 결과를 보면 동일합니다.

그렇다면 바깥을 `'`로 감싸고 `"`을 안에 보면하면 어떻게 되는지 알아보겠습니다.

```
print('"Hello, Python!"')
```

```
> "Hello, Python!"
```

반대로 감싸면 당연히 반대의 경우가 나옴을 알 수 있습니다.

```
print("'Hello, Python!'")
```

```
> 'Hello, Python!'
```

`,`를 이용하여 print문을 사용할 수도 있습니다.

```
print('Hello,', 'Python!')
```

```
> Hello, Python!
```

`,`로 구분해주면 자동으로 한 칸 띄어쓰기가 되어 뒤에 값이 연결됨을 볼 수 있습니다.

그러면 이제 아래와 같은 문구를 출력해보도록 하겠습니다.

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

```
print("The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!")
```

```
File "<ipython-input-5-a0384dd3f4b7>", line 1
    print("The Zen of Python, by Tim Peters
                                           ^
SyntaxError: EOL while scanning string literal
```

위의 결과를 보면 Error가 발생하였음을 볼 수 있습니다.

```
SyntaxError: EOL while scanning string literal
```

문자열(string)을 검사하면서 문법적인 에러가 발생했음을 나타내고 있습니다.

그렇다면 어떻게 써야할까요?

지금 배운 내용으로만 출력해보록 하겠습니다.

```
print("The Zen of Python, by Tim Peters")
print()
print("Beautiful is better than ugly.")
print("Explicit is better than implicit.")
print("Simple is better than complex.")
print("Complex is better than complicated.")
print("Flat is better than nested.")
print("Sparse is better than dense.")
print("Readability counts.")
print("Special cases aren't special enough to break the rules.")
print("Although practicality beats purity.")
print("Errors should never pass silently.")
print("Unless explicitly silenced.")
print("In the face of ambiguity, refuse the temptation to guess.")
print("There should be one-- and preferably only one --obvious way to do it.")
print("Although that way may not be obvious at first unless you're Dutch.")
print("Now is better than never.")
print("Although never is often better than *right* now.")
print("If the implementation is hard to explain, it's a bad idea.")
print("If the implementation is easy to explain, it may be a good idea.")
print("Namespaces are one honking great idea -- let's do more of those!")
```

```
> The Zen of Python, by Tim Peters
>
> Beautiful is better than ugly.
> Explicit is better than implicit.
> Simple is better than complex.
> Complex is better than complicated.
> Flat is better than nested.
> Sparse is better than dense.
> Readability counts.
> Special cases aren't special enough to break the rules.
> Although practicality beats purity.
> Errors should never pass silently.
> Unless explicitly silenced.
> In the face of ambiguity, refuse the temptation to guess.
> There should be one-- and preferably only one --obvious way to do it.
```

```
> Although that way may not be obvious at first unless you're Dutch.
> Now is better than never.
> Although never is often better than *right* now.
> If the implementation is hard to explain, it's a bad idea.
> If the implementation is easy to explain, it may be a good idea.
> Namespaces are one honking great idea -- let's do more of those!
```

위의 결과로 보았을 때,

`print()`만을 썼을 때는 한 번 엔터를 친것과 동일하다는 것을 알 수 있습니다.

또한, 한줄 한줄씩 출력되고 있습니다.

확실한건, 이렇게 하나씩 하기에는 너무도 불편합니다.

한번에 긴 문장을 작성하는 방법에 대해서 알아보겠습니다.

```
print("""The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!""")
```

```
> The Zen of Python, by Tim Peters
>
> Beautiful is better than ugly.
> Explicit is better than implicit.
> Simple is better than complex.
> Complex is better than complicated.
> Flat is better than nested.
> Sparse is better than dense.
> Readability counts.
> Special cases aren't special enough to break the rules.
> Although practicality beats purity.
> Errors should never pass silently.
> Unless explicitly silenced.
```

```
> In the face of ambiguity, refuse the temptation to guess.
> There should be one-- and preferably only one --obvious way to do it.
> Although that way may not be obvious at first unless you're Dutch.
> Now is better than never.
> Although never is often better than *right* now.
> If the implementation is hard to explain, it's a bad idea.
> If the implementation is easy to explain, it may be a good idea.
> Namespaces are one honking great idea -- let's do more of those!
```

`"""문자열 구문"""`으로 작성하게 된다면 띄어쓰기와 줄바꿈이 해결되어 문자열 그대로 작성됨을 확인할 수 있습니다.

`"`을 문자열 좌/우로 3번씩 사용할 수도 있고, `'`를 좌/우로 3번 사용할수도 있습니다.

간혹, 프로그래밍을 하다 보면 띄어쓰지 않고 연이어 써야할 때도 있습니다.

이는 print함수 안에 있는 `end`라는 파라미터를 통하여 할 수 있습니다. (파라미터의 개념은 함수에서 다루도록 하겠습니다.)

```
print('안녕', end='')
print('하세요!')
```

```
> 안녕하세요!
```

end를 사용하지 않은 출력문구들은 자동으로 줄바꿈이 되었는데, `end=''`으로 하니 줄바꿈이 되지 않았습니다.

여기서 `end`는 앞에 작성한 문자열이 끝나고 다음으로 이뤄지는 출력입니다.

`end`를 작성하지 않을 경우에는 초기값(default)이 `\n`(줄바꿈)으로 처리되어 있습니다.

(\n과 같은 표현을 이스케이프 문자(Escape sequence)라고 합니다. 이는 **str** 타입을 다룰 때 같이 다루도록 하겠습니다.)

Remark, 주석

코드를 작성하는 중에 코드에 영향을 주지 않고 설명을 쓰고 싶은 경우가 있습니다.

이러한 경우에 주석을 사용합니다.

주석은 한 줄 주석과 여러줄 주석이 있습니다.

한줄 주석 : # 내용

여러줄 주석 : """내용"""

(큰 따옴표가 아닌 작은 따옴표로 할수도 있습니다.)

```
# 주석 사용법 배우기
"""
여러줄 주석은 이와 같이 사용할 수 있습니다.
이렇게 하면 코드에 영향을 주지 않습니다.
다만, 주석도 byte로 용량에는 영향이 주어지고
너무 방대한 경우에 속도에 영향이 다소 있습니다.
"""
print("주석") #이렇게도 주석을 쓸수 있습니다.
```

```
> 주석
```

```
print("주석 #이렇게는 주석을 사용할 수 없습니다.")
```

```
> 주석 #이렇게는 주석을 사용할 수 없습니다.
```