mahtabsyed@Mahtabs-MacBook-Pro 21 Agentic Patterns % python3 11\_Goal\_Setting\_Iteration\_2.py

📡 Initializing OpenAI LLM (gpt-4o)...

🧠 Welcome to the AI Code Generation Agent

🎯 Use Case: Write code which takes a command line input of a word doc or docx file and opens it and counts the number of words, characters and numbers of pages in it and prints all

🎯 Goals:

- Code simple to understand

- Functionally correct

- Handles edge cases

=== 🔁 Iteration 1 of 5 ===

📝 Constructing prompt for code generation...

🚧 Generating code...

🧾 Generated Code:

--------------------------------------------------

import sys

from docx import Document

import os

def count\_words\_characters\_pages(doc\_path):

if not os.path.exists(doc\_path):

print("File does not exist.")

return

if not doc\_path.lower().endswith(('.doc', '.docx')):

print("Invalid file format. Please provide a .doc or .docx file.")

return

try:

doc = Document(doc\_path)

except Exception as e:

print(f"Error opening document: {e}")

return

word\_count = 0

char\_count = 0

page\_count = len(doc.element.xpath('//w:sectPr'))

for paragraph in doc.paragraphs:

text = paragraph.text

word\_count += len(text.split())

char\_count += len(text)

print(f"Words: {word\_count}")

print(f"Characters: {char\_count}")

print(f"Pages: {page\_count}")

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

if len(sys.argv) != 2:

print("Usage: python script.py <path\_to\_doc\_or\_docx\_file>")

else:

count\_words\_characters\_pages(sys.argv[1])

--------------------------------------------------

📤 Submitting code for feedback review...

🔍 Evaluating code against the goals...

📥 Feedback Received:

--------------------------------------------------

The provided code snippet is a Python script designed to count the number of words, characters, and pages in a `.doc` or `.docx` file using the `python-docx` library. Let's evaluate it based on the specified goals:

### 1. Code Simplicity and Clarity

- \*\*Imports\*\*: The code imports necessary modules (`sys`, `os`, and `Document` from `python-docx`). These are appropriate for the task.

- \*\*Function Definition\*\*: The function `count\_words\_characters\_pages` is clearly defined with a descriptive name that indicates its purpose.

- \*\*Error Messages\*\*: The error messages are clear and informative, helping the user understand what went wrong.

- \*\*Code Structure\*\*: The code is structured logically, with checks for file existence and format before attempting to process the document.

### 2. Functional Correctness

- \*\*Word and Character Count\*\*: The script correctly counts words and characters by iterating over each paragraph in the document.

- \*\*Page Count\*\*: The page count is derived from the number of `sectPr` elements, which is a reasonable approach for `.docx` files.

### 3. Edge Case Handling

- \*\*File Existence\*\*: The script checks if the file exists before proceeding, which is good.

- \*\*File Format\*\*: It checks if the file has the correct extension, which helps prevent processing unsupported formats.

- \*\*Error Handling\*\*: The script includes a try-except block to handle exceptions when opening the document, which is good practice.

### Areas for Improvement

- \*\*File Format Check\*\*: While the script checks for `.doc` and `.docx` extensions, it only processes `.docx` files. The check for `.doc` is misleading since `python-docx` does not support `.doc` files.

- \*\*Character Count\*\*: The character count currently includes spaces. If the intent is to count only non-space characters, this should be adjusted.

- \*\*Page Count\*\*: The method for counting pages might not be accurate for complex documents with multiple sections or custom page layouts. Consider using a library that can accurately render and count pages.

- \*\*Test Coverage\*\*: The script does not include any tests. Adding unit tests would improve reliability and ensure the function behaves as expected across different scenarios.

### Suggested Improvements

1. \*\*Remove Unsupported Format Check\*\*: Remove `.doc` from the file format check or add support for it using a different library.

2. \*\*Clarify Character Count\*\*: Specify whether spaces should be included in the character count and adjust the code if necessary.

3. \*\*Improve Page Count Accuracy\*\*: Consider using a more robust method or library for counting pages if accuracy is critical.

4. \*\*Add Tests\*\*: Implement unit tests to cover various edge cases, such as empty documents, documents with no text, and documents with complex formatting.

### Revised Code Snippet

```python

import sys

from docx import Document

import os

def count\_words\_characters\_pages(doc\_path):

if not os.path.exists(doc\_path):

print("File does not exist.")

return

if not doc\_path.lower().endswith('.docx'):

print("Invalid file format. Please provide a .docx file.")

return

try:

doc = Document(doc\_path)

except Exception as e:

print(f"Error opening document: {e}")

return

word\_count = 0

char\_count = 0

page\_count = len(doc.element.xpath('//w:sectPr'))

for paragraph in doc.paragraphs:

text = paragraph.text

word\_count += len(text.split())

char\_count += len(text) # Adjust if spaces should not be counted

print(f"Words: {word\_count}")

print(f"Characters: {char\_count}")

print(f"Pages: {page\_count}")

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

if len(sys.argv) != 2:

print("Usage: python script.py <path\_to\_docx\_file>")

else:

count\_words\_characters\_pages(sys.argv[1])

```

This revised version addresses the misleading file format check and clarifies the character count logic.

--------------------------------------------------

✅ LLM confirms goals are met. Stopping iteration.

💾 Saving final code to file...

✅ Code saved to: /Users/mahtabsyed/Documents/VSCode Cursor/21 Agentic Patterns/docstats\_8452.py

mahtabsyed@Mahtabs-MacBook-Pro 21 Agentic Patterns %