**Final Project:**

**Step 1: Plan Your Project (30 Minutes)**

1. **Define the Scope**:
   * The project will analyze legal documents to find specific legal terms or phrases.
   * Features:
     + Upload text or input a sample text.
     + Identify and highlight legal keywords or citations.
     + Output the results in a readable format.
2. **Technology Choices**:
   * Language: **Python** (simple and powerful for text processing).
   * Libraries:
     + re (for regex-based keyword searches).
     + sys (to take command-line input).
   * Optional: argparse (for improved command-line input parsing).
3. **Project Folder Setup**:
   * Create a folder named project.
   * Inside, create files:
     + analyzer.py (main Python script).
     + keywords.txt (file containing keywords to search for).
     + Sample legal document files (e.g., sample1.txt, sample2.txt).

**Step 2: Build Your Program (3-4 Hours)**

**Part 1: Implement the Core Functionality**

1. **Set Up the Main Script** (analyzer.py):
   * Import necessary libraries (re, sys).
   * Write code to load a text file provided as a command-line argument.
2. **Add Keyword Detection**:
   * Load keywords from keywords.txt.
   * Use Python’s re module to search for these keywords in the document.
   * Count and display the frequency of each keyword found.
3. **Output the Results**:
   * Print the results to the console.
   * Optionally save the results to a new file (results.txt)

**Part 2: Test and Refine**

1. Test with a small sample document (sample1.txt) to verify it works.
2. Add error handling:
   * Check if the file exists.
   * Handle cases where no keywords are found.

**Example Code (Starter)**

python

Copiar código

import re

import sys

def load\_keywords(file\_path):

with open(file\_path, "r") as file:

return [line.strip() for line in file.readlines()]

def analyze\_document(document\_path, keywords):

try:

with open(document\_path, "r") as file:

text = file.read()

results = {}

for keyword in keywords:

matches = re.findall(rf"\b{keyword}\b", text, re.IGNORECASE)

results[keyword] = len(matches)

return results

except FileNotFoundError:

print(f"Error: File {document\_path} not found.")

sys.exit(1)

def main():

if len(sys.argv) != 2:

print("Usage: python analyzer.py <document\_path>")

sys.exit(1)

document\_path = sys.argv[1]

keywords = load\_keywords("keywords.txt")

results = analyze\_document(document\_path, keywords)

print("Keyword Analysis Results:")

for keyword, count in results.items():

print(f"{keyword}: {count}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Part 3: Polish Your Project**

1. Add a few sample legal terms to keywords.txt (e.g., "reasonable doubt", "burden of proof").
2. Create two sample text files with legal content for testing (sample1.txt, sample2.txt).
3. Test thoroughly to ensure it handles edge cases.

**Step 3: Create the README.md File (1 Hour)**

1. Open a file named README.md in your project folder.
2. Use the following structure:

markdown

Copiar código

# Legal Document Analyzer

#### Video Demo: [INSERT YOUR VIDEO LINK HERE]

#### Description:

The Legal Document Analyzer is a Python-based tool that allows users to search for and analyze specific legal terms in a document.

## Features:

- Input legal documents in `.txt` format.

- Search for keywords like "burden of proof" or "reasonable doubt."

- Outputs frequency counts for each keyword.

## How It Works:

- Keywords are listed in a `keywords.txt` file.

- The program reads a document, searches for matches, and prints the results to the console.

## Files:

- `analyzer.py`: The main program.

- `keywords.txt`: List of legal terms to search for.

- `sample1.txt` and `sample2.txt`: Sample legal documents for testing.

**Step 4: Record the Video (1 Hour)**

1. Prepare the following:
   * Briefly explain your project’s goal and features.
   * Show the folder structure in your IDE.
   * Demonstrate how the program works:
     + Run it with a sample document.
     + Display the output.
2. Tools to record:
   * Windows: Use Xbox Game Bar (Win + G).
   * Mac: Use QuickTime Player.
   * Linux: Use SimpleScreenRecorder.
3. Include:
   * Project Title: Legal Document Analyzer
   * Your Name
   * GitHub and edX usernames
   * City and Country
   * Date
4. Upload the video to YouTube as “unlisted” and save the URL.

**Step 5: Submit the Project (30 Minutes)**

1. Test submit50 in the terminal to ensure the project folder is ready.

bash

Copiar código

submit50 cs50/problems/2024/x/project

1. Check your gradebook at [cs50.me/cs50x](https://cs50.me/cs50x" \t "_new) to confirm submission.

You’re done! Let me know if you need further help or clarification

Video:

you need to submit "a short video (that's no more than 3 minutes in length) in which you present your project to the world, as with slides, screenshots, voiceover, and/or live action. Your video should somehow include your project’s title, your name and year, your city/state/country, and any other details that you’d like to convey to viewers. Upload your video to YouTube and take note of its URL."   When uploading to YouTube, it's fine to flag your video as "unlisted", but don't flag it as "private," else we won't be able to view it.  And be sure not to disable embedding. What's the URL of your video on YouTube?\*

Tu respuesta

Hi everyone. My name is Steffany Tovar Villa. I live in Armenia, Quindío, Colombia. I'm recording this video the 30th of December of 2024. My GitHub user is steffanytovarvilla and my edx user name is steffanytovar25. My project name is Legal Document Keyword Analyzer.

My project is a Python-based tool designed to analyze legal documents for key terms and phrases. It helps legal professionals identify and count occurrences of critical legal concepts within text files.

The main features of my tool are the keyword Detection, which allows me to Scan a document for a list of pre-defined keywords provided in `keywords.txt`.

Also the Frequency Analysis, which permits to Count and displays the number of times each keyword appears in the document.

And finally, how customizable it is, since Users can modify `keywords.txt` to include their own list of terms to analyze.