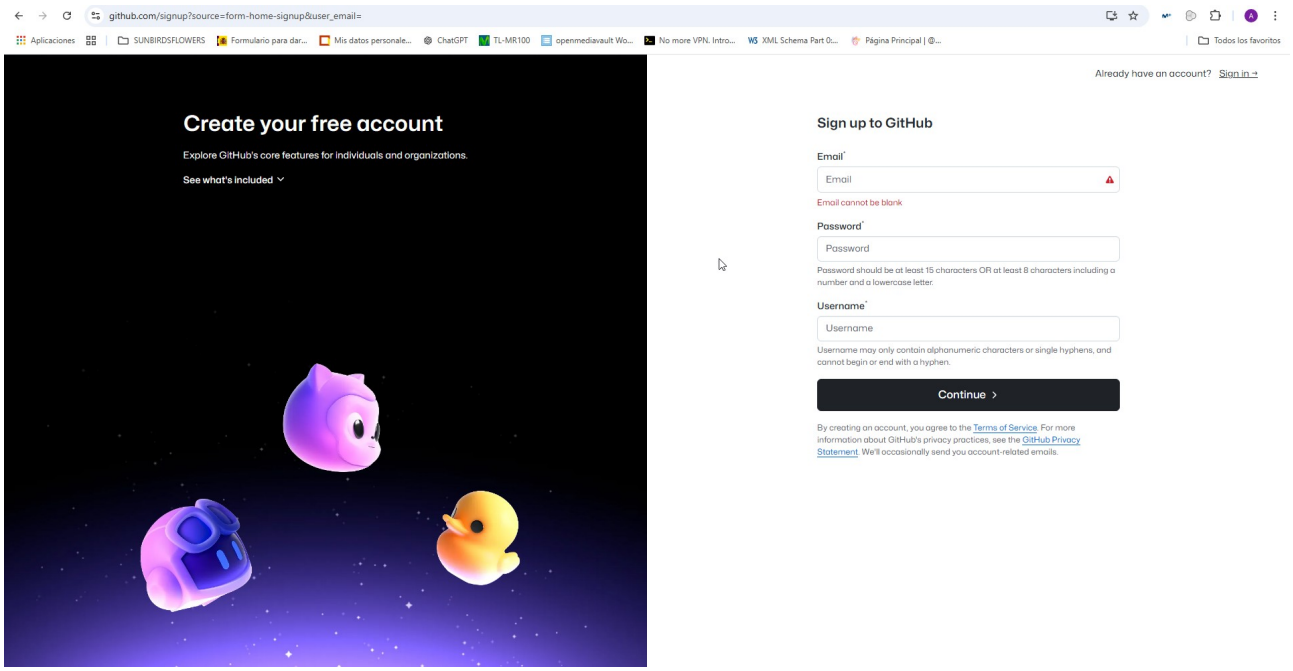
PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL

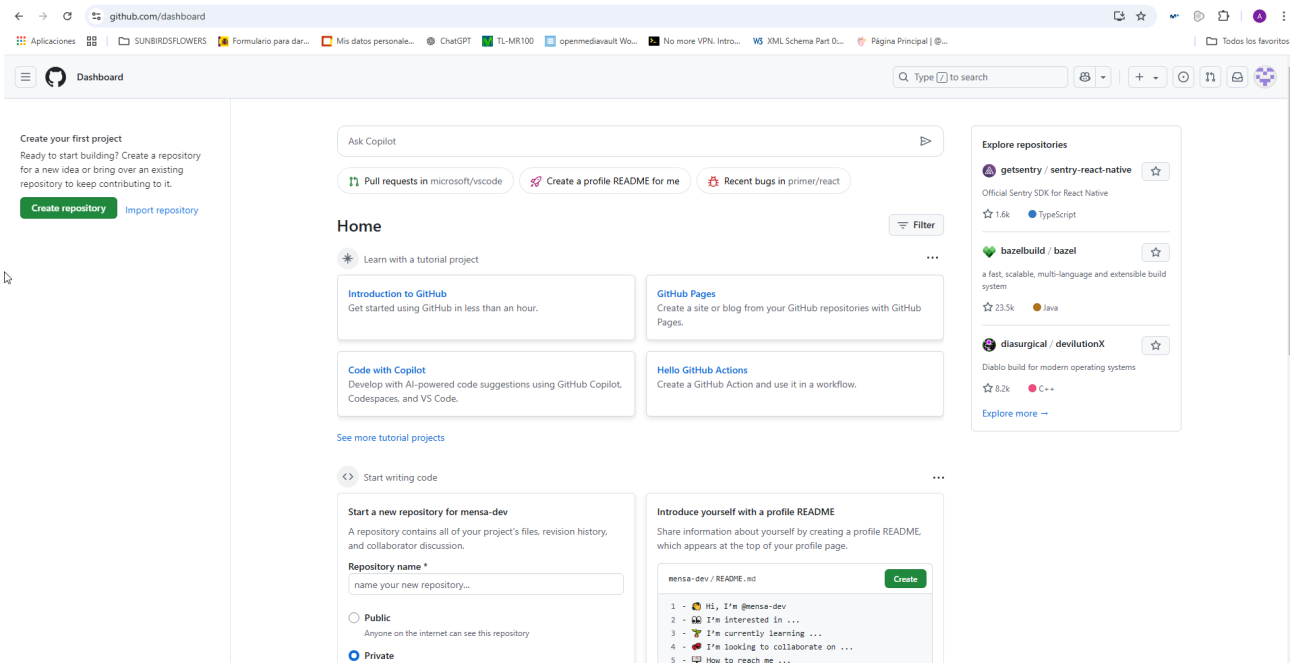
TAREA 1

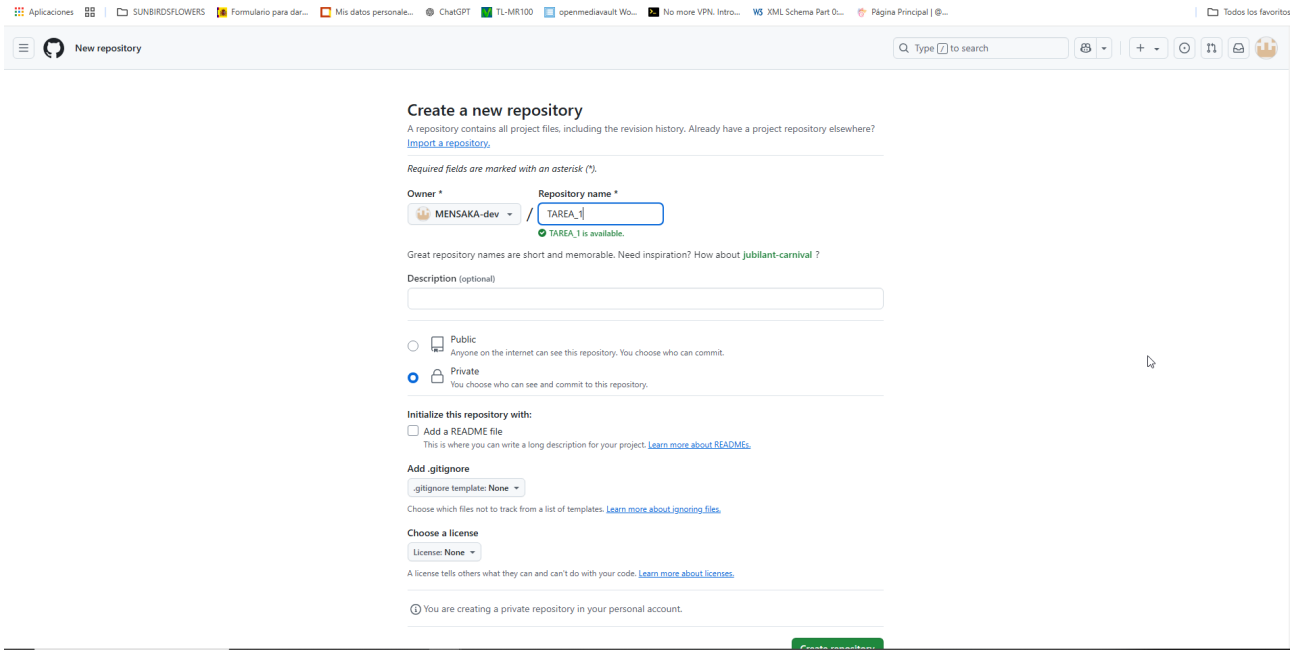
Apartado 1: Crear cuenta en GitHub y crear un repositorio

Accedemos a la página de GITHUB y nos registramos.

