

# Py Reivew App

최재원

부산대학교 정보컴퓨터공학부 인공지능전공

# Background

- 동기들과의 Python 스터디에서 코드 리뷰를 진행하며 '빠른 피드백'의 중요성을 체감
- 정해진 시간에만 리뷰가 가능하여 발생하는 **피드백 지연** 문제를 해결하고자 함
- LLM을 활용하여 코드 리뷰를 자동화하는 웹 앱 개발을 구상

# Problem Specification

- **입력 (Input):**
  - 리뷰가 필요한 문제 설명
  - 사용자가 작성한 Python 코드
- **출력 (Output):**
  - Linter, Formatter, LLM 등을 활용한 자동화된 코드 리뷰 결과

# Design

- 웹 서버: FastAPI
- 데이터베이스: TinyDB
  - ▶ 유저 정보, 리뷰 로그 저장을 위한 간단한 JSON 기반 NoSQL
- 로그인 및 회원가입:
  - ▶ 세션(Session) 기반으로 로그인 상태 유지
  - ▶ 회원가입 시, pbkdf2\_sha256 알고리즘으로 패스워드를 해싱 후 저장

# Design

- 코드 리뷰 기능:
  - ▶ Linter/Formatter: [Ruff](#) (subprocess로 실행)
  - ▶ AI 리뷰: [OpenAI o1-mini](#) (자체 프롬프트와 조합)
- 프론트엔드: [TailwindCSS](#), [DaisyUI](#)
  - ▶ '페이지 당 단일 HTML' 원칙으로 단순성 추구

# Implementation

- 디렉터리 구조:

- ▶ /app: FastAPI 서버 로직 (Python 코드)
- ▶ /frontend: HTML 등 프론트엔드 리소스

- 계층적 설계:

- ▶ `main.py`의 라우터가 요청을 받고, `auth_helper`, `db_helper` 등 기능별로 분리된 모듈을 호출하여 로직을 처리

# Implementation

로그인

아이디

아이디를 입력해주세요.

비밀번호

비밀번호를 입력해주세요.

로그인 하기

가입이 필요하신가요? [회원가입](#) [회기](#)

# Implementation

회원가입

아이디

아이디를 입력해주세요.

비밀번호

비밀번호를 입력해주세요.

회원가입 하기

이미 계정이 있으신가요? [로그인 하기](#)



# Implementation

py review app

로...마...

| 시간 제한 | 메모리 제한 | 제출 | 정답 | 맞힌 사람 | 정답 비율 |

| --- | --- | --- | --- | --- | --- |

| 1 초 | 1024 MB | 10517 | 4796 | 3568 | 43.779% |

## 문제

작년에 이어 새로운 문자열 게임이 있다. 게임의 진행 방식은 아래와 같다.

1. 알파벳 소문자로 이루어진 문자열 W가 주어진다.

"""

알고리즘 분류에는 슬라이딩 윈도우라고 되어있는데, 왜 슬라이딩 윈도우인  
지는 잘 모르겠다.

최소, 최대 문자열의 모두 시작점과 끝점이 무조건 같아야한다는 사실을 발견  
함.

처음에는 이중 for문을 이용하여 풀었더니 시간 초과가 나서  
미리 전처리를 해주고 for문을 돌리는 코드를 작성하니 시간 안에 풀 수 있었  
다.

"""

import sys

from collections import defaultdict

T = int(input()) # T: testcase

for i in range(T):

W = sys.stdin.readline().rstrip()

K = int(input())

# K=1 일 때는 특수 케이스이므로 처리

if K == 1:

print(1, 1)

continue

min\_len = float('inf')

max\_len = -float('inf')

코드 리뷰

🔍 린터

실행 옵션

☒ ruff linter

☒ ruff formatter

☒ ai code review

⚡ 코드 리뷰 중입니다.

실행하기

# Implementation

py review app

로그아웃

| 시간 제한 | 메모리 제한 | 제출 | 정답 | 맞힌 사람 | 정답 비율 |

| --- | --- | --- | --- | --- | --- |

| 1 초 | 1024 MB | 10517 | 4796 | 3568 | 43.779% |

## 문제

작년에 이어 새로운 문자열 게임이 있다. 게임의 진행 방식은 아래와 같다.

1. 알파벳 소문자로 이루어진 문자열 W가 주어진다.

