**UNIVERSIDAD PRIVADA DOMINGO SAVIO**

# FACULTAD DE INGENIERÍA



**PROYECTO FORMATIVO MATERIA:** INTELIGENCIA ARTIFICIAL

**DOCENTE:** Ing. Jaime Zambrana Chacon

**INTEGRANTES:**

* **Jesus Gabriel Justiniano Enriquez**
* **Wilmer Michel Condori**
* **Alejandro López Molina**
* **Esnaider juan Aguilera**
* **Sergio Garafulic Baldiviezo**
* **Edward alexander viruez roca**

# Extractor y Resolvedor de Ecuaciones Introducción

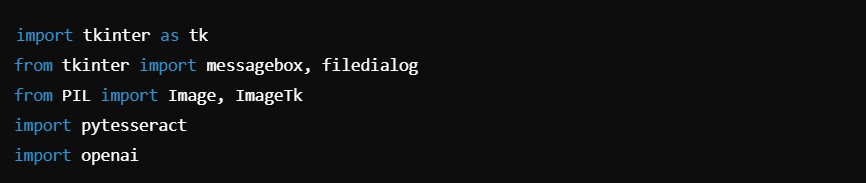
Este proyecto tiene como objetivo desarrollar una aplicación que permite a los usuarios cargar imágenes que contengan ecuaciones matemáticas, extraer el texto de estas imágenes mediante técnicas de reconocimiento óptico de caracteres (OCR), y resolver las ecuaciones utilizando la API de OpenAI. La aplicación está diseñada con una interfaz gráfica utilizando Tkinter, facilitando su uso y acceso.

Objetivos

1. **Cargar una imagen de ecuación**: Permitir al usuario seleccionar una imagen desde su sistema.
2. **Extraer texto de la imagen**: Usar Tesseract para convertir la imagen en texto legible.
3. **Resolver la ecuación**: Utilizar OpenAI para obtener la solución de la ecuación extraída y mostrarla al usuario.

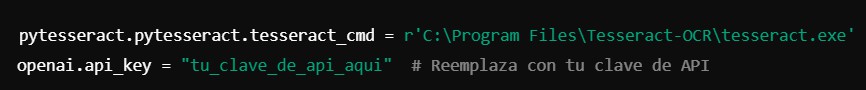
# Descripción del Código Importaciones

Se importan las librerías necesarias para el funcionamiento del proyecto:



# Configuración de Tesseract y OpenAI

La ruta de Tesseract se establece para permitir el acceso al ejecutable, y se proporciona la clave de API de OpenAI para utilizar sus servicios:



**LIBRERIAS UTILIZADAS DENTRO DEL PROGRAMA:**

**1. Tkinter**

Tkinter es la biblioteca estándar de Python para la creación de interfaces gráficas de usuario (GUI). Es una herramienta simple y potente que permite a los desarrolladores construir ventanas, botones, cuadros de texto y otros elementos interactivos para aplicaciones de escritorio. Algunas características clave incluyen:

Facilidad de uso: Tkinter proporciona una forma intuitiva de crear GUIs con un código mínimo.

Compatibilidad multiplataforma: Las aplicaciones creadas con Tkinter pueden ejecutarse en Windows, macOS y Linux sin modificaciones.

Gestión de layouts: Incluye métodos como pack(), grid(), y place() para organizar los widgets dentro de las ventanas.

Soporte para eventos: Tkinter permite la interacción del usuario mediante eventos, como hacer clic en botones o ingresar texto, gestionados a través de "handlers" o controladores.

En el proyecto, Tkinter se utilizó para crear la interfaz principal, permitiendo al usuario cargar imágenes, mostrar resultados y realizar otras interacciones visuales.

**2. Pillow**

Pillow es una biblioteca poderosa y flexible para la manipulación de imágenes en Python. Es una bifurcación de la antigua biblioteca PIL (Python Imaging Library), con muchas más funcionalidades y actualizaciones. Proporciona una amplia gama de herramientas para abrir, modificar y guardar imágenes en diferentes formatos.

Compatibilidad con múltiples formatos: Pillow permite trabajar con imágenes en formatos como PNG, JPEG, BMP, GIF, entre otros.

Manipulación de imágenes: Facilita operaciones como redimensionar, rotar, cortar y aplicar filtros a imágenes.

Conversión: Soporta la conversión entre diferentes modos de color, como RGB, CMYK y escalas de grises.

