

About Python underscore(_)

Usage of underscore(_) 5 cases

- 인터프리터(Interpreter)에서 마지막 값을 저장할 때
- 값을 무시하고 싶을 때 (흔히 'I don't care'이라 부른다.)
- 변수나 함수명에 특별한 의미 또는 기능을 부여하고자 할 때
- 국제화(Internationalization, i19n) / 지역화(Localization, l10n) 함수로써 사용할 때
- 숫자 리터럴값의 자릿수 구분을 위한 구분자로써 사용할 때

1. 인터프리터에서 사용되는 경우

파이썬 인터프리터에선 마지막으로 실행된 결과값이 _라는 변수에 저장된다. 이는 표준 CPython 인터프리터에서 먼저 사용되었으며 다른 파이썬 인터프리터 구현체에서도 똑같이 사용할 수 있다.

```
>>> 10
10
>>> _
10
>>> _ * 3
30
>>> _ * 20
600
```

2. 값을 무시하고 싶은 경우

_는 또한 어떤 특정값을 무시하기 위한 용도로 사용되기도 한다. 값이 필요하지 않거나 사용되지 않는 값을 _에 할당하기만 하면 된다.

```
# 언패킹시 특정값을 무시
```

```
x, _, y = (1, 2, 3) # x = 1, y = 3
```

```
# 여러개의 값 무시
```

```
x, *_ , y = (1, 2, 3, 4, 5) # x = 1, y = 5
```

```
# 인덱스 무시
```

```
for _ in range(10):  
    do_something()
```

```
# 특정 위치의 값 무시
```

```
for _, val in list_of_tuple:  
    do_something()
```

3. 특별한 의미의 네이밍을 하는 경우

파이썬에서 _가 가장 많이 사용되는 곳은 아마 네이밍일 것이다. 파이썬 컨벤션 가이드라인인 PEP8에는 다음과 같은 4가지의 언더스코어를 활용한 네이밍 컨벤션을 소개하고 있다.

코딩 컨벤션(Coding Convention)이란 읽고 관리하기 쉬운 코드를 작성하기 위한 일종의 코딩 스타일 약속이다.

(1) `_single_leading_underscore`

- 주로 한 모듈 내부에서만 사용하는 private 클래스/함수/변수/메서드를 선언할 때 사용하는 컨벤션이다.
- 이 컨벤션으로 선언하게 되면 `from module import *` 시 `_`로 시작하는 것들은 모두 import에서 무시된다.
- 그러나 파이썬은 진정한 의미의 private를 지원하고 있지는 않기 때문에 private를 완전히 강제할 수는 없다.
- 즉 import 문에서는 무시되지만 직접 가져다 쓰거나 호출을 할 경우에는 사용이 가능하다.
 - "weak internal use indicator" 라고 부르기도 한다.

```
_internal_name = 'one_module' # private 변수  
_internal_version = '1.0' # private 변수
```

```
def __init__(self, price):  
    self._price = price  
  
def _double_price(self): # private 메서드  
    return self._price * self._hidden_factor  
  
def get_double_price(self):  
    return self._double_price()
```

- single_trailing_underscore_ : 파이썬 키워드와의 충돌을 피하기 위해 사용되는 컨벤션이다. 많이 사용하지 않는다.

```
Tkinter.Toplevel(master, class_ = 'ClassName') # class  
list_ = List.objects.get(1) # list
```

(2) `__double_leading_underscores`

- 이는 컨벤션이라기보단 하나의 문법적인 요소이다.
- 더블 언더스코어(`__`)는 클래스 속성명을 망글링하여 클래스 간 속성명의 충돌을 방지하기 위한 용도로 사용된다.
- 파이썬의 망글링 규칙은 더블 언더스코어로 지정된 속성명 앞에 `_ClassName`을 결합하는 방식이다.
- `ClassName`이라는 클래스에서 `method`라는 메서드를 선언했다면 이는 `_ClassNamemethod`로 망글링 된다.

망글링(Mangling)이란 소스 코드에 선언된 함수 또는 변수의 이름을 컴파일 단계에서 컴파일러가 일정한 규칙을 가지고 변형하는 것이다.


```
class A:
    def _single_method(self):
        pass

    def __double_method(self): # 맹글링을 위한 메서드
        pass

class B(A):
    def __double_method(self): # 맹글링을 위한 메서드
        pass

print(dir(A()))
# ['_A__double_method', ..., '_single_method']

print(dir(B()))
# ['_A__double_method', '_B__double_method', ..., '_single_method']

# 서로 같은 이름의 메서드를 가지지만 오버라이드가 되지 않는다.
```

python dir() 내장함수는 어떤 객체를 인자로 넣어주면 해당 객체가 어떤 변수와 메소드(method)를 가지고 있는지 나열해준다.

Reference

파이썬 언더스코어(_)에 대하여