



# Integración: Git, GitHub y VSC

Ing. Luis Guillermo Molero Suárez



## Git y GitHub

### ¿Qué es Git?

Es un sistema de control de versiones, es distribuido, es decir que múltiples personas pueden trabajar en equipo, es open source y también se adapta a todo tipo de proyectos desde pequeños hasta grandes, además, se pueden fusionar archivos, guarda una línea de tiempo a lo largo de todo el proyecto. Maneja una interfaz tipo Bash. GIT, es el software de control de versiones en el que se basa GitHub

Sitio de descarga: <https://git-scm.com/downloads>

Como instalar Git: <https://www.youtube.com/watch?v=ExdLS6lZaAY>

### ¿Qué es Github?

A diferencia de Git, Github es un sitio web y un servicio en la nube que ayuda a los desarrolladores a almacenar y administrar su código, al igual que llevar un registro y control de cualquier cambio sobre este código. En otras palabras, es una plataforma de desarrollo colaborativo, o también llamada la red social de los desarrolladores donde se alojan los repositorios, el código se almacena de forma pública pero se puede hacer privado con una cuenta de pago.

La interfaz de GitHub es bastante fácil de usar para el desarrollador novato que quiera aprovechar las ventajas del Git. Sin GitHub, usar un Git generalmente requiere de un poco más de conocimientos de tecnología y uso de una línea de comando (Bash).

Sitio de descarga: <https://desktop.github.com/>

Como instalar Github: <https://www.youtube.com/watch?v=tN6tloweTU>



## ¿Qué Es una Versión de Control?

Una Versión de Control ayuda a los desarrolladores a llevar un registro y administrar cualquier cambio en el código del proyecto de software. A medida que crece este proyecto, la versión de control se vuelve esencial.

Con la bifurcación, un desarrollador duplica parte del código fuente (llamado repositorio). Este desarrollador, luego puede, de forma segura, hacer cambios a esa parte del código, sin afectar al resto del proyecto.

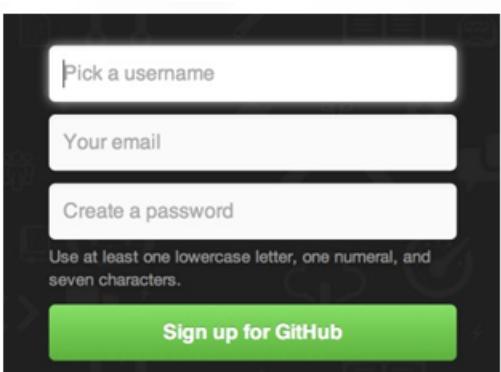
Luego, una vez que el desarrollador logre que su parte del código funcione de forma apropiada, esta persona podría fusionar este código al código fuente principal para hacerlo oficial. Todos estos cambios luego son registrados y pueden ser revertidos si es necesario.

**Documentación de Github:** <https://docs.github.com/es/github>

**Documentación Git:** <https://git-scm.com/book/es/v2>

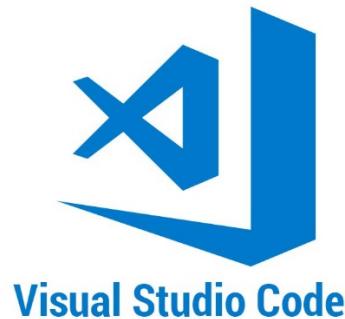
## Creación de una cuenta en GitHub

Lo primero que necesitas es una cuenta de usuario gratuita. Simplemente visita <https://github.com>, elige un nombre de usuario que no esté ya en uso, proporciona un correo y una contraseña, y pulsa el botón verde grande “Sign up for GitHub”.



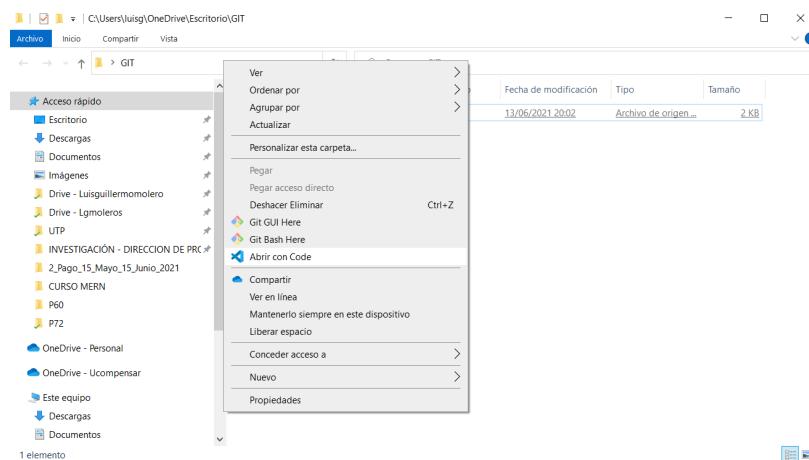
Lo siguiente que verás es la página de precios para planes mejores, pero lo puedes ignorar por el momento. GitHub te enviará un correo para verificar la dirección que les has dado. Confirmar la dirección ahora, es bastante importante (como veremos después).