코드가 포맷팅 되었어요

```python\n\n알고리즘 분류에는 슬라이딩 윈도우라고 되어있는데, 왜 슬라이딩 윈도우인\n지는 잘 모르겠다.\n\n최소, 최대 문자열의 모두 시작점과 끝점이 무조건 같아야한다는 사실을 발견\n함.\n\n처음에는 이중 for문을 이용하여 풀었더니 시간 초과가 나서\n미리 전처리를 해주고 for문을 돌리는 코드를 작성하니 시간 안에 풀 수 있었\n다.\n\n```\n\nimport sys\nfrom collections import defaultdict\n\nT = int(input()) # T: testcase\n\nfor i in range(T):\n W = sys.stdin.readline().rstrip()\n K = int(input())\n\n # K=1 일 때는 특수 케이스이므로 처리\n if K == 1:\n print(1, 1)\n continue\n\n min\_len = float("inf")\n max\_len = -float("inf")

코드 리뷰

안녕하세요! 문제와 코드를 공유해 주셔서 감사합니다. 주어진 문제를 해결하기 위해 접근하신 방법은 올바른 방향으로 가고 있지만, 약간의 최적화가 필요해 보입니다. 아래에서 문제의 요구사항과 효율적인 해결 방법에 대해 자세히 설명드리겠습니다.

문제 이해

주어진 문제는 다음과 같습니다:  
문자열 W와 정수 K가 주어집니다.  
3번 문제: 어떤 문자를 정확히 K개 포함하는 가장 짧은 연속 문자열의 길이를 구합니  
다.  
4번 문제: 어떤 문자를 정확히 K개 포함하고, 그 문자열의 첫 번째와 마지막 문자가  
해당 문자와 같은 가장 긴 연속 문자열의 길이를 구합니다.  
만약 만족하는 문자열이 없으면 -1을 출력합니다.

제한 사항:

문자열의 길이 |W| 는 최대 10,000입니다.  
테스트 케이스는 최대 100개입니다.  
접근 방법

현재 사용하신 접근 방식은 다음과 같습니다:  
각 문자에 대해 인덱스를 미리 저장 (pos dictionary 사용).  
각 문자에 대해 연속된 K개의 인덱스를 찾아서, 해당 범위의 길이를 계산하여 최소 및  
최대 길이를 업데이트.  
이 접근 방식은 자체적으로 효율적이지만 코드 내에서 max(r1, index(i))를 사

린터

--:1:1: D212 [\*] Multi-line docstring summary should start at the first line  
--:21:5: T201 'print' found --:23:1: W293 [\*] Blank line contains  
whitespace --:24:19: Q000 [\*] Single quotes found but double quotes  
preferred --:25:20: Q000 [\*] Single quotes found but double quotes  
preferred --:31:7: PLW2901 Outer 'for' loop variable 'i' overwritten by  
inner 'for' loop target --:34:7: PLW2901 Outer 'for' loop variable 'i'  
overwritten by inner 'for' loop target --:37:49: E701 Multiple statements  
on one line (colon) --:42:23: Q000 [\*] Single quotes found but double  
quotes preferred --:42:51: Q000 [\*] Single quotes found but double  
quotes preferred --:43:5: T201 'print' found --:45:5: T201 'print' found

실행 옵션

✓ ruff linter

✓ ruff formatter

✓ ai code review

실행하기

# Analysis & Discussion

- 데이터베이스와 ORM:

- ▶ 고민: ORM 도입을 고려했으나, 프로젝트 규모에 비해 학습/설정 비용이 크다고 판단.
- ▶ 해결: `TinyDB (NoSQL)` + `Pydantic (데이터 검증)` 조합을 선택.
- ▶ 구조: `Service <-> Pydantic <-> NoSQL` 데이터 흐름을 만들어 ORM의 장점(데이터 유효성 검증)과 NoSQL의 단순함을 모두 취함.
- ▶ 한계: `TinyDB`는 단일 스레드 기반이므로, 멀티 프로세스 환경에서는 사용 불가.

# Analysis & Discussion

- **프론트엔드:**

- ▶ **목표:** 백엔드 로직에 집중하기 위해 프론트엔드는 최대한 단순하게 유지.
- ▶ **구현:** React 같은 무거운 도구 대신, 페이지 당 단일 HTML 파일과 CSS 라이브러리(Tailwind, DaisyUI)를 활용하여 개발 속도 향상.

# Analysis & Discussion

- **인증 구현 상세:**

- ▶ **패스워드 보안:** 회원가입 시, 각 패스워드마다 salt 값을 부여하고 pbkdf2\_sha256로 해싱. (레인보우 테이블 공격 방어 및 최신 GPU 환경에서 bcrypt보다 안전)
- ▶ **성능:** 해싱은 CPU-bound 작업으로, 사용자 증가 시 서버 병목이 될 수 있음. 대규모 서비스에서는 별도 스레드에서 비동기 처리 필요. 현 프로젝트에선 미적용.
- ▶ **세션 관리:** 메모리에 세션을 저장하므로 멀티 프로세스/스레드 환경에서는 세션 공유 불가. 확장 시, Redis 같은 외부 인메모리 DB를 세션 스토어로 사용해야 함.

# Analysis & Discussion

- **LLM 비용 관리:**

- ▶ 토큰 당 비용 문제를 해결하기 위해 서비스에 제한을 적용.
- ▶ (1)관리자 인증
- ▶ (2)하루 5회
- ▶ (3)문제/코드 각 1500자 이하.

# Conclusion

- Python 코드 리뷰 웹앱을 성공적으로 구현 완료.
- **FastAPI**, **Pydantic** 등을 활용하며 Python 웹 생태계의 견고함과 매력을 느낌.
- 단순한 앱이라도 모듈 분리 등 좋은 설계의 중요성을 체감.
- 알고리즘 문제 풀이를 넘어, 실제 서비스를 만들며 언어에 대한 이해도를 높이는 계기가 됨.

Source Code: <https://github.com/cjaewon/pyreview-app>