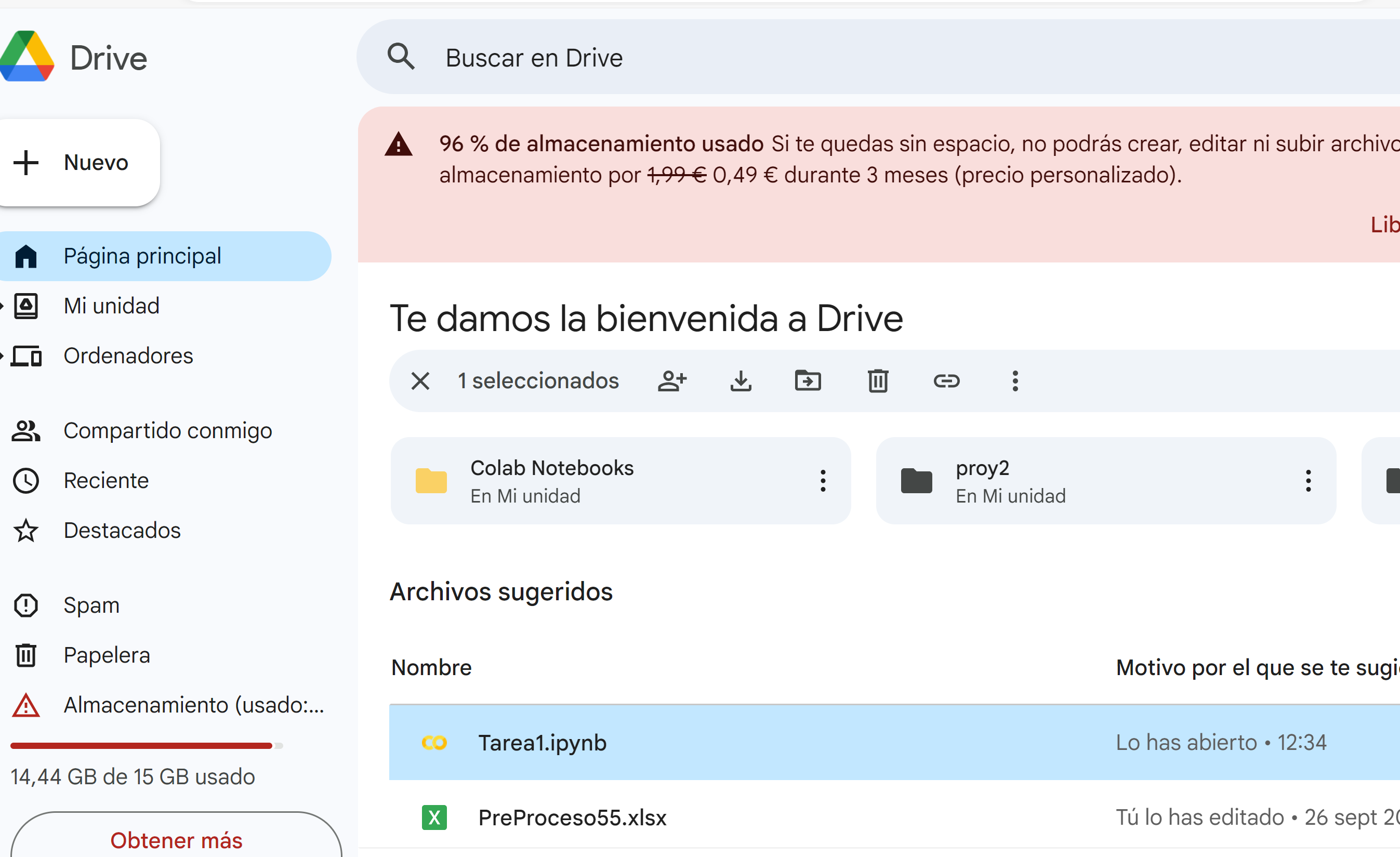
* **Apartado 2: Resolver ciertos problemas en Python (2 puntos cada uno)**

Dado que a lo largo del año vamos a tener que trabajar bastante con Python, es necesario tener cierta base sobre los aspectos básicos del lenguaje. Para ello se propone la realización de los siguientes ejercicios que deberán ser subidos al **repositorio GitHub** del Apartado 1.

