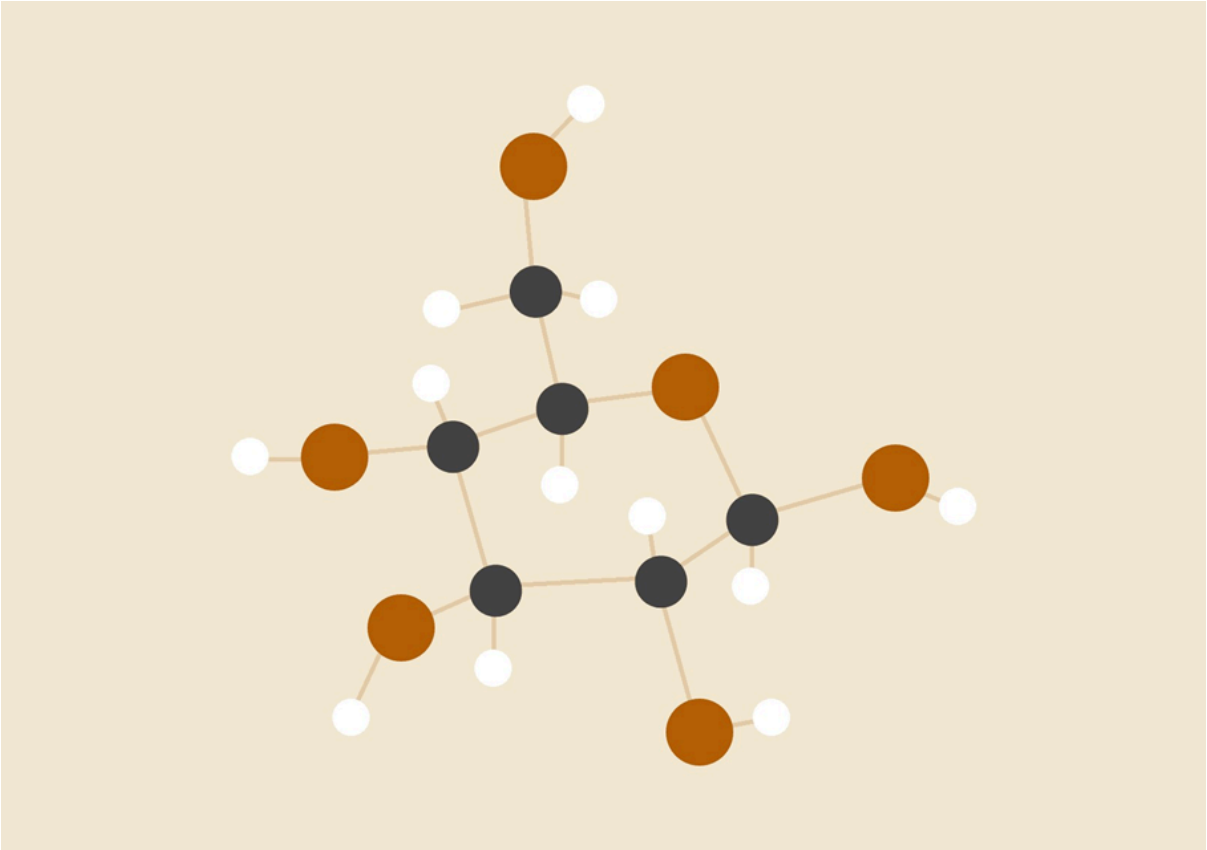
TRABAJO PRÁCTICO 1

IA4.4 Procesamiento Digital de Imágenes
Tecnicatura Universitaria en Inteligencia Artificial



Alumnos:
Arce, Sofía
Gauto, Lucas
Rizzotto, Camila

30/04/2023
Universidad Nacional de Rosario
Facultad de Ciencias Exactas, Ingeniería y Agrimensura

INTRODUCCIÓN

Nuestro trabajo se basa en la resolución de dos ejercicios: el primero, consta de desarrollar una función para implementar la ecualización local de un histograma, que reciba como parámetros de entrada una imágen a procesar, y el tamaño de una ventana de procesamiento (M x N). El segundo, se trata de generar un script que corrija exámenes de tipo múltiple choice de forma automática, además de verificar los datos personales de cada alumno.