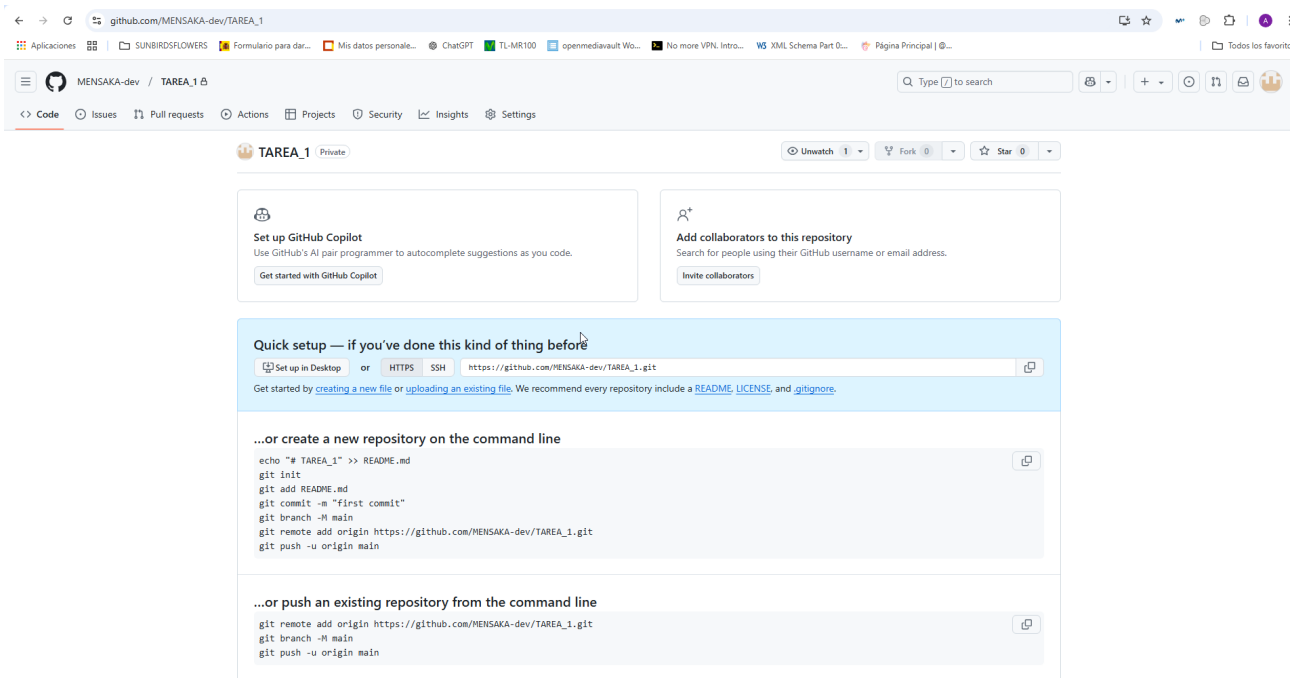


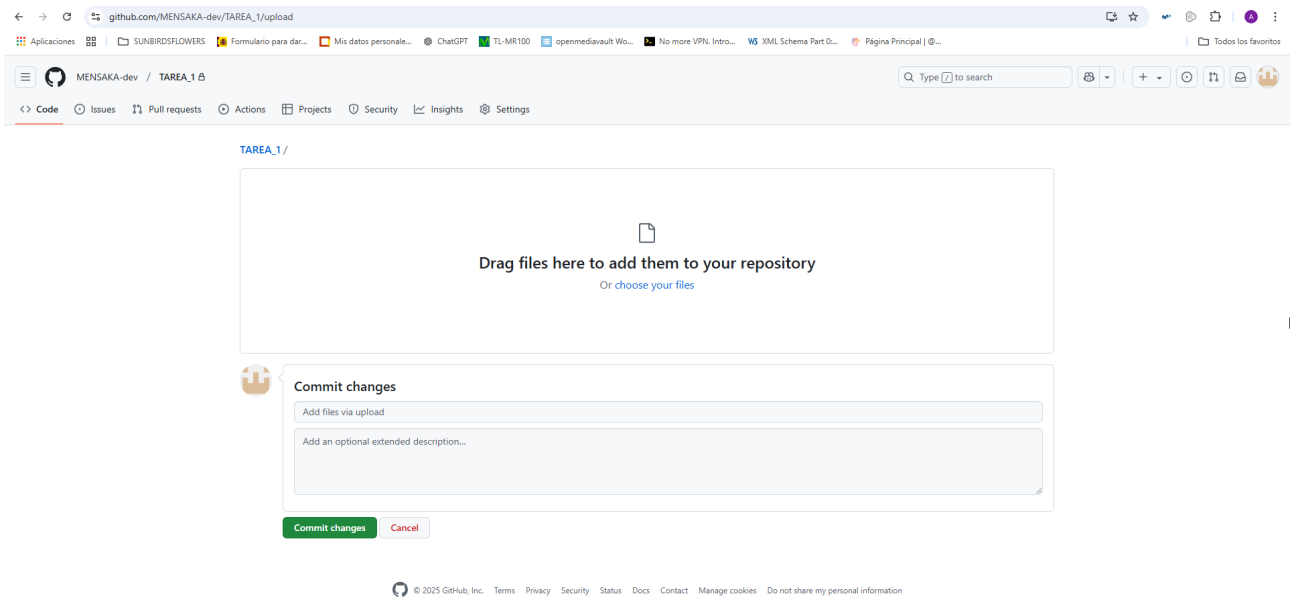
Creamos nuestro nuevo repositorio.