**Para ampliar esta información:** <https://n9.cl/nqu9>

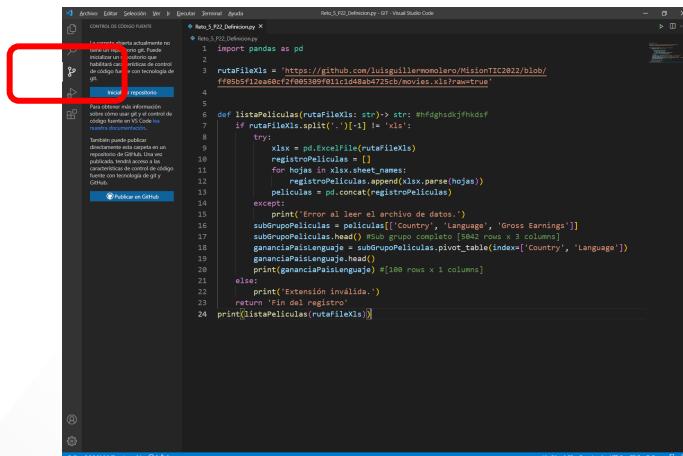


## Integración de GIT, GitHub y Visual Studio Code

Paso 1: Abrimos nuestro proyecto en VSC, de la forma:

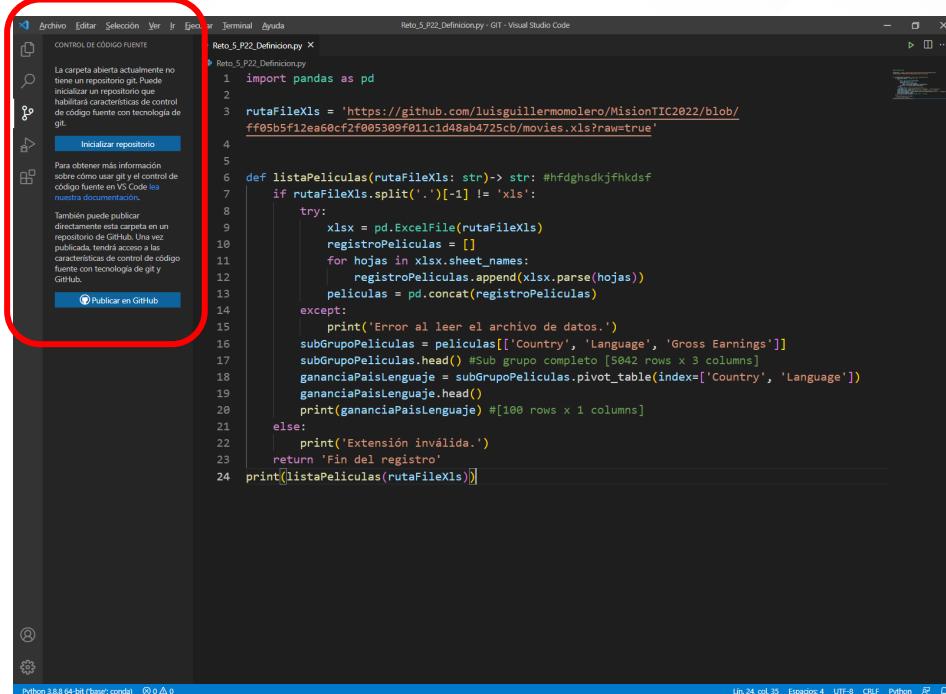


2.- Desde nuestra carpeta de código hacemos clic en el botón “Control de código fuente” de VSC .





