Indentation

- 4 space per indentation level

Arguments 들여쓰기시 다른 코드와 구별되게 적어야 한다.
Arguments끼리 같은 줄에서부터 시작하도록 들여쓰기 해야한다.
Hanging indents는 4칸일 필요는 없다.

If 문의 조건에서도 앞서 말한 들여쓰기를 해도 되나 Optional이다.(나는 Add some extra indentation on the conditional continuation line이 가독성 좋을 것 같다.)

괄호 열고 닫을 때 요소들의 열거가 끝난 다음 줄에 적는 것이 가독성에 좋다. 나는 이 중에서 요소들의 열거 시작지점에 괄호를 놓는 것이 더 읽기 쉬워보인다.

```
my_list = [
    1, 2, 3,
    4, 5, 6,
    ]
result = some_function_that_takes_arguments(
    'a', 'b', 'c',
    'd', 'e', 'f',
    )
```

Tabs or Spaces?
들여쓰기를 할 때 Tab을 쓰는 것이 편하다. Space를 써도 되지만 Tab과 혼용해서 사용하지마라

Maximum Line Length
한 줄은 79자 이하로 제한해라, Docstrings나 주석은 72자까지 줄여라.
With문의 경우 3.10 버전 이하에서는 백슬레쉬로 이어쓸 수 있었다.
Assert문도 줄바꿈한 뒤에 이어쓸려면 적절한 방법을 써야한다.

Should a Line Break Before or After a Binary Operator?
연산자를 적고 줄바꿈하면 피연산자와 너무 멀리 떨어져서 가독성이 떨어지므로 줄바꿈을 한 뒤에 연산자, 피연산자를 적어라. 그리고 중간에 괄호가 있는 연산은 한 번에 이어서 적어야 가독성이 좋다.

Blank Lines
Top-level function/Class 정의 시 공백 2줄
Class 안의 method는 공백 1줄

관련 있는 함수들끼리 모아놓은 것처럼 보이도록 1줄 추가 또는 없애도 된다.
함수 내에서도 논리적으로 이해되도록 코드 1줄 추가 가능
파이썬에서 ^L은 공백줄로 인식한다. 하지만 일부 에디터나 웹의 코드 뷰어에서는 인식 못할 수도 있다.

Source File Encoding
UTF-8, ASCII, English 사용

Imports
Import할 때 한 번에X, 나눠서 O
From을 통해 같은 파일에서 import할 때는 한 번에 O
코드에서 import의 위치는 Docstrings, comments 뒤, 모듈의 전역 변수와 상수 앞에 있어야한다.
Import시 순서

1. 표준 라이브러리
2. 관련 third party
3. 로컬 애플리케이션과 라이브러리 특정 import

각 import 그룹 줄 사이에는 줄 바꿈이 필수
Absolute import 추천, 에러 메세지 덜 발생
너무 장황하면 Explicit relative imports 사용
Wildcard imports(from <module> import *)는 피해라. 어떤 것을 import 했는지 불분명하게 만든다.

Modul Level Dunder Names
Dunders란 앞 뒤로 __가 있는 것.
(참고)__all__의 경우 from package import * 시에 *에 포함되는 것을 정해줄 수 있다. *하더라도 전체 다 import 되는 것이 아니고 작성자가 공개범위를 정하여 보여줄 수 있는 것이다.
Dunders의 위치는 Docstrings와 comments, __future__ , 뒤 다른 import 앞이다.
(참고) __future__를 통해 python 이전 버전 코드에서도 python 현재 버전처럼 코드를 돌릴 수 있도록 만든 것이다.

String Quotes
', ''는 같은 의미이다. 하나 선택해서 일관성있게 작성해라. 문자열에서 ', ''를 사용하고 싶으면 다른 문자를 써라.

Whitespace in Expressions and Statements
Pet Peeves
띄어쓰기 할 때: 괄호의 가장 자리에는 X, 쉼표와 바로 오는 괄호에 X, 쉼표, 세미콜론, 콜론 바로 앞에 공백X
':' 양 옆에는 동일한 공백 두기, 매개변수가 생략되었으면 공백X

괄호 시작하기 전에 띄어쓰기 x, 할당시 다른 변수들과 굳이 정렬 맞추지 말아라.

Other Recommendations

후행 공백 피하기, =, +=, -=, 비교, Booleans는 양쪽 공백을 사용해라. 우선 순위 연산에는 공백x, 낮은 연산자 양쪽에 공백을 넣어서 가독성을 높이자.

