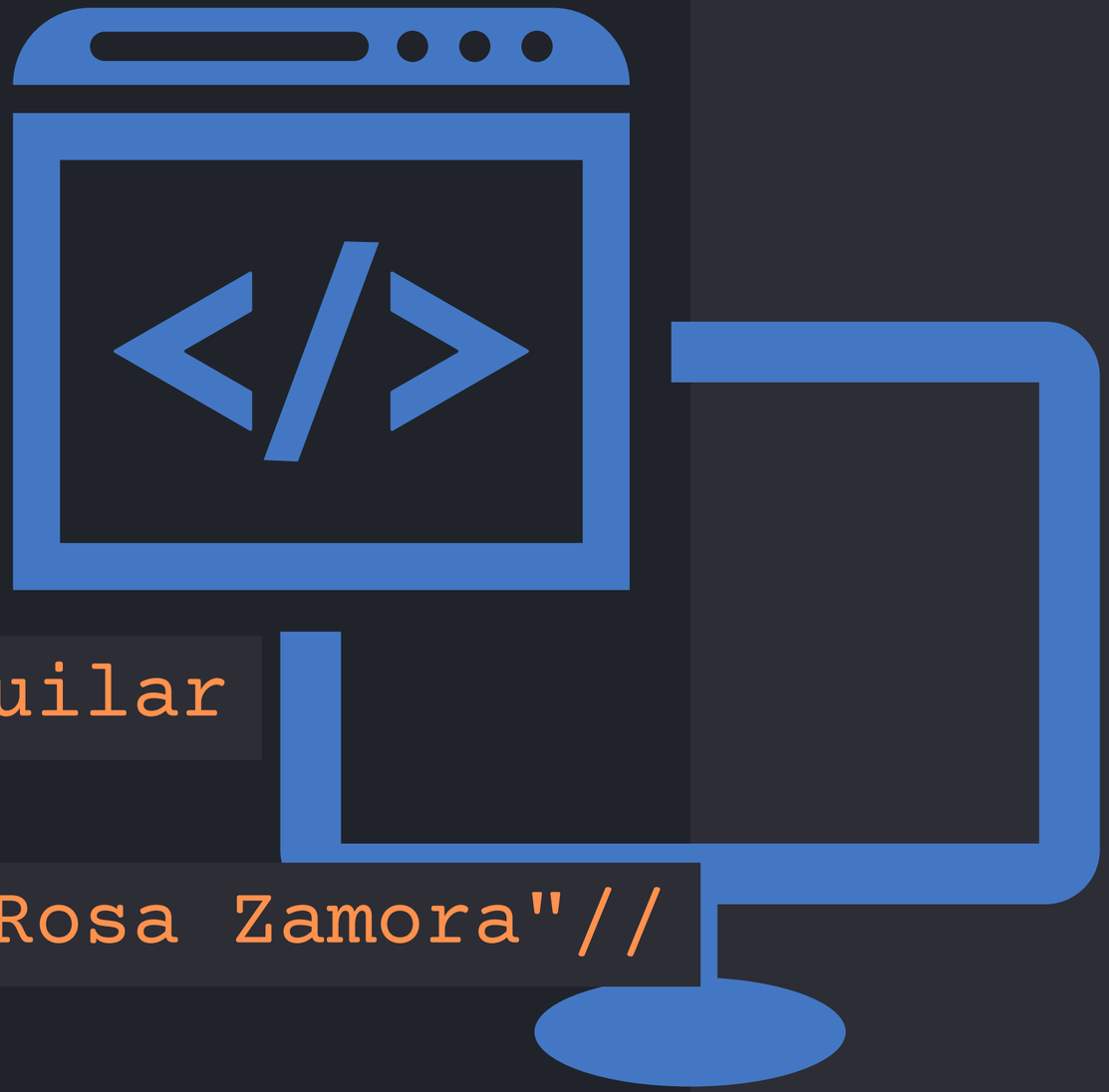
<!--Estudio Shonos-->

{OCRMath

<Por="Fátima Nivreh Aguilar Tomás//A00344922"/>
<"Miguel Alonso De La Rosa Zamora"//A01646106>
}



Problemática{

- En entornos educativos y profesionales, las operaciones matemáticas escritas a mano en papel o pizarras son difíciles de digitalizar y evaluar rápidamente.

- La transcripción manual a formato digital es un proceso tedioso y propenso a errores.

- Existe una necesidad de automatizar la extracción y evaluación de estas operaciones para agilizar procesos, especialmente en educación, donde se manejan grandes volúmenes de cálculo.

- La solución emplea tecnologías de OCR y procesamiento de imágenes para transformar imágenes de operaciones matemáticas en texto digital, permitiendo una evaluación automática, lo que ahorra tiempo y minimiza errores.