Nos aparece la siguiente interacción

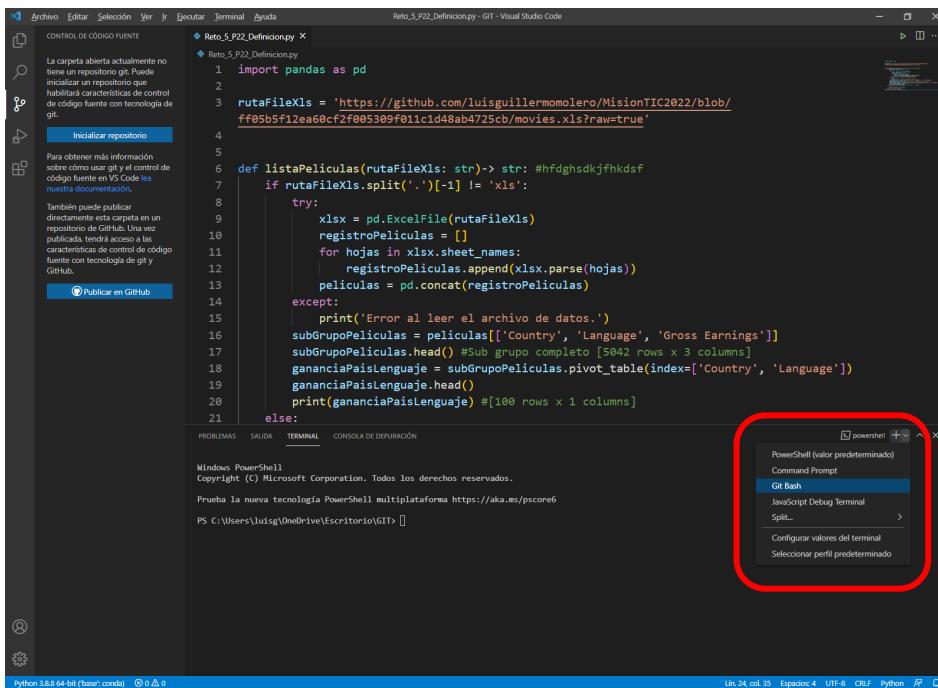


The screenshot shows a Visual Studio Code interface with a Python script named 'Reto\_5\_P2\_Definicion.py' open. A red circle highlights the GitHub integration sidebar on the left, which includes buttons for 'Iniciar repositorio' (Start repository), 'Publicar en GitHub' (Publish to GitHub), and 'Para obtener más información sobre cómo usar git y el control de código fuente en VS Code' (Information about using git and code control in VS Code).

```
import pandas as pd
rutaFileXls = 'https://github.com/luisguillermomolero/MisionTIC2022/blob/ff05b5f12ea60cf2f005309f011c1d48ab4725cb/movies.xls?raw=true'

def listaPeliculas(rutaFileXls: str)-> str: #hfdfghsdkjfhkdsf
    if rutaFileXls.split('.')[1] != 'xls':
        try:
            xlsx = pd.ExcelFile(rutaFileXls)
            registroPeliculas = []
            for hojas in xlsx.sheet_names:
                registroPeliculas.append(xlsx.parse(hojas))
            peliculas = pd.concat(registroPeliculas)
        except:
            print('Error al leer el archivo de datos.')
        subGrupoPeliculas = peliculas[['Country', 'Language', 'Gross Earnings']]
        subGrupoPeliculas.head() #Sub grupo completo [5042 rows x 3 columns]
        gananciaPaislenguaje = subGrupoPeliculas.pivot_table(index=['Country', 'Language'])
        gananciaPaislenguaje.head()
        print(gananciaPaislenguaje) #[100 rows x 1 columns]
    else:
        print('Extensión inválida.')
    return 'Fin del registro'
print(listaPeliculas(rutaFileXls))
```

3.- Abrimos la terminal de Git en VSC y escribimos git init



The screenshot shows the Visual Studio Code interface with the terminal tab active. A red circle highlights a context menu for the terminal, which includes options like 'PowerShell (valor predeterminado)', 'Command Prompt', 'Git Bash' (which is selected), 'JavaScript Debug Terminal', 'Split...', 'Configurar valores del terminal', and 'Seleccionar perfil predeterminado'. The terminal window displays a PowerShell prompt with the command 'git init' entered.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/psscnew6

PS C:\Users\luisg\OneDrive\Escritorio\GIT> [
```



```
import pandas as pd

rutaFileXls = 'https://github.com/luisguillermomolero/MisionTIC2022/blob/_ff05b5f12ea60cf2f005309f011c1d4ab4725cb/movies.xls?raw=true'

def listaPeliculas(rutaFileXls: str)-> str: #hfdfghsdkjfhkdsf
    if rutaFileXls.split('.')[1] != 'xls':
        try:
            xlsx = pd.ExcelFile(rutaFileXls)
            registroPeliculas = []
            for hojas in xlsx.sheet_names:
                registroPeliculas.append(xlsx.parse(hojas))
            peliculas = pd.concat(registroPeliculas)
        except:
            print('Error al leer el archivo de datos.')
        subGrupoPeliculas = peliculas[['Country', 'Language', 'Gross Earnings']]
        subGrupoPeliculas.head() #Sub grupo completo [5042 rows x 3 columns]
        gananciaPaislenguaje = subGrupoPeliculas.pivot_table(index=['Country', 'Language'])
        gananciaPaislenguaje.head()
        print(gananciaPaislenguaje) #[100 rows x 1 columns]
    else:
        print('Formato de archivo incorrecto')

PROBLEMAS SALIDA TERMINAL CONSOLA DE DEPURACIÓN

luisg@DESKTOP-4C8B4GB MINGW64 ~/OneDrive/Escritorio/GIT
$ git init
Initialized empty Git repository in C:/Users/luisg/OneDrive/Escritorio/GIT/.git/
luisg@DESKTOP-4C8B4GB MINGW64 ~/OneDrive/Escritorio/GIT (master)
$
```

Una vez ejecutado el comando, desaparece el área de interacción y ya podemos trabajar con el código fuente, ya que se inicializó un repositorio en local.

```
import pandas as pd

rutaFileXls = 'https://github.com/luisguillermomolero/MisionTIC2022/blob/_ff05b5f12ea60cf2f005309f011c1d4ab4725cb/movies.xls?raw=true'

def listaPeliculas(rutaFileXls: str)-> str: #hfdfghsdkjfhkdsf
    if rutaFileXls.split('.')[1] != 'xls':
        try:
            xlsx = pd.ExcelFile(rutaFileXls)
            registroPeliculas = []
            for hojas in xlsx.sheet_names:
                registroPeliculas.append(xlsx.parse(hojas))
            peliculas = pd.concat(registroPeliculas)
        except:
            print('Error al leer el archivo de datos.')
        subGrupoPeliculas = peliculas[['Country', 'Language', 'Gross Earnings']]
        subGrupoPeliculas.head() #Sub grupo completo [5042 rows x 3 columns]
        gananciaPaislenguaje = subGrupoPeliculas.pivot_table(index=['Country', 'Language'])
        gananciaPaislenguaje.head()
        print(gananciaPaislenguaje) #[100 rows x 1 columns]
    else:
        print('Formato de archivo incorrecto')

PROBLEMAS SALIDA TERMINAL CONSOLA DE DEPURACIÓN

luisg@DESKTOP-4C8B4GB MINGW64 ~/OneDrive/Escritorio/GIT
$ git init
Initialized empty Git repository in C:/Users/luisg/OneDrive/Escritorio/GIT/.git/
luisg@DESKTOP-4C8B4GB MINGW64 ~/OneDrive/Escritorio/GIT (master)
$
```

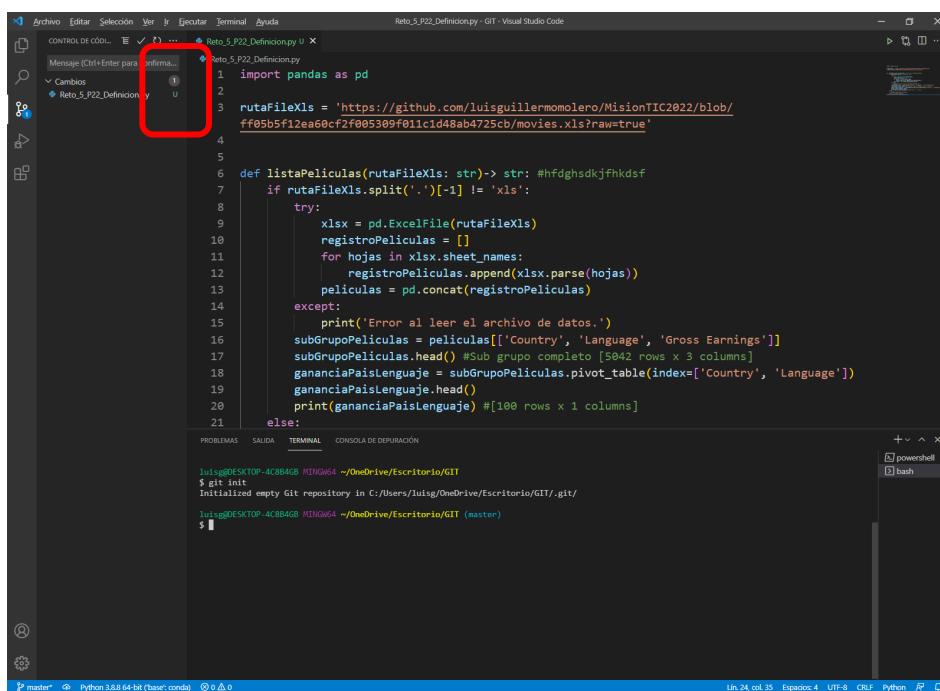


Podemos confirmar que ya está inicializado el repositorio de la siguiente manera:  
Nos vamos a la carpeta donde se ubica el proyecto y observaremos una carpeta oculta .git  
(Cada vez que queramos ocultar una carpeta colocamos el “.” antes del nombre).

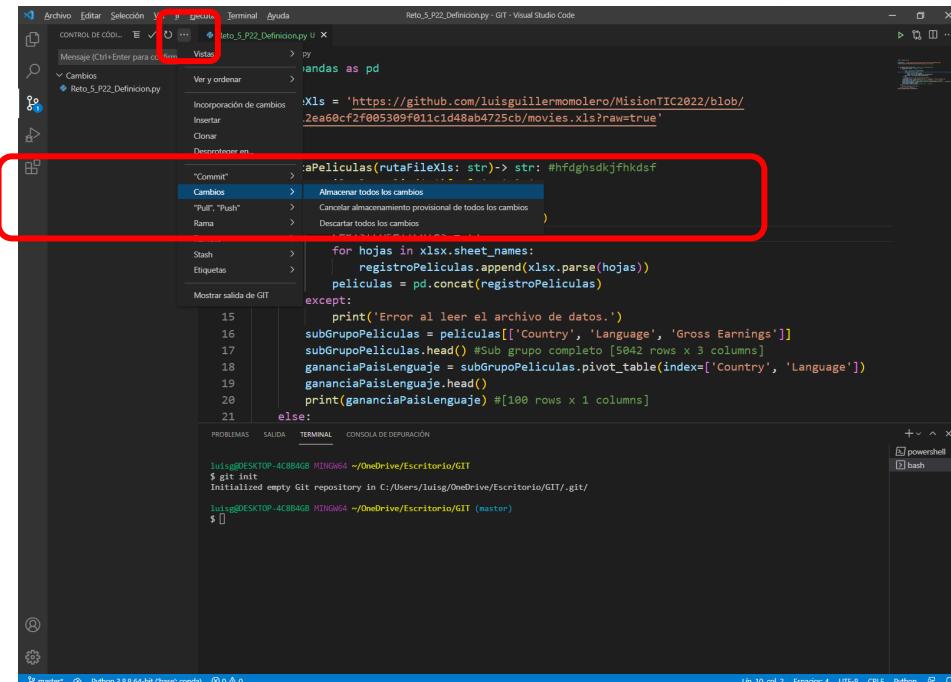


Luego de verificar que tenemos control de cambios en el proyecto, revisemos los siguientes:

En nuestro proyecto, el archivo abierto aparece con el identificador “U” que significa sin seguimiento

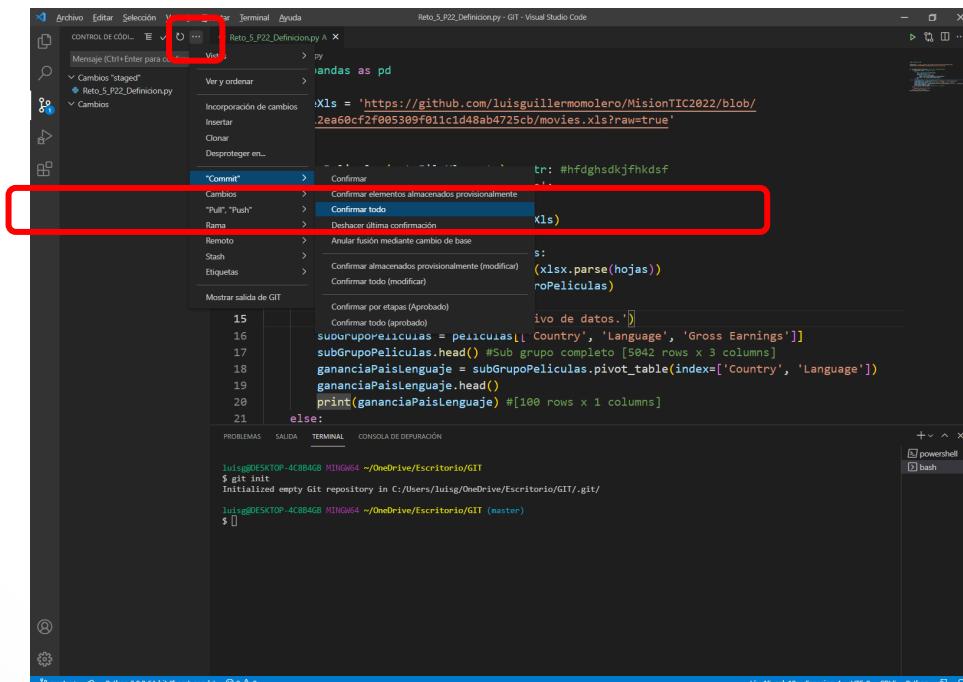


4.- Entonces, hacemos clic en “almacenar todos los cambios”



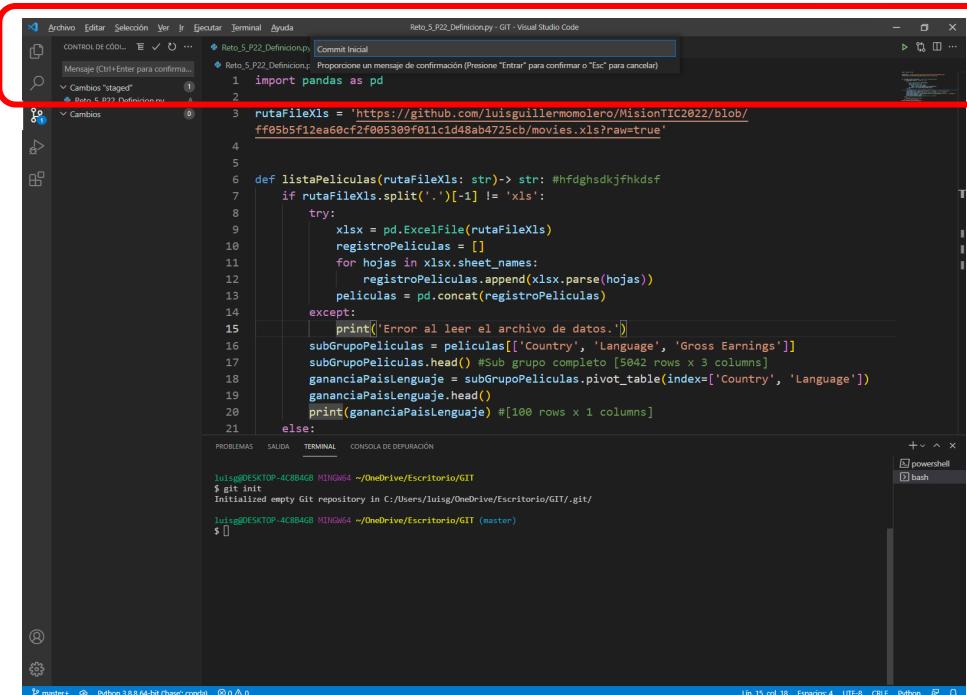
Una vez hecho esto, nos aparece el identificador “A” (Add).

4.- Como paso seguido, procedemos a “confirmar todo”





5.- Nos abrirá la “paleta de comandos” y le colocamos un mensaje para el primer commit (Este mensaje debe hacerse tantas veces hagamos una actualización del proyecto para llevar el control de cambios), que para el caso de este ejemplo es: “Commit Inicial”.



