## ****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](https://www.python.org/downloads/" \t "C:\\Users\\vc\\AppData\\Local\\Temp\\_new) 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](https://www.python.org/downloads/" \t "C:\\Users\\vc\\AppData\\Local\\Temp\\_new) if needed.
  1. In

**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.
  1. **Navigate to the Directory**:
* cd path/to/your/script
  1. **Execute the Script**:
* python combine\_txt\_to\_docx.py
  1. **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.

1. **Open Script:**:

* Open combine\_txt\_to\_docx.py in your IDE.

1. **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.

1. **Select Files:**

* Follow the same file selection process as described above.

## ****Code Details and Function Descriptions****

### ****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.