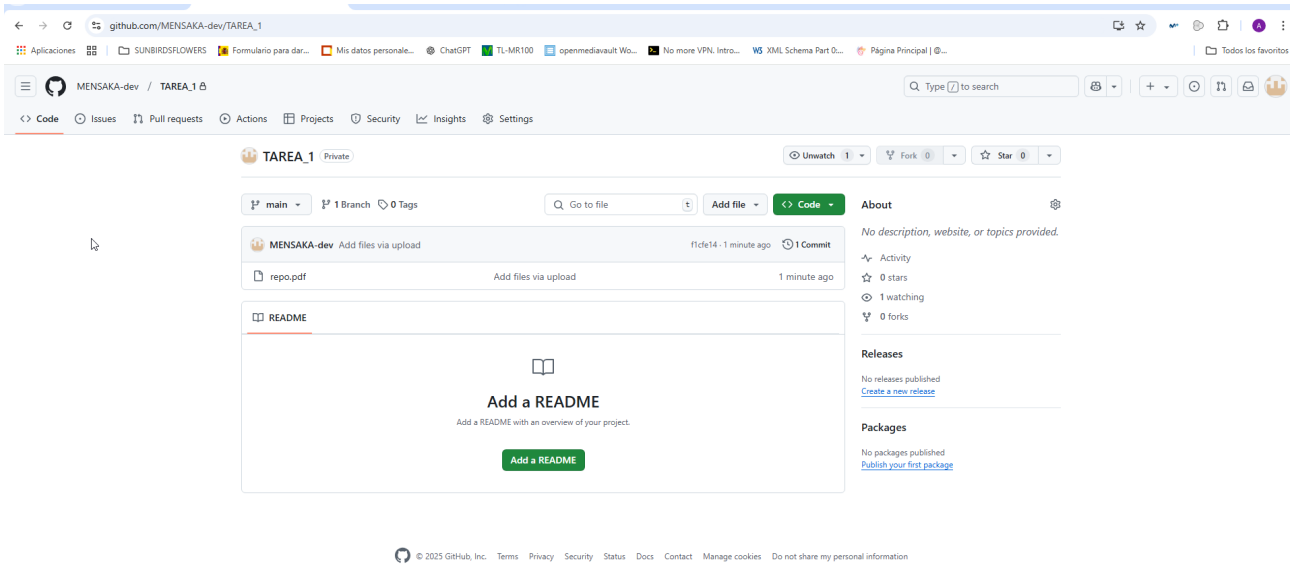


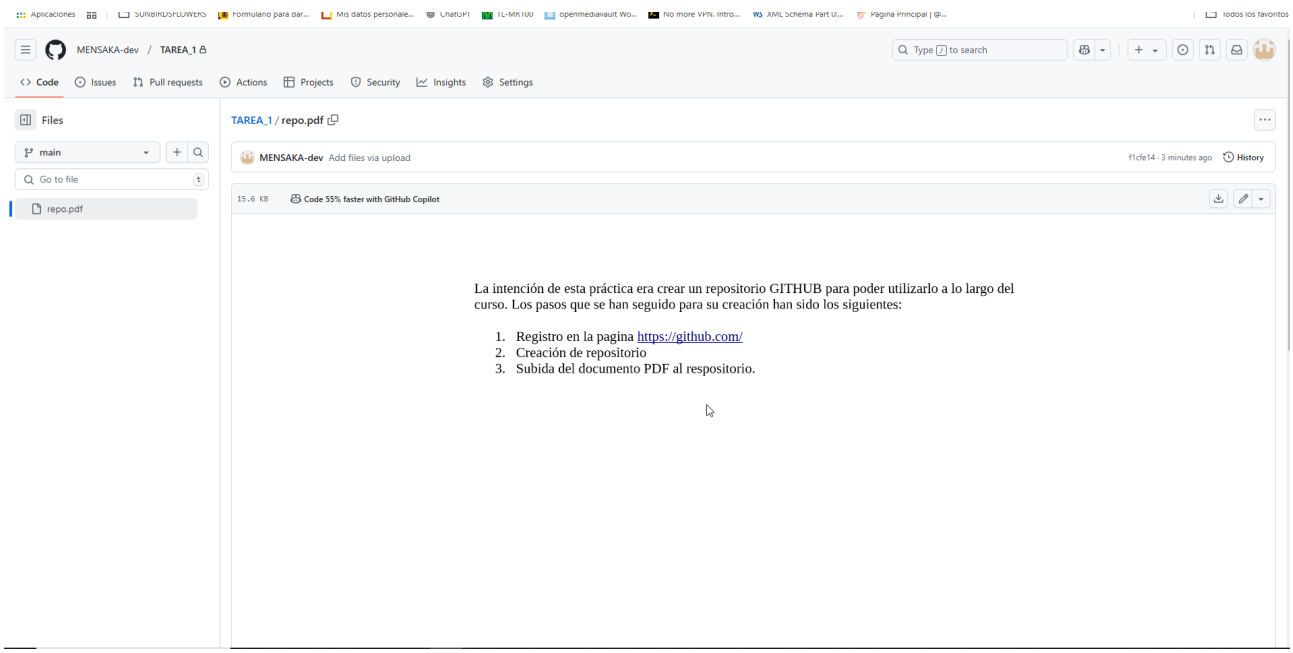


Añadimos un archivo PDF a la rama Principal.




github.com/MENSAKA-dev/TAREA_1/upload
MENSAKA-dev / TAREA_1
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
TAREA_1 /
Drag files here to add them to your repository
Or choose your files
Commit changes
Add files via upload
Add an optional extended description...
Commit changes
Cancel
© 2025 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

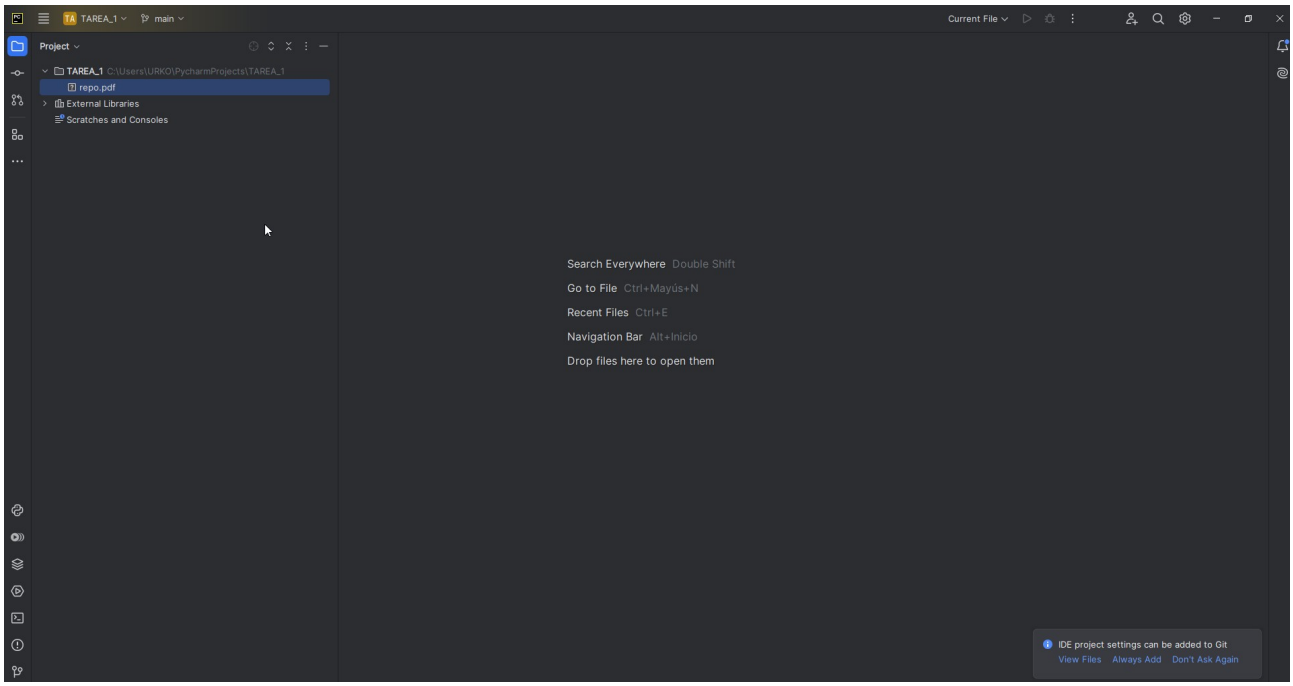

github.com/MENSAKA-dev/TAREA_1
MENSAKA-dev / TAREA_1
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
TAREA_1
Private
Unwatch 1
Fork 0
Star 0
main
1 Branch
0 Tags
Go to file
Add file
Code
About
No description, website, or topics provided.
MENSAKA-dev Add files via upload
f1cfe14 · 1 minute ago
1 Commit
repo.pdf
Add files via upload
1 minute ago
README
Add a README
Add a README with an overview of your project.
Add a README
Activity
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
© 2025 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information



Apartado 2: Resolver ciertos problemas en Python

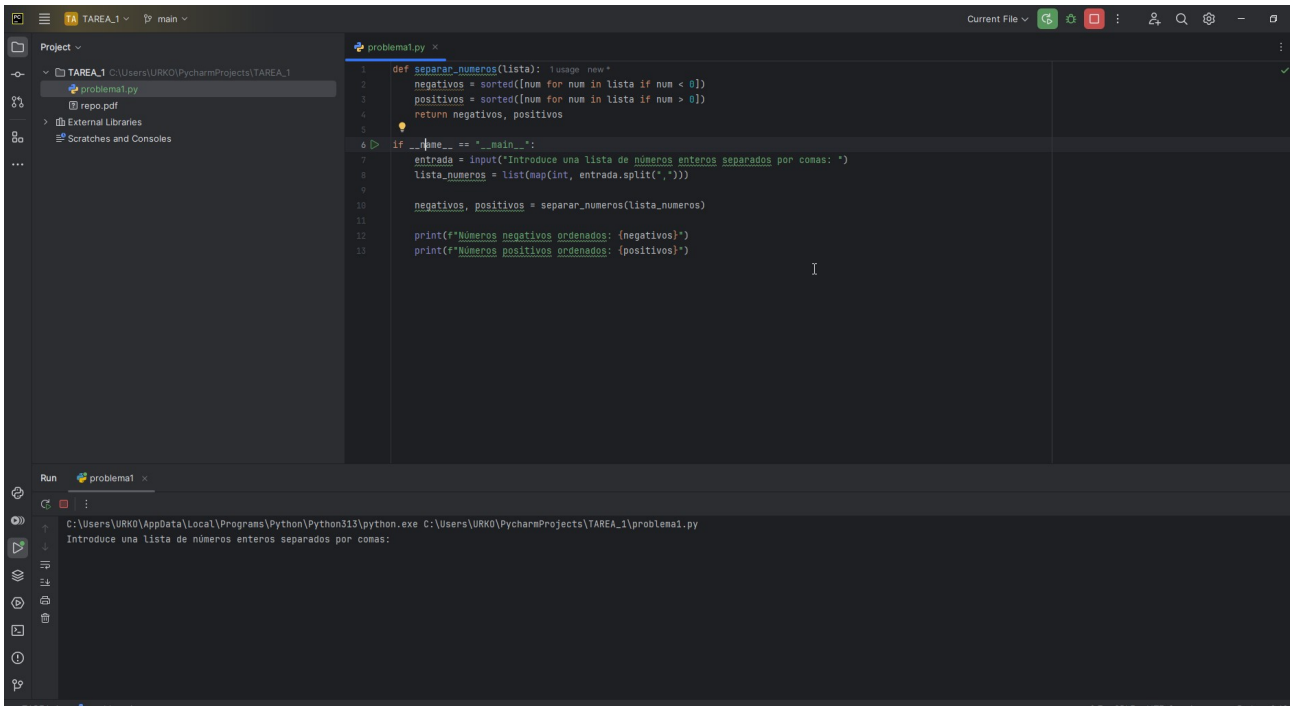
Problema 1. División de una lista de enteros.