```
import pandas as pd
rutaFileXls = 'https://github.com/luisguillermomolero/MisionTIC2022/blob/_ff05bf12ea60fcf005309f011c1d48ab4725cb/movies.xls?raw=true'
def listaPeliculas(rutafilexls: str)-> str: #hfdfghsdfkjfhkdsf
    if rutafilexls.split('.')[1] != 'xls':
        try:
            xlsx = pd.ExcelFile(rutafilexls)
            registroPeliculas = []
            for hojas in xlsx.sheet_names:
                registroPeliculas.append(xlsx.parse(hojas))
            peliculas = pd.concat(registroPeliculas)
        except:
            print('Error al leer el archivo de datos.')
            subGrupoPeliculas = peliculas[['Country', 'Language', 'Gross Earnings']]
            subGrupoPeliculas.head() #Sub grupo completo [5042 rows x 3 columns]
            gananciaPaisLenguaje = subGrupoPeliculas.pivot_table(index=['Country', 'Language'])
            gananciaPaisLenguaje.head()
            print(gananciaPaisLenguaje) #[100 rows x 1 columns]
    else:
```

luis@DESKTOP-4C8B4GB MINGW64 ~/OneDrive/Escritorio/GIT\$ git init  
Initial empty Git repository in C:/Users/luis/OneDrive/Escritorio/GIT/.git/  
luis@DESKTOP-4C8B4GB MINGW64 ~/OneDrive/Escritorio/GIT (master)

## IMPORTANTE:

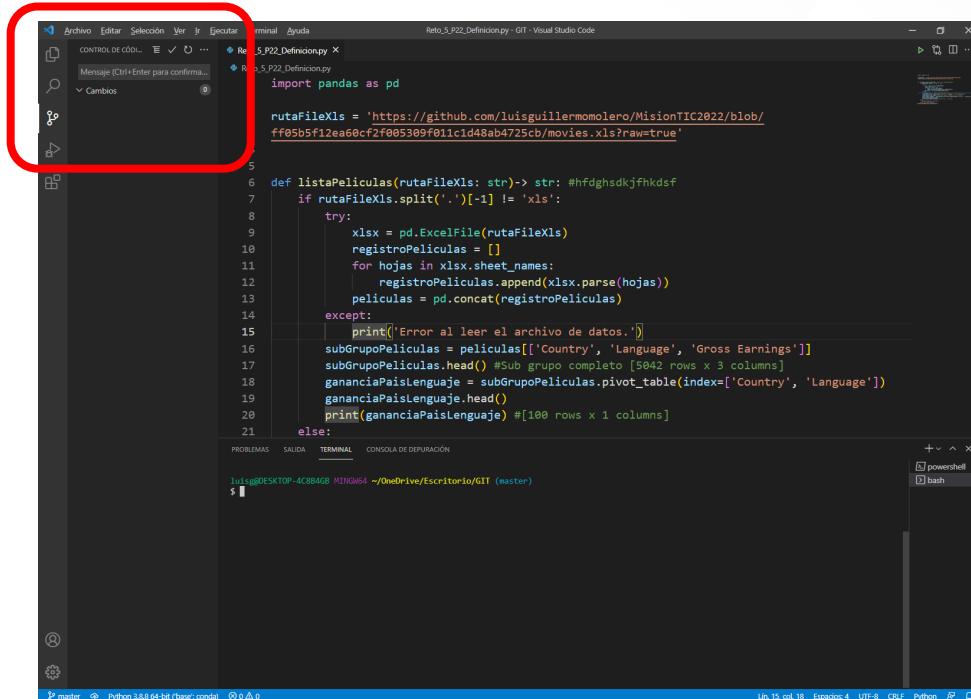
Si al momento de teclear <ENTER> para confirmar el mensaje del “commit” hecho genera un error de autenticación, introduzca en la terminal GIT las siguientes líneas de comandos:  
git config --global user.email "tu\_correo\_de\_GITHUB"  
git config --global user.name "tu\_usuario\_de\_GITHUB"

## RECUERDA:

esta configuración de GIT, GitHub en VSC debe hacerse luego de haber instalado GIT en tu Pc y creado tu cuenta en GITHUB.



Paso seguido, veremos que no hay cambios que guardar en nuestro repositorio



```
Reto_5_P22_Definicion.py - GIT - Visual Studio Code

Archivo Editar Selección Ver Ir Ejecutar
CONTROL DE CÓDIGO... 🔍 ⌂ ⌂ ... Reto_5_P22_Definicion.py
Mensaje (Ctrl+Enter para confirmar...)
Cambiros

