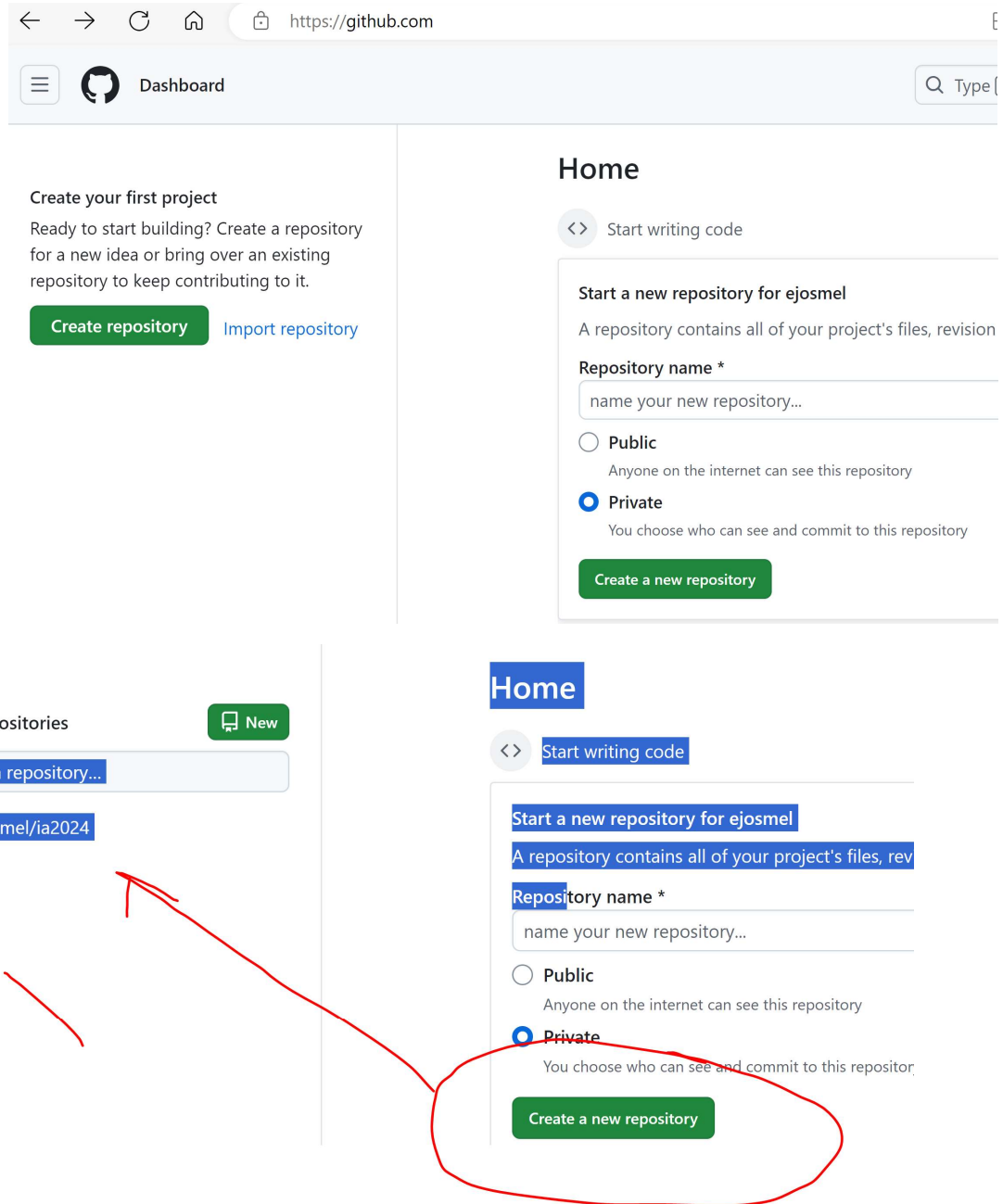


Melendez_Vazquez_josemanuel_PIA01_Tarea.pdf

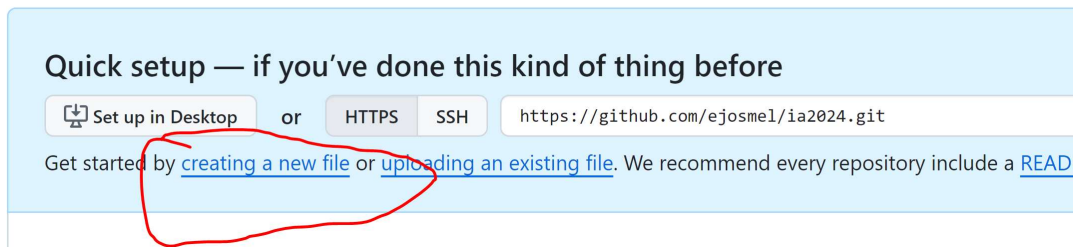
Solucion: url del Repositorio: <https://github.com/ejosmel/ia2025>

