

Contenido

Tarea de la unidad 1 de PIA.....	2
1. ¿Qué contenidos y resultados de aprendizaje trabajaremos?	2
2. Descripción de la tarea.....	2
Apartado 1: Crear cuenta en GitHub y crear un repositorio. (2 puntos).....	2
Apartado 2: Resolver ciertos problemas en Python (2 puntos cada uno)	2
Apartado 3: Consultar competición en plataforma de IA Kaggle. (2 puntos) ...	3
3. Recursos necesarios.	3
4. Recomendaciones.	4
5. Indicaciones de entrega.....	4
6. ¿Cómo valoramos y puntuamos tu tarea?	5
7. Respuestas.....	5
Apartado 1: Crear cuenta en GitHub y crear un repositorio. (2 puntos).....	5
Apartado 2: Resolver ciertos problemas en Python (2 puntos cada uno)	12
Apartado 3: Consultar competición en plataforma de IA Kaggle. (2 puntos) .	19



Tarea de la unidad 1 de PIA

1. ¿Qué contenidos y resultados de aprendizaje trabajaremos?

Resultados de aprendizaje:

- RA1. Caracteriza lenguajes de programación valorando su idoneidad en el desarrollo de inteligencia artificial.

Contenidos:

1. Perfil del programador de IA.
 1. Conocimientos técnicos.
 2. Conocimientos matemáticos.
2. Ecosistema en el ámbito de la inteligencia artificial.
 1. Comunidad de desarrolladores.
 2. Plataformas.
 3. Competiciones y eventos.
 4. Empleo.
3. Lenguajes de programación.
 1. Octave.
 2. Python.
 3. R.
 4. Java.
 5. Otros.

2. Descripción de la tarea.

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

Accede a la página web de GitHub y sigue los pasos para registrarte y crearte una cuenta. Cuando se te pida que especifiques si eres alumno o profesor, pulsa en la opción que aparece en la parte inferior de la pantalla "skip personalization". Tras concluir el proceso de registro, crea tu primer repositorio, incluyendo, de momento, un archivo pdf en el que añadas, una breve explicación de cómo lo has hecho.

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. 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 negativos que tuviera y otra con los positivos. Ambas listas deben estar ordenadas ascendentemente.

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 convertir 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.

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).

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

Crea una cuenta en Kaggle y haz las siguientes tareas:

- Accede a una competición activa.
- Descarga el dataset usado para esa competición.
- Sube al repositorio Github del primer apartado, un documento con pantallazos de cómo has realizado el proceso y del dataset descargado.

3. Recursos necesarios.

- Ordenador personal con, al menos, 4 Gigabytes de memoria RAM
- Conexión a Internet.
- Navegador web.
- Documentación de Github: <https://docs.github.com/es>



- Si no tienes conocimientos de Python, puedes consultar cualquier tutorial de los muchos que hay en Internet, por ejemplo:
<https://www.w3schools.com/python/default.asp>
- Puedes ayudarte de herramientas como chatGPT o Copilot para ayudarte a resolver aquellas dudas puntuales que no te permitan avanzar. Este tipo de herramientas son maravillosas bien utilizadas. No tiene mucho sentido, pedirles que te hagan la tarea completa, si no entendemos lo que se está haciendo.

4. Recomendaciones.

- Antes de abordar la tarea:
 - Lee con detenimiento la unidad, consulta los enlaces para saber más, examina el material proporcionado por el profesor y aclara las dudas que te surjan con él.
 - Realiza el examen online de la unidad, y consulta nuevamente las dudas que te surjan. Solo cuando lo tengas todo claro, debes abordar la realización de la tarea.
- No olvides elaborar los documentos explicativos.

5. Indicaciones de entrega.

Una vez realizada la tarea, el envío se realizará a través de la plataforma. El archivo se nombrará siguiendo las siguientes pautas:

Apellido1_Apellido2_Nombre_PIA01_Tarea

Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños. Así por ejemplo la alumna Begoña Sánchez Mañas para la primera unidad del MP de PIA, debería nombrar esta tarea como...

sanchez_manas_begona_PIA01_Tarea

El contenido del fichero para esta primera tarea debería constar únicamente de un fichero de texto con la url del repositorio Github del apartado 1. Ahí dentro, estarán los contenidos de los tres apartados.



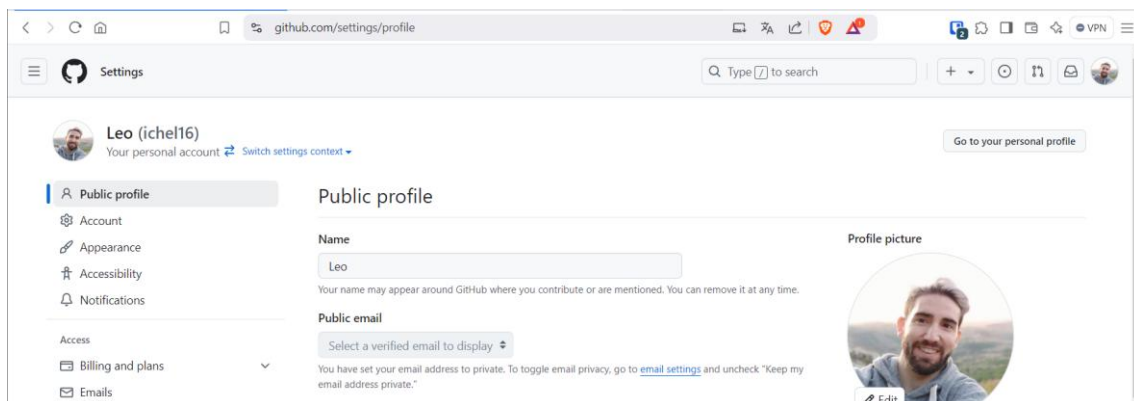
6. ¿Cómo valoramos y puntuamos tu tarea?

Rúbrica de la tarea	
Apartado 1: Se crea una cuenta en Github.	0.75 puntos
Apartado 1: Se crea un repositorio Github e incluye un documento con breve descripción del proceso.	1.25 puntos
Apartado 2: Realiza correctamente el problema 1 y lo sube al repositorio Github	2 puntos
Apartado 2: Realiza correctamente el problema 2 y lo sube al repositorio Github	2 puntos
Apartado 2: Realiza correctamente el problema 3 y lo sube al repositorio Github	2 puntos
Apartado 3: Se crea una cuenta en Kaggle	0.75 puntos
Apartado 3: Accede a una competición y se descarga el dataset, mostrando pantallazos de cómo se ha hecho el proceso y subiendo su contenido al repositorio Github.	1.25 puntos

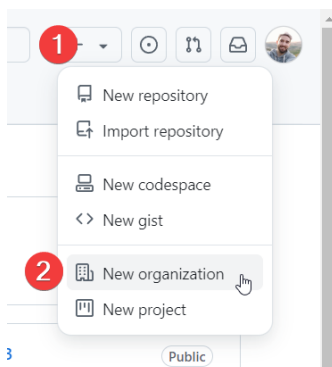
7. Respuestas.