import pandas as pd

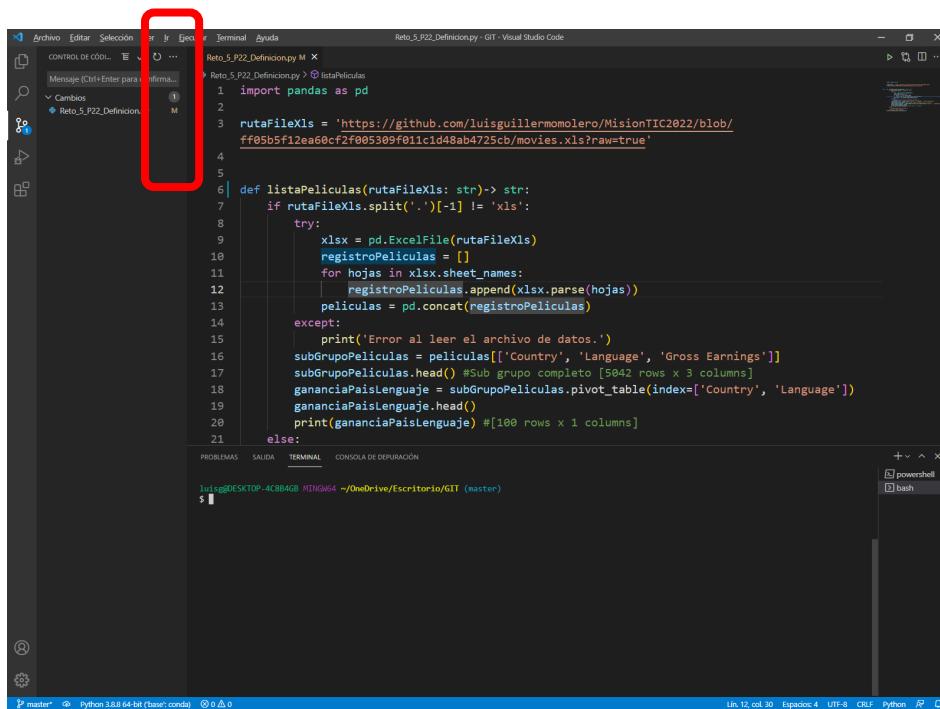
rutaFileXls = 'https://github.com/luisguillermomolero/MisionTIC2022/blob/_ff05b5f12ea60cf2f005309f011c1d48ab4725cb/movies.xls?raw=true'

def listaPeliculas(rutaFileXls: str)-> str: #hfdfghsdkjfhkdsf
    if rutaFileXls.split('.')[1] != 'xls':
        try:
            xlsx = pd.ExcelFile(rutaFileXls)
            registroPeliculas = []
            for hojas in xlsx.sheet_names:
                registroPeliculas.append(xlsx.parse(hojas))
            peliculas = pd.concat(registroPeliculas)
        except:
            print('Error al leer el archivo de datos.')
            subGrupoPeliculas = peliculas[['Country', 'Language', 'Gross Earnings']]
            subGrupoPeliculas.head() #Sub grupo completo [5042 rows x 3 columns]
            gananciaPaislenguaje = subGrupoPeliculas.pivot_table(index=['Country', 'Language'])
            gananciaPaislenguaje.head()
            print(gananciaPaislenguaje) #[100 rows x 1 columns]
    else:
        print('El archivo es de otro tipo')

PROBLEMAS SALIDA TERMINAL CONSOLA DE DEPURACIÓN

luisg@DESKTOP-4CB84GB MINGW64 ~/OneDrive/Escritorio/GIT (master)
$
```

Una vez que hallamos modificado nuestro archivo del proyecto, aparecerá el identificador “M” en la ventana de “Control de cambios”



```
Reto_5_P22_Definicion.py M - GIT - Visual Studio Code

Archivo Editar Selección Ver Ir Ejecutar
CONTROL DE CÓDIGO... 🔍 ⌂ ⌂ ... Reto_5_P22_Definicion.py M
Mensaje (Ctrl+Enter para confirmar...)
Cambiros

import pandas as pd

rutaFileXls = 'https://github.com/luisguillermomolero/MisionTIC2022/blob/_ff05b5f12ea60cf2f005309f011c1d48ab4725cb/movies.xls?raw=true'

def listaPeliculas(rutaFileXls: str)-> str:
    if rutaFileXls.split('.')[1] != 'xls':
        try:
            xlsx = pd.ExcelFile(rutaFileXls)
            registroPeliculas = []
            for hojas in xlsx.sheet_names:
                registroPeliculas.append(xlsx.parse(hojas))
            peliculas = pd.concat(registroPeliculas)
        except:
            print('Error al leer el archivo de datos.')
            subGrupoPeliculas = peliculas[['Country', 'Language', 'Gross Earnings']]
            subGrupoPeliculas.head() #Sub grupo completo [5042 rows x 3 columns]
            gananciaPaislenguaje = subGrupoPeliculas.pivot_table(index=['Country', 'Language'])
            gananciaPaislenguaje.head()
            print(gananciaPaislenguaje) #[100 rows x 1 columns]
    else:
        print('El archivo es de otro tipo')

PROBLEMAS SALIDA TERMINAL CONSOLA DE DEPURACIÓN

luisg@DESKTOP-4CB84GB MINGW64 ~/OneDrive/Escritorio/GIT (master)
$
```



En ese sentido, al hacer clic sobre el archivo modificado, aparecerá otra pantalla donde se observan los cambios.

The screenshot shows two code editors in Visual Studio Code. The left editor contains the file `Reto_5_P22_Definicion.py`, and the right editor contains the file `Reto_5_P22_Definicion.py (órbito de trabajo)`. Both files are identical, displaying the following code:

```
import pandas as pd

rutaFileXls = 'https://github.com/luisguillermomolero/MisionTIC2022/blob/ff05b5f12ea60cf2f0e05309f011c1d48ab4725cb/movies.xls?raw=true'

def listaPeliculas(rutaFileXls: str) -> str:
    if rutaFileXls.split('.')[-1] != 'xls':
        try:
            xlsx = pd.ExcelFile(rutaFileXls)
            registroPeliculas = []
            for hojas in xlsx.sheet_names:
                registroPeliculas.append(xlsx.parse(hojas))
            peliculas = pd.concat(registroPeliculas)
        except:
            print('Error al leer el archivo de datos.')
    subGrupoPeliculas = peliculas[['Country', 'Language', 'Gross Earnings']]
    return subGrupoPeliculas.to_string()

if __name__ == '__main__':
    listaPeliculas(rutaFileXls)
```

Una vez observado estos cambios y estar conforme con ellos, nos vamos de nuevo a “almacenar todos los cambios” y luego procedemos a “confirmar todo “de nuevo (pasos anteriores). Para este ejemplo el mensaje será “Eliminación de comentarios”

```
Reto_5_P22_Definicion.py (índice de trabajo) - GIT - Visual Studio Code

CONTROL DE CÓDIGO...  Eliminación de comentarios
Mensaje (Ctrl+Enter para confirmar...) Cambios "staged"
Cambiando

Reto_5_P22_Definicion.py Proporcione un mensaje de confirmación (Presione "Entrar" para confirmar o "Esc" para cancelar)

1 import pandas as pd
2
3 rutaFileXls = 'https://github.com/luisguillermomolero/MisionTIC2022/blob/ff05b5f12ea60cf2#005309f011c1d48ab4725cb/movies.xls?raw=true'
4
5
6 def listaPeliculas(rutaFileXls: str) -> str:
7     if rutaFileXls.split('.')[-1] != 'xls':
8         try:
9             xlsx = pd.ExcelFile(rutaFileXls)
10            registroPeliculas = []
11            for hojas in xlsx.sheet_names:
12                registroPeliculas.append(xlsx.parse(hojas))
13            peliculas = pd.concat(registroPeliculas)
14        except:
15            print('Error al leer el archivo de datos.')
16        subGrupoPeliculas = peliculas[['Country', 'Language', 'Gross Earnings']]
17        subGrupoPeliculas.head() #Sub grupo

1 import pandas as pd
2
3 rutaFileXls = 'https://github.com/luisguillermomolero/MisionTIC2022/blob/ff05b5f12ea60cf2#005309f011c1d48ab4725cb/movies.xls?raw=true'
4
5
6 def listaPeliculas(rutaFileXls: str) -> str:
7     if rutaFileXls.split('.')[-1] != 'xls':
8         try:
9             xlsx = pd.ExcelFile(rutaFileXls)
10            registroPeliculas = []
11            for hojas in xlsx.sheet_names:
12                registroPeliculas.append(xlsx.parse(hojas))
13            peliculas = pd.concat(registroPeliculas)
14        except:
15            print('Error al leer el archivo de datos.')
16        subGrupoPeliculas = peliculas[['Country', 'Language', 'Gross Earnings']]
17        subGrupoPeliculas.head() #Sub grupo

PROBLEMAS SALIDA TERMINAL CONSOLA DE DEPURACIÓN
luisgg@DESKTOP-4CBB46B MINGW64 ~/OneDrive/Escritorio/GIT (master)
$ [ ]
```

Finalmente, confirmamos que no se vean cambios en nuestro “Control de cambios”.



## RECUERDA:

Todos estos cambios se están haciendo de manera local, ahora vamos a configurar nuestro GITHUB para poder hospedar estos cambios en la nube.

1.- Nos vamos a nuestra cuenta de GITHUB en la web y creamos un “Nuevo Repositorio”

The screenshot shows a GitHub user profile for 'luisguillermomolero'. The top navigation bar includes 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. A red box highlights the 'New repository' option in the dropdown menu of the top right corner. Below the navigation, there's a section for 'Popular repositories' featuring 'Estructura\_Datos' and 'CSS'. Further down is a chart titled '9 contributions in the last year' and a 'Contribution activity' section.

Le colocamos un nombre, una descripción a nuestro repositorio y hacemos clic en “Create repository”

The screenshot shows the 'Create a new repository' form. The 'Repository name' field is set to 'luisguillermomolero / Repositorio\_Prueba\_GIT\_VSC'. The 'Description (optional)' field contains the text 'Repositorio de prueba para configurar control de cambios en VSC'. The 'Public' radio button is selected. At the bottom, the 'Create repository' button is highlighted with a red box.

2.- Una vez creado el repositorio, copiamos las siguientes líneas de código que se encuentra en GitHub, en nuestra terminal de GIT de VSC.



The screenshot shows a GitHub repository page for 'luisguillermomolero / Repositorio\_Prueba\_GIT\_VSC'. The 'Code' tab is selected. A red box highlights the 'Quick setup — if you've done this kind of thing before' section, which contains a command-line script for initializing a new repository:

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

Below this, there are sections for pushing an existing repository and importing code from another repository.

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



```
def listaPeliculas(rutaFileXls: str)-> str:
    if rutaFileXls.split('.')[ -1 ] != 'xls':
        try:
            xlsx = pd.ExcelFile
            (rutaFileXls)
            registroPeliculas =
            []
        
```