- **Apartado 1: Crear cuenta en GitHub y crear un repositorio. (2 puntos)**

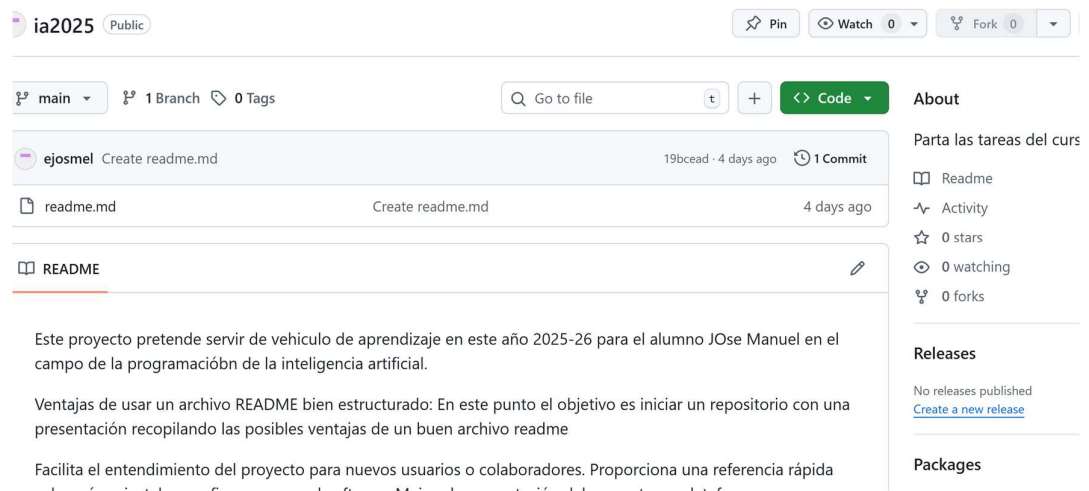
Accedemos a la página web de **GitHub** y seguimos los pasos para registrarte (crear usuario, correo (ejosmel@gmail.com) y password) y crear una cuenta. Usamos la opción "skip personaliza on".



Parece que de lo primero que hay que hacer es dar un nombre al repositorio y crear un fichero con la intención (tipo MD: Readme file)



Incluimos un archivo de presentación readme. md como se sugiere en GitHub



Posteriormente subo un archivo pdf explicando muy brevemente el paso0 (tal como se solicita en la tarea)

Nota: En primer lugar no es intuitivo crear estructuras de ficheros ni tampoco la subida que hay que realizarla con el botón + y la creación de estructuras insertando la carpeta en el nombre seguida de "/"

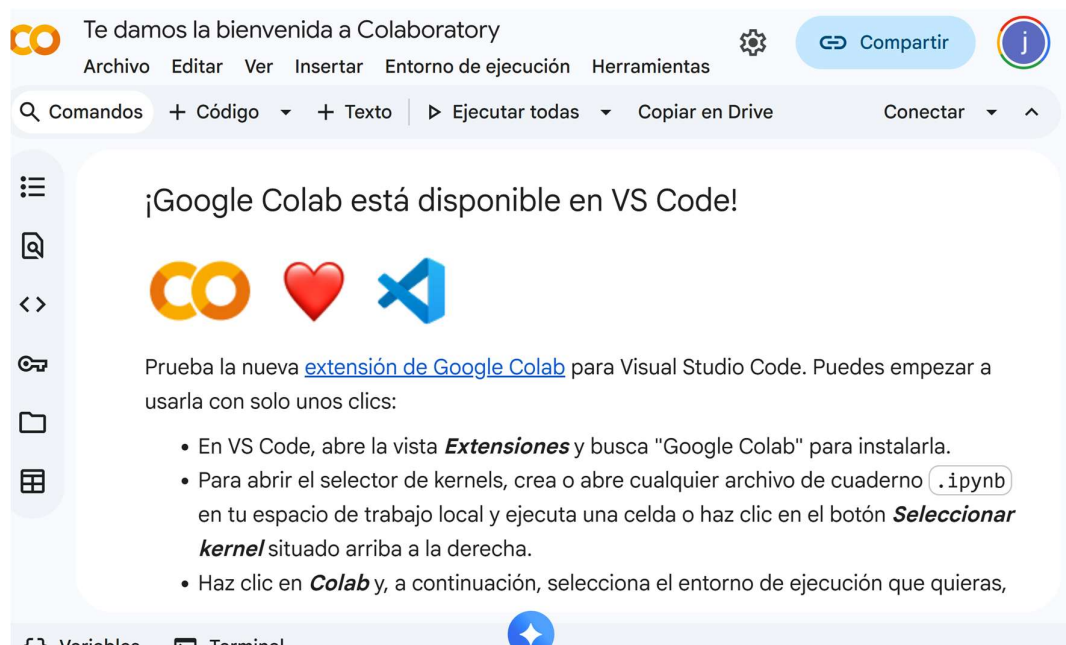
- **Apartado 2: Resolver ciertos problemas en Python (2 puntos cada uno)**

