Word Frequency Counter - CLI Challenge

Time Limit: 45 minutes
Difficulty: Intermediate
Skills: File I/O, Data Structures, Command Line Arguments

Challenge Overview

Build a command-line tool that analyzes text files and reports word frequency statistics. Your program should read a text file, count how often each word appears, and display the results in a user-friendly format.

Project Structure

```
word_frequency_challenge/
├── word_counter.py          # Your main implementation
├── sample_text.txt          # Provided test file
├── README.md               # This file
└── tests/
    └── test_word_counter.py # Optional unit tests
```

Core Requirements (35 minutes)

1. Command Line Interface (5 minutes)

- Accept filename as command line argument
- Display usage message if no argument provided
- Handle file not found errors gracefully

Example Usage:

bash

```bash
python word_counter.py sample_text.txt
python word_counter.py /path/to/any/textfile.txt
```

2. File Processing (10 minutes)

- Read entire text file into memory
- Handle encoding issues (UTF-8)
- Strip whitespace and normalize text

3. Word Counting Logic (15 minutes)

- Split text into individual words
- Convert to lowercase for case-insensitive counting
- Remove punctuation from words
- Store word counts in a dictionary/hash map
- Ignore empty strings

Text Processing Rules:

- "Hello," and "hello" should count as the same word
- "don't" should be treated as "dont" (remove apostrophes)
- Numbers can be treated as words
- Minimum word length: 1 character

4. Results Display (5 minutes)

- Show total word count
- Display top 10 most frequent words
- Format: `word: count`
- Sort by frequency (highest first)

Expected Output Format:

```
=== Word Frequency Analysis ===
File: sample_text.txt
Total words: 1,247
Unique words: 456

Top 10 Most Frequent Words:
1. the: 89
2. and: 67
3. to: 45
4. of: 41
5. a: 38
6. in: 32
7. is: 28
8. that: 24
9. for: 22
10. with: 19
```

Bonus Features (10 minutes)

Choose **ONE** of these bonus features if you finish early:

Option A: Sorting Modes

Add command line flag to change sorting:

- `--sort=frequency` (default)
- `--sort=alphabetical`

Option B: Export Results

Add ability to save results to CSV file:

- `--output=results.csv`
- Format: `word,count`

Option C: Stop Words Filter

Filter out common English stop words:

- Create list: `["the", "and", "to", "of", "a", "an", "in", "is", "it", "you", "that", "he", "was", "for", "on", "are", "as", "with", "his", "they", "i", "at", "be", "this", "have", "from", "or", "one", "had", "by", "word", "but", "not", "what", "all", "were", "we", "when", "your", "can", "said", "there", "each", "which", "she", "do", "how", "their", "if", "will", "up", "other", "about", "out", "many", "then", "them", "these", "so", "some", "her", "would", "make", "like", "into", "him", "has", "two", "more", "go", "no", "way", "could", "my", "than", "first", "been", "call", "who", "its", "now", "find", "long", "down", "day", "did", "get", "come", "made", "may", "part"]`
- Add `--filter-stopwords` flag

Option D: Statistics Summary

Add detailed statistics:

- Average word length
- Longest/shortest words
- Words appearing only once

Technical Requirements

Python Standard Library Only

- Use only built-in Python modules
- No external dependencies (no pandas, numpy, etc.)
- Suggested modules: `sys`, `argparse`, `collections`, `string`, `os`

Error Handling

Your program should handle these errors gracefully:

- File not found
- Permission denied
- Empty files
- Invalid command line arguments

Code Quality

- Use meaningful variable names
- Add docstrings to functions
- Keep functions focused and small
- Follow PEP 8 style guidelines

Starter Code Template

python

```python
#!/usr/bin/env python3
"""
Word Frequency Counter
Analyzes text files and reports word frequency statistics.
"""

import sys
import string
from collections import Counter


def read_file(filename):
    """
    Read and return the contents of a text file.

    Args:
        filename (str): Path to the text file

    Returns:
        str: Contents of the file

    Raises:
        FileNotFoundError: If file doesn't exist
        PermissionError: If file can't be read
    """
    # TODO: Implement file reading with error handling
    pass


def clean_word(word):
    """
    Clean a word by removing punctuation and converting to lowercase.

    Args:
        word (str): Raw word from text

    Returns:
        str: Cleaned word, or empty string if invalid
    """
    # TODO: Remove punctuation, convert to lowercase
    pass
```

```python
def count_words(text):
    """
    Count frequency of words in text.

    Args:
        text (str): Input text to analyze

    Returns:
        dict: Dictionary mapping words to their frequencies
    """
    # TODO: Split text, clean words, count frequencies
    pass


def display_results(word_counts, filename, top_n=10):
    """
    Display word frequency results in formatted output.

    Args:
        word_counts (dict): Word frequency data
        filename (str): Name of analyzed file
        top_n (int): Number of top words to display
    """
    # TODO: Format and print results
    pass


def main():
    """
    Main program entry point.
    Handle command line arguments and coordinate the analysis.
    """
    # TODO: Parse command line arguments
    # TODO: Read file
    # TODO: Count words
    # TODO: Display results
    pass


if __name__ == "__main__":
    main()
```

Sample Test File

Create `sample_text.txt` with this content for testing:

```
The quick brown fox jumps over the lazy dog. The dog was not amused by the fox's
behavior, but the fox didn't care. The quick brown fox is known for being quick, and
the lazy dog is known for being lazy. This is a simple test file to help you test
your word frequency counter. The word "the" appears many times in this text, making
it likely to be the most frequent word. Other common words like "and", "is", "for",
and "to" should also appear frequently in the results.
```

Evaluation Criteria

Functionality (60%)

- ✅ Reads file from command line argument
- ✅ Correctly counts word frequencies
- ✅ Displays top 10 results
- ✅ Handles errors gracefully

Code Quality (25%)

- ✅ Clean, readable code
- ✅ Proper function organization
- ✅ Good variable names
- ✅ Basic error handling

User Experience (15%)

- ✅ Clear output formatting
- ✅ Helpful error messages
- ✅ Intuitive command line interface

Time Management Tips

Minutes 0-5: Read requirements, understand the problem
Minutes 5-15: Set up file reading and command line parsing
Minutes 15-30: Implement word counting logic
Minutes 30-40: Add output formatting and error handling
Minutes 40-45: Test with sample file and debug

Common Pitfalls to Avoid

1. **Over-engineering:** Don't spend time on complex regex or advanced text processing
2. **Perfect punctuation handling:** Simple string methods are sufficient
3. **Memory optimization:** Don't worry about large files for this challenge
4. **Complex argument parsing:** Basic `sys.argv` checking is enough for core requirements

Success Indicators

You've successfully completed the challenge when:

- Your program runs without errors on the sample file
- Word counts are accurate (test with a few manual counts)
- Output is clean and readable
- Basic error handling works (try with non-existent file)

Good luck! Remember: working code is better than perfect code.