DESCRIPCIÓN DEL ENTORNO DE TRABAJO

Para la realización de estos ejercicios utilizamos el lenguaje Python y generamos un entorno virtual en el cual instalamos y utilizamos los siguientes paquetes:

- Numpy
- Open CV
- Matplotlib

PROCEDIMIENTO EJERCICIO 1

El ejercicio número uno pedía la elaboración de una función que implemente una ecualización local de histograma a cierta imagen dada para poder revelar las figuras que llevaba ocultas.
Se ideó así una función que, dada una imagen y un tamaño de ventana, devuelve la ecualización local del histograma de la misma. Para ello el algoritmo recorre la imagen de entrada píxel por píxel calculando la ventana correspondiente a cada uno de ellos y posteriormente obteniendo la ecualización del histograma de dicha ventana para luego cargarla en una imagen en blanco previamente creada. De esta forma cada imagen ecualizada se carga sobre la imagen en blanco y al final de la iteración se obtiene la imagen completa.
La función lleva el nombre de `ecualizacionLocal`.

PROCEDIMIENTO EJERCICIO 2

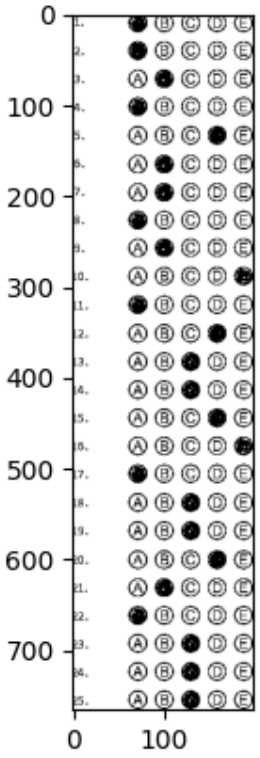
Parte a

‘’Debe tomar únicamente como entrada la imagen de un examen y mostrar por pantalla cuáles de las respuestas son correctas y cuáles incorrectas.”

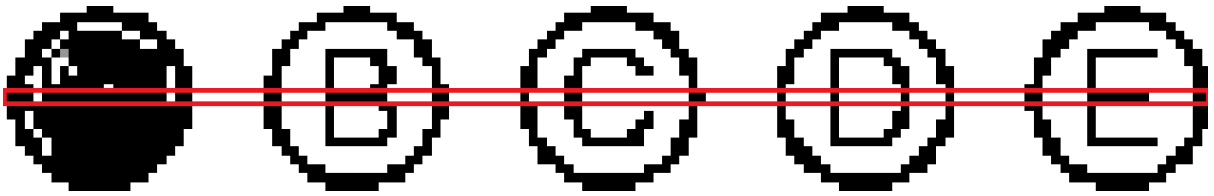
Para resolver el punto a del ejercicio número dos tuvimos que dividir el problema en dos funciones distintas:

- `adecuar`
- `corregir`

Como su nombre lo sugiere, la función `adecuar` toma la ruta de una imagen y la prepara para ser tratada posteriormente (esto es, obtiene sólo el recorte donde están las respuestas del exámen). El resultado de la función es el siguiente:



Posteriormente mediante la función `corregir` se determina si la respuesta estuvo correcta o no. Para analizar esto, se toma de cada renglón el píxel del medio y para cada letra se realiza la suma de sus píxeles interiores. Aquella letra que posea la menor suma será que posea más píxeles en negro (0), por lo tanto esta será la opción marcada. Esta información será guardada en un diccionario que luego será retornado como resultado de la función.



En rojo la selección de píxeles que se toman para realizar la suma de cada letra. Como se puede observar, en este caso la letra marcada es la A, la cual conserva una suma de aproximadamente 510, ya que todos sus otros valores son 0.