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 python sin instalar nada en mi pc. Voy a usar Google Colab, que (una vez logged in con credenciales Google) permite guardar en mi disco en la nube Google-drive el trabajo, pero creo unas celdas de ejecución (idea de jupyter notebooks donde se ejecuta el programa python dentro del navegador).

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. El entorno es un espacio web en <https://colab.research.google.com/>

Ayuda en [Google Colab Basics](#)



Te damos la bienvenida a Colaboratory

Archivo Editar Ver Insertar Entorno de ejecución Herramientas

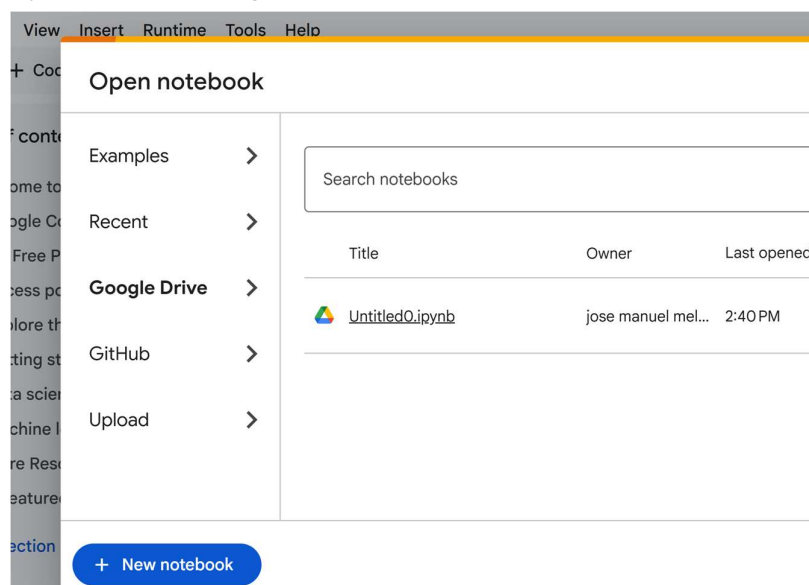
Comandos + Código + Texto ▶ Ejecutar todas Copiar en Drive Conectar

¡Google Colab está disponible en VS Code!

Prueba la nueva [extensión de Google Colab](#) para Visual Studio Code. Puedes empezar a usarla con solo unos clics:

- En VS Code, abre la vista **Extensiones** y busca "Google Colab" para instalarla.
- Para abrir el selector de kernels, crea o abre cualquier archivo de cuaderno `.ipynb` en tu espacio de trabajo local y ejecuta una celda o haz clic en el botón **Seleccionar kernel** situado arriba a la derecha.
- Haz clic en **Colab** y, a continuación, selecciona el entorno de ejecución que quieras,