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

Contamos con una cuenta de Github creada hace unos años:



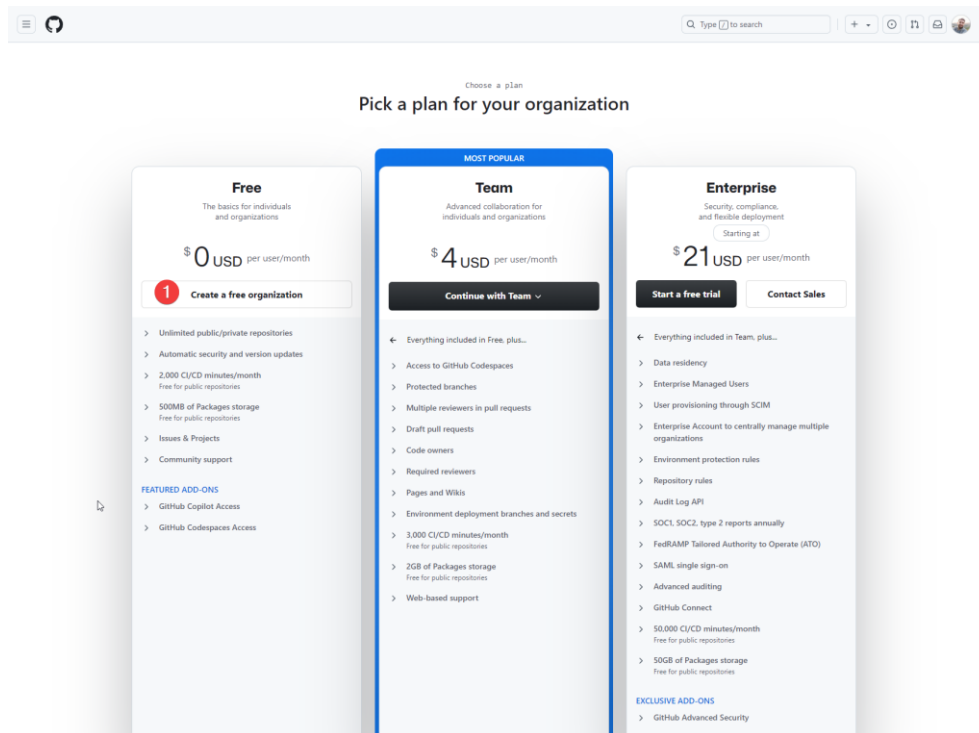
Para la creación de un repositorio, vamos a crear primero una organización, para ello:



Pulsaremos en el mas y en “New organization”



Pulsaremos en la opción sin coste:



En la siguiente ventana seguiremos los pasos:

Tell us about your organization

Set up your organization

Organization name *

ljcPIA 1

This will be the name of your account on GitHub.
Your URL will be: <https://github.com/ljcPIA>.

Contact email *

ljaraizc@gmail.com 2

This organization belongs to:

☒ My personal account 3
i.e., ichel16 (Leo)

☐ A business or institution
For example: GitHub, Inc., Example Institute, American Red Cross

Verify your account

4

Add-ons

☐ Get GitHub Copilot Business in this organization
Boost developer productivity for \$19/user/month. Pay only for assigned seats after setup.
[See Copilot Business docs.](#)

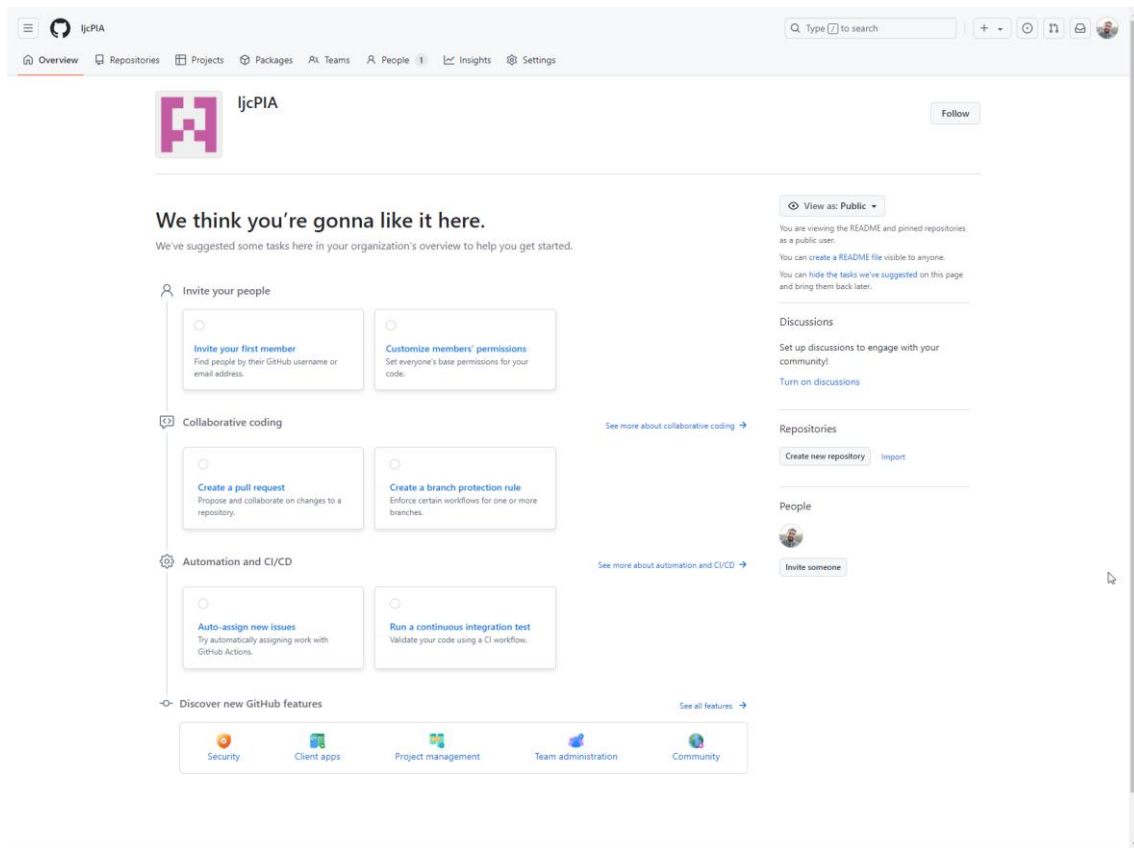
5 ☒ I hereby accept the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#).

6 Next

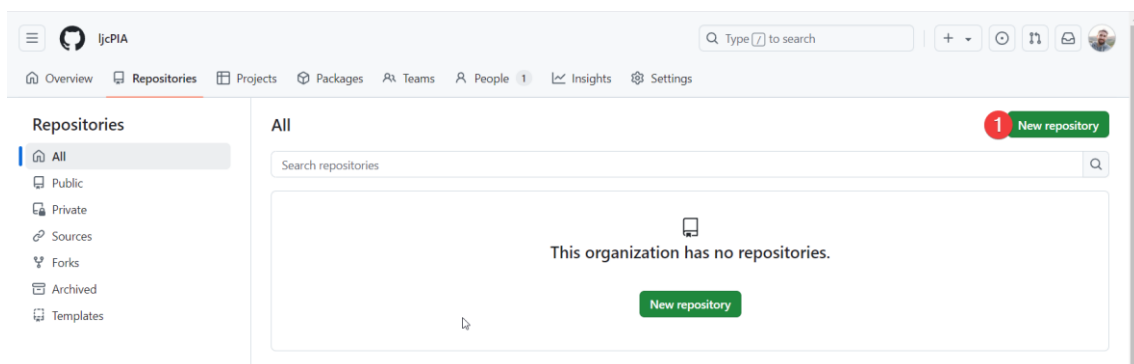
Escribimos el nombre de la organización, a mi me gusta indicar ljc+”siglas de la asignatura”, indicamos el mail, que la organización me pertenece, verificamos el captcha, aceptamos términos y verificamos.