Problema0 o previo: establecer un entorno de trabajo y pruebas de phyton:

Voy a usar Google colab, que es una extensión de Google drive que habilita un entorno de prueba para programas Python creando una referencia como un simple fichero en la carpeta “colab notebooks” en Google Drive



* + **Problema 1. División de una lista de enteros.**

Escribe una función que reciba por parámetro una lista de enteros y devuelva dos listas: una con los valores nega&vos que tuviera y otra con los posi&vos. Ambas listas deben estar ordenadas ascendentemente

def separar\_positivos\_negativos(numeros):

numeros.sort()

negativos = [num for num in numeros if num < 0]

positivos = [num for num in numeros if num >= 0]

return negativos, positivos

Forma de probarlo:

numeros = [5, -3, 9, -8, 2, -7, 1, -1]

negativos, positivos = separar\_positivos\_negativos(numeros)

print("Números negativos:", negativos)

print("Números positivos:", positivos)

Nota aquí usamos la función sort() de una lista, la asignación a una nueva lista, la forma de crear una lista a partir de recorrer otra con una variable elemento temporas usando una clausula if aplicada al elemento.

Usamos la función print utilizando texto y listas como argumentos.

* + **Problema 2. Frecuencia de palabras en un texto.**

Escribe un programa que pida al usuario ingresar una frase o párrafo. Luego, el programa debe contar cuántas veces aparece cada palabra en el texto y mostrar las palabras junto con su frecuencia.

**Requisitos**:

* + 1. Eliminar los signos de puntuación y conver&r todas las palabras a minúsculas para evitar diferencias.
    2. Usar un diccionario donde la clave sea la palabra y el valor sea su frecuencia.
    3. Mostrar las palabras y sus frecuencias de forma ordenada por la palabra.

def contar\_palabras(texto):

# Convertir el texto a minúsculas para hacer la comparación de palabras más fácil

texto = texto.lower()

# Reemplazar signos de puntuación comunes con espacios

for signo in [",", ".", ":", ";", "!", “¿”,"?", "(", ")"]:

texto = texto.replace(signo, " ")

# Separar el texto en palabras

palabras = texto.split()

# Crear un diccionario para contar las palabras

conteo\_palabras = {}

for palabra in palabras:

if palabra in conteo\_palabras:

conteo\_palabras[palabra] += 1

else:

conteo\_palabras[palabra] = 1

# Obtener una lista de palabras ordenadas alfabéticamente

palabras\_ordenadas = sorted(conteo\_palabras.keys())

# Imprimir cada palabra y su conteo

for palabra in palabras\_ordenadas:

print(f"{palabra}: {conteo\_palabras[palabra]}")

Para Probarlo:

texto = "Hola, hola! ¿Cómo estás? Hola de nuevo. ¿Todo bien? Todo bien."

contar\_palabras(texto)

Exploramos aquí principalmente la estructura diccionario de phyton, y como imprimir datos en diccionarios.

* + **Problema 3. Intersección y unión de conjuntos**

Escribe un programa que permita al usuario crear dos conjuntos de números enteros. Luego, el programa debe calcular y mostrar:

* + 1. La **intersección** de ambos conjuntos (elementos comunes).
    2. La **unión** de ambos conjuntos (todos los elementos sin duplicados).
    3. La **diferencia simétrica** (elementos que están en uno u otro conjunto, pero no en ambos).

def procesar\_conjuntos():

# Solicitar el primer conjunto de números

conjunto1 = input("Introduce el primer conjunto de números separados por comas: ")

# Convertir la entrada en un conjunto de enteros

conjunto1 = set(map(int, conjunto1.split(',')))

# Solicitar el segundo conjunto de números

conjunto2 = input("Introduce el segundo conjunto de números separados por comas: ")

# Convertir la entrada en un conjunto de enteros

conjunto2 = set(map(int, conjunto2.split(',')))

# Calcular la intersección (elementos comunes)

interseccion = conjunto1 & conjunto2

# Calcular la unión (todos los elementos sin duplicados)

union = conjunto1 | conjunto2

# Calcular la diferencia simétrica (elementos que están en uno u otro, pero no en ambos)

diferencia\_simetrica = conjunto1 ^ conjunto2

# Mostrar los resultados

print(f"Intersección: {sorted(interseccion)}")

print(f"Unión: {sorted(union)}")

print(f"Diferencia simétrica: {sorted(diferencia\_simetrica)}")

Como Probarlo

procesar\_conjuntos()

Exploramos la interacción por consola con el usuario (función input creando una lista)

La creación de elementos set

La operación con elementos set

Subimos los tres programas:

