Fundamentos de Python

Unidad 3. LIBRERÍA PANDAS

PROJECTOS

1. Automatización de calificaciones

ESTUDIANTE

Armando Gabriel Jacinto López

INSTITUCION

INFOTEC

FECHA

18/09/2025

1.- Introducción

Este informe presenta un análisis detallado de un programa en Python diseñado para automatizar el resultado de un grupo de alumnos. El código utiliza la biblioteca pandas para leer, procesar y comparar las respuestas de los estudiantes con las respuestas correctas. El objetivo principal del programa es calcular la puntuación de cada estudiante y generar un reporte claro y ordenado que indique su rendimiento.

La estructura del programa se basa en la manipulación de DataFrames, una de las estructuras de datos más potentes de la biblioteca pandas. A través de este proceso, el script es capaz de manejar grandes cantidades de datos de manera eficiente, lo que lo hace ideal para tareas de evaluación a gran escala. Además, el programa no solo calcula las puntuaciones, sino que también genera un archivo CSV con un reporte más detallado, lo que facilita el seguimiento y la distribución de los resultados. El enfoque del código es convertir una tarea repetitiva en un proceso estructurado y fácil de entender, lo que permite que cada paso contribuya a un código que pueda ser utilizado múltiples veces para facilitar esta tarea.

2.- Desarrollo

El programa está dividido en varias secciones lógicas que trabajan en conjunto para lograr el objetivo final. A continuación, se detalla el propósito de cada una de estas secciones.

2.1.- Importación y Carga de Datos

La primera parte del código se encarga de preparar el entorno. La línea import pandas as pd es crucial, ya que importa la biblioteca pandas, asignándole el alias pd para facilitar su uso posterior. Este paso es fundamental, ya que sin esta biblioteca el programa no podría funcionar.

Luego, el código lee los archivos de entrada que contienen los datos del examen:

df_students = pd.read_csv("./Pandas/Files/respuestas_estudiantes.csv"): Carga un archivo CSV que contiene las respuestas enviadas por cada estudiante en un DataFrame llamado df_students. Cada fila representa a un estudiante y cada columna una pregunta.

df_answers = pd.read_excel("./Pandas/Files/respuestas_correctas.xlsx"): Carga un archivo de Excel que contiene las respuestas correctas a cada pregunta en un DataFrame llamado df_answers.

```python
import pandas as pd
# Extrar los archivos, y guardarlos en un dataframe
df_students = pd.read_csv("./Pandas/Files/respuestas_estudiantes.csv")
df_answers = pd.read_excel("./Pandas/Files/respuestas_correctas.xlsx")
```

Ilustración 1 Cargando los datos en dataframes

2.2.- Preparación de las Respuestas Correctas

Para facilitar la comparación, el programa extrae las respuestas correctas y las almacena de una forma más accesible.

questions = df_answers['Pregunta'].values: Extrae la columna de preguntas del DataFrame de respuestas correctas y la guarda en una lista.

answers = df_answers['Respuesta'].values: Extrae la columna de respuestas correctas y la guarda en otra lista.

n_questions = len(questions): Obtiene el número total de preguntas, lo que será útil para el bucle posterior.

right_answers = {}: Se crea un diccionario vacío que servirá para almacenar las preguntas y sus respuestas correctas.

El bucle for n in range(n_questions): recorre las listas de preguntas y respuestas, poblando el diccionario right_answers con pares clave-valor (pregunta: respuesta correcta). Esta estructura de datos es ideal para buscar la respuesta correcta de una pregunta específica de forma rápida y eficiente.

```
# Guardar las preguntas y respuestas en listas
questions = df_answers['Pregunta'].values
answers = df_answers['Respuesta'].values
# Obtener el numero total de preguntas
n_questions = len(questions)
# Crear un diccionario para guardar las preguntas, y sus respuestas
right_answers = {}
# Ciclo para agregar la informacion al diccionario
for n in range(n_questions):
    right_answers[questions[n]] = answers[n]
# Crear una nueva columna en el df de estudiantes
df_students['Puntuacion'] = 0
```

Ilustración 2 Creación del diccionario de respuestas correctas

2.3.- Lógica de Puntuación

Esta es la sección más importante del código, donde se calcula la puntuación de cada estudiante.

df_students['Puntuacion'] = 0: Se crea una nueva columna llamada 'Puntuacion' en el DataFrame de estudiantes y se inicializa con el valor 0 para todos.

El bucle for q in questions: itera sobre cada pregunta.

Dentro del bucle, right_answer = right_answers[q] busca la respuesta correcta para la pregunta actual (q) en el diccionario creado previamente.

La línea df_students['Puntuacion'] = df_students['Puntuacion'].add((df_students[q] == right_answer).astype(int)) es la pieza central de la lógica. df_students[q] == right_answer compara las respuestas de cada estudiante para la pregunta q con la respuesta correcta, generando una Serie de valores booleanos (True si la respuesta es correcta, False si no lo es).

.astype(int) convierte estos booleanos a enteros (1 para True, 0 para False).

.add() suma estos valores a la columna 'Puntuacion' existente. De esta manera, cada respuesta correcta añade 1 punto a la puntuación acumulada de cada estudiante.

```
for q in questions:
    # Obtener la respuesta correcta
    right_answer = right_answers[q]
    # Guardar la nueva serie creada en la column 'Puntuacion'
    df_students['Puntuacion'] = df_students['Puntuacion'].add((df_students[q] == right_answer).astype(int))
```

Ilustración 3 Nueva serie con las puntuaciones de cada estudiante

2.4.- Generación del Reporte Detallado

Después de calcular las puntuaciones, el programa crea un reporte más visual y detallado.

df_report = df_students.copy(): Se crea una copia del DataFrame de estudiantes para evitar modificar el original.