Parte b

‘’Con la misma imagen de entrada, validar los datos del encabezado y mostrar por pantalla el estado de cada campo teniendo en cuentas las siguientes restricciones:
i. Name: debe contener al menos dos palabras y no más de 25 caracteres.
ii. ID: 8 caracteres formando una sola palabra.
iii. Code: un único caracter.
iv. Date: 8 caracteres formando una sola palabra.
Asuma que todos los campos ocupan un solo renglón y que se utilizan caracteres alfanuméricos, guión medio “ - ” y barra inclinada “ / ”. ‘’

Para cumplir con este punto, desarrollamos un conjunto de funciones que procesan una imagen de entrada para validar los datos del encabezado del examen. A continuación, describimos cada función:

Extracción del renglón de datos del encabezado:

1. Primero creamos la función *‘’obtener_renglon_de_datos”* para extraer el renglón de datos del encabezado de un examen. Le pasamos el exámen y obtenemos otra imágen con dicho renglón. Utilizamos técnicas de Open CV como umbralado y detección de contornos para identificar el área del mismo y recortarlo.

Extracción de datos de los campos:

2. Luego desarrollamos *”obtener_datos_de_campos”* para extraer los crops de los campos individuales del encabezado del examen. Volvimos a usar umbralado y detección de contornos para identificar y recortar los campos individuales del renglón de datos. Se le pasa el encabezado a la función y nos devuelve un diccionario con los componentes del mismo, en donde cada clave es el campo y como valor los datos respectivos de cada alumno.

Conteo de caracteres:

3. La función *‘’contar_componentes”* cuenta los caracteres en cada campo del encabezado del examen y determina la cantidad de espacios entre ellos. Esto se logra detectando los componentes conectados en cada campo de la imagen con la función **cv2.connectedComponentsWithStats(),** tal y como nos fue aconsejado en la presentación del trabajo. Los resultados se almacenan en un diccionario para su posterior validación según las restricciones especificadas en la consigna.

Validación de los datos de los campos:

4. Implementamos una función *”validar_caracteres”* para validar los datos de los campos extraídos del encabezado del examen. La función verifica si los datos cumplen con las restricciones especificadas para cada campo (Nombre, ID, Código y Fecha) y muestra el estado de cada campo por pantalla.

Convergencia de funciones:

5. Por último, implementamos todas las anteriores en una función principal denominada ‘’*main”* a la que se le pasa una lista con todos los exámenes deseados para ser procesados según la consigna.

Parte c

“Utilice el algoritmo desarrollado para evaluar las imágenes de exámenes resueltos (archivos multiple_choice_x.png) e informe los resultados obtenidos.”

En el algoritmo ya desarrollado *‘’main”* agregamos ahora una lista ‘’*resultados”* para ser retornada al usuario y cumplir con lo requerido en este punto, informar los resultados obtenidos. La lista se compone de tuplas con el exámen que fue corregido y la cantidad de respuestas correctas que tuvo.

Parte d

‘’Generar una imagen de salida informando los alumnos que han aprobado el examen (con al menos 20 respuestas correctas) y aquellos alumnos que no. Esta imagen de salida debe tener los “crop” de los campos Name del encabezado de todos los exámenes del punto anterior y diferenciar de alguna manera aquellos que corresponden a un examen aprobado de uno desaprobado.”

En esta parte, lo que hicimos fue re-utilizar las funciones generadas en la parte b. Usamos *‘’obtener_renglon_de_datos” y ‘’obtener_datos_de_campos”* para realizar una nueva función llamada *’’obtener_campo_nombre”*. Luego, con *‘’main”* desarrollamos una función llamada *‘’generar_imagen_salida”* que toma una lista (el resultado de los exámenes) y guarda como output la imagen .png que nos fue pedida en este punto. Con numpy creamos una imágen blanca de base, y con Open CV logramos pegar sobre la imágen los campos pertinentes del exámen y un correspondiente texto que indica si están aprobados o no.