Clonamos en pycharm el repositorio que hemos creado.

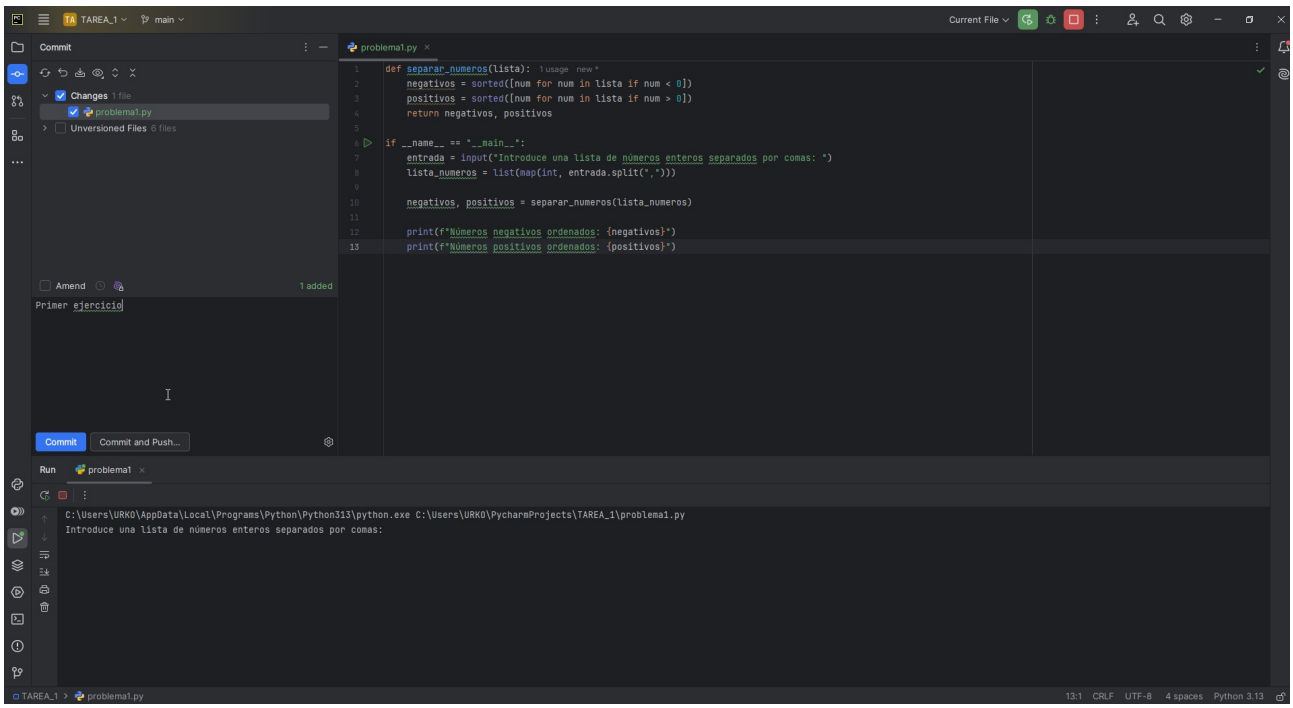


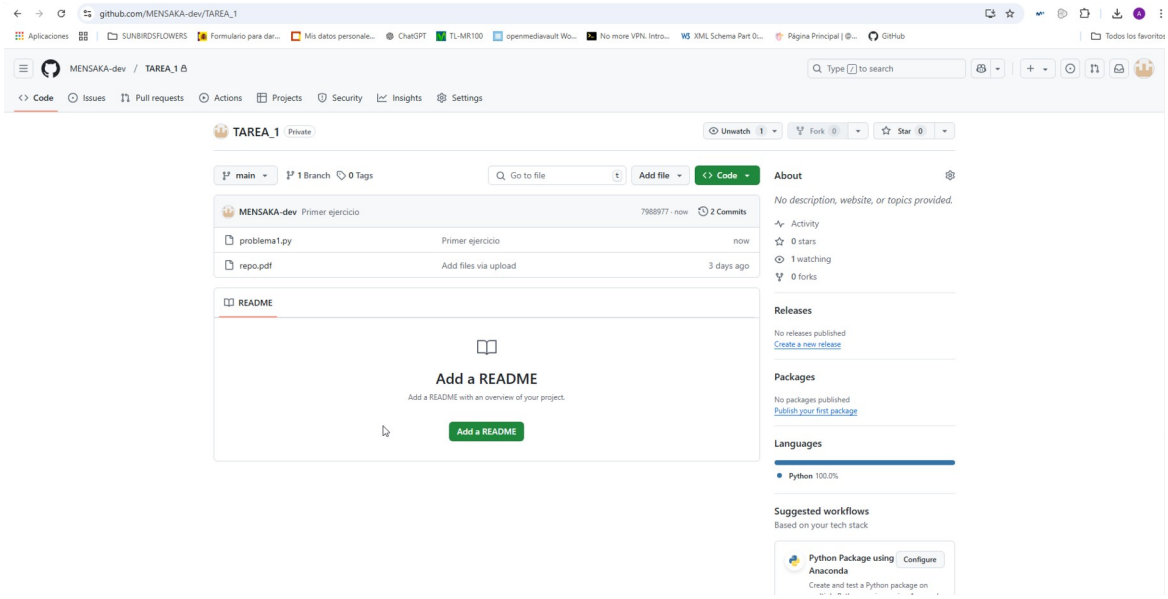
Añadimos un fichero llamado problema1.py y creamos el código para el primer ejercicio.

```python
def separar_numeros(lista):
    negativos = sorted([num for num in lista if num < 0])
    positivos = sorted([num for num in lista if num > 0])
    return negativos, positivos

if __name__ == "__main__":
    entrada = input("Introduce una lista de números enteros separados por comas: ")
    lista_numeros = list(map(int, entrada.split(",")))

    negativos, positivos = separar_numeros(lista_numeros)

    print(f"Números negativos ordenados: {negativos}")
    print(f"Números positivos ordenados: {positivos}")
```



Realizamos un Commit and Push





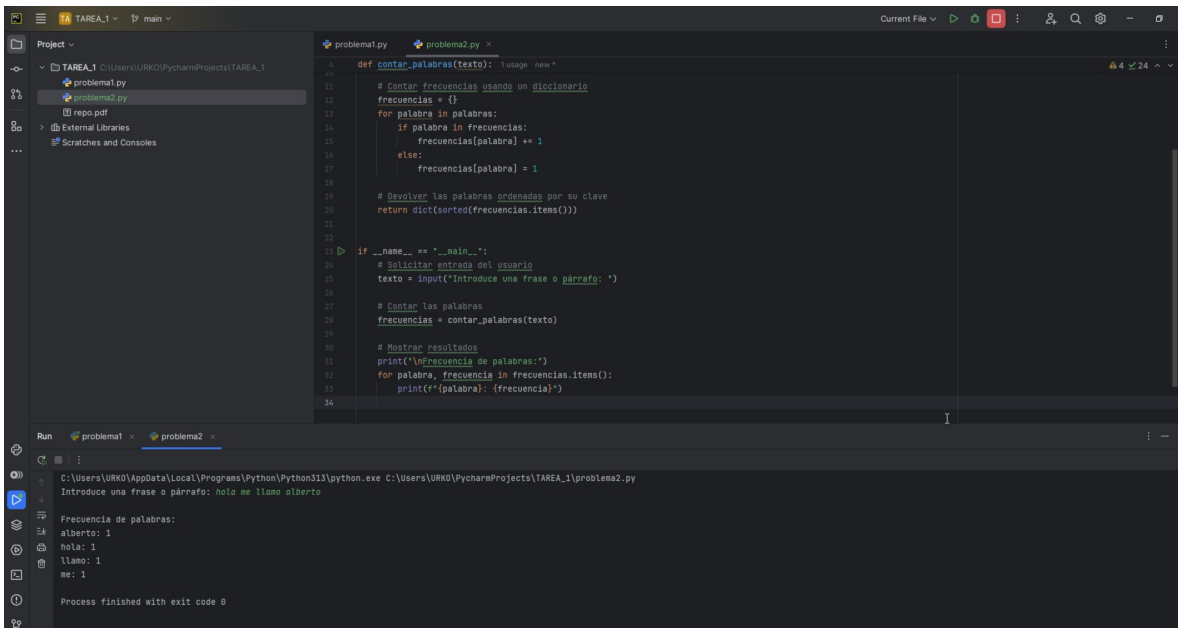
Problema 2. Frecuencia de palabras en un texto.