El bucle for q in questions: se usa de nuevo para recorrer las preguntas.

df_report[q] = df_report[q].where(df_report[q] == right_answers[q], df_report[q] + 'x'): Este comando es muy ingenioso. Utiliza el método .where() de pandas, que evalúa una condición y modifica los valores que no cumplen con ella. En este caso, si la respuesta de un estudiante (df_report[q]) es diferente de la respuesta correcta (right_answers[q]), el valor en el DataFrame de reporte se reemplaza por la respuesta del estudiante seguida de una 'x', marcando así las respuestas incorrectas de manera visual. Si la respuesta es correcta, no se hace nada.

df_report = df_report.sort_values('Puntuacion', ascending=False): El DataFrame de reporte se ordena de mayor a menor puntuación, facilitando la visualización de los mejores resultados.

```
# Crear una copia del dataframe de estudiantes
df_report = df_students.copy()
# En el nuevo dataframe, se usa el metodo 'where', para modificar los valores donde la respuesta no fue correcta
for q in questions:
    df_report[q] = df_report[q].where(df_report[q] == right_answers[q], df_report[q] + 'x')
# Ordenar el dataframe de manera descendente
df_report = df_report.sort_values('Puntuacion', ascending=False)
# Limpiar el formate del dataframe
df_final = df_report[['Nombre', 'Puntuacion']].sort_values('Puntuacion', ascending=False).to_string(index=False)
```

Ilustración 4 Modificación de los dataframe para mostrar, y crear el nuevo documento

2.5.- Salida y Exportación de Resultados

La última parte del código se encarga de presentar los resultados al usuario y de guardar el reporte para uso futuro.

df_final = df_report[['Nombre', 'Puntuacion']].sort_values('Puntuacion', ascending=False).to_string(index=False): Crea una versión simplificada del reporte que solo muestra el nombre y la puntuación de cada estudiante, también ordenada de forma descendente. .to_string(index=False) convierte este DataFrame en una cadena de texto, lista para ser impresa.

df_report.to_csv('resultados_examen.csv', index=False): Exporta el DataFrame de reporte completo (incluyendo las preguntas y respuestas marcadas) a un nuevo archivo CSV llamado 'resultados_examen.csv'. Esto permite que el reporte detallado sea guardado y accesible en un archivo.

print(df_final): Muestra en la consola la tabla final con los nombres y las puntuaciones, proporcionando una visualización inmediata de los resultados principales.

```
# Crear un archivo csv con los resultados obtenidos
df_report.to_csv('resultados_examen.csv', index=False)
# Mostrar los resultados
print(df_final)
```

Ilustración 5 Creación del nuevo documento

```
    Sofía  10
    Diego  10
    Mario  10
    Elena  10
    Laura  10
    Jorge  10
  Roberto  10
    Diana  10
  Ricardo  10
 Fernanda  10
  Claudia  10
    María   9
   Carlos   9
    Pedro   6
(.venv)
```

Ilustración 6 Resultados finales

resultados_examen.csv

	A	B	C	D	E	F	G	H	I	J	K	L
1	Nombre	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Puntuacio
2	Juan	A	B	C	D	A	B	C	D	A	B	10
3	Luisa	A	B	C	D	A	B	C	D	A	B	10
4	Luis	A	B	C	D	A	B	C	D	A	B	10
5	Ana	A	B	C	D	A	B	C	D	A	B	10
6	Gabriel	A	B	C	D	A	B	C	D	A	B	10
7	Patricia	A	B	C	D	A	B	C	D	A	B	10
8	SofÃa	A	B	C	D	A	B	C	D	A	B	10
9	Diego	A	B	C	D	A	B	C	D	A	B	10
10	Mario	A	B	C	D	A	B	C	D	A	B	10
11	Elena	A	B	C	D	A	B	C	D	A	B	10
12	Laura	A	B	C	D	A	B	C	D	A	B	10
13	Jorge	A	B	C	D	A	B	C	D	A	B	10
14	Roberto	A	B	C	D	A	B	C	D	A	B	10
15	Diana	A	B	C	D	A	B	C	D	A	B	10
16	Ricardo	A	B	C	D	A	B	C	D	A	B	10
17	Fernanda	A	B	C	D	A	B	C	D	A	B	10
18	Claudia	A	B	C	D	A	B	C	D	A	B	10
19	MarÃa	Bx	B	C	D	A	B	C	D	A	B	9
20	Carlos	A	B	C	D	A	B	C	D	A	Cx	9
21	Pedro	A	B	Ax	D	Cx	B	Ax	D	Cx	B	6

Ilustración 7 Archivo final

3.- Conclusiones

El programa de Python desarrollado demuestra una aplicación práctica y eficiente de la biblioteca pandas para la automatización de tareas académicas. El código es robusto, claro y sigue una lógica bien estructurada que se puede dividir en pasos distintos: carga de datos, preparación de la información, cálculo de la puntuación, generación de reportes y, finalmente, la exportación de los resultados.

La utilización de DataFrames y el método .where() son ejemplos de cómo pandas simplifica tareas complejas. En lugar de escribir bucles largos y complicados, el código aprovecha las operaciones de la librería pandas, lo que lo hace más rápido y legible. El resultado es un reporte simple y completo que no solo proporciona las puntuaciones, sino que también marca visualmente las respuestas incorrectas, lo que facilita el análisis de errores. Este programa es un ejemplo excelente de cómo la programación puede ser utilizada para la automatización de tareas.