INSTRUCCIONES DE USO

Se debe contar con Python y algún IDE como Visual Studio Code o Pycharm.

Primero se debe clonar nuestro repositorio:

git clone https://github.com/LucasGauto/PDI.git

Luego, abrimos la carpeta en nuestro IDE, y en su consola creamos un entorno virtual:

python -m venv “directorio”

Activar entorno virtual ejecutando el script:

directorio/Scripts/activate.bat

Instalar paquetes:

pip install numpy
pip install matplotlib
pip install opencv-contrib-Python

Finalmente, vamos a **‘’main.py”** y ejecutamos las líneas pertinentes de cada ejercicio.

CÓDIGO EJERCICIO 1

```
def ecualizacionLocal(ruta,window_size = 25):
    '''
    Toma la ruta de una imagen y devuelve la misma imagen pero
ecualizada localmente.
    '''
    #Primero realizamos una carga y visualización de la imagen
    imagen = cv2.imread(ruta, cv2.IMREAD_GRAYSCALE)

    #Funcion para ecualizar localmente
    def local_histogram_equalization(img, window_size):
        '''
        Calcula la ecualizacion local de histograma
        '''
        h, w = img.shape #Toma el alto y ancho de la imagen
        img_equalized = np.zeros((h, w), dtype=np.uint8) #Se crea una
imagen base en blanco

        #Se toma la mitad de la imagen para calcular el desplazamiento
        #de la ventana desde el centro, donde el centro es i,j
        half_size = window_size // 2

        for i in range(h):
            for j in range(w):
                i_min = max(0, i - half_size) #Para que no tome una
fila menor a 0
                i_max = min(h, i + half_size + 1) #Lo mismo pero mayor
a 255
                j_min = max(0, j - half_size) #Lo mismo para las
columnas
                j_max = min(w, j + half_size + 1) #etc

                # Obtener la subimagen dentro de la ventana
                window = img[i_min:i_max, j_min:j_max]

                # Ecualizar el histograma localmente
                window_equalized = cv2.equalizeHist(window)

                # Asignar el valor ecualizado al píxel central de la
ventana en la imagen resultante
```

```
                img_equalized[i, j] = window_equalized[i - i_min, j -
j_min]

        return img_equalized

    #window_size = 25

    imagen_ecualizada_localmente = local_histogram_equalization(imagen,
window_size)

    #plt.imshow(imagen_ecualizada_localmente, cmap = 'gray')
    #plt.show(block=False)
    return imagen_ecualizada_localmente

######## Ejercicio 1 ##############
imagen = funciones.ecualizacionLocal(ruta =
'Imagen_con_detalles_escondidos.tif',window_size=25)
plt.imshow(imagen)
plt.show()
```

