

## Apartado 1 : Crear cuenta en GitHub y crear un repositorio (paso a paso)

### A. Crear la cuenta en GitHub

#### Pasos que realicé:

1. Abrí el navegador para ir a <https://github.com>.
2. Pulse **Sign up** (registrarse) y rellené:
  - **Email:** mi correo.
  - **Password:** añadí una contraseña segura.
  - **Username:** mi nombre de usuario.
3. Pasé a verificar el correo: GitHub me envió un email; hice clic en el enlace para verificar la cuenta.
4. **Cuando GitHub preguntó “Are you a student or a teacher?”:**
  - Seleccioné **“Skip personalization”**
5. Configuré el **Two-factor authentication** para añadir más seguridad.

### Create your free account

Explore GitHub's core features for individuals and organizations.  
See what's included ▾



### Sign up for GitHub

[Continue with Google](#)  
[Continue with Apple](#)  
or

Email<sup>\*</sup>

Password<sup>\*</sup>  
  
Password should be at least 15 characters OR at least 8 characters including a number and a lowercase letter.

Username<sup>\*</sup>  
  
Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.

Your Country/Region<sup>\*</sup>  
Slovakia ▾

## B. Pasos que realice para crear el repositorio

1. Inicié sesión en GitHub.
2. En la esquina superior derecha, pulsé el botón **+** → **New repository**.
3. Rellené:
  - **Repository name:** PIA01\_tarea (en este caso, escogí este).
  - **Description:** Tarea 1 - Unidad PIA: repositorio con ejercicios y documentación.
  - **Public:** marqué **Public** (para que la tarea sea revisada).
  - **Initialize this repository with:** marque **Add a README file** (para que el repositorio cuente con un README).
4. Para finalizar, pulsé **Create repository**.

## C. Añadir el PDF con la breve explicación del proceso

En este caso lo subí desde la web aunque también podría haberlo realizado desde por ejemplo Visual Studio o similares.

1. Desde el repositorio.
2. Pulsé **Add file** → **Upload files**.
3. Arrastré el archivo PDF (llamado **proceso\_github.pdf**).
4. En **Commit changes** escribí: **Añadido proceso\_github.pdf — Apartado 1.**
5. Pulsé **Commit changes**.

The screenshot shows a GitHub repository page for 'PIA01\_tarea'. The repository was created by 'escalartica' and has 2 commits. It contains files such as 'proceso\_github.pdf', 'README.md', and a 'README' file. The 'README' file is described as 'Tarea 1 - Unidad PIA: repositorio con ejercicios y documentación'.

## 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.

### Problema 1. Procesamiento de una lista de enteros.

Crea una función que reciba una lista de enteros por parámetro y devuelva otra lista, de acuerdo a las siguientes acciones:

1. Eliminar los números negativos de la lista.
2. Eliminar los valores que están repetidos, quedándonos con uno de ellos.
3. Ordenar los números resultantes de menor a mayor.

Por ejemplo, si le pasara [4, -1, 2, 4, 3, -5, 2], debería retornar [2,3,4].

#### Captura del problema 1.

```

def procesar_lista_enteros(lista):
    """
    Recibe una lista de enteros y devuelve otra lista:
    1. Sin números negativos.
    2. Sin valores duplicados.
    3. Ordenada de menor a mayor.
    """
    # 1. Filtrar negativos
    positivos = [n for n in lista if n >= 0]

    # 2. Eliminar duplicados usando un set
    sin_duplicados = set(positivos)

    # 3. Ordenar de menor a mayor
    resultado = sorted(sin_duplicados)

    return resultado

# Ejemplo de prueba
if __name__ == "__main__":
    ejemplo = [4, -1, 2, 4, 3, -5, 2]
    print("Entrada:", ejemplo)

```

## Problema 2. Frecuencia de palabras en un texto.

Escribe una función que reciba por parámetro una lista de palabras y la ruta a un fichero de texto y devuelva un diccionario que muestre cuantas veces aparecen las distintas palabras de la lista en el fichero de texto. Haz un pequeño programa que la ponga a prueba.

