

Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de ciencias y sistemas

Ing. Damaris Campos

Aux. Enrique Pínula

Sección: A-



PROYECTO 1- MANUAL TECNICO

Nombre: Frander Ovelto Carreto Gómez

Carné: 201901371

Fecha: 18/09/2023

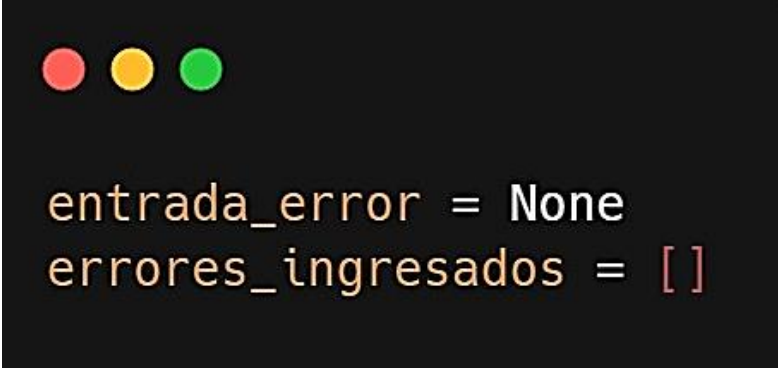
INTRODUCCION

El presente manual técnico tiene como finalidad proporcionar una descripción exhaustiva y detallada del código fuente de una aplicación desarrollada en el lenguaje de programación Python, utilizando la biblioteca gráfica Tkinter. La aplicación en cuestión se ha diseñado con el propósito de permitir a los usuarios llevar a cabo análisis de texto, identificar errores específicos y generar informes visuales de las operaciones realizadas.

El desarrollo de esta aplicación se basa en una arquitectura modular y orientada a objetos, lo que facilita su comprensión, mantenimiento y extensión. Este manual se enfocará en explicar de manera sistemática las principales funcionalidades y componentes del código fuente subyacente.

1. Definición de variables.

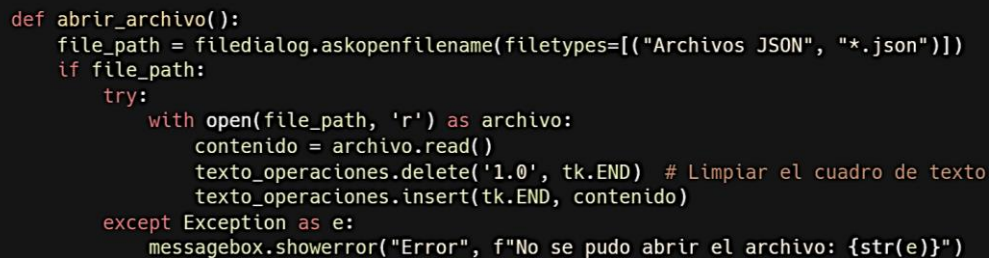
En esta sección, se definen las variables que se utilizarán a lo largo de la aplicación. Estas variables incluyen entrada_error para la entrada de errores, y errores_ingresados para almacenar los errores detectados.

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The code defines two variables: 'entrada_error' set to 'None' and 'errores_ingresados' set to an empty list '[]'.

```
entrada_error = None
errores_ingresados = []
```

2. Funciones para el menú archivo.

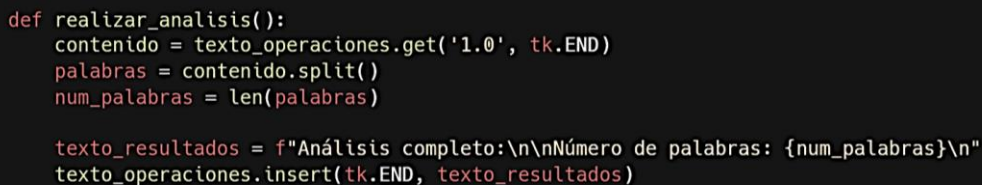
En esta sección, se definen varias funciones que manejan las operaciones relacionadas con archivos, como abrir, guardar y guardar como. Estas funciones utilizan el módulo filedialog de tkinter para interactuar con el sistema de archivos del usuario.

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The code defines a function 'abrir_archivo()' that uses 'filedialog.askopenfilename()' to select a file, opens it, reads its content, and inserts it into a text widget. It also includes an exception handler for file opening errors.

```
def abrir_archivo():
    file_path = filedialog.askopenfilename(filetypes=[("Archivos JSON", "*.json")])
    if file_path:
        try:
            with open(file_path, 'r') as archivo:
                contenido = archivo.read()
                texto_operaciones.delete('1.0', tk.END) # Limpiar el cuadro de texto
                texto_operaciones.insert(tk.END, contenido)
        except Exception as e:
            messagebox.showerror("Error", f"No se pudo abrir el archivo: {str(e)}")
```

3. Función para el análisis.

La función realizar_analisis() se encarga de realizar un análisis de palabras en el texto ingresado por el usuario y muestra el número de palabras en el cuadro de texto.

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The code defines a function 'realizar_analisis()' that retrieves text from a text widget, splits it into words, counts them, and displays the result in the text widget.

```
def realizar_analisis():
    contenido = texto_operaciones.get('1.0', tk.END)
    palabras = contenido.split()
    num_palabras = len(palabras)

    texto_resultados = f"Análisis completo:\n\nNúmero de palabras: {num_palabras}\n"
    texto_operaciones.insert(tk.END, texto_resultados)
```

4. Función para agregar y analizar errores.

Estas secciones incluyen funciones para agregar errores al texto y analizar el contenido en busca de errores específicos. Los errores detectados se almacenan en una lista y se proporciona la opción de guardar los resultados en un archivo JSON.

```
def mostrar_errores():
    error = entrada_error.get()
    if error:
        errores_ingresados.append(error)
        texto_operaciones.insert(tk.END, f"Error agregado: {error}\n")
        entrada_error.delete(0, tk.END)
```

5. Función para generar reporte.

La función generar_reporte1() crea un informe visual de las operaciones realizadas utilizando la biblioteca graphviz. El informe se genera como un archivo PNG y se muestra al usuario.

```
def generar_reporte1():
    contenido = texto_operaciones.get('1.0', tk.END)

    lineas = contenido.split('\n')

    # Crear un objeto de grafo de Graphviz
    dot = graphviz.Digraph(format='png', engine='dot', graph_attr={'rankdir': 'TB'}, node_attr={
        'shape': 'box', 'style': 'filled', 'fillcolor': 'lightblue1', 'fontname': 'Helvetica'})

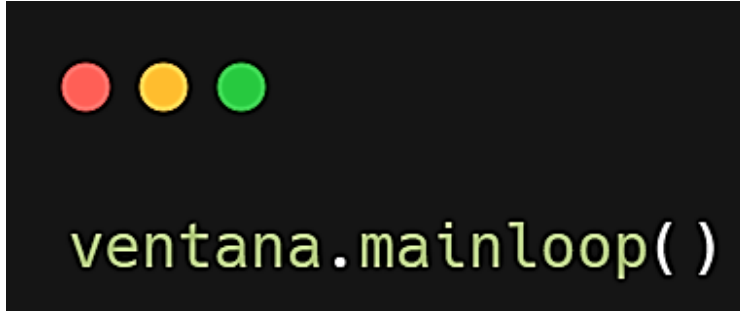
    for linea in lineas:
        partes = linea.split(':')
        if len(partes) == 2:
            operacion = partes[0].strip()
            resultado = partes[1].strip()
            dot.node(operacion, operacion, color='lightblue2')
            dot.node(resultado, resultado, color='lightblue2')
            dot.edge(operacion, resultado, color='lightblue3')

    dot.render('grafo_operaciones')

    import os
    os.system('start grafo_operaciones.png')
    texto_operaciones.insert(tk.END, "Generando reporte...\n")
```