CÓDIGO EJERCICIO 2

Parte a

```
def adecuar(NombreImagen:str):
    '''
    Toma la imagen leida y extrae solamente la parte del examen
    '''
    img = cv2.imread(NombreImagen, cv2.IMREAD_GRAYSCALE)
    #---------- UMBRALADO ------------
    '''
    Las siguientes líneas llevan los píxeles menores de 150 a 0 y,
    los mayores, a 255, quedando una imagen solo con blancos y negros.
    '''
    img[img < 150] = 0 #umbralado negro
    img[img > 150] = 255 #Umbralado blanco

    #----- RECORTAR EL ENCABEZADO -----
    y = 140
    imgRecortada = img[y:]

    #------ SELECCIONAR SOLO LAS RESPUESTAS -------
```

```
    '''
    Razonamiento:

    Como la imagen tiene 916 pixeles en sus columnas
    podemos decir que una fila toda blanca tendra una suma
    de 255*816. Por lo tanto, si la suma de la fila da un valor
    menor a ese numero, podemos decir que ya hay algun valr igual
    a 0
    '''
    #Recortar filas
    def sumar_filas(matriz):
        '''
        Recibe un array de mas de una dimensión, recorre la dimensión
        inferior y realiza la suma de sus elementos.
        '''
        suma_filas = []
        for fila in matriz:
            suma = sum(fila)
            suma_filas.append(suma)
        return suma_filas

    sumaFilas = sumar_filas(imgRecortada) #lista de longitud = filas
matriz con la suma de cada fila

    cant_filas, cant_columnas = imgRecortada.shape

    contador = 1
    for i in sumaFilas:
        if i != 255*cant_columnas:
            #print(f'La fila n° {contador} es la primera con un valor
distinto de blanco')
            break
        contador += 1


    examenRecorte1 = imgRecortada[contador-1:]
    examenRecorte2 = examenRecorte1[:766] #766 es el alto del examen

    #Recortar columnas

    imgTraspuesta = examenRecorte2.T #trasponemo'

    sumaColumnas = sumar_filas(imgTraspuesta)
```

```
    cant_filas,cant_columnas = examenRecorte2.shape

    contador = 1
    for i in sumaColumnas:
        if i != 255*cant_filas:
            #print(f'La columna n° {contador} es la primera con un
valor distinto de blanco')
            break
        contador += 1

    imagenRecorte3 = examenRecorte2[:,contador-1:]
    ancho_examen = 197

    examen = imagenRecorte3[:,:ancho_examen] #Imagen sobre la que
trabajar

    return examen

def corregir(examen):

    #img = cv2.imread(examen, cv2.IMREAD_GRAYSCALE)
    img = examen

    respuestasCorrectas = {
        1: 'A',
        2: 'A',
        3: 'B',
        4: 'A',
        5: 'D',
        6: 'B',
        7: 'B',
        8: 'C',
        9: 'B',
        10: 'A',
        11: 'D',
        12: 'A',
        13: 'C',
        14: 'C',
        15: 'D',
        16: 'B',
        17: 'A',
        18: 'C',
```

```
        19: 'C',
        20: 'D',
        21: 'B',
        22: 'A',
        23: 'C',
        24: 'C',
        25: 'C'
    }

    separacion_entre_letras = 6
    ancho_de_las_letras = 23

    lista = []
    cantidad_preguntas = 25
    longitud_renglon = 20
    espacio_entre_renglones = 11

    for i in range(cantidad_preguntas):
        inicio = longitud_renglon * i + espacio_entre_renglones * i
        fin = inicio + 20
        lista.append(img[inicio:fin,:])

    '''
    Las preguntas ya estan separadas.
    Ahora hay que ver como comprobar cual fue la que se seleccionó.
    Para ello se nos ocurrió hacerlo por posición y sumando los
valores.
    Es evidente que una letra marcada tendrá una suma de sus valores
muy baja
    porque la mayoria de sus valores son = 0.
    Se nos ocurrió tomar la fila de en medio de cada respuesta, separar
por posicion
    y realizar la suma de sus filas. Aquella que posea una suma más
baja será la
    seleccionada.
    '''

    respuestasAlumno = {}

    contador = 1
    for respuesta in lista:
        #Seteo las posiciones para cada letra
        A = sum(respuesta[10:11,61:81][0])
```

```
        B = sum(respuesta[10:11,90:111][0])
        C = sum(respuesta[10:11,119:140][0])
        D = sum(respuesta[10:11,148:168][0])
        E = sum(respuesta[10:11,177:197][0])

        opciones = [A,B,C,D,E]

        menorValor = min(opciones)

        if menorValor == opciones[0]:
            respuestasAlumno[contador] = 'A'
            contador+=1
        elif menorValor == opciones[1]:
            respuestasAlumno[contador] = 'B'
            contador+=1
        elif menorValor == opciones[2]:
            respuestasAlumno[contador] = 'C'
            contador+=1
        elif menorValor == opciones[3]:
            respuestasAlumno[contador] = 'D'
            contador+=1
        else:
            respuestasAlumno[contador] = 'E'
            contador+=1

    #COMPARACION CON LAS RESPUESTAS CORRECTAS
    correccion = {}

    for i in range(1,25+1):
        if respuestasAlumno[i] == respuestasCorrectas[i]:
            correccion[f'Pregunta {i}'] = 'OK'
        else:
            correccion[f'Pregunta {i}'] = 'MAL'

    return correccion




######## Ejercicio 2 ################
#a)
examen = funciones.adecuar('multiple_choice_1.png')
print(funciones.corregir(examen))
```

