1주차 과제) PEP 8 – Style Guide for Python Code

요약

코드 레이아웃

- 들여쓰기: 4번의 띄어쓰기로 요소 구분

```
def long_function_name(
        var_one, var_two, var_three,
        var_four):
    print(var_one)
```

- if 조건문에서 () 활용해서 조건문 여러 줄에 쓰기

```
if (this_is_one_thing
        and that_is_another_thing):
    do_something()
```

- 여러 줄의 리스트 작성 시, 닫는 괄호는 '바로 전 줄의 첫 번째 글자' 또는 '문장 구조의 첫 번째 문자'에 정렬

```
my_list = [
    1, 2, 3,
    4, 5, 6,
    ]
my_list = [
    1, 2, 3,
    4, 5, 6,
]
```

- 한 줄 최대 길이: 79자, docstring/주석은 72자
  - 괄호로 묶어 줄바꿈(안 되면 백슬래시)
  - 이항연산자 앞에서 줄바꿈
- 공백 라인
  - 최상위 레벨의 함수와 클래스의 def문은 공백 두 줄
  - 클래스 내부 메소드의 def문은 공백 한 줄
  - 가끔은 연관된 함수 구분, 논리적 구분을 위해 공백 사용
- imports
  - 한 줄에 하나씩, 하나의 모듈에서 여러 메소드는 불러오는 건 ㄱㅊ
  - 파일의 가장 위에 작성
  - 정렬: 1. 표준 라이브러리 import문 (공백 한 줄) 2. 연관된 써드파티 import문 (공백 한 줄) 3. 특정 로컬 어플리케이션이나 라이브러리 import문
  - 절대경로 사용 권장. 복잡한 패키지 경로의 경우 상대경로(.) 사용
- Dunder 레벨 모듈의 이름
  - docstring 이후, 다른 import문 이전에 작성 (from__future__ 제외)
    - from__future__는 docstring을 제외하고 제일 이전에 작성

```
"""This is the example module.

This module does stuff.
"""

from __future__ import barry_as_FLUFL

__all__ = ['a', 'b', 'c']
__version__ = '0.1'
__author__ = 'Cardinal Biggles'
```

```
import os
import sys
```

이름 붙이기

- 소스 파일 인코딩
  - UTF-8 사용
  - 모든 식별자는 반드시 ASCII, 영어로만 이름 붙이기
- 네이밍
  - I, l, O 단일 사용 금지 <- 헷갈림
  - 함수 이름, 변수 이름, 메소드 이름, 인스턴스 변수: short_lower_case
    - non-public의 경우에만: _short_lower_case
  - 모듈 이름: shortlowercase/short_lower_case
  - 패키지 이름: shortlowercase
  - 클래스 이름: CapWords
  - 타입변수 이름: 짧은 CapWords, _co(<-covariant), _contra(<-contravariant)

    (ex) T, AnyStr, Num
  - 전역변수 이름: __all __
  - 클래스 메소드 인수: self, cls, class_
  - 모듈 상수: ALL_CAPS

간결화

- 따옴표 종류 통일
- 필요 없는 공백 제거: 괄호 안쪽 공백, 콤마 전 공백, 괄호 바로 앞 공백, 후행 공백, 매개변수에 기본값 표시할 때 '=' 주변 공백

```
spam(ham[1], {eggs: 2})

foo = (0,)

if x == 4: print(x, y); x, y = y, x

ham[1:9], ham[1:9:3], ham[:9:3], ham[1::3], ham[1:9:]
ham[lower:upper], ham[lower:upper:], ham[lower::step]
ham[lower+offset : upper+offset]
ham[: upper_fn(x) : step_fn(x)], ham[:: step_fn(x)]
ham[lower + offset : upper + offset]

spam(1)

dct['key'] = lst[index]

def complex(real, imag=0.0):
    return magic(r=real, i=imag)
```

- 필요한 공백: 이항 연산자 양쪽 한 칸씩 공백, 우선순위가 가장 낮은 연사자들 양쪽 공백, : 뒤 공백, -> 주변 공백, 함수 주석과 기본값에서 = 주변 공백

```
i = i + 1
submitted += 1
x = x*2 - 1
hypot2 = x*x + y*y
c = (a+b) * (a-b)

def munge(input: AnyStr): ...
def munge() -> PosInt: ...

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

- 같은 줄에는 하나의 명령어만
  - 가끔 if/for/while문에 간단한 명령문은 같은 줄에 써도 ㄱㅊ
- 요소를 하나만 가지고 있는 튜플을 만들 때만 (괄호 안에) 뒤에 콤마 붙이기
  - 값이 여러 개일 때 각각의 값은 반드시 한 줄식 입력해야 함

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

- 프로그램을 짜기 전에 클래스의 멤버들을 public으로 할지 non-public으로 할지 정하기. 애매하다면, 우선 non-public으로. (반대의 경우보다 추후 수정이 용이)
- startswith() 및 endswith() 사용
- 객체 유형 비교는 isinstance()
- boolean 값을 ==로 비교하지 않기

```
if greeting:
```

- 값이 비어있는지 아닌지를 검사하기 위해 길이를 확인하지 말고 그냥 if 나 if not으로 확인

```
if not seq:
if seq:
```

주석

- 대문자로 시작하는 완전한 영어 문장.(문장 끝 마침표 뒤에 1~2개의 공백 사용)
- 코드 내용을 되풀이하는 인라인 주석은 사용하지 말 것
- 코드랑 모순되거나 명료한 얘기 되풀이하는 주석 쓰지 말 것

소감

파이썬은 역시 언어라는 생각을 많이 하면서 가이드라인을 읽었다. 코드는 컴퓨터와의 소통만을 위한 수단이 아니라, 여러 사람들이 공유하여 읽을 수 있는 글이다. 그러므로 정말 글을 쓰듯이 가독성을 필히 고려할 필요가 있을 것이다. PEP의 스타일 가이드는 여러 사람이 들여쓰기, 띄어쓰기, 대소문자 등과 같이 간단한 방식으로 자신의 코드를 더욱 명료하게 표현하고 다른 사람의 코드를 더 쉽게 이해할 수 있게 해준다. 많은 사람들이 이와 같은 같은 가이드라인을 공유하고 사용할수록 코드를 읽고 쓰는 일의 전반적인 효율이 제고되리라 생각한다. 나도 스타일 가이드라인을 참고하여 읽기 좋은 코드를 쓰는 프로그래머가 되어 효율적인 파이썬 생태계에 조금이나마 기여하고 싶다고 다짐해본다.

개인적으로는 뒷부분으로 갈수록 클래스, 메소드, 인코딩 등과 관련하여 명확히 알지 못하는 개념들과 써보지 않은 형식의 코드들이 많이 나와 스스로의 부족함을 느끼기도 했다. 방학 동안 스터디와 세션을 중심으로 공부하며 나중에 PEP 8 가이드라인을 다시 읽을 때는 더 명확히 기능과 문법의 의미를 이해할 수 있게 되면 좋겠다.