Facilidad de integración: Se integra perfectamente con otras bibliotecas y frameworks de Python.

En el proyecto, Pillow fue utilizado para cargar y redimensionar imágenes antes de procesarlas con OCR, lo que optimizó su presentación y manejo dentro de la aplicación.

**3. pytesseract**

pytesseract es una biblioteca de Python que actúa como un wrapper para Tesseract OCR, un motor de reconocimiento óptico de caracteres (OCR) desarrollado por Google. Permite la extracción de texto de imágenes y es compatible con varios formatos de archivo.

OCR potente: Tesseract es conocido por su alta precisión en la extracción de texto, especialmente cuando se combina con imágenes de buena calidad.

Soporte para múltiples idiomas: Puede extraer texto en varios idiomas mediante el uso de paquetes de entrenamiento adicionales.

Integración sencilla: La biblioteca pytesseract permite el uso de Tesseract OCR directamente desde scripts de Python con una sintaxis sencilla.

En este proyecto, pytesseract fue utilizado para extraer texto de imágenes cargadas por el usuario, interpretando las ecuaciones presentes en dichas imágenes.

**4. OpenAI**

OpenAI proporciona una API avanzada de inteligencia artificial que permite interactuar con modelos de lenguaje como GPT (Generative Pre-trained Transformer). Estos modelos pueden realizar tareas como generación de texto, traducción, y en este caso, resolver ecuaciones matemáticas.

Modelos avanzados: Los modelos como GPT-3 y GPT-4 son capaces de comprender y generar texto de manera precisa y fluida en diferentes contextos.

Fácil integración: La API de OpenAI puede integrarse fácilmente en aplicaciones Python mediante la biblioteca openai.

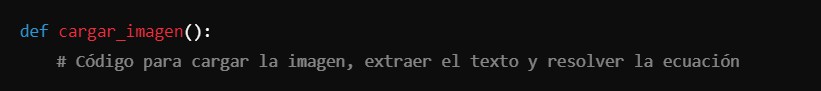
Amplias capacidades: OpenAI permite resolver ecuaciones, responder preguntas, y generar respuestas detalladas basadas en entradas de texto proporcionadas.

En este proyecto, OpenAI fue utilizado para resolver las ecuaciones extraídas de las imágenes, proporcionando una solución inteligente y rápida para los usuarios.

# Función cargar\_imagen()

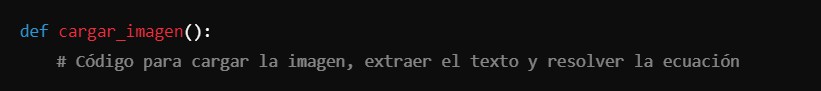
Esta función permite al usuario seleccionar una imagen y realiza las siguientes tareas:

* Carga y muestra la imagen seleccionada en el canvas.
* Extrae el texto de la imagen utilizando Tesseract.
* Muestra la ecuación extraída al usuario.
* Llama a la función resolver\_ecuacion para obtener la solución de la ecuación.



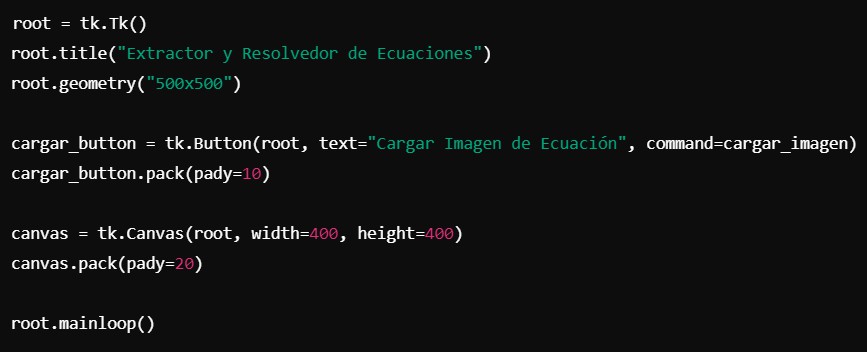
# Función resolver\_ecuacion(ecuacion)

Esta función utiliza la API de OpenAI para resolver la ecuación extraída y presenta la solución al usuario. Se maneja cualquier excepción que pueda surgir durante el proceso:



# Configuración de la Interfaz Gráfica

Se establece la ventana principal de la aplicación y se configuran los elementos de la interfaz, incluyendo un botón para cargar la imagen y un canvas para mostrarla:



**Participación y Colaboración**

Durante el desarrollo del proyecto, se promovió una discusión activa y constante entre todos los miembros del equipo, asegurando una colaboración sólida y un flujo continuo de ideas. Cada integrante tuvo la oportunidad de participar no solo en la codificación, sino también en la toma de decisiones técnicas, lo que contribuyó a un sentido compartido de responsabilidad y éxito del proyecto. En las sesiones de trabajo colaborativo, se revisaron y optimizaron distintas secciones del código, promoviendo la mejora continua en cada etapa del desarrollo.