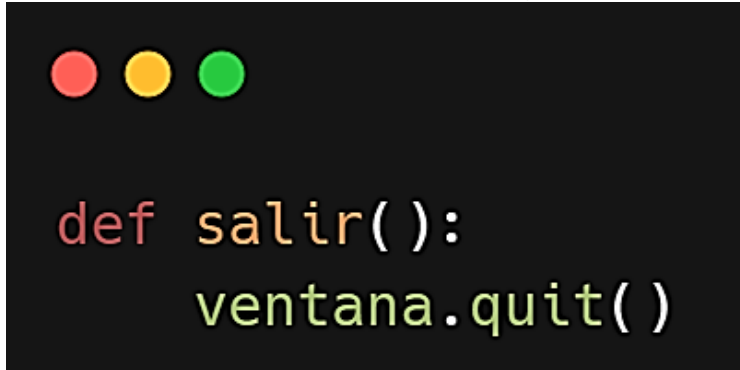
6. Función creación de ventana.

En esta sección, se crea la ventana principal de la aplicación y se define un menú con opciones relacionadas con archivos, análisis y generación de informes.



7. Función salir

Esta sección ayuda a finalizar el programa y cerrar la ventana del menú.



IMPORTANCIA DE LIBRERIAS.

1. Este import te permite utilizar la biblioteca tkinter, que es una biblioteca gráfica estándar de Python utilizada para crear interfaces gráficas de usuario (GUI). La importación con `as tk` es una convención común para abreviar el nombre de la biblioteca, facilitando su uso en el código.



2. Esta línea importa dos módulos específicos de tkinter: `filedialog` y `messagebox`.
`filedialog` proporciona ventanas de diálogo para seleccionar archivos, guardar archivos y navegar por el sistema de archivos del usuario. `messagebox` se utiliza para mostrar cuadros de diálogo de mensaje, como cuadros de error, información o confirmación, en la interfaz gráfica.

A screenshot of a code editor with a dark background. At the top left, there are three colored circles (red, yellow, green) representing window control buttons. The code `from tkinter import filedialog, messagebox` is displayed in a light green monospace font.

```
from tkinter import filedialog, messagebox
```

3. Este importa el módulo `re`, que es la biblioteca de expresiones regulares de Python. Las expresiones regulares son patrones utilizados para buscar y manipular cadenas de texto de manera eficiente. En tu código, se usa para buscar patrones específicos en el texto.

A screenshot of a code editor with a dark background. At the top left, there are three colored circles (red, yellow, green) representing window control buttons. The code `import re` is displayed in a light green monospace font.

```
import re
```

4. Esta importación permite utilizar el módulo `json`, que se utiliza para trabajar con datos en formato JSON (JavaScript Object Notation). JSON es un formato de intercambio de datos común en la programación web y se utiliza para serializar y deserializar datos estructurados.

A screenshot of a code editor with a dark background. At the top left, there are three colored circles (red, yellow, green) representing window control buttons. The code `import json` is displayed in a light green monospace font.

```
import json
```

5. Esta línea importa la biblioteca `graphviz`, que es una biblioteca de Python para generar gráficos y diagramas en formato DOT. El formato DOT es un lenguaje de descripción de gráficos que se utiliza comúnmente para representar relaciones entre elementos en un formato

legible por máquina. Graphviz se usa para crear y renderizar gráficos basados en datos estructurados, lo que es especialmente útil para representar relaciones entre operaciones y resultados en tu aplicación.



CONCLUSION

Este manual técnico busca dotar al lector de una comprensión profunda de la aplicación, su estructura, y su funcionalidad subyacente. Cada sección se encuentra diseñada para brindar una visión clara de cómo se han implementado las diversas funcionalidades y cómo pueden ser modificadas o ampliadas según las necesidades del proyecto.

El desarrollo de esta aplicación en Python con Tkinter es un testimonio de la versatilidad y potencia de esta combinación de herramientas para crear interfaces gráficas de usuario y procesar datos de manera eficiente. Por lo tanto, este manual pretende servir tanto como guía para la comprensión del código fuente como fuente de inspiración para futuros proyectos de desarrollo de aplicaciones similares.