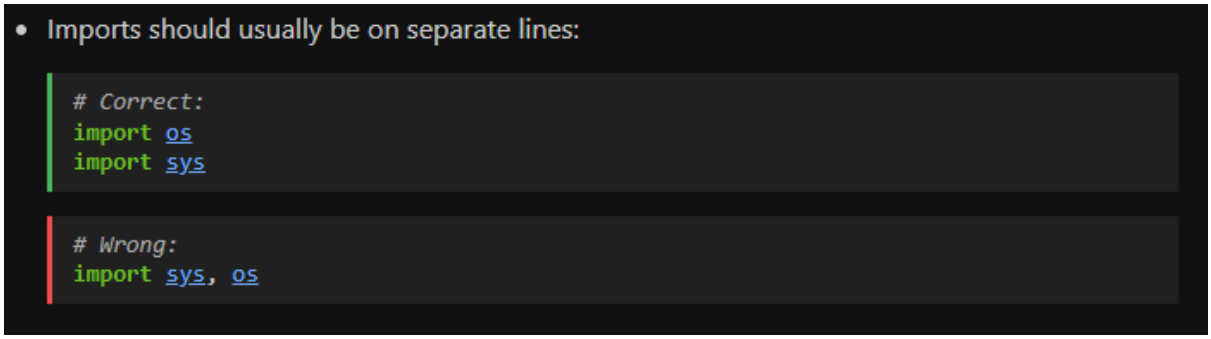
과제1 PEP 8 – Style Guide for Python Code 정독

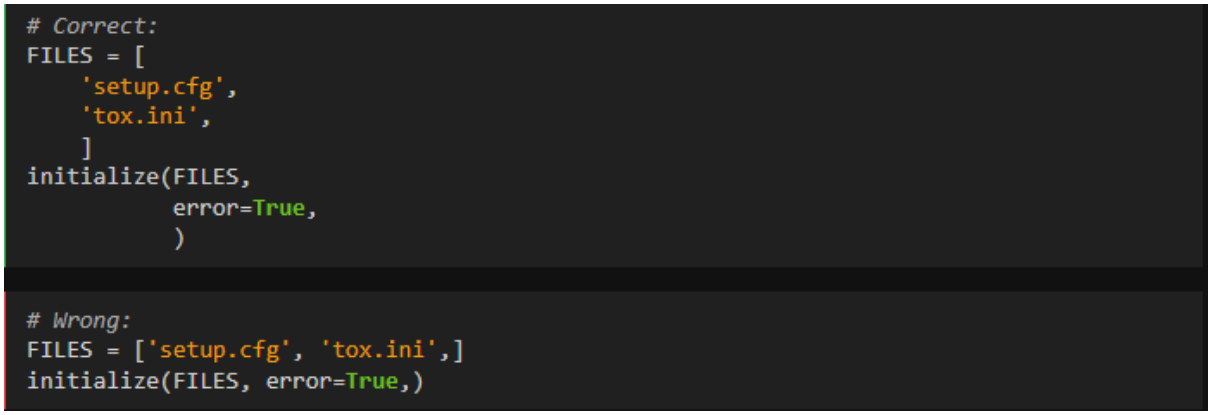
해당 문서에는 파이썬 코딩 스타일에 대한 가이드라인이 담겨 있었다. 대부분 지키던 것들이나,



- Imports should usually be on separate lines:

```
# Correct:
import os
import sys
```

```
# Wrong:
import sys, os
```



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

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

이 두 가지는 잘 지키지 못한 적도 있는 것 같다. 1학년 때 컴퓨터프로그래밍이라는 수업을 들었었다. 교수님께서도 코딩 스타일에 관한 얘기도 이유를 포함해서 여러 번 설명해주셨는데, 솔직히 와닿지 않아서 그냥 쓰기 편한 코딩을 했던 것 같다. 이후 5년 동안 파이썬 뿐만 아니라 다양한 PL의 코딩을 하면서 조금씩 나에게 맞춰 변한 코딩 스타일이, 최근 협업을 하며 남이 읽기 편한 코드는 아니라는 것을 알게 되었었는데, 이 문서를 정독하면서 이러한 점들을 되돌아 볼 수 있었던 것 같다.

과제2 보고서

1. 각 class 디자인

   BPETokenizer class 디자인

   기존 BPE 알고리즘의 문제점: 한번의 merge마다 pair들의 빈도를 전부 다시 계산해야 한다는 것

   해결 방안: merge 횟수 만큼의 best N pair를 뽑아서 한번에 merge하되,

   해당 iteration에서 이미 merge한 character를 제외

   (예시의 [low, lower, newest, widest) 에서처럼 we가 이미 merge해버렸는데 l과 o가 또 merge하는 경우를 방지하기 위해)

   WordTokenizer 디자인

   white space

2. 각 method 작동 원리 (tokenizer.py 주석 참고)

```python
def merge_n_best(pairs, vocab, n):
    if vocab is None:
        raise ValueError("vocab is not initialized. Train tokenizer first!")

    # n 개의 가장 높은 빈도를 가진 pair 선택
    best_n = sorted(pairs, key=pairs.get, reverse=True)[:n]

    # 이미 선택된 문자를 저장하는 집합
    selected_chars = set()

    # 중복되지 않는 새로운 pair를 찾아서 merge
    num_merged = 0  # 중복되지 않아서 merge한 pair의 수를 계산하는 변수
    for best in best_n:
        if num_merged >= n:  # 이미 목표한 수만큼 merge 했으면 종료
            break
        if not any(char in selected_chars for char in best):
            vocab = merge_vocab(best, vocab)
            selected_chars.update(best)
            num_merged += 1

    return num_merged, vocab
```

   ➔ merge_n_best 함수가 이 알고리즘의 메인 아이디어

```python
def train(self, n_iter: int) -> None:
    vocab = get_vocab(self.corpus)
    # n_iter만큼 merge
    for i in range(n_iter):
        pairs = get_stats(vocab)
        num_merged, vocab = merge_n_best(pairs, vocab, n_iter)
        # merge한 갯수를 i에 더해줌
        i += num_merged
    self.vocab = vocab
```

➔ merge_n_best에서 return한 merge한 개수만큼 iteration 횟수에 더해주어 n_iter만큼 merge하도록 조정

3. 제시된 조건들 충족

   typing -> 함수 인자 받을 때 Union, Optional 사용, WordTokenizer의 train에서 n_iter 인자 안 받을 때 사용

   입력 타입에 따른 분기 처리-> isinstance(corpus, list) 활용해서 분기

4. 협업한 방식

   파트너가 일이 많아 과제를 못 하실거 같다고 하셔서 혼자 했습니다.