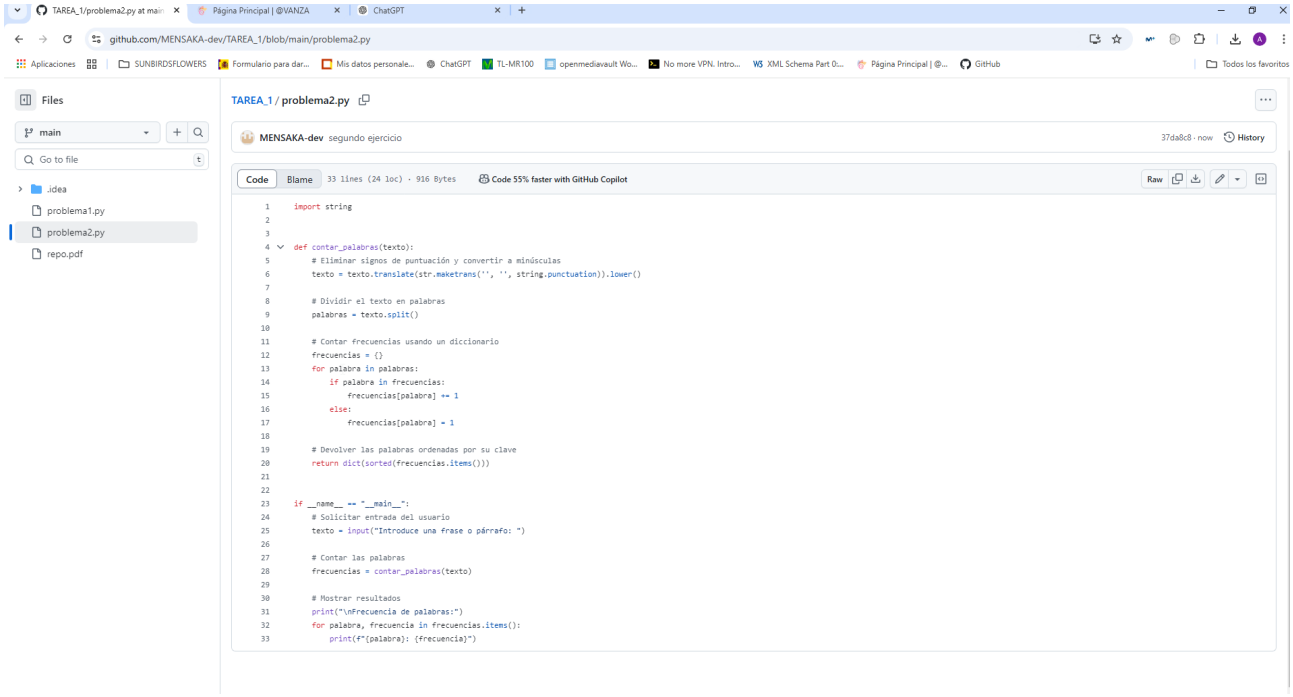
Añadimos el código para el segundo ejercicio.

```
import string


def contar_palabras(texto):
    # Eliminar signos de puntuación y convertir a minúsculas
    texto = texto.translate(str.maketrans('', '', string.punctuation)).lower()

    # Dividir el texto en palabras
    palabras = texto.split()
```

```
    # Contar frecuencias usando un diccionario
    frecuencias = {}
    for palabra in palabras:
        if palabra in frecuencias:
            frecuencias[palabra] += 1
        else:
            frecuencias[palabra] = 1

    # Devolver las palabras ordenadas por su clave
    return dict(sorted(frecuencias.items()))


if __name__ == "__main__":
    # Solicitar entrada del usuario
    texto = input("Introduce una frase o párrafo: ")

    # Contar las palabras
    frecuencias = contar_palabras(texto)

    # Mostrar resultados
    print("\nFrecuencia de palabras:")
    for palabra, frecuencia in frecuencias.items():
        print(f"{palabra}: {frecuencia}")
```

Realizamos un Commit and Push.


TAREA_1
main
Current File
Project
TAREA_1 C:\Users\URKO\PycharmProjects\TAREA_1
problema1.py
problema2.py
repo.pdf
External Libraries
Scratches and Consoles
problema1.py
problema2.py
def contar_palabras(texto): 1 usage new *
# Contar frecuencias usando un diccionario
frecuencias = {}
for palabra in palabras:
if palabra in frecuencias:
frecuencias[palabra] += 1
else:
frecuencias[palabra] = 1
# Devolver las palabras ordenadas por su clave
return dict(sorted(frecuencias.items()))
if __name__ == "__main__":
# Solicitar entrada del usuario
texto = input("Introduce una frase o párrafo: ")
# Contar las palabras
frecuencias = contar_palabras(texto)
# Mostrar resultados
print("\nFrecuencia de palabras:")
for palabra, frecuencia in frecuencias.items():
print(f"{palabra}: {frecuencia}")
Run
problema1
problema2
C:\Users\URKO\AppData\Local\Programs\Python\Python313\python.exe C:\Users\URKO\PycharmProjects\TAREA_1\problema2.py
Introduce una frase o párrafo: hola me llamo alberto
Frecuencia de palabras:
alberto: 1
hola: 1
llamo: 1
me: 1
Process finished with exit code 0

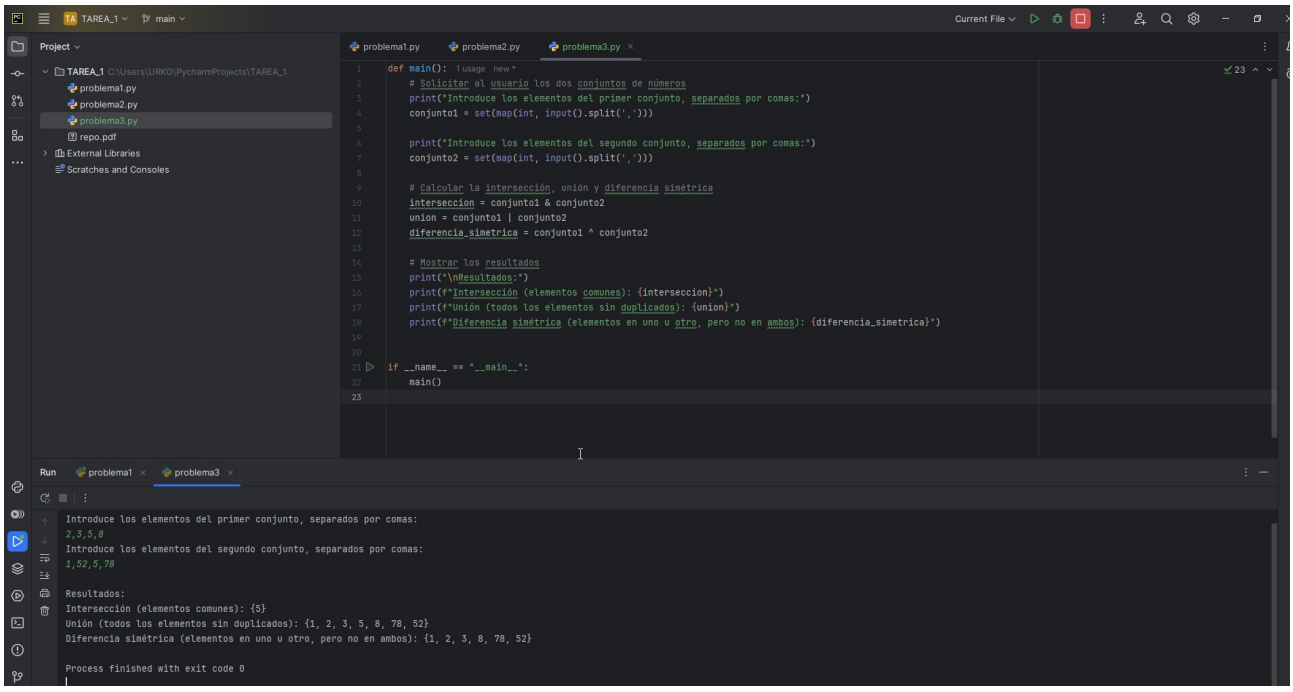

TAREA_1/problema2.py at main
Página Principal | @VANZA
ChatGPT
github.com/MENSAKA-dev/TAREA_1/blob/main/problema2.py
Files
main
Go to file
.idea
problema1.py
problema2.py
repo.pdf
TAREA_1 / problema2.py
MENSAKA-dev segundo ejercicio
37da8c8 · now
History
Code
Blame
33 lines (24 loc) · 916 Bytes
Code 55% faster with GitHub Copilot
Raw
import string
def contar_palabras(texto):
# Eliminar signos de puntuación y convertir a minúsculas
texto = texto.translate(str.maketrans('', '', string.punctuation)).lower()
# Dividir el texto en palabras
palabras = texto.split()
# Contar frecuencias usando un diccionario
frecuencias = {}
for palabra in palabras:
if palabra in frecuencias:
frecuencias[palabra] += 1
else:
frecuencias[palabra] = 1
# Devolver las palabras ordenadas por su clave
return dict(sorted(frecuencias.items()))
if __name__ == "__main__":
# Solicitar entrada del usuario
texto = input("Introduce una frase o párrafo: ")
# Contar las palabras
frecuencias = contar_palabras(texto)
# Mostrar resultados
print("\nFrecuencia de palabras:")
for palabra, frecuencia in frecuencias.items():
print(f"{palabra}: {frecuencia}")

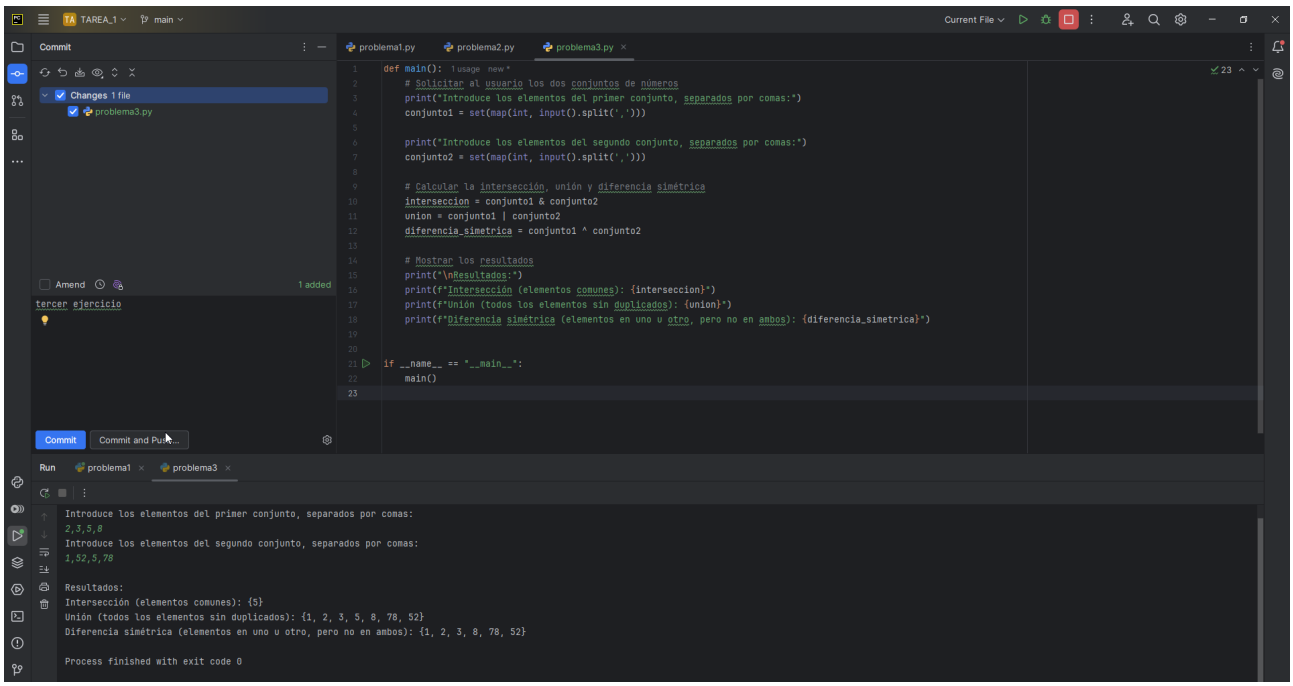
Problema 3. Intersección y unión de conjuntos

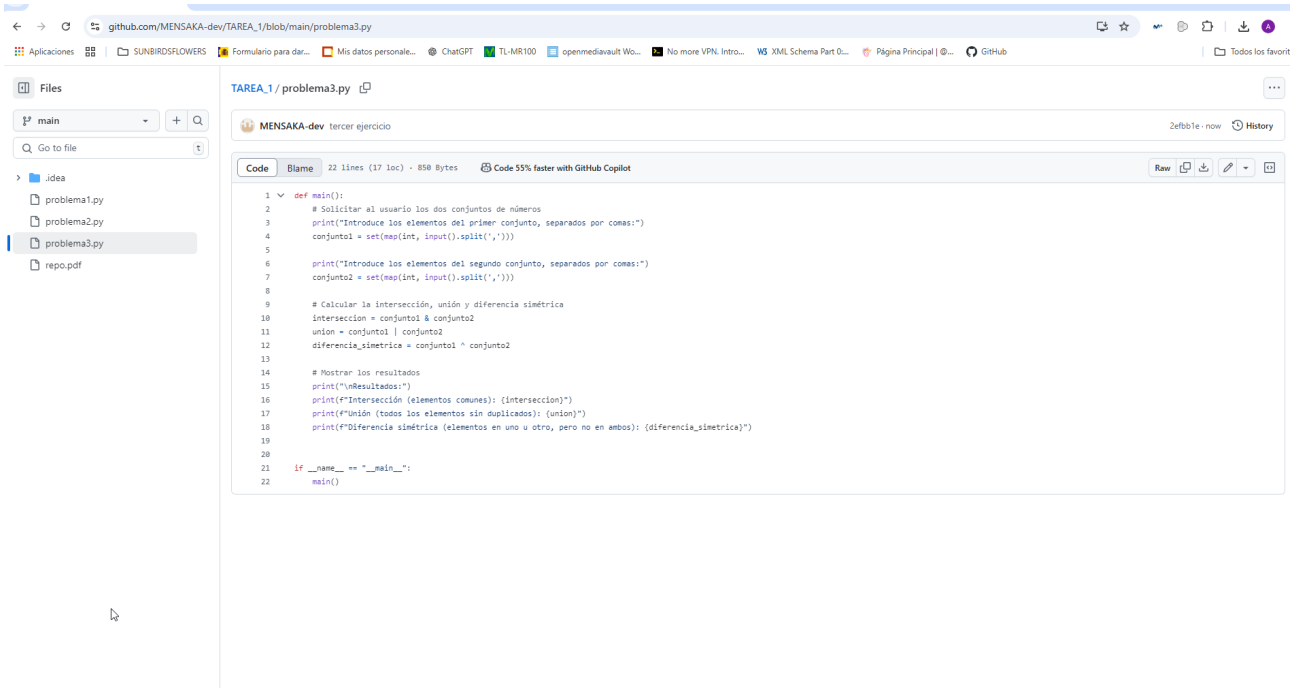
Añadimos el código para el tercer ejercicio.

```
def main():
    # Solicitar al usuario los dos conjuntos de números
    print("Introduce los elementos del primer conjunto, separados por comas:")
    conjunto1 = set(map(int, input().split(',')))

    print("Introduce los elementos del segundo conjunto, separados por comas:")
    conjunto2 = set(map(int, input().split(',')))

    # Calcular la intersección, unión y diferencia simétrica
    interseccion = conjunto1 & conjunto2
    union = conjunto1 | conjunto2
    diferencia_simetrica = conjunto1 ^ conjunto2

    # Mostrar los resultados
    print("\nResultados:")
    print(f"Intersección (elementos comunes): {interseccion}")
    print(f"Unión (todos los elementos sin duplicados): {union}")
    print(f"Diferencia simétrica (elementos en uno u otro, pero no en ambos): {diferencia_simetrica}")


if __name__ == "__main__":
    main()
```

Realizamos Commit and Push.