La integración de la API de OpenAI resultó fundamental para el éxito del proyecto. A lo largo del proceso, todos los miembros tuvieron la oportunidad de interactuar directamente con la API, experimentando de primera mano sus capacidades para resolver ecuaciones complejas. Esta experiencia no solo enriqueció el conocimiento del equipo sobre el uso de inteligencia artificial en aplicaciones reales, sino que también fomentó una mayor confianza en el uso de herramientas avanzadas. El equipo se dividió tareas estratégicamente, lo que permitió que algunos miembros se enfocaran en la integración de OpenAI, mientras que otros trabajaban en el procesamiento de imágenes utilizando **Pillow** y **Tesseract** para la extracción de texto.

Se generó un ambiente donde las sugerencias fueron valoradas y aplicadas, lo que facilitó la resolución de problemas y el ajuste de detalles técnicos. Además, se incentivó el intercambio de conocimientos técnicos entre los miembros, mejorando el aprendizaje mutuo y fortaleciendo la cohesión del equipo.

**Resultados**

La aplicación final cumplió con los objetivos planteados, demostrando una integración eficaz entre el reconocimiento óptico de caracteres (OCR) y la inteligencia artificial. Se logró cargar imágenes de ecuaciones matemáticas, extraer texto con alta precisión a través de Tesseract OCR y resolver las ecuaciones utilizando la API de OpenAI. Las respuestas proporcionadas por la IA fueron precisas y se mostraron de manera clara y concisa al usuario en la interfaz gráfica, lo que facilitó una experiencia de usuario satisfactoria.

El éxito del proyecto no solo reside en la correcta ejecución de las funcionalidades, sino también en la forma en que el equipo colaboró para superar los desafíos técnicos. La aplicación fue probada extensamente y cumplió con los requisitos de funcionalidad y eficiencia. Cada miembro del equipo contribuyó de manera significativa, asegurando que la aplicación no solo funcionara correctamente, sino que también proporcionara una base sólida para futuras mejoras o expansiones.

# Conclusiones

El proyecto ha demostrado ser exitoso en la creación de una herramienta útil para la extracción y resolución de ecuaciones matemáticas a partir de imágenes. La implementación de Tesseract y OpenAI ha permitido integrar eficazmente el OCR y la resolución matemática en una interfaz gráfica amigable.

Se recomienda continuar con el desarrollo de la aplicación para incluir características adicionales, como la capacidad de resolver diferentes tipos de ecuaciones y mejorar el proceso de limpieza del texto extraído. También sería beneficioso asegurar la privacidad y seguridad al manejar la clave de API de OpenAI.

# Código Fuente

A continuación, se incluye el código fuente utilizado durante el proyecto: import tkinter as tk

from tkinter import messagebox, filedialog from PIL import Image, ImageTk

import pytesseract import openai

# Configura la ruta de Tesseract

pytesseract.pytesseract.tesseract\_cmd = r'C:\Program Files\Tesseract-

OCR\tesseract.exe' # Cambia la ruta si es necesario

# Proporciona tu clave de API de OpenAI

openai.api\_key = "tu\_clave\_de\_api\_aqui" # Reemplaza con tu clave de API

# Función para cargar la imagen y extraer la ecuación def cargar\_imagen():

# Abre un cuadro de diálogo para seleccionar una imagen

file\_path = filedialog.askopenfilename(title="Selecciona una imagen", filetypes=[("Archivos de imagen", ".png;.jpg;.jpeg;.bmp")])

if file\_path: try:

# Carga la imagen

img = Image.open(file\_path)

img.thumbnail((400, 400)) # Ajusta el tamaño para mostrarla img\_tk = ImageTk.PhotoImage(img)