De esta forma, tenemos un espacio reservado, donde podemos crear repositorios y cada repositorio será una tarea;



Para crear el primer repositorio, iremos a la pestaña “Repositories” y pulsamos en “New repository”



Y ahora le indicaremos un nombre, una descripción, la visibilidad la pondremos como pública para que sea accesible por terceros y pulsaremos en “Create repository”

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * ljcPIA / Repository name * Tarea01 1 Tarea01 is available.

Great repository names are short and memorable. Need inspiration? How about [animated-computing-machine](#)?

Description (optional) Primera tarea de la asignatura Programación de Inteligencia Artificial. 2

3 ☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore
.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license
License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

1 You are creating a public repository in the ljcPIA organization.

4 **Create repository**

Ya tenemos nuestro repositorio: <https://github.com/ljcPIA/Tarea01>

Tarea01 Public

[Edit Pins](#) [Watch](#) [Fork](#) [Star](#)

Set up GitHub Copilot
Use GitHub's AI pair programmer to autocomplete suggestions as you code.
[Get started with GitHub Copilot](#)

Give access to the people you work with
Ensure the right people and teams have access to this repository.
[Manage access](#)

Quick setup — if you've done this kind of thing before
[Set up in Desktop](#) or [HTTPS](#) [SSH](#) <https://github.com/ljcPIA/Tarea01.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# Tarea01" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/ljcPIA/Tarea01.git
git push -u origin main
```

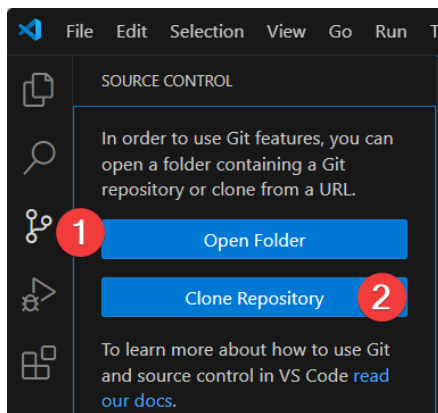
...or push an existing repository from the command line

