Script for Combining Text Files into a Word Document

Overview

This script combines multiple text files into a single Microsoft Word document. It automatically generates a title for each section based on the content of the files and formats the document according to specified styles.

Features

- **Title Generation**: Automatically generates a title for each section in the format "Appendix [Letter] Configuration for [SiteName]: [Hostname]".
- **Content Formatting**: Sets the title to bold and uses a specified font and size. The body text is formatted with a different font size and single spacing.
- **Error Checking**: Verifies if the specific string "no ip http server" is present in each file and reports errors if not found.
- **File Selection**: Allows users to select multiple text files through a graphical file dialog.

Requirements

- Python 3.x
- python-docx library
- tkinter library (included with Python standard library)

Installation

1. **Install Python**: Ensure latest version Python 3.x is installed on your machine. Download from python.org if needed.
2. **Install Dependencies**: Install the python-docx library using pip. In terminal write as follows:

```
pip install python-docx
```

Script Usage

1. Script Overview

- **File Selection**: Opens a file dialog to select multiple .txt files.
- **Processing**: For each file, the script:
    - Extracts the hostname.
    - Determines the site name from the hostname.
    - Generates a title with an appendix letter (starting from D and incrementing alphabetically).
    - Adds the content of the file to the Word document.
    - Checks for the presence of the string "no ip http server".
- **Output**: Saves the combined document as combined_document.docx.

2. How to Run the Script

Save the Script:

- **File Name**: Save the provided Python script as combine_txt_to_docx.py on your local machine.
- **Location**: Choose a convenient directory where you will store and run the script.

Install Python and Dependencies:

- Ensure Python 3.x is installed on your machine. Download from python.org if needed.

Run the Script Using Command Line:

1. **Open Command Prompt (Windows) or Terminal (macOS/Linux)**:
   - **Windows**: Search for "Command Prompt" or "cmd" in the Start menu.
   - **macOS/Linux**: Open the Terminal application.
2. **Navigate to the Directory**:
   - cd path/to/your/script
3. **Execute the Script**:
   - python combine_txt_to_docx.py
4. **Select Files**:
   - A file dialog will appear. Navigate to and select the .txt files you want to combine.
   - Click "Open" to process the selected files.

Run the Script Using an Integrated Development Environment (IDE):

1. **Open Your IDE**:
   - Use an IDE like PyCharm, Visual Studio Code, or any other that supports Python.

2. Open Script::

- Open combine_txt_to_docx.py in your IDE.

3. Run the Script:

- Use the "Run" or "Execute" option provided by your IDE. This usually involves clicking a green "Run" button or selecting "Run" from the menu.

4. Select Files:

- Follow the same file selection process as described above.

Code Details and Function Descriptions

1. Import Statements

Code:

```
import re
from docx import Document
from docx.shared import Pt
from docx.oxml.ns import qn
import tkinter as tk
from tkinter import filedialog
import os
```

Explanation:

- **re:** Used for regular expression operations, such as extracting the hostname.
- **docx:** Provides functionalities for creating and manipulating Word documents.
- **tkinter:** Used to create a graphical file dialog for selecting files.
- **os:** Provides a way to interact with the operating system, although it’s not used in this script.

2. Function: extract_hostname

```
def extract_hostname(content):
    match = re.search(r'hostname\s+(\S+)', content)
    if match:
        return match.group(1)
    return 'Unknown'
```

- **Purpose**: Extracts the hostname from the content of a .txt file.
- **How It Works**: Uses a regular expression to find the text following the keyword hostname. Returns the hostname if found, otherwise returns 'Unknown'.

3. Function: extract_site_name

```
def extract_site_name(hostname):
    parts = hostname.split('_')
    if parts:
        return parts[0]
    return 'Unknown'
```

- **Purpose**: Extracts the site name from the hostname.
- **How It Works**: Splits the hostname by the underscore character _ and returns the first part. If no underscore is found, returns 'Unknown'.

