

Trabajo práctico grupal

Lector de datos tabulares

Tabla de contenidos

1	Introducción	1
2	Objetivo	2
3	Pautas	2
3.1	Organización del código	2
3.2	Lectura del archivo	3
3.3	Tipos de datos	3
3.4	Selección de filas y columnas	3
3.5	Manejo de errores	4
3.6	Ejemplos y pruebas	4
4	Visualización del resultado	4
5	Evaluación	5

1 Introducción

Una tabla de datos permite organizar información en filas y columnas. En general, cada fila representa una observación y cada columna una variable. Por ejemplo, en una tabla de productos, cada fila puede corresponder a un producto y las columnas a su nombre, precio y cantidad disponible. En el análisis de datos se trabaja frecuentemente con este tipo de tablas. A partir de ellas podemos, por ejemplo, calcular el precio promedio de los productos o identificar aquellos que tienen pocas unidades disponibles.

Existen distintas formas de almacenar tablas en una computadora. Se pueden utilizar planillas de cálculo, bases de datos o archivos de texto plano, entre otras opciones. En este trabajo nos vamos a concentrar en los archivos de texto plano, cuyo contenido se puede leer con cualquier editor de texto.

Uno de los formatos más utilizados es CSV, del inglés *comma-separated values*. En su forma más sencilla, cada línea del archivo representa una fila y los valores se separan mediante comas. La primera línea suele contener los nombres de las columnas.

Por ejemplo, el archivo `productos.csv` contiene la siguiente tabla:

```
nombre,precio,cantidad
Cuaderno,2500.5,10
Lapicera,800.0,25
Carpeta,3200.0,8
```

En este caso, hay tres filas de datos y tres columnas. La primera fila de datos corresponde al producto Cuaderno, cuyo precio es 2500.5 y del que hay 10 unidades disponibles. La línea nombre,precio,cantidad es el encabezado y permite identificar qué representa cada columna.

También es posible utilizar otros separadores. La misma tabla, con punto y coma, se ve de la siguiente manera:

```
nombre;precio;cantidad
Cuaderno;2500.5;10
Lapicera;800.0;25
Carpeta;3200.0;8
```

En ambos ejemplos, las filas y las columnas representan la misma información. Lo que cambia es el carácter utilizado para separar los valores. Otros archivos utilizan tabulaciones o espacios.

2 Objetivo

El objetivo de este trabajo práctico es implementar un módulo de Python que permita leer tablas de datos almacenadas en archivos de texto plano. El lector deberá soportar como mínimo archivos CSV, y se valorará positivamente que permita utilizar otros separadores, como punto y coma o tabulaciones. Además, quienes quieran ampliar esta funcionalidad pueden permitir que el usuario indique un delimitador arbitrario, incluso de más de un carácter.

El resultado de la lectura será una lista de diccionarios. Cada diccionario representa una fila, sus claves son los nombres de las columnas y sus valores son los datos correspondientes. Por ejemplo, al leer el archivo productos.csv se obtiene:

```
[
  {"nombre": "Cuaderno", "precio": 2500.5, "cantidad": 10},
  {"nombre": "Lapicera", "precio": 800.0, "cantidad": 25},
  {"nombre": "Carpeta", "precio": 3200.0, "cantidad": 8},
]
```

Notar que los precios y las cantidades se representan mediante números, aunque en el archivo original todos los valores están almacenados como texto.

3 Pautas

3.1 Organización del código

Pueden organizar la implementación en un único módulo o distribuirla en varios módulos. También pueden utilizar programación orientada a objetos, programación funcional o combinar ambos enfoques. Los nombres de las funciones, las clases y sus argumentos quedan a criterio de cada grupo.

Cada grupo deberá decidir cómo organizar su solución y cómo se utilizará el lector desde otro programa. Para ello, tendrán que considerar distintas alternativas y evaluar sus ventajas y

desventajas. Durante la presentación deberán poder explicar por qué eligieron una alternativa y no otra.

3.2 Lectura del archivo

El análisis del texto y la separación de los campos deben ser implementados por el grupo. No se permite utilizar el módulo `csv`, los lectores de `pandas` ni otras bibliotecas que resuelvan esta tarea. Sí pueden utilizar expresiones regulares, si lo consideran conveniente.

El lector utilizará la primera fila del archivo como encabezado. Deberán definir qué sucede si alguno de los nombres de las columnas está vacío o si hay nombres repetidos. Opcionalmente, pueden admitir la lectura de archivos sin encabezado.

3.3 Tipos de datos

El lector deberá soportar valores enteros (`int`), flotantes (`float`), de texto (`str`) y booleanos (`bool`). También deberá contemplar valores nulos cuando una celda esté vacía.