- Automatizar este proceso mejora la eficiencia y facilita la integración de datos matemáticos en sistemas digitales, optimizando el flujo de trabajo tanto educativo como profesional.

}

Implementación {

-OpenCV: gestiona la carga de imágenes con operaciones matemáticas, preparándolas para el procesamiento.

-Pytesseract:convierte las imágenes en texto mediante OCR, permitiendo la detección de operaciones escritas a mano.

- Utilizamos expresiones regulares para limpiar y validar las operaciones extraídas, asegurándonos de que sean aptas para evaluación.

- Las operaciones se evalúan en Python para obtener los resultados rápidamente y sin intervención manual.

- El código y el control de versiones del proyecto se gestionan en GitHub, facilitando la colaboración y el desarrollo continuo.

Esta solución automatiza la extracción y evaluación de operaciones matemáticas, optimizando procesos académicos y profesionales que utilizan datos escritos a mano.

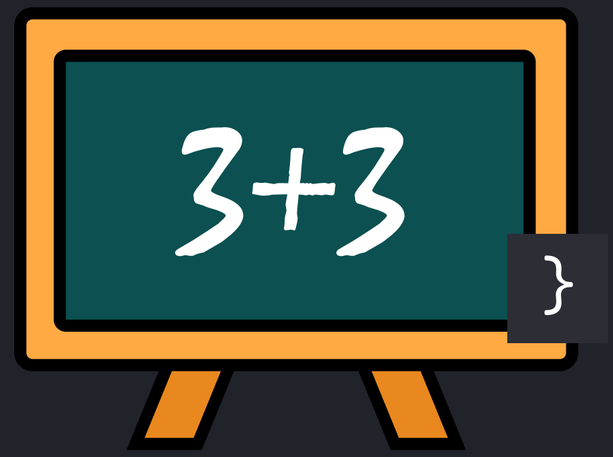
}

Filtros {

OpenCV: Carga y manipulación de imágenes.

Pytesseract: Extracción de texto de las imágenes (Optical Character Recognition OCR).

Regular Expressions (re): Filtrado de caracteres no deseados en operaciones detectadas.



}

Archivos de entrada {

Imágenes de operaciones matemáticas, por ejemplo,

"Dos_operacions.jpg".

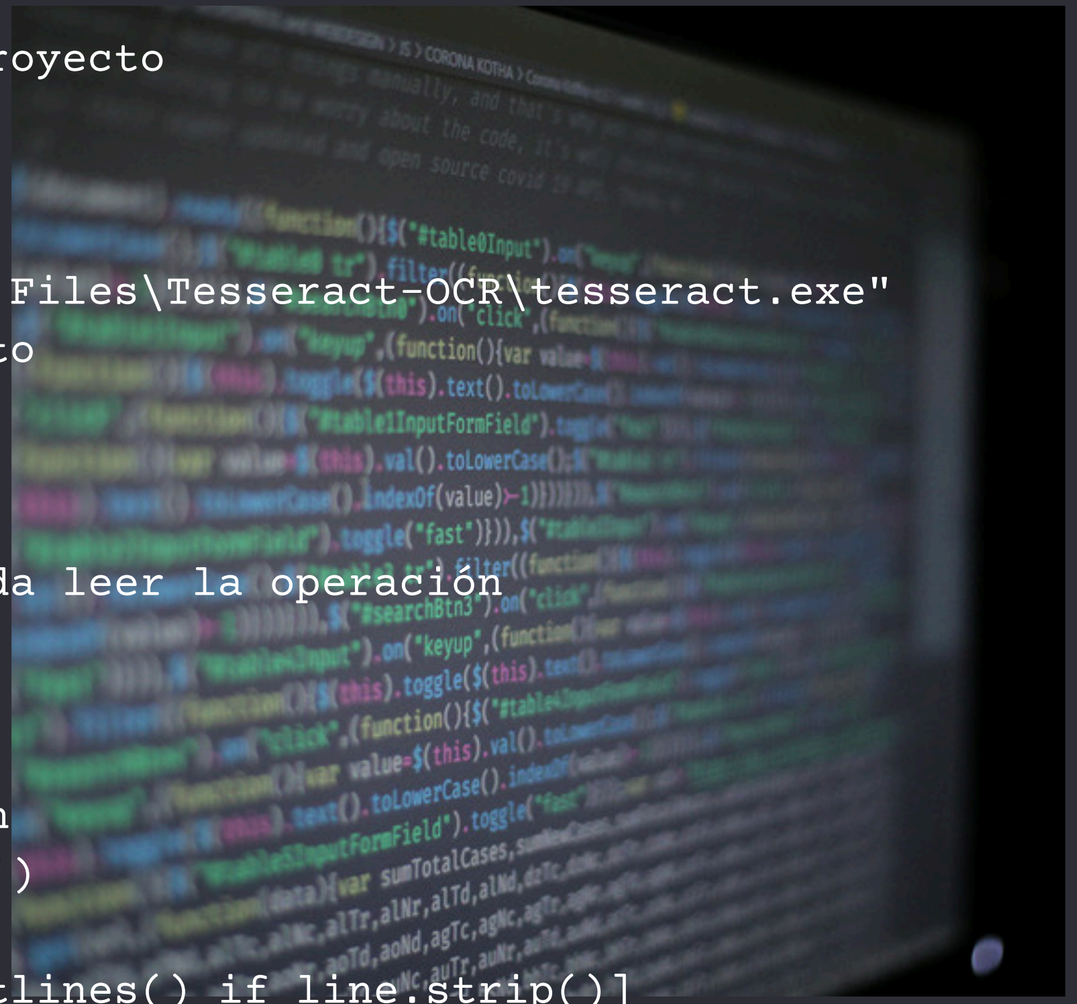
$$X = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Archivos de Salida:
Lista de operaciones con sus resultados evaluados, que se despliega en la consola.