4. Function: get_next_appendix_letter

```
def get_next_appendix_letter(current_letter):
    # Function to convert a letter (e.g., 'A') to its corresponding position (e.g., 0 for 'A')
    def letter_to_number(letter):
        return ord(letter) - ord('A')

    # Function to convert a number (e.g., 0) to its corresponding letter (e.g., 'A')
    def number_to_letter(number):
        return chr(number + ord('A'))

    # Convert current letter to a list of its positions in the alphabet (e.g., 'AB' -> [0, 1])
    positions = [letter_to_number(c) for c in current_letter]

    # Increment the letter positions
    for i in reversed(range(len(positions))):
        if positions[i] < 25:  # 25 corresponds to 'Z'
            positions[i] += 1
            break
        positions[i] = 0  # Reset current position to 'A' if it overflows to 'Z'

    # If all positions have overflowed (e.g., from 'Z' -> 'AA'), add a new letter
    if all(p == 0 for p in positions):
        positions.insert(0, 0)

    # Convert positions back to letters and return as a string
    next_letter = ''.join(number_to_letter(p) for p in positions)
    return next_letter
```

Explanation:

Helper Functions:

- letter_to_number(letter): Converts a letter ('A'-'Z') to its corresponding position (0-25).
- number_to_letter(number): Converts a number (0-25) to its corresponding letter ('A'-'Z').

Convert Current Letter:

- Converts the current appendix letter to a list of its positions in the alphabet. For example, 'AB' becomes [0, 1].

Increment Positions:

- The function iterates over the positions in reverse order to increment the last letter. If it overflows from 'Z' to 'A', it carries over to the next letter.

Handle Overflow:

- If all positions overflow (e.g., from 'ZZ' to 'AAA'), it inserts a new letter at the beginning.

Convert Back to String:

- Converts the positions back to letters and joins them into the next appendix letter.

5. Function: add_file_to_document

```
def add_file_to_document(doc, file_path, appendix_letter, separator_style):
    with open(file_path, 'r') as file:
        content = file.read()

        if 'no ip http server' not in content:
            print(f"Error: 'no ip http server' not found in {file_path}")

        hostname = extract_hostname(content)
        site_name = extract_site_name(hostname)

        title_paragraph = doc.add_paragraph()
        run = title_paragraph.add_run(f"Appendix {appendix_letter} Configuration for
{site_name}: {hostname}")
        run.bold = True
        run.font.size = Pt(separator_style['size'])
        run.font.name = separator_style['font']
        title_paragraph.style.font.name = separator_style['font']
        title_paragraph.style.element.rPr.rFonts.set(qn('w:eastAsia'),
separator_style['font'])

        doc.add_paragraph(content, style='Normal')
        doc.add_paragraph()
```

- **Purpose**: Adds content from a .txt file to the Word document with a formatted title.
- **How It Works**:
  - Opens and reads the file content.
  - Checks if the string "no ip http server" is present, prints an error if not.
  - Extracts the hostname and site name.
  - Adds a formatted title with appendix letter and site name to the document.
  - Adds the file content and inserts spacing between sections.

6. Function: combine_txt_files_to_docx

```
def combine_txt_files_to_docx(txt_files, output_docx, separator_style):
    doc = Document()
    doc.styles['Normal'].font.size = Pt(8)

    appendix_letter = 'D'
    for file_path in txt_files:
        add_file_to_document(doc, file_path, appendix_letter, separator_style)
        appendix_letter = get_next_appendix_letter(appendix_letter)

    doc.save(output_docx)
```

- **Purpose**: Combines multiple text files into a single Word document.
- **How It Works**:
    - Creates a new Word document.
    - Sets the font size for normal text.
    - Iterates over each text file, adding its content and updating the appendix letter.
    - Saves the final document as output_docx.

7. Function: select_files

```
def select_files():
    root = tk.Tk()
    root.withdraw()
    file_paths = filedialog.askopenfilenames(
        title='Select text files',
        filetypes=[('Text files', '*.txt')]
    )
    return list(file_paths)
```

- **Purpose**: Opens a file dialog to allow users to select multiple .txt files.
- **How It Works**:
    - Initializes a hidden root window (necessary for tkinter).
    - Opens a file dialog where users can select multiple text files.
    - Returns the list of selected file paths.

8. Main Execution Block

```
if __name__ == "__main__":
    txt_files = select_files()
    if not txt_files:
        print("No files selected.")
    else:
        output_docx = 'combined_document.docx'
        separator_style = {'font': 'Calibri', 'size': 16}
        combine_txt_files_to_docx(txt_files, output_docx, separator_style)
        print(f"Document saved as {output_docx}")
```

- **Purpose**: Executes the main logic of the script.
- **How It Works**:
    - Calls select_files() to get the list of selected .txt files.
    - If files are selected, it combines them into a single Word document using combine_txt_files_to_docx().
    - Prints a confirmation message with the name of the saved document.