Parte b

```
# Funcion para obtener el renlgon de datos del examen --> devuelve la
iamgen ya recortada
def obtener_renglon_de_datos(examen):
    """
    Devuelve la imagen del renglón de los campos a analizar
    """
    img = cv2.imread(examen, cv2.IMREAD_GRAYSCALE)
    umbral, umbralizada = cv2.threshold(img, 120, 255,
cv2.THRESH_BINARY)
    img_neg = umbralizada==0  #True -> blanco, False --> negro

    img_row_zeros = img_neg.any(axis=1)
    x = np.diff(img_row_zeros)
    renglones_indxs = np.argwhere(x)  # me devuelve donde empieza y
termina el renglon, me interesa la pos 2 y 3
    renglon_de_datos = [renglones_indxs[2], renglones_indxs[3]]
    # Genero imagen para pasar como argumento a la otra que analiza el
texto
    recorte_renglon =
img[renglon_de_datos[0][0]:renglon_de_datos[1][0], :]
    return recorte_renglon

def obtener_datos_de_campos(imagen):
    """
    Funcion que devuelve una lista con las imagenes de los campos
completados
    """
    campos = imagen

    _, umbral = cv2.threshold(campos, 220, 255, cv2.THRESH_BINARY)
    contornos, _ = cv2.findContours(umbral, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    campos = []

    for c in contornos:
        x, y, w, h = cv2.boundingRect(c)

        if w > 77:
```

```
          # chequeo que la posiscion de x no sea 569 porque coincide
con el ancho del campo de codigo
          if x == 569:
            continue

          campos.append((x, y, w, h))

    # Genero imagenes para pasar como argumento a la otra que analiza
los caracteres:
    indv_datos_del_examen=[]
    campos_a_retornar=imagen.copy()
    for x, y, w, h in campos:
      indv_datos_del_examen.append(campos_a_retornar[y+3:y+h-3,
x+3:x+w-3]) # Agrego los recortes de los campos, el +3, -3 para
descartar los borde

    return indv_datos_del_examen

def contar_componentes(campos):
    """
    Función que cuenta los caracteres de mi imagen
    """
    componentes={}
    con = 0

    for imagen in campos:
      ret, thresh = cv2.threshold(imagen, 127, 255, 0)

      #cv2 Componets detecta los blancos como porciones de componentes
--> hay que invertir los bits
      img = cv2.bitwise_not(thresh)
      output = cv2.connectedComponentsWithStats(img)
      caracteres = output[0]-1

      stats = output[2]
      sort_index = np.argsort(stats[:, 0])
      stats = stats[sort_index]

      # Descartar las componentes de ancho pequeño
      for i in range(len(stats)):
        if i >= 1:
          anchura = stats[i][2]
          if anchura <= 2:
```

```
            caracteres = caracteres -1

      espacios =  []
      for i in range(len(stats)):
        if i > 1: # para calcular la diferencia con el anterior
          val_espacio = stats[i][0]-(stats[i-1][0]) # calculo la
diferencia entre la cordenada x de mi componente siguiente y la
anterior
          if val_espacio > 9 and  i > 2: # > 2 Es para descartar el
vector de mi primer componente. Porque las masyusculas tienden a ser
mas anchas y no corresponden a espacios
            espacios.append(val_espacio)

      clave = f"campo_{con}"
      componentes[clave] = (caracteres, len(espacios))
      con = con + 1

    return componentes

def validar_caracteres(componentes):

  for val, keys in componentes.items():
    n_caracteres = keys[0]
    espacios = keys[1]

    if val == "campo_1":
       if n_caracteres == 1:
          print("CODE:OK")
       else:
          print("CODE: MAL")

    if val == "campo_2" or val == "campo_0":
       if n_caracteres == 8:
          if val == "campo_0":
            print("DATE:OK")
          else:
            print("ID:OK")
       else:
          if val == "campo_0":
            print("DATE:MAL")
          else:
            print("ID: MAL")
```

```
    if val == "campo_3":
       if n_caracteres > 1 and  n_caracteres <= 25 and espacios == 1:
          print("NAME:OK")
       else:
          print("NAME: MAL")
```

```
def main(multiple_choice):
  '''Función para retornar el nombre del exámen y el número de
respuestas correctas'''
  lista_de_examenes = multiple_choice
  resultados = []  # Lista para almacenar los resultados de los
exámenes
  ex_id = 0

  for examen in lista_de_examenes:
     print(f"Examen: {ex_id}-{examen}")
     renglon = obtener_renglon_de_datos(examen)
     #plt.figure(), plt.imshow(renglon, cmap='gray'),
plt.show(block=True)
     datos_de_los_campos = obtener_datos_de_campos(renglon)
     #plt.figure(), plt.imshow(datos_de_los_campos[0], cmap='gray'),
plt.show(block=True)
     componentes = contar_componentes(datos_de_los_campos)
     #print(componentes)
     validar_caracteres(componentes)
     # Calcular el número de respuestas correctas
     correccion_exam = corregir(adecuar(examen))
     respuestas_correctas = sum(1 for estado in
correccion_exam.values() if estado == 'OK')
     resultados.append((examen, respuestas_correctas))
     ex_id += 1

  return resultados


#b)
#Correccion de encabezados
m_choice=['multiple_choice_1.png','multiple_choice_2.png',
'multiple_choice_3.png', 'multiple_choice_4.png',
'multiple_choice_5.png']
resultados_examenes = funciones.main(m_choice)
print(resultados_examenes)
```