}

El Código {

```
# Importamos las librerías necesarias para nuestro proyecto
import cv2
import pytesseract
import re
pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"
# Definimos la función con la que extraeremos el texto
def extraccion_texto(img):
  # Leemos la imagen
  img = cv2.imread(img)
  # Agregamos una condición en caso de que no se pueda leer la operación
  if img is None:
    print(f"No se pudo cargar la imagen: {img}")
    return ""
  # Usamos tesseract para detectar texto en la imagen
  texto = pytesseract.image_to_string(img, lang='eng')
  # Separa cada línea (operación)
  operaciones = [line.strip() for line in texto.splitlines() if line.strip()]
  return operaciones
```



}

El Código {

```
# Definimos la función con la realizaremos la operacion
def evaluar(operaciones):
    # Variable para guardar los resultados
    resultados = []
    # Ciclo for para evaluar cada operacion
    for op in operaciones:
        # Usamos la libreria re para que en la cadena de string extraiga
los operadores y numeros
        extraer_elementos_operacion = re.sub(r'[^0-9]\+\-\*/\(\) ]', '',
op)
        try:
            # evaluamos la operacion
            resultado = eval(extraer_elementos_operacion)
            # La guardamos en resultados con la operacion correspondiente
            resultados.append((extraer_elementos_operacion, resultado))
        except Exception as e:
            # Un excepcion por si da error y no puede evaluar la operacion
            resultados.append((extraer_elementos_operacion, f"Error:
{e}"))
    return resultados

operacion = extraccion_texto("Dos_operacions.jpg")
resultado = evaluar(operacion)
print(resultado)
```

((Se inicia una lista vacía resultados donde se almacenarán los resultados de cada operación.))

Se utiliza un bucle for para recorrer cada operación en la lista operaciones, que incluye operaciones matemáticas en formato de texto.

La función re.sub() de la librería de expresiones regulares re limpia cada operación.
Esta línea elimina cualquier carácter que no sea un dígito del 0 al 9, operadores matemáticos (+, -, *, /), paréntesis, o espacios. Así, nos aseguramos de que la cadena solo contenga elementos que forman una expresión matemática válida.

Dentro de un bloque try, se utiliza eval() para evaluar la operación limpia. eval() ejecuta la cadena como una expresión matemática de Python.

Si eval() tiene éxito y no lanza un error, el resultado de la operación junto con la operación limpia se añaden a la lista resultados.

Si eval() lanza una excepción (es decir, un error), el bloque except captura este error.
En tal caso, se agrega un mensaje a resultados que indica que hubo un error al evaluar la operación junto con el mensaje del error.

Después de evaluar todas las operaciones, la función retorna la lista resultados, que contiene pares de cada expresión matemática y su resultado (o el mensaje de error, si la evaluación falló).

}

Conclusión {

$$ax^2 + bx + c = 0$$

La solución que desarrollamos permite extraer texto y evaluar operaciones matemáticas de imágenes de manera automática, lo que sustituye procesos manuales que suelen ser laboriosos. Esta herramienta es especialmente útil para quienes manejan grandes cantidades de datos matemáticos escritos a mano.
Es importante mencionar que la precisión del reconocimiento del texto (OCR) depende de la calidad de las imágenes. Si las imágenes son claras y bien definidas, los resultados serán mejores. El preprocesamiento de imágenes, como ajustar el brillo o el contraste, puede ayudar a obtener mejores resultados.
Además, al evaluar las operaciones con eval(), debemos ser cautelosos; es fundamental garantizar que las entradas sean seguras y controladas, ya que eval() puede ejecutar cualquier código si no se gestiona adecuadamente.

}

<!--Estudio Shonos-->

Gracias {

<Print=”Hola mundo"/>

}