```
git remote add origin https://github.com/ljcPIA/Tarea01.git
git branch -M main
git push -u origin main
```

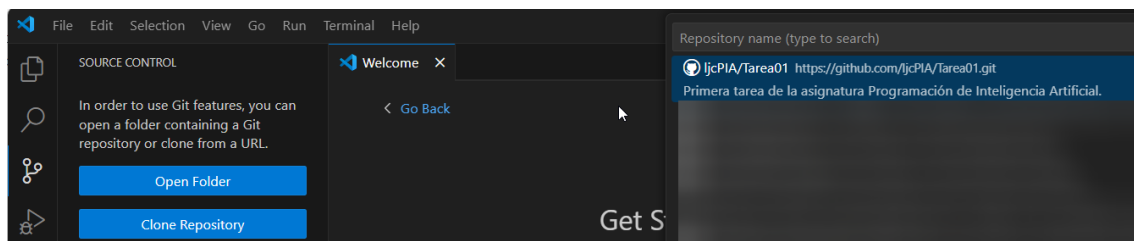
ProTip! Use the URL for this page when adding GitHub as a remote.



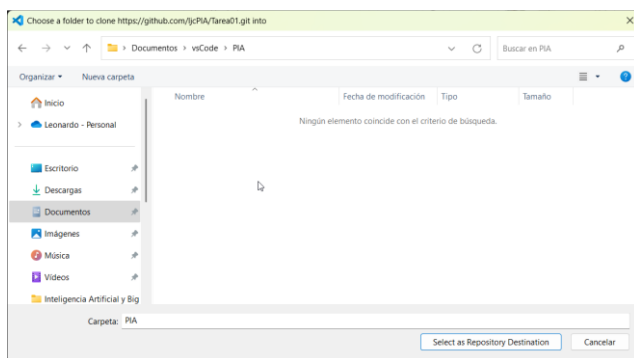
Ahora solo tenemos que inicializar nuestro repositorio, para ello nos aseguraremos de tener Git instalado en nuestro equipo, y con vsCode por ejemplo haremos lo siguiente:



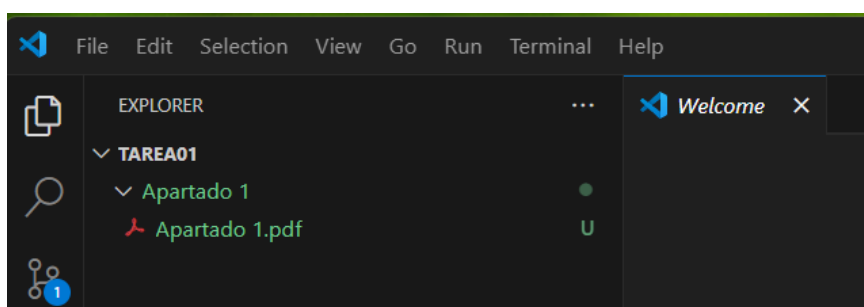
Seleccionamos el repositorio que queremos



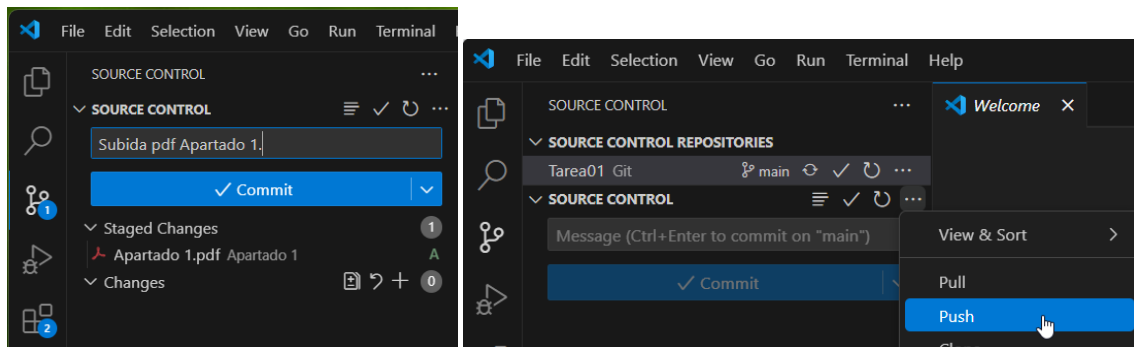
Una carpeta cualquiera de nuestro equipo:



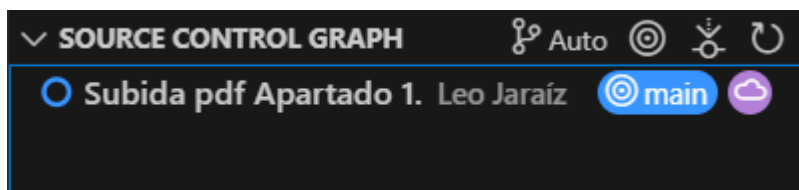
Y ahora guardamos este fichero ahí, hacemos un commit y push para subir el contenido.



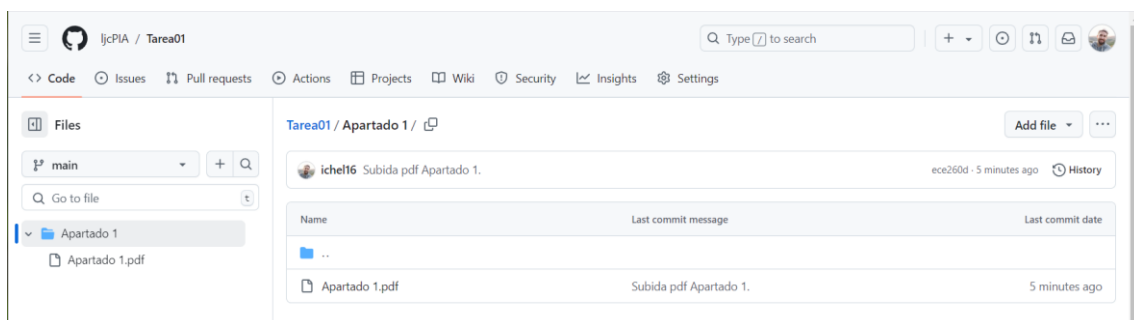
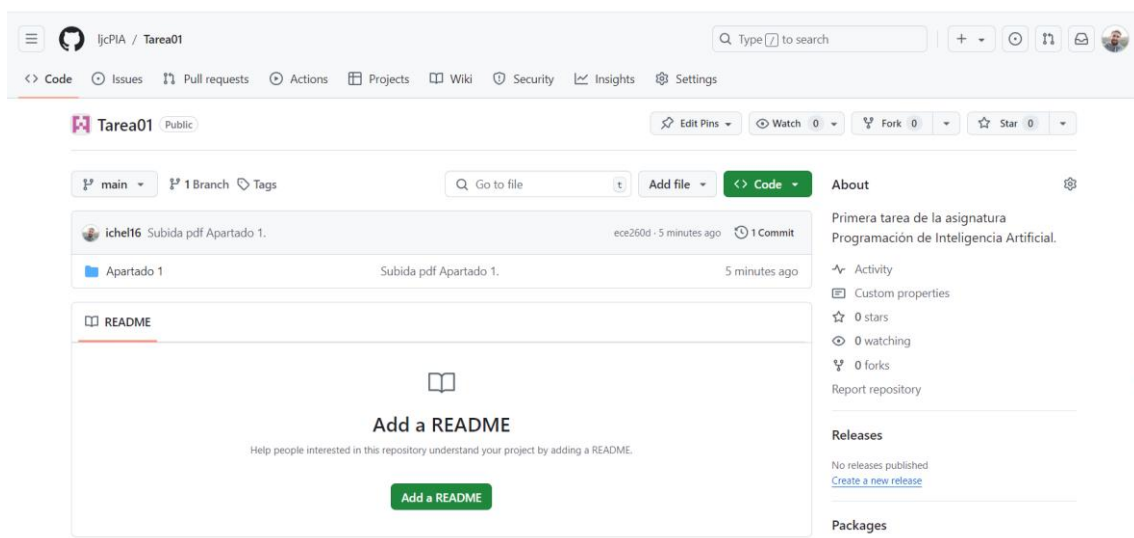
Escribimos un commit indicando los cambios que hemos hecho y hacemos push (subimos a la rama en la que estamos)

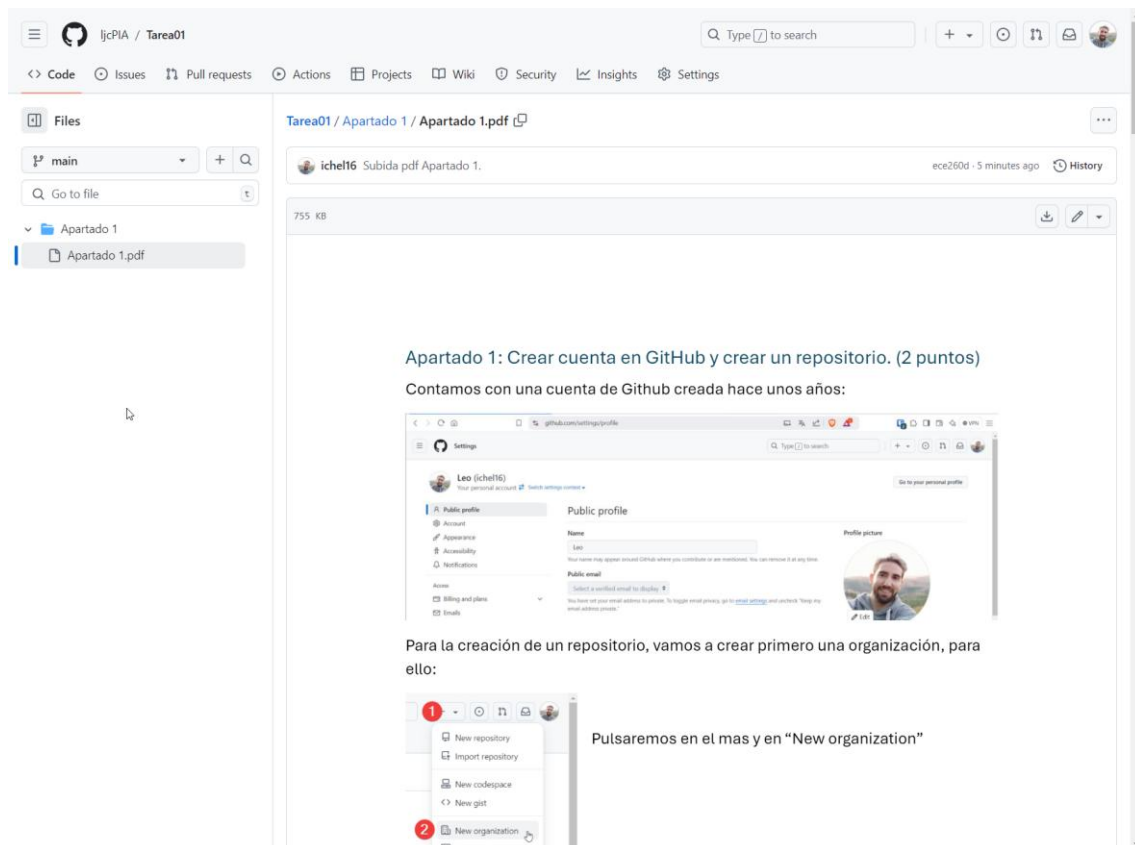


Vemos la descripción del commit, quién lo ha hecho, y a qué rama.



Si recargamos la web de github, vemos que el repositorio ya no está vacío, sino que hay una carpeta y dentro el pdf.





Subimos este cambio al pdf y hacemos commit para tener todo actualizado.



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.

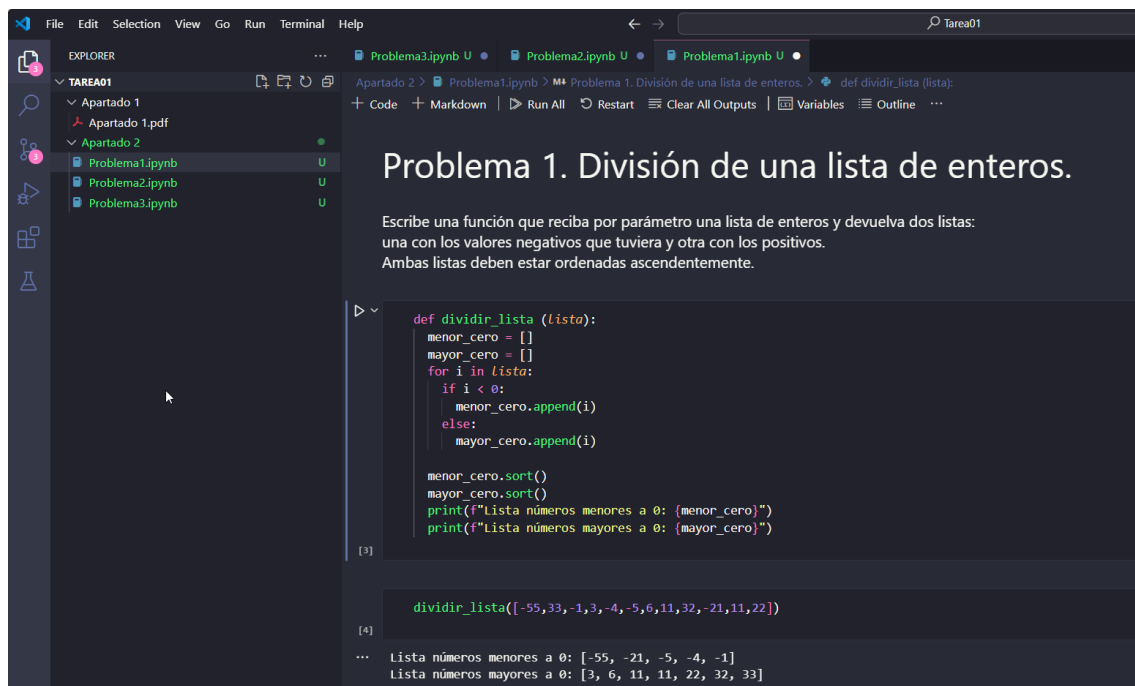
Problema 1. División de una lista de enteros.

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

Creemos una nueva carpeta para este apartado y creamos un fichero “ipynb”, donde indicamos la descripción, un método que hace lo que se pide y una muestra.

El código simplemente pregunta si el número es menor o mayor a cero, y lo añade a una lista, luego las ordena y las muestra por pantalla.

NOTA: No se tiene en cuenta el número 0, si se introduce, se descarta y no es mostrado en ninguna de las dos listas, ya que ni es mayor ni es menor a sí mismo.