Open notebook /Google Drive



View Insert Runtime Tools Help

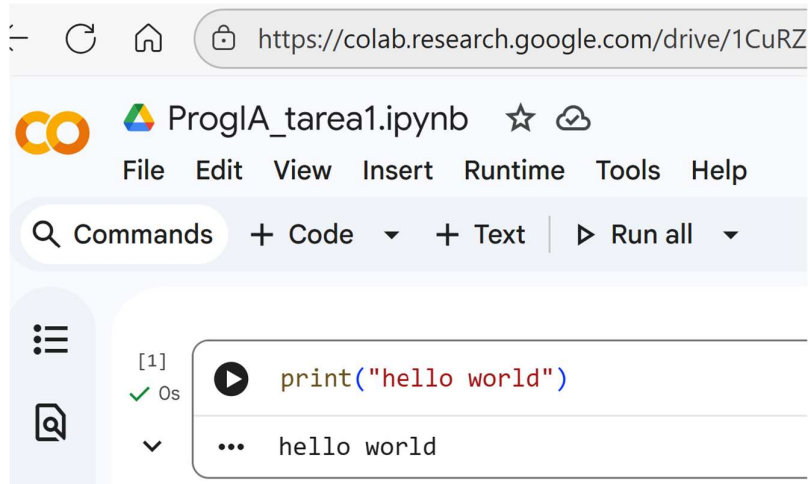
Open notebook

Search notebooks

Title	Owner	Last opened
Untitled0.ipynb	jose manuel mel...	2:40 PM

+ New notebook

Que renombramos y probamos esto de las celdas de ejecución...



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]`.

```
def procesar_lista_enteros(lista_original):
```

```
    # 1. Eliminar los números negativos de la lista.
```

```
    # Filtramos solo los números que son mayores o iguales a cero.
```

```
    numeros_positivos = [num for num in lista_original if num >= 0]
```

```
    # 2. Eliminar los valores que están repetidos, quedándonos con uno de ellos.
```

```
    # Convertimos la lista a un conjunto para eliminar duplicados automáticamente.
```

```
    # Luego, la convertimos de nuevo a una lista.
```

```
    lista_sin_duplicados = list(set(numeros_positivos))
```

```
    # 3. Ordenar los números resultantes de menor a mayor.
```

```
    # Usamos la función sorted() para ordenar la lista.
```

```
    lista_final_ordenada = sorted(lista_sin_duplicados)
```

```
    return lista_final_ordenada
```

Nota: La función esta entre def y return. Una secuencia de números separados por “,” no es una lista si no está metido en []. Se puede hacer un recorrido por una lista seleccionando elementos a partir de una condición con una forma compacta de for, in, (sin :) if generando otra lista []. La función set() se aplica a una lista y la función list() se a

```

+
+# Ejemplo de uso:
+mi_lista = [4, -1, 2, 4, 3, -5, 2]
+# Se corrigió el error: los argumentos deben pasarse como una lista.
+resultado = procesar_lista_enteros(mi_lista)
+print(f"La lista original es: {mi_lista}")
+print(f"La lista procesada es: {resultado}")
+
+

```

```

.. La lista original es: [4, -1, 2, 4, 3, -5, 2]
   La lista procesada es: [2, 3, 4]

```

○ 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. 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.

```

def contar_palabras_fichero(lista_palabras, ruta_fichero):
    # 1. recuperamos el contenido del fichero en la variable contenido pasandolo a
    minúsculas
    try:
        with open(ruta_fichero, 'r', encoding='utf-8') as f:
            contenido = f.read().lower()
    except FileNotFoundError:
        print(f"Error: El archivo '{ruta_fichero}' no fue encontrado.")
        return {}

    # Reemplazar signos de puntuación comunes con espacios y dividir
    for signo in [",", ".", ":", ";", "!", "?", "(", ")"]:
        contenido = contenido.replace(signo, " ")

    # creamos diccionario frecuencias a 0 para las palabras de interés
    lista_palabras_lower = [palabra.lower() for palabra in lista_palabras]
    frecuencias = {palabra: 0 for palabra in lista_palabras_lower}

    # 3. Usar un diccionario donde la clave sea la palabra y el valor su frecuencia.
    lista_contenido = contenido.split()
    for palabra in lista_contenido:
        if palabra in frecuencias:

```

```
frecuencias[palabra] += 1
```

```
return frecuencias
```

```
# --- Pequeño programa para poner a prueba la función ---
```

```
# 1. Crear un archivo de texto de ejemplo
```

```
nombre_fichero_prueba = "mi_fichero_ejemplo.txt"
```

```
contenido_ejemplo = """
```

```
Hola mundo, este es un ejemplo de texto para probar la función.
```

```
El mundo es grande y hola es una palabra común.
```

```
Ejemplo, ejemplo, MUNDO.
```

```
"""
```

```
with open(nombre_fichero_prueba, 'w', encoding='utf-8') as f:
```

```
    f.write(contenido_ejemplo)
```

```
print(f"Archivo '{nombre_fichero_prueba}' creado con éxito.")
```

```
# 2. Definir la lista de palabras a buscar
```

```
palabras_a_buscar = input("palabras a buscar separadas por comas").split()
```

```
# 3. Llamar a la función
```

```
resultado_frecuencias = contar_palabras_fichero(  
    palabras_a_buscar, nombre_fichero_prueba  
)
```

```
# 4. Mostrar las palabras y sus frecuencias de forma ordenada por la palabra.
```

```
print("\nFrecuencia de palabras en el archivo (ordenado alfabéticamente):")
```

```
if resultado_frecuencias:
```

```
    for palabra, frecuencia in sorted(resultado_frecuencias.items()):
```

```
        print(f"'{palabra}': {frecuencia} veces")
```

```
else:
```

```
    print("No se encontraron las palabras buscadas o el archivo está vacío/no existe.")
```

```
+
```

```
✓ Archivo 'mi_fichero_ejemplo.txt' creado con éxito.
```

```
palabras a buscar separadas por coma hola mundo
```

```
Frecuencia de palabras en el archivo (ordenado alfabéticamente):
```

```
'hola': 2 veces
```

```
'mundo': 3 veces
```

○ 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).

```
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) función and
```

```
    interseccion = conjunto1 & conjunto2
```

```
    # Calcular la unión (todos los elementos sin duplicados) función or
```

```
    union = conjunto1 | conjunto2
```

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

```
    diferencia_simetrica = conjunto1 ^ conjunto2 (función xor)
```

```
    # 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