Requisitos:

1. Eliminar signos de puntuación y convertir todo a minúsculas.
2. Usar un diccionario donde la clave sea la palabra y el valor su frecuencia.
3. Mostrar las palabras y sus frecuencias de forma ordenada por la palabra.

### -Captura problema 2

The screenshot shows a code editor interface with two tabs open. On the left, there is a file explorer window titled 'EXPLORADOR' showing a folder structure under 'PIAO1\_TAREA [GITHUB]'. The files listed are: Apartado3\_kaggle\_dataset.pdf, Captura de pantalla 2025-11-05 a las 17.5..., problema1\_lista\_enteros.py (marked with a 'M' icon), problema2\_frecuencia\_palabras.py (marked with a 'M' icon), problema3\_conjuntos.py, proceso\_github.pdf, README.md, and texto\_prueba.txt. The right side of the screen displays the content of the 'problema2\_frecuencia\_palabras.py' file. The code is as follows:

```
problema2_frecuencia_palabras.py > [?] palabra
import string
def frecuencia_palabras(lista_palabras, ruta_fichero):
    """
    Recibe una lista de palabras y la ruta a un fichero de texto.
    Devuelve un diccionario con la frecuencia de aparición de
    esas palabras en el texto (en minúsculas y sin puntuación).
    """
    frecuencias = {}

    # Leer el contenido del archivo
    with open(ruta_fichero, 'r', encoding='utf-8') as f:
        texto = f.read().lower() # convierte a minúsculas

    # Elimina signos de puntuación
    for signo in string.punctuation:
        texto = texto.replace(signo, "")

    # Convierte el texto en una lista de palabras
    palabras_texto = texto.split()

    # Calcula la frecuencia
    for palabra in lista_palabras:
        frecuencias[palabra] = palabras_texto.count(palabra.lower())
```

## Problema 3. Trabajo con conjuntos.

Escribe una función que reciba dos listas de enteros y devuelva un diccionario con la siguiente información (ES OBLIGATORIO USAR CONJUNTOS PARA CALCULARLOS)

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).

```
git add .
problem1_lista_enteros.py
problem2_frecuencia_palabras.py
problem3_conjuntos.py
README.md
proceso_github.pdf
```

The screenshot shows a terminal window with the following command and output:

```
git add .
problem1_lista_enteros.py
problem2_frecuencia_palabras.py
problem3_conjuntos.py
README.md
proceso_github.pdf
```

The terminal is part of a larger interface, likely a code editor or IDE, showing a file explorer on the left and code editors on the right. The file explorer lists the following files in the directory "PIA01\_TAREA":

- Captura de pantalla 2025-11-05 a las 17.5...
- problema1\_lista\_enteros.py
- problema2\_frecuencia\_palabras.py
- problema3\_conjuntos.py**
- proceso\_github.pdf
- README.md
- texto\_prueba.txt

The "problema3\_conjuntos.py" file is currently selected and its content is displayed in the main editor area:

```
def operaciones_conjuntos(lista1, lista2):
    resultado = {}
    for clave in lista1:
        if clave in lista2:
            resultado[clave] = lista1.count(clave)
        else:
            resultado[clave] = -1
    for clave in lista2:
        if clave not in resultado:
            resultado[clave] = lista2.count(clave)
        else:
            resultado[clave] -= lista2.count(clave)
    return resultado

# ---- Programa de prueba ---
if __name__ == "__main__":
    lista_a = [1, 2, 3, 4, 5]
    lista_b = [4, 5, 6, 7, 8]

    resultado = operaciones_conjuntos(lista_a, lista_b)

    print("Resultados de las operaciones con conjuntos:\n")
    for clave, valor in resultado.items():
        print(f"{clave.capitalize()}: {valor}")
```

\*Los 3 problemas fueron subidos al repositorio de GitHub