```
def dividir_lista (lista):
    menor_cero = []
    mayor_cero = []
    for i in lista:
        if i < 0:
            menor_cero.append(i)
        else:
            mayor_cero.append(i)

    menor_cero.sort()
    mayor_cero.sort()
    print(f"Lista números menores a 0: {menor_cero}")
    print(f"Lista números mayores a 0: {mayor_cero}")

dividir_lista([-55,33,-1,3,-4,-5,6,11,32,-21,11,22])

...
Lista números menores a 0: [-55, -21, -5, -4, -1]
Lista números mayores a 0: [3, 6, 11, 11, 22, 32, 33]
```



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 convertir 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.

Creemos una lista con todos los signos de puntuación que se me ocurren, si encontrásemos alguno en algún texto, sería tan fácil como añadirlo a la lista y sería eliminado.

Lo primero que hacemos es eliminar los signos de puntuación, luego pasamos las palabras a minúscula, dividimos el texto en palabras aprovechándonos de los espacios y las guardamos.

Ahora recorremos la lista de palabras, si no está en el diccionario la añadimos y si está en el diccionario, actualizamos el contador sumando 1.

Ordenamos el diccionario, lo recorremos y mostramos la palabra y el número de veces que se repite.

```

Apartado 2 > Problema2.ipynb > ...
+ Code + Markdown | ▶ Run All ↺ Restart | Clear All Outputs | Variables | Outline ...

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 convertir 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):
    signos_puntuacion = [",", ".", ";", ":", "...", "-", "_", "?", "¿", "!", "!", "/", "(", ")", "<", ">", "[", "]"]
    diccionario_palabras = {}
    for i in signos_puntuacion:
        texto = texto.replace(i, "")

    texto = texto.lower()
    palabras = texto.split(" ")
    for i in palabras:
        try:
            diccionario_palabras[i] = diccionario_palabras[i] + 1
        except:
            diccionario_palabras[i]=1

    diccionario_ordenado = dict(sorted(diccionario_palabras.items()))

    for k,v in diccionario_ordenado.items():
        print(f"La palabra '{k}': {v} vez/veces en el texto.")

[4] ✓ 0.0s
```



Si ejecutamos el método, nos muestra el resultado esperado:

```
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."
contar_palabras(texto)
```

```
[4] ✓ 0.0s Python
```

```
La palabra 'al': 1 vez/veces en el texto.
La palabra 'aparece': 1 vez/veces en el texto.
La palabra 'añadimos': 1 vez/veces en el texto.
La palabra 'cada': 1 vez/veces en el texto.
La palabra 'con': 1 vez/veces en el texto.
La palabra 'contar': 1 vez/veces en el texto.
La palabra 'cuántas': 1 vez/veces en el texto.
La palabra 'debe': 1 vez/veces en el texto.
La palabra 'el': 2 vez/veces en el texto.
La palabra 'en': 1 vez/veces en el texto.
La palabra 'escribe': 1 vez/veces en el texto.
La palabra 'frase': 1 vez/veces en el texto.
La palabra 'frecuencia': 1 vez/veces en el texto.
La palabra 'ingresar': 1 vez/veces en el texto.
La palabra 'junto': 1 vez/veces en el texto.
La palabra 'las': 1 vez/veces en el texto.
La palabra 'luego': 1 vez/veces en el texto.
La palabra 'mostrar': 1 vez/veces en el texto.
La palabra 'mis': 1 vez/veces en el texto.
La palabra 'o': 1 vez/veces en el texto.
La palabra 'palabra': 1 vez/veces en el texto.
La palabra 'palabras': 2 vez/veces en el texto.
La palabra 'pida': 1 vez/veces en el texto.
La palabra 'programa': 2 vez/veces en el texto.
La palabra 'párrafo': 1 vez/veces en el texto.
La palabra 'que': 1 vez/veces en el texto.
La palabra 'su': 1 vez/veces en el texto.
La palabra 'texto': 1 vez/veces en el texto.
La palabra 'un': 1 vez/veces en el texto.
```



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).

Tendríamos un método que lo hace todo;

Para el primero, recorremos el primer conjunto, y vamos comparando cada número si aparece en el segundo conjunto, si se cumple lo añade un nuevo conjunto para mostrarlo.

Para el segundo, recorremos el primer conjunto, y vamos comparando cada valor con un nuevo conjunto, si no está lo añadimos a ese nuevo conjunto, repetimos ahora la operación, pero recorriendo el segundo conjunto, de esa forma nos quedan todos los elementos de ambos conjuntos sin duplicados.

Para el tercero, recorremos el primer conjunto y vamos verificamos uno a uno que no están en el segundo conjunto, si es así lo añadimos a un nuevo conjunto. Repetimos para el segundo conjunto, así conseguimos los elementos que están en uno u otro conjunto, pero no en ambos.

```
Apartado 2 > Problema3.ipynb > M4 Problema 3. Intersección y unión de conjuntos > def conjunto_numeros(a, b):
+ Code + Markdown | ▶ Run All ⌂ Restart ⌂ Clear All Outputs | 📄 Variables 📄 Outline ...

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 conjunto_numeros(a, b):
    conjunto_comun = []
    conjunto_union = []
    conjunto_diferencia_simetrica = []

    for i in a:
        if i in b:
            conjunto_comun.append(i)

    print(f"Mostrando los elementos comunes de ambos conjuntos -> {conjunto_comun}")

    for i in a:
        if i not in conjunto_union:
            conjunto_union.append(i)
    for i in b:
        if i not in conjunto_union:
            conjunto_union.append(i)
    print(f"Mostrando la union de ambos conjuntos -> {conjunto_union}")

    for i in a:
        if i not in b:
            conjunto_diferencia_simetrica.append(i)
    for i in b:
        if i not in a:
            conjunto_diferencia_simetrica.append(i)
    print(f"Mostrando la diferencia simétrica de ambos conjuntos -> {conjunto_diferencia_simetrica}")
```



Hay otra forma más simple de hacerlo, que es usando métodos predefinidos de Python.

```
"""Esta sería la forma rápida de hacerlo, usando métodos ya predefinidos por python en vez de bucles for"""
#interseccion = list(set(a).intersection(set(b)))
#print(f"Mostrando los elementos comunes de ambos conjuntos -> {interseccion}")

#union = list(set(a).union(set(b)))
#print(f"Mostrando la union de ambos conjuntos -> {union}")

#diferencia_simetrica = list(set(a).symmetric_difference(set(b)))
#print(f"Mostrando la diferencia simétrica de ambos conjuntos -> {diferencia_simetrica}")
```

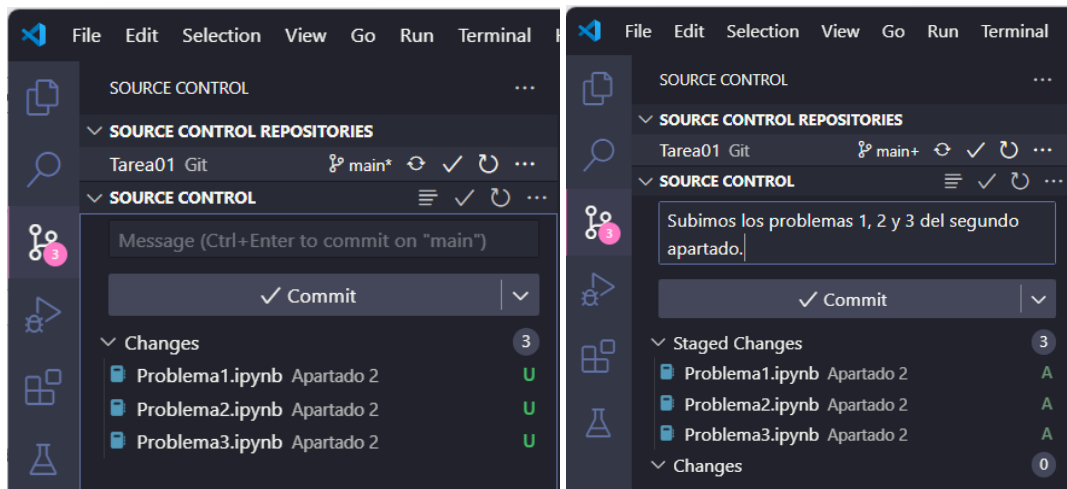
Si ejecutamos el código podemos visualizar el resultado.

```
[4] conjunto_numeros([1,2,3,4,5,6], [4,7,3,8,9,0])

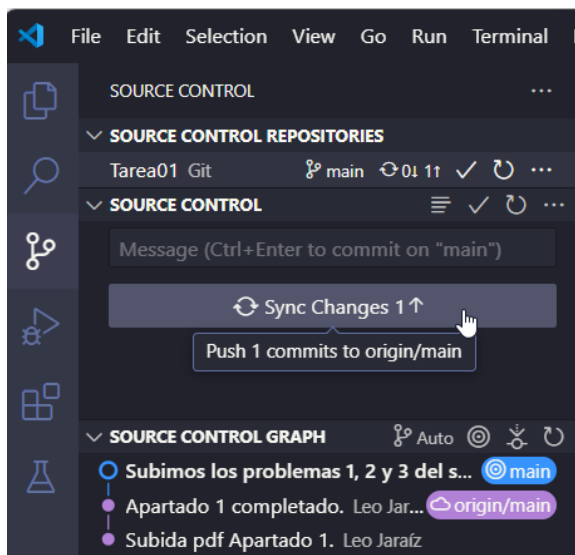
... Mostrando los elementos comunes de ambos conjuntos -> [3, 4]
    Mostrando la union de ambos conjuntos -> [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
    Mostrando la diferencia simétrica de ambos conjuntos -> [1, 2, 5, 6, 7, 8, 9, 0]
```



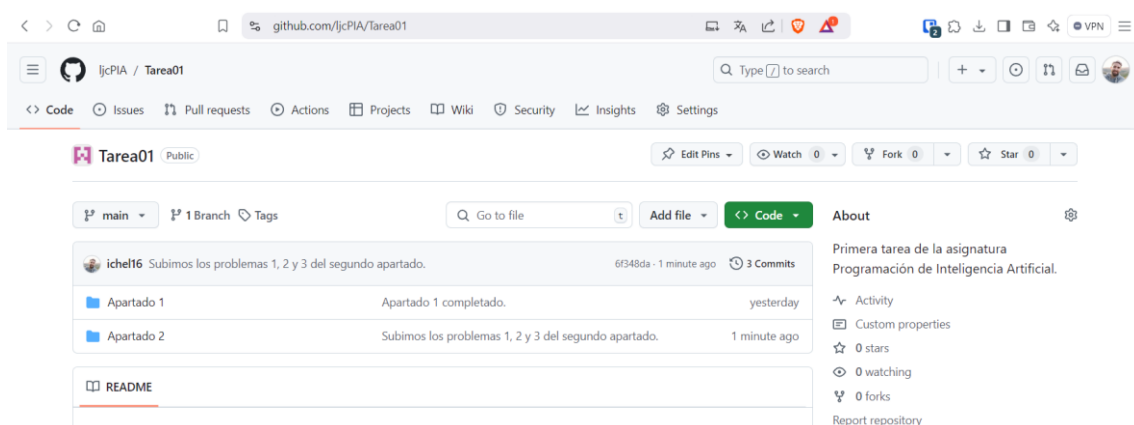
Subimos los cambios a Github para ello, nos vamos al apartado de GIT, hacemos un staged de los cambios, y escribimos un commit:



Pulsamos en Commit y una vez hecho pulsamos en “sync changes” para hacer un “push” y que los cambios se suban a Github.



Ya tenemos los ficheros guardado en Github.



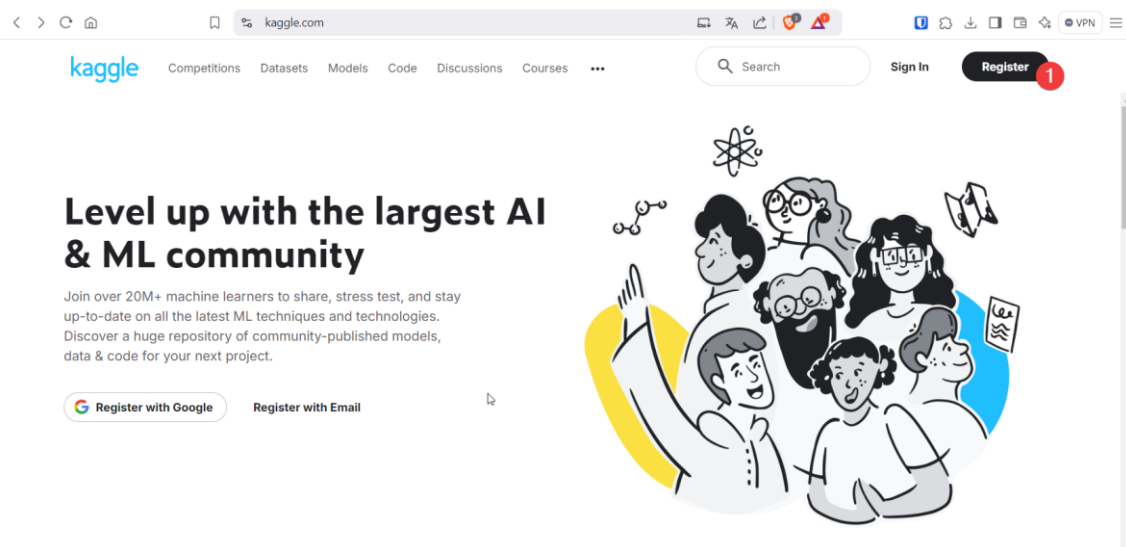


Apartado 3: Consultar competición en plataforma de IA Kaggle. (2 puntos)

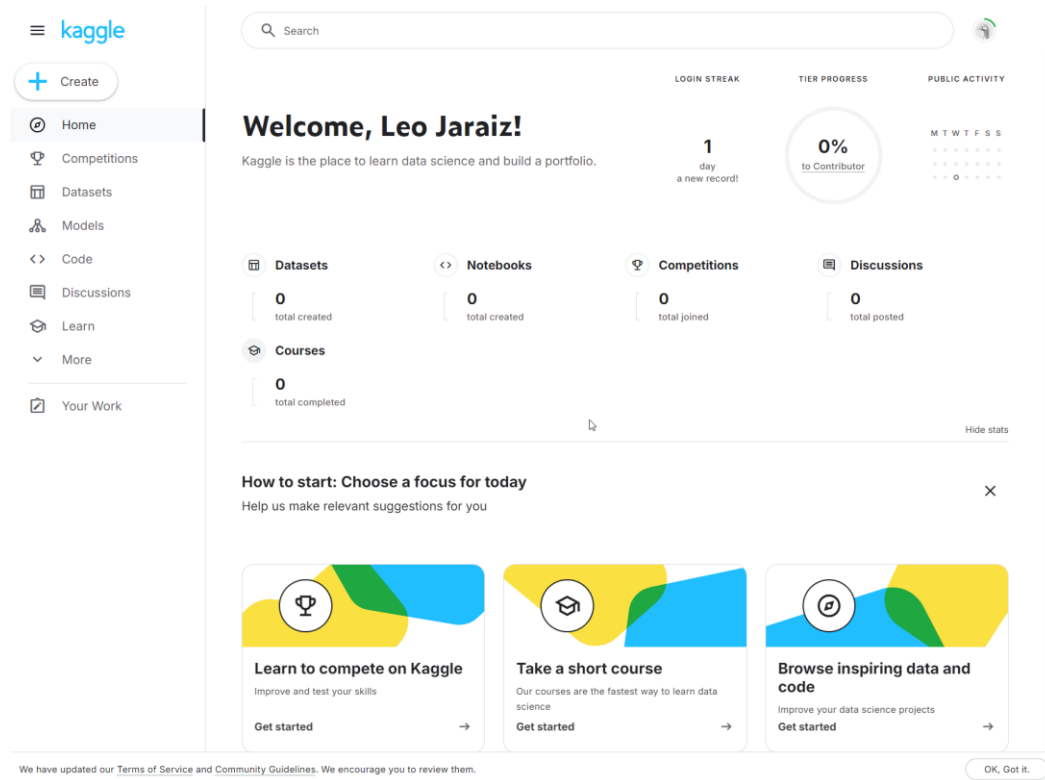
Crea una cuenta en Kaggle y haz las siguientes tareas:

- Accede a una competición activa.
- Descarga el dataset usado para esa competición.
- Sube al repositorio Github del primer apartado, un documento con pantallazos de cómo has realizado el proceso y del dataset descargado.

Creemos la cuenta en Kaggle:

A mockup of the Kaggle registration process. On the left is a 'Welcome!' screen with 'Sign In' and 'Register' tabs, and buttons for 'Register with Google' and 'Register with Email' (marked with a red '1'). Below is a link to 'Sign in' for existing users. On the right is the 'Register with email' form. It includes fields for 'EMAIL' (filled with 'iche116@gmail.com'), 'PASSWORD' (masked with dots), and 'FULL NAME' (filled with 'Leo Jaraiz'). It also features a reCAPTCHA verification step with a green checkmark and a checkbox for 'Email me Kaggle news and tips'. Navigation buttons 'Back' and 'Next' are at the bottom.

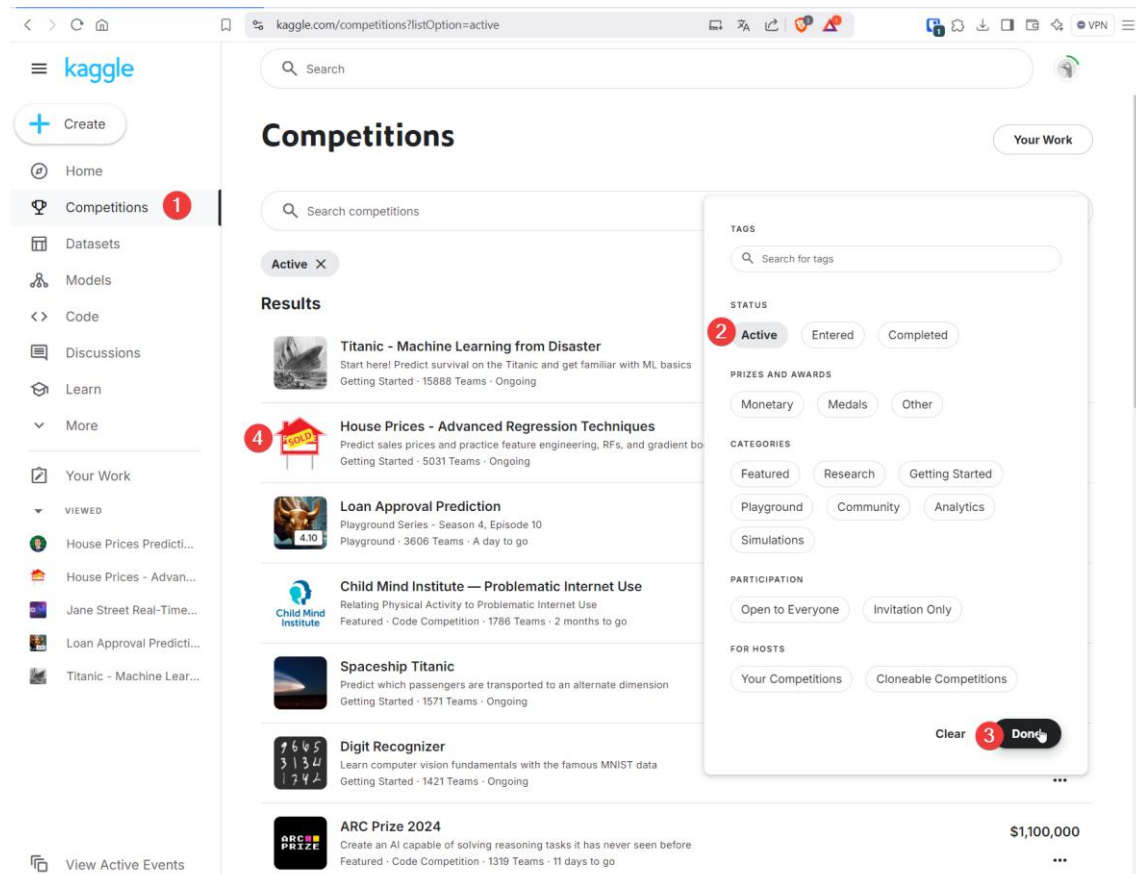
Ya tenemos nuestra cuenta creada:



The image shows the Kaggle user dashboard for a new user, Leo Jaraiz. The dashboard is divided into several sections. On the left, there is a sidebar with navigation links: Home, Competitions, Datasets, Models, Code, Discussions, Learn, More, and Your Work. The main content area features a welcome message, a search bar, and a progress bar showing a login streak of 1 day and a tier progress of 0% to Contributor. Below this, there are four cards: Datasets (0 total created), Notebooks (0 total created), Competitions (0 total joined), and Discussions (0 total posted). A 'Courses' card shows 0 total completed. A 'How to start' section suggests choosing a focus for today, with three options: 'Learn to compete on Kaggle', 'Take a short course', and 'Browse inspiring data and code'. At the bottom, there is a note about updated Terms of Service and Community Guidelines, and an 'OK, Got it.' button.

Paso 1:

buscamos y accedemos a una competición activa:



The image shows the Kaggle Competitions page. The page is titled 'Competitions' and has a search bar. A sidebar on the left shows navigation links: Home, Competitions (with a red notification badge), Datasets, Models, Code, Discussions, Learn, More, and Your Work. The main content area displays a list of competitions. The first competition is 'Titanic - Machine Learning from Disaster'. The second is 'House Prices - Advanced Regression Techniques'. The third is 'Loan Approval Prediction'. The fourth is 'Child Mind Institute - Problematic Internet Use'. The fifth is 'Spaceship Titanic'. The sixth is 'Digit Recognizer'. The seventh is 'ARC Prize 2024'. A filter panel on the right side of the page allows users to filter competitions by tags, status, prizes and awards, categories, participation, and for hosts. The 'Status' filter is set to 'Active'. The 'Prizes and Awards' filter is set to 'Monetary'. The 'Categories' filter is set to 'Featured'. The 'Participation' filter is set to 'Open to Everyone'. The 'For Hosts' filter is set to 'Your Competitions'. The 'ARC Prize 2024' competition is highlighted with a red badge and a price tag of \$1,100,000.



Al final de la web tenemos el enlace de descarga, pero deberemos unirnos a la competición para poder acceder a ellos.

The screenshot shows a file download interface. At the top, a file named **data_description.txt** (13.37 kB) is listed with download, share, and expand icons. Below this, a section titled **Competition Rules** features a laptop icon with a checkmark and the text: "To see this data you need to agree to the competition rules. Join the competition to view the data." A prominent **Join the competition** button is centered below the text. To the right, the **Data Explorer** section shows a list of files: **data_description.txt**, **sample_submission.csv**, **test.csv**, and **train.csv**. Below the file list, a **Summary** section indicates there are **4 files** and **163 columns**. At the bottom right, a **Download All** button is displayed with a red notification badge showing the number **1**.

Leemos las reglas y nos unimos.

The screenshot displays the Kaggle competition page for **House Prices - Advanced Regression Techniques**. A modal window titled **Review competition terms and conditions** is open in the center. The modal text states: "By clicking on the 'I Understand and Accept' button below, you agree to be bound by the competition rules for House Prices - Advanced Regression Techniques." It includes sections for **COMPETITION-SPECIFIC RULES** and **18. GOVERNING LAW**, which specifies that the competition is governed by California law. At the bottom of the modal, there are two radio buttons: one for **I agree to the Competition-specific rules above and Kaggle's Foundational Competition Rules...** (which is selected) and another for **I understand joining this Competition creates a direct relationship between the Host and me...**. Below these options are **Decline** and **I Understand and Accept** buttons. The background shows the competition details, including a **Join Competition** button and a list of rules on the right side of the page.



Ahora sí podemos descargar los ficheros:

House Prices - Advanced Regression Techniques

Overview Data Code Models Discussion Leaderboard Rules Team Submissions

data_description.txt (13.37 kB)

MSSubClass: Identifies the type of dwelling involved in the sale.

20	1-STORY 1946 & NEWER ALL STYLES
30	1-STORY 1945 & OLDER
40	1-STORY W/FINISHED ATTIC ALL AGES
45	1-1/2 STORY - UNFINISHED ALL AGES
50	1-1/2 STORY FINISHED ALL AGES
60	2-STORY 1946 & NEWER
70	2-STORY 1945 & OLDER
75	2-1/2 STORY ALL AGES
80	SPLIT OR MULTI-LEVEL
85	SPLIT FOYER
90	DUPLEX - ALL STYLES AND AGES
120	1-STORY PUD (Planned Unit Development) - 1946 & NEWER
150	1-1/2 STORY PUD - ALL AGES
160	2-STORY PUD - 1946 & NEWER
180	PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
190	2 FAMILY CONVERSION - ALL STYLES AND AGES

MSZoning: Identifies the general zoning classification of the sale.

A	Agriculture
C	Commercial
FV	Floating Village Residential
I	Industrial
RH	Residential High Density
RL	Residential Low Density
RP	Residential Low Density Park
RM	Residential Medium Density

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access to property

Grv1 Gravel

Data Explorer

957.39 kB

- data_description.txt
- sample_submission.csv
- test.csv
- train.csv

Summary

- 4 files
- 163 columns

Download All

house-prices-advanced-regres

Descargas > house-prices-advanced-regression-techniques

Nombre	Fecha de modificación	Tipo	Tamaño
data_description	30/10/2024 19:04	Documento de tex...	14 KB
sample_submission.csv	30/10/2024 19:04	Excel.CSV	32 KB
test.csv	30/10/2024 19:04	Excel.CSV	441 KB
train.csv	30/10/2024 19:04	Excel.CSV	450 KB

Paso 3:

Ya solo queda subir esos ficheros al repositorio, para no tener que hacer varios commit mostrando el proceso y subir este mismo fichero varias veces, los pasos son los mismos que hemos hecho en el segundo apartado, crear una carpeta, ahí subimos el dataset, en el directorio raíz del proyecto subiré este pdf y haré un commit & push.