Parte c

```
#c)
#Correccion de cada examen
for imagen in m_choice:
    respuesta = funciones.corregir(funciones.adecuar(imagen))
    print(imagen)
    #print(respuesta)
    for punto in respuesta:
        print(punto, respuesta[punto])
    print('\n')
```

Parte d

```
def obtener_campo_nombre(examen):
    '''Función que devuelve los crop de los campos name'''
    renglon = obtener_renglon_de_datos(examen)
    # Como sé que el ultimo campo es el nombre, me quedo con ese
    campos_datos = obtener_datos_de_campos(renglon)
    name = campos_datos[3]
    #plt.figure(), plt.imshow(renglon, cmap='gray'),
plt.show(block=True)
    return name

def generar_imagen_salida(resultados):

    '''Función para generar la imágen de salida'''
    # Crear una imagen en blanco para la salida
    height = len(resultados) * 60
    width = 400
    output_image = np.ones((height, width, 3), np.uint8) * 255

    # Iterar sobre los resultados y generar los crops de los campos
Name
    y = 20
    for examen, respuestas_correctas in resultados:
        # Obtener el campo Name del examen
        campo_name = obtener_campo_nombre(examen)
        #plt.figure(), plt.imshow(campo_name, cmap='gray'),
plt.show(block=True)
        h, w = campo_name.shape[:3]

        # Dibujar el crop del campo Name en la imagen de salida
        output_image[y:y+h, :w, 1] = campo_name
```

```
        # Escribir el nombre del examen y el número de respuestas
correctas
        if respuestas_correctas >= 20:
            text = f"Examen: APROBADO"
        else:
            text = f"Examen: DESAPROBADO"

        cv2.putText(output_image, text, (w + 5, y + h // 2),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1)
        y += h + 10

    # Guardar la imagen de salida
    cv2.imwrite('output_image.png', output_image)
    plt.imshow(output_image)
    plt.axis('off')
    plt.show()

#d)
funciones.generar_imagen_salida(resultados_examenes)
```