El lector deberá determinar automáticamente el tipo de las columnas para las que el usuario no haya indicado un tipo. Para ello, tendrán que definir y documentar los criterios utilizados. Por ejemplo, una columna puede contener los valores 10, 20 y 30.5. En ese caso, deberán decidir qué tipo permite representar todos sus valores. También deberán considerar qué sucede si aparecen números y texto en una misma columna, o si todas sus celdas están vacías.

También deberán indicar qué representaciones se reconocen como booleanos y cómo se representan los valores nulos en el resultado.

El lector deberá permitir que el usuario especifique el tipo de una o más columnas. Los tipos indicados por el usuario tendrán prioridad sobre la inferencia automática. Esto puede ser útil cuando la interpretación automática no coincide con lo que se necesita. Por ejemplo, un código como 00123 puede contener solo dígitos, pero al convertirlo a un entero se pierden los ceros iniciales. Si el usuario indica que esa columna es de texto, el lector deberá conservar "00123".

Si un valor no puede convertirse al tipo indicado, deberán tratarlo de acuerdo con el manejo de errores definido por el grupo. No deberán reemplazar el tipo solicitado por otro inferido automáticamente.

3.4 Selección de filas y columnas

El lector deberá permitir:

1. Especificar la cantidad de filas a leer.
2. Especificar un rango de filas a leer.
3. Seleccionar las columnas que se quieren obtener.
4. Filtrar las filas mediante una función que evalúe los valores de una o más columnas y devuelva un booleano. Se conservarán las filas para las que la función devuelva `True`.

Por ejemplo, para la tabla de productos, el usuario podría querer obtener solamente los nombres y los precios de aquellos productos que tienen menos de diez unidades disponibles. La función de

filtro permite expresar esa condición, mientras que la selección de columnas determina qué datos se incluyen en el resultado.

Deberán documentar cómo se utiliza cada opción y qué sucede cuando se combinan. Por ejemplo, si se solicitan diez filas y se aplica un filtro, pueden examinar las primeras diez filas del archivo o continuar leyendo hasta obtener diez filas que cumplan la condición. Cada grupo deberá decidir cuál será el comportamiento de su lector.

Del mismo modo, deberán indicar cómo se numeran las filas, si los extremos del rango están incluidos y si el rango se aplica antes o después del filtro. También tendrán que definir si es posible filtrar por una columna que no se incluye en el resultado.

3.5 Manejo de errores

Deberán definir cómo se comportará el lector ante un archivo vacío, filas vacías, filas con distinta cantidad de campos y valores que no puedan convertirse al tipo esperado.

No es necesario continuar la lectura ante cualquier problema. Pueden decidir detenerla e informar un error. En ese caso, el mensaje debe permitir identificar qué ocurrió y, cuando corresponda, en qué fila y columna se encuentra el problema.

3.6 Ejemplos y pruebas

Para comprobar el correcto funcionamiento del lector, deberán incluir archivos pequeños y ejemplos de uso con sus resultados esperados. Las pruebas deben contemplar la lectura habitual, la combinación de opciones y las situaciones problemáticas mencionadas anteriormente.

4 Visualización del resultado

Para mostrar los datos de forma tabular, pueden convertir la lista de diccionarios a un `DataFrame` de pandas. Esta conversión debe realizarse en el programa utilizado para la demostración, fuera del código del lector.

Por ejemplo, si la función que implementaron se llama `leer_tabla`, pueden utilizar:

```
import pandas as pd

filas = leer_tabla("productos.csv")
tabla = pd.DataFrame(filas)
print(tabla)
```

El nombre `leer_tabla` se utiliza solo como ejemplo. Cada grupo puede definir su propia interfaz.

También deberán poder mostrar la lista de diccionarios original, ya que pandas puede cambiar la representación de los tipos y de los valores nulos al construir el `DataFrame`.

5 Evaluación

La evaluación contempla no solo la correcta ejecución del programa, sino también la calidad del código desarrollado, considerando aspectos como la organización, claridad, robustez y el uso de estructuras de datos y algoritmos apropiados.

Además, se tendrá en cuenta que puedan justificar las decisiones tomadas y que el lector se comporte de acuerdo con lo que documentaron.

Cada grupo realizará una presentación en una reunión con los docentes. Durante la reunión se les proporcionarán archivos CSV que no hayan utilizado previamente, dentro del alcance establecido en estas pautas. Deberán leerlos con su implementación y explicar los resultados obtenidos.

Todos los integrantes deberán poder explicar el funcionamiento del código y responder preguntas sobre las decisiones del grupo, las alternativas consideradas y las limitaciones de la solución.