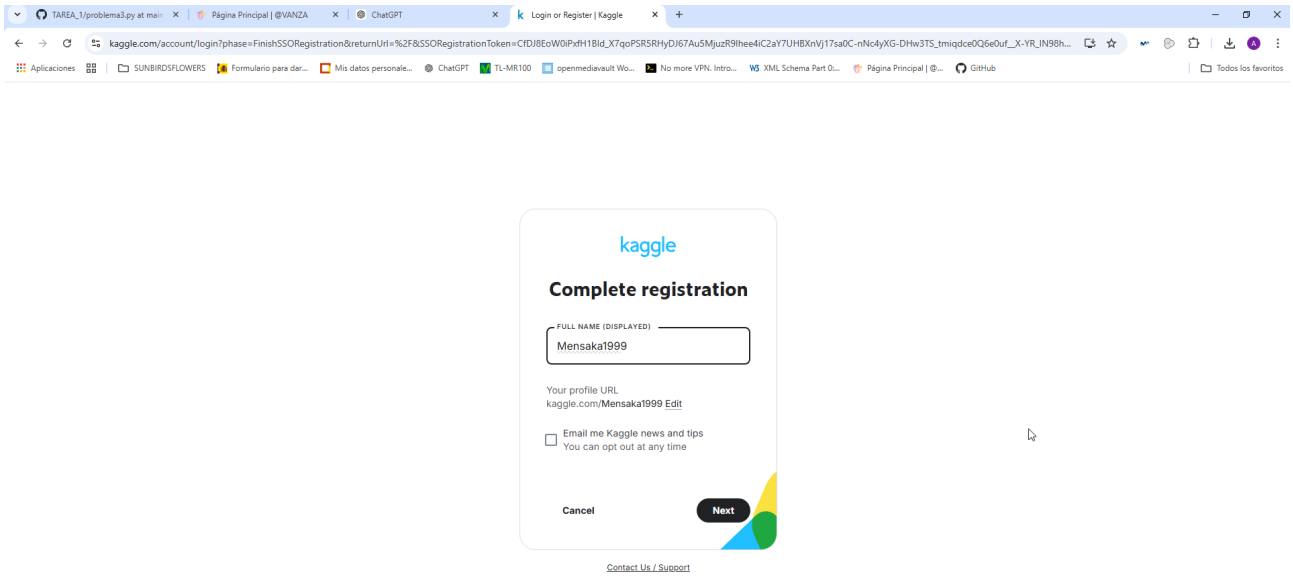



TAREA_1
main
Current File
Commit
problema1.py
problema2.py
problema3.py
Changes 1 file
problema3.py
Amend
1 added
tercer ejercicio
Commit
Commit and Push...
def main(): 1 usage new *
# Solicitar al usuario los dos conjuntos de números
print("Introduce los elementos del primer conjunto, separados por comas:")
conjunto1 = set(map(int, input().split(',')))
print("Introduce los elementos del segundo conjunto, separados por comas:")
conjunto2 = set(map(int, input().split(',')))
# Calcular la intersección, unión y diferencia simétrica
interseccion = conjunto1 & conjunto2
union = conjunto1 | conjunto2
diferencia_simetrica = conjunto1 ^ conjunto2
# Mostrar los resultados
print("\nResultados:")
print(f"Intersección (elementos comunes): {interseccion}")
print(f"Unión (todos los elementos sin duplicados): {union}")
print(f"Diferencia simétrica (elementos en uno u otro, pero no en ambos): {diferencia_simetrica}")
if __name__ == "__main__":
main()
Run
problema1
problema3
Introduce los elementos del primer conjunto, separados por comas:
2,3,5,8
Introduce los elementos del segundo conjunto, separados por comas:
1,52,5,78
Resultados:
Intersección (elementos comunes): {5}
Unión (todos los elementos sin duplicados): {1, 2, 3, 5, 8, 78, 52}
Diferencia simétrica (elementos en uno u otro, pero no en ambos): {1, 2, 3, 8, 78, 52}
Process finished with exit code 0

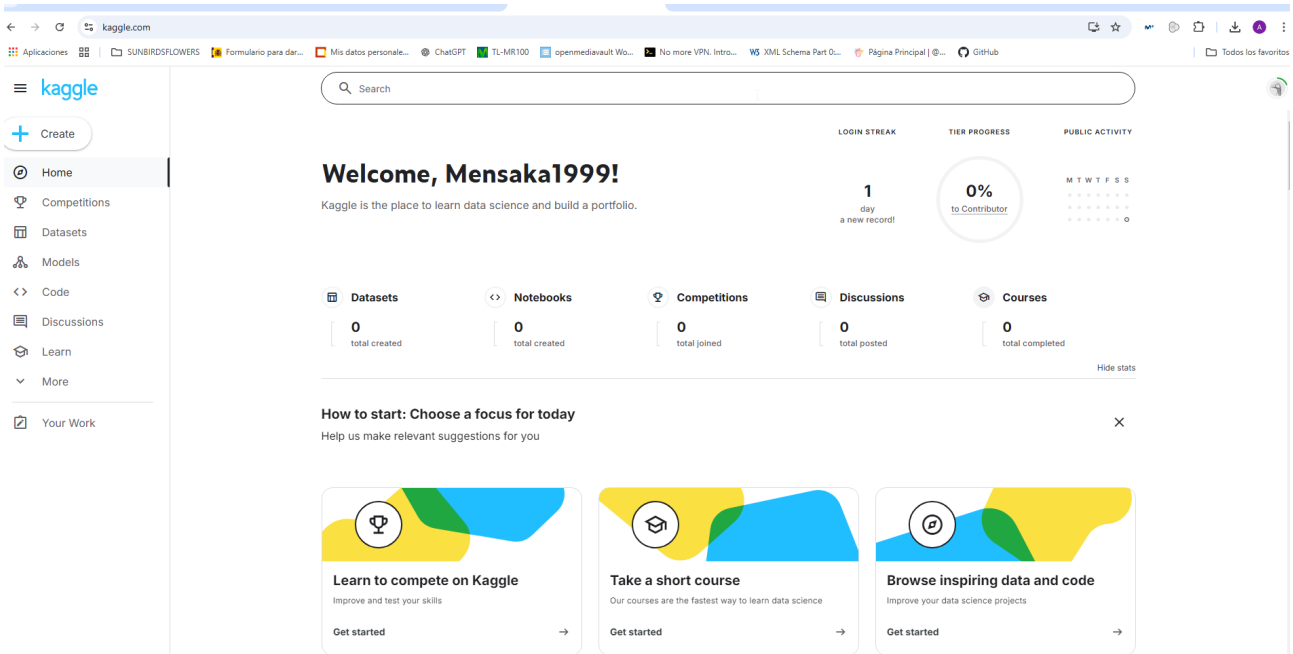

github.com/MENSAKA-dev/TAREA_1/blob/main/problema3.py
Files
main
Go to file
.idea
problema1.py
problema2.py
problema3.py
repo.pdf
TAREA_1 / problema3.py
MENSAKA-dev tercer ejercicio
2efbb1e · now
History
Code
Blame
22 lines (17 loc) · 850 Bytes
Code 55% faster with GitHub Copilot
Raw
def main():
# Solicitar al usuario los dos conjuntos de números
print("Introduce los elementos del primer conjunto, separados por comas:")
conjunto1 = set(map(int, input().split(',')))
print("Introduce los elementos del segundo conjunto, separados por comas:")
conjunto2 = set(map(int, input().split(',')))
# Calcular la intersección, unión y diferencia simétrica
interseccion = conjunto1 & conjunto2
union = conjunto1 | conjunto2
diferencia_simetrica = conjunto1 ^ conjunto2
# Mostrar los resultados
print("\nResultados:")
print(f"Intersección (elementos comunes): {interseccion}")
print(f"Unión (todos los elementos sin duplicados): {union}")
print(f"Diferencia simétrica (elementos en uno u otro, pero no en ambos): {diferencia_simetrica}")
if __name__ == "__main__":
main()