# Muestra la imagen en el canvas canvas.create\_image(200, 200, image=img\_tk)

canvas.image = img\_tk # Mantiene una referencia a la imagen

# Extrae el texto de la imagen (la ecuación) texto = pytesseract.image\_to\_string(img)

messagebox.showinfo("Ecuación extraída", texto) # Muestra la ecuación extraída

# Resuelve la ecuación usando OpenAI resolver\_ecuacion(texto.strip()) # Elimina espacios en blanco

except Exception as e:

messagebox.showerror("Error", f"Error al procesar la imagen: {str(e)}")

# Función para resolver la ecuación con OpenAI def resolver\_ecuacion(ecuacion):

try:

response = openai.ChatCompletion.create( model="gpt-3.5-turbo", # O el modelo que prefieras messages=[

{"role": "user", "content": f"Resuelve la siguiente ecuación:

{ecuacion}"}

]

)

respuesta = response.choices[0].message['content'] messagebox.showinfo("Solución", respuesta) # Muestra la solución

except Exception as e:

messagebox.showerror("Error", f"Error al resolver la ecuación: {str(e)}")

# Configuración de la interfaz gráfica root = tk.Tk()

root.title("Extractor y Resolvedor de Ecuaciones") root.geometry("500x500")

# Botón para cargar la imagen

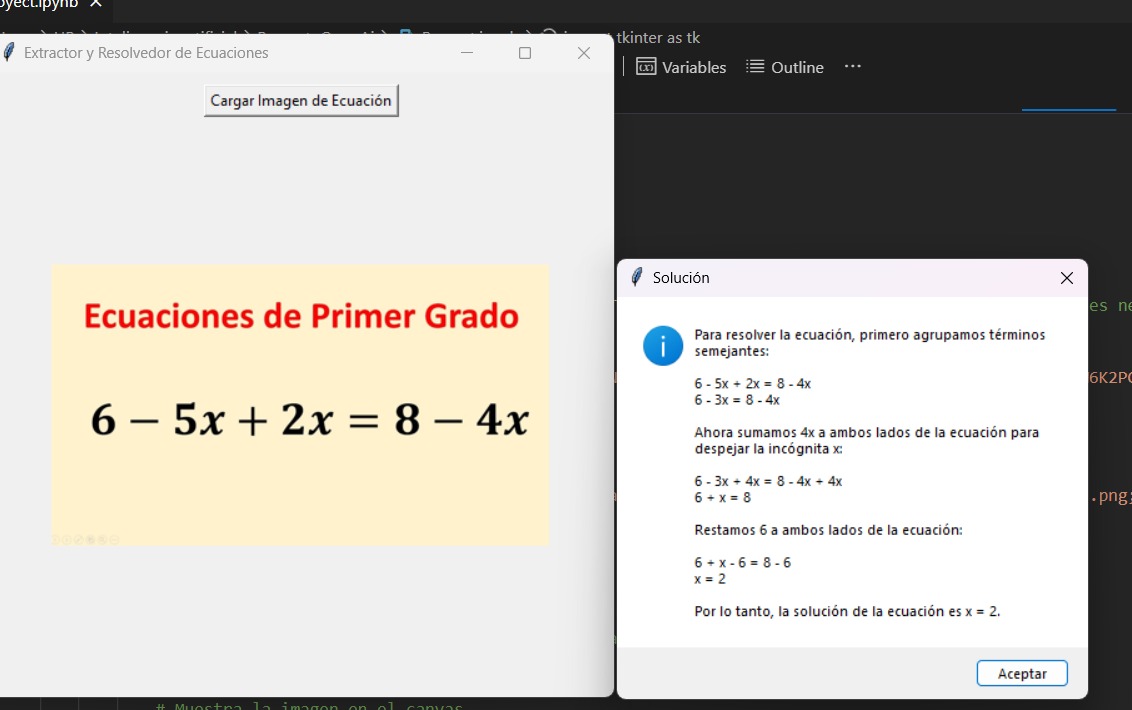
cargar\_button = tk.Button(root, text="Cargar Imagen de Ecuación", command=cargar\_imagen)

cargar\_button.pack(pady=10)

# Canvas para mostrar la imagen cargada canvas = tk.Canvas(root, width=400, height=400) canvas.pack(pady=20)

# Iniciar la interfaz gráfica root.mainloop()

**IMAGENES DEL PROGRAMA CORRIENDO…**



**Imágenes del código:**