함수 쓰는 형식도 앞서 말한 규칙이 적용되며 -> 양쪽 옆에는 공백을 넣어주자. 또한 arguments의 기본값을 설정할 때는 공백을 넣지X

하지만 type힌트를 미리 적는 경우 = 사이에 공백O

```
# Correct:
def munge(sep: AnyStr = None): ...
def munge(input: AnyStr, sep: AnyStr = None, limit=1000): ...

# Wrong:
def munge(input: AnyStr=None): ...
def munge(input: AnyStr, limit = 1000): ...
```

함수 작성시 굳이 다른 줄을 한 줄에 적지마라.

If/for/while 문의 경우 작은 body면 적을 수 있지만 2줄 이상부터는 절대 한 줄에 적지마라.

When to Use Trailing Commas

대부분 후행 commas는 선택이지만 하나의 요소만 가진 tuple은 필수이다.

버전 관리나 인자의 확장을 할 경우에는 후행 쉼표를 사용하기도 한다. 만약, 줄바꿈을 이용해 요소들을 나열한다면 마지막 줄에도 ,를 추가하지만 같은 줄에 있을 때는 ,를 추가하지 않는다.

```
# Correct:
FILES = [
    'setup.cfg',
    'tox.ini',
    ]
initialize(FILES,
           error=True,
           )

# Wrong:
FILES = ['setup.cfg', 'tox.ini',]
initialize(FILES, error=True,)
```

Comments

코드와 모순되는 주석 적지 않기(주석 매번 업데이트 해야 한다.)

주석은 완전한 문장이어야하고 첫 단어는 대문자지만 변수 이름이 소문자로 시작할 경우 소문자로 시작해라.

블럭 주석은 하나 이상의 문단으로 구성되고 각 문장은 마침표로 끝나야한다.

주석은 영어로 적어야한다.

Block Comments
여러 문장이 있는 주석이고 모두 마침표로 끝나야한다. 마지막 줄이 아닌 이상 마침표 뒤에 한 두 개의 공백을 두는 것이 일반적이다. 모두 #과 띄어쓰기 1개로 구성된다. 주석이 적용되는 코드와 동일한 들여쓰기 수준을 유지. 코드의 목적과 동작을 명확하게 설명하는데 중요하다.

Inline Comments
인라인 주석은 절제해서 사용해라. 문장과 같은 줄에 있는 주석을 의미하며 최소 공백 2칸 이상 놔두라. 당연한 것처럼 보이는 주석은 쓰지마라 산만하다.

Documentation Strings
모든 모듈, 함수, 메서드에 대해 Docstrings를 작성하라. Def 다음에 작성하며 원래는 """ 줄을 마지막에 따로 줄 바꿈해야 하지만 한 줄일 경우 줄 바꿈할 필요없다.

Naming Conventions
Overriding Principle
구현 방식보다 사용 방식을 반영하는 관례를 따라 이름을 지어야한다.

Descriptive: Naming Styles

Public and Internal Interfaces
Public interfaces: 문서화된 인터페이스는 공개된 것으로 간주, 하지만 문서에서 잠정적이거나 internal interfaces라고 선언한 경우 예외다.
Internal Interfaces: 모든 문서화되지 않은 interfaces는 내부로 간주
__all__를 사용해야한다. __all__에는 공개 API에 포함되는 이름을 모두 적어야한다.
__all__을 통해 공개 API를 나눠놨어도 비공개 API는 꼭 접두사에 _를 적어야한다.
패키지, 모듈, 클라스 등을 포함하는 네임스페이스가 internal이면 그 안의 interfaces도 모두 internal이다.

Python Recommendations
None 사용시 = 대신 is, is not 사용해야한다. Not .. is 사용하지 마라 읽기 힘들다. If x랑 if x is not None을 구별해라

__eq__, __ne__ 등 직접 구현하는 것이 좋다. 구현시 functools.total_ordering()을 사용하면 편하다.

Lambda 대신 def를 사용하는 것이 가독성에 좋다.

예외처리를 명확하게 해라. Except시에 자세하게 적어줘야 한다. 너무 포괄적으로 적으면 어떤 error인지 찾아내기 힘들다.

With문을 사용하면 자원은 신속하게 사용하고 정리할 수 있다. With 사용할 때 자원 관리 이상의 작업을 할 때 별도의 함수나 메소드로 구현하는 것이 바람직하다.

함수에서 return을 하도록 설정한 경우 값이 반환되지 않을 때는 return None을 명시적으로 적어줘야한다.

문자열의 접미사, 접두사 확인시 .startswith(), .endswith() 사용하면 오류발생할 확률도 줄고 편하다.

변수형 비교시 isinstance() 사용하면 편하다.

문자열, 배열, 튜플이 비어있으면 False이므로 if not seq: 나 if seq로 확인하면 된다.

값 비교시 boolean이면 ==, !=을 사용할 필요없이 if 문만 사용하면 된다.

Variable Annotations
: 앞에는 뛰어쓰기 x, = 사용시 양쪽에 한 번씩 띄어쓰기