Why This Approach Is Optimal for Smart Text Analyzer

1. **Simplicity**: Uses Python's built-in libraries instead of external dependencies
2. **Performance**: str.translate() is faster than regex alternatives for punctuation
3. **Efficiency**: collections.Counter provides O(n) frequency counting
4. **Maintainability**: Clear 10-step process makes code easy to modify
5. **Reliability**: Explicit error handling prevents silent failures
6. **Consistency**: Deterministic output through predictable sorting

Design Decisions Made (Documented)

- **Why collections.Counter**: O(n) performance, purpose-built, cleaner than manual loops
- **Why str.translate()**: Single-pass efficiency, faster than regex, more readable
- **Why alphabetical + frequency sorting**: Identifies important words first, deterministic output
- **Why explicit error handling**: Type checking, specific exceptions, descriptive messages
- **Why case-insensitive**: Standard NLP practice, words like "The" and "the" should count as same

Problems Faced & Solutions

- **Challenge**: Empty input after punctuation removal → Solution: Validation check with ValueError
- **Challenge**: Division by zero risk → Solution: Verify words list before average calculation
- **Challenge**: Type safety → Solution: isinstance() check with TypeError
- **Challenge**: Unpredictable output order → Solution: Sort by frequency then alphabetically