```
procesar_conjuntos()
```

```
... Introduce el primer conjunto de números separados por comas: 1,2,3,5,7,9
Introduce el segundo conjunto de números separados por comas: 2,3,4,5,8,9,0
Intersección: [2, 3, 5, 9]
Unión: [0, 1, 2, 3, 4, 5, 7, 8, 9]
Diferencia simétrica: [0, 1, 4, 7, 8]
```

Subimos los tres +1 programas:

Files

main

Go to file

tarea1

Paso0_ Abrir Cuenta GitHub, cre...

contar_palabras_fichero.py

procesar_conjuntos.py

procesar_lista_enteros.py

separar_ordenar_numeros.py

test_contar_file.py

readme.md

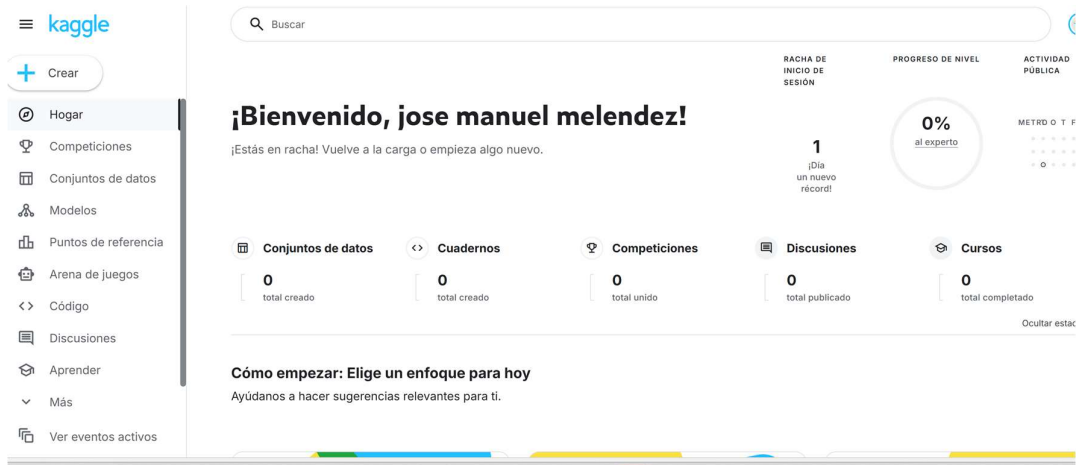
ia2025 / tarea1 /

ejosmel Add files via upload

Name
..
Paso0_ Abrir Cuenta GitHub, c
contar_palabras_fichero.py
procesar_conjuntos.py
procesar_lista_enteros.py
separar_ordenar_numeros.py

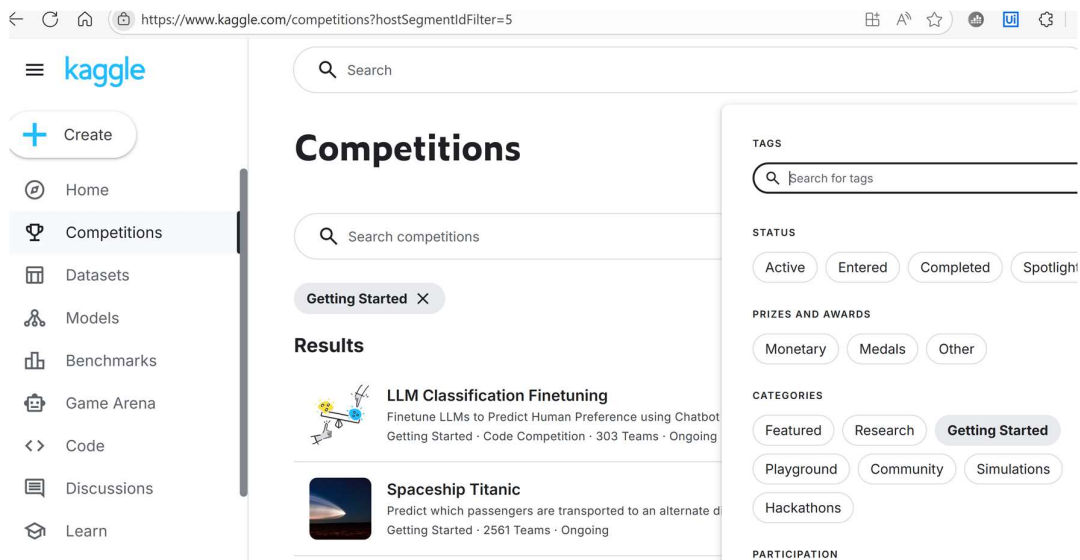
- **Apartado 3: Consultar competición en plataforma de IA Kaggle. (2 puntos)**

1.-Crea una cuenta en Kaggle



2-Accede a una competición activa

Desde el desconocimiento primeramente vemos que la plataforma tiene un buscador para las competiciones, y buscamos alguna filtrando por getting Started



Y accedemos por ejemplo a esta primera Spaceship Titanic

https://www.kaggle.com/competitions/spaceship-titanic/data

kaggle

Create

Home

Competitions

Datasets

Models

Benchmarks

Game Arena

Code

Discussions

Learn

More

Search

Spaceship Titanic

Predict which passengers are transported to an alternate dimension

Overview **Data** Code Models Discussion Leaderboard Rules

Dataset Description

In this competition your task is to predict whether a passenger was transported to an alternate dimension during the *Spaceship Titanic*'s collision with the spacetime anomaly. To help you make these predictions, you're given a set of personal records recovered from the ship's damaged computer system.

File and Data Field Descriptions

- train.csv** - Personal records for about two-thirds (~8700) of the passengers, to be used as training data.
 - PassengerId - A unique Id for each passenger. Each Id takes the form gggg_pp where gggg indicates a group

Files
3 files

Size
1.24 MB

Type
csv

License
[Attribution 4](#)

3- Descarga el dataset usado para esa competición. En la parte inferior de la página tenemos los datos y un botón de descarga previamente aceptando las condiciones y uniéndome a la competición

Spaceship Titanic

Overview **Data** Code Models Discussion Leaderboard Rules

sample_submission.csv (33.9 KB)

1.24 MB

Competition Rules

To see this data you need to agree to the [competition rules](#).
Join the competition to view the data.

Join the competition

Summary

- 3 files
- 29 columns

Download All

DOWNLOAD DATA

kaggle competitions download -c spaceship-titanic

Search

Spaceship Titanic

Overview Data Code Models Discussion Leaderboard Rules Team Submissions

sample_submission.csv (59.9 kB) [Download](#) [Details](#) [Share](#)

Data Explorer
1.24 MB
sample_submission.csv
test.csv
train.csv

Summary
3 files
29 columns

[Download All](#)

Descargas

- spaceship-titanic.zip
[Abrir archivo](#)
- EnergíaXXI-factura-6
Quitados
- fact_num_17812110
Quitados
- fact_num_17703493
Quitados
- fact_num_17595059
Quitados
- fact_num_17485415
Quitados
- WhatsApp image 20
Quitados
- WhatsApp image 20
Quitados

spaceship-titanic

WCDMA-COURSE

Whiteboards

Escritorio

Nombre

- sample_submission
- test
- train

En este mismo documento quedo reflejado el proceso y lo subo al propio repositorio Github