Apartado 3: Consultar competición en plataforma de IA Kaggle

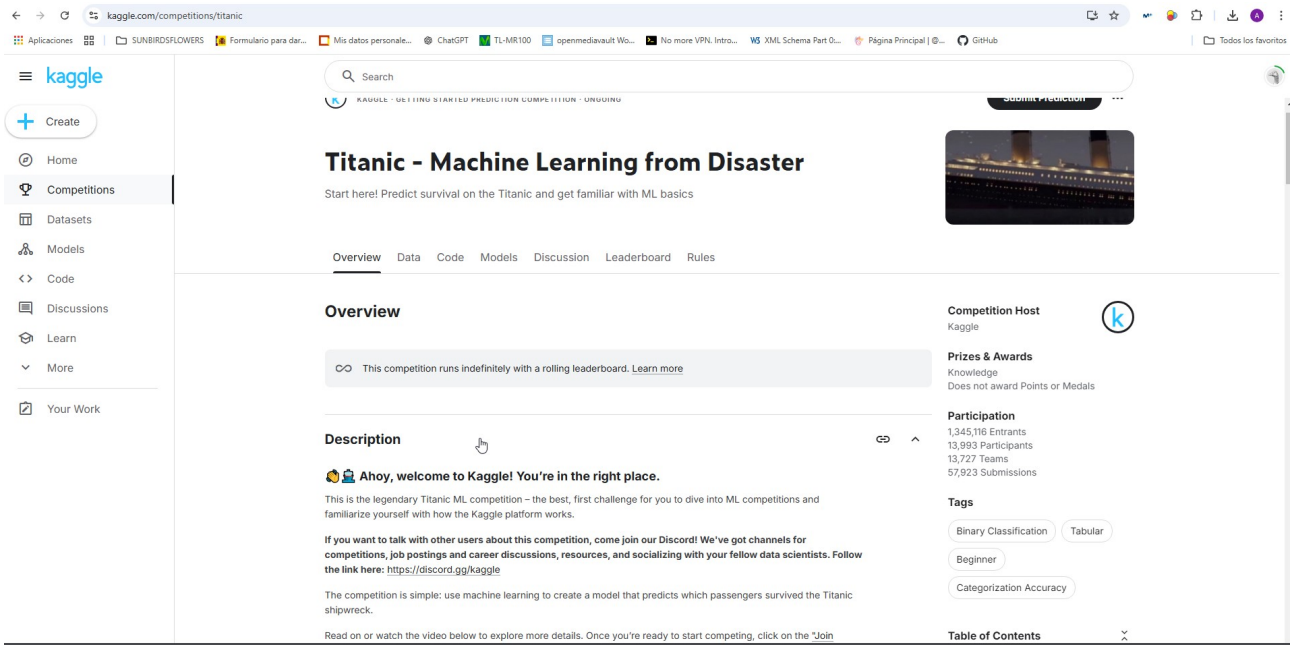
Iniciamos el registro en Kaggle y lo hacemos a través de nuestra cuenta de GOOGLE.



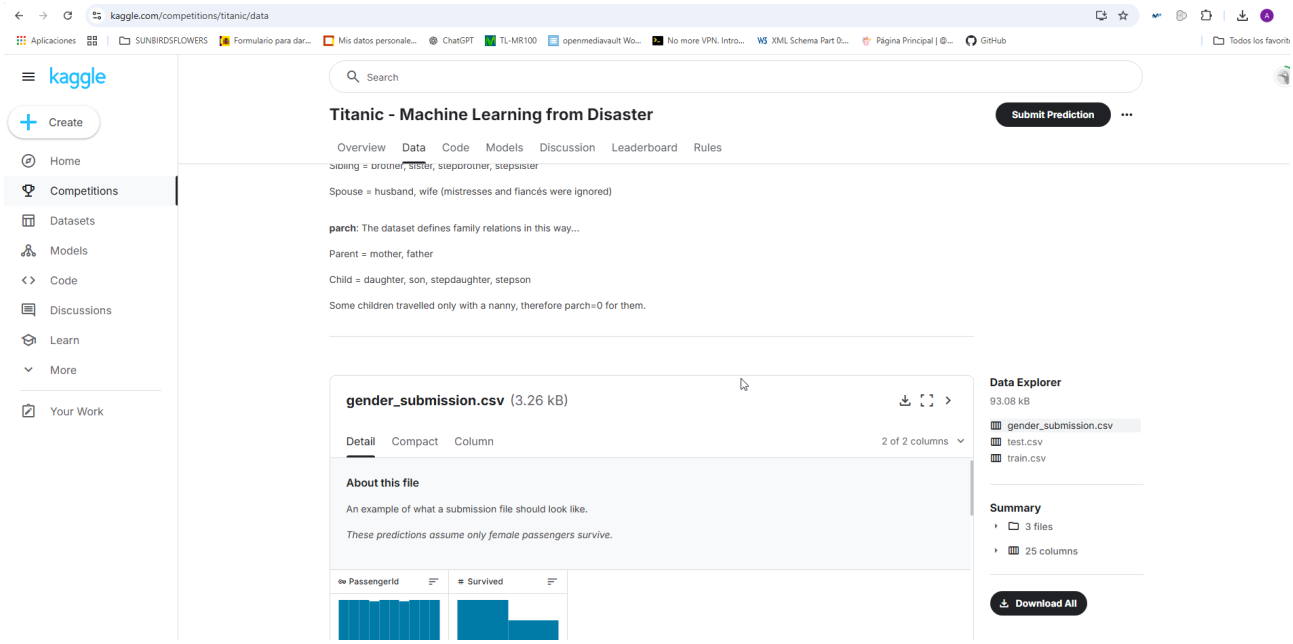
Aparece la pantalla de bienvenida.



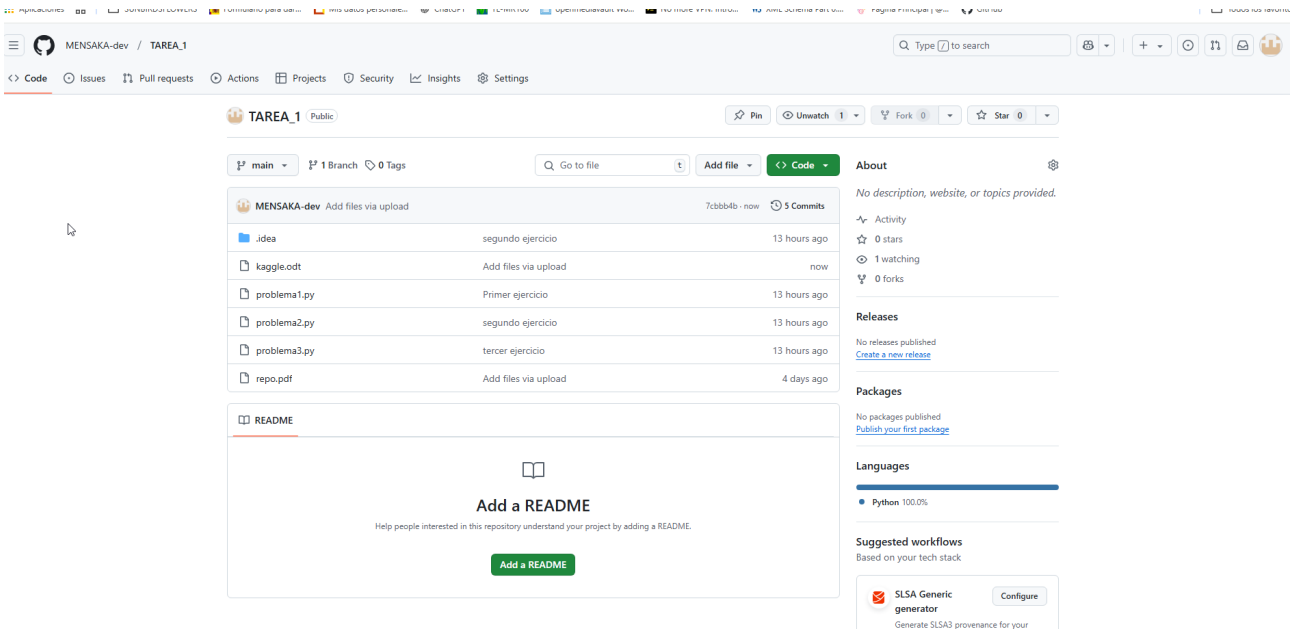
Pulsamos en la pestaña Competitions.



Dentro de la competición, vamos a la sección de data y descargamos el dataset



Subimos el documento de como hemos realizado el proceso a Github



Y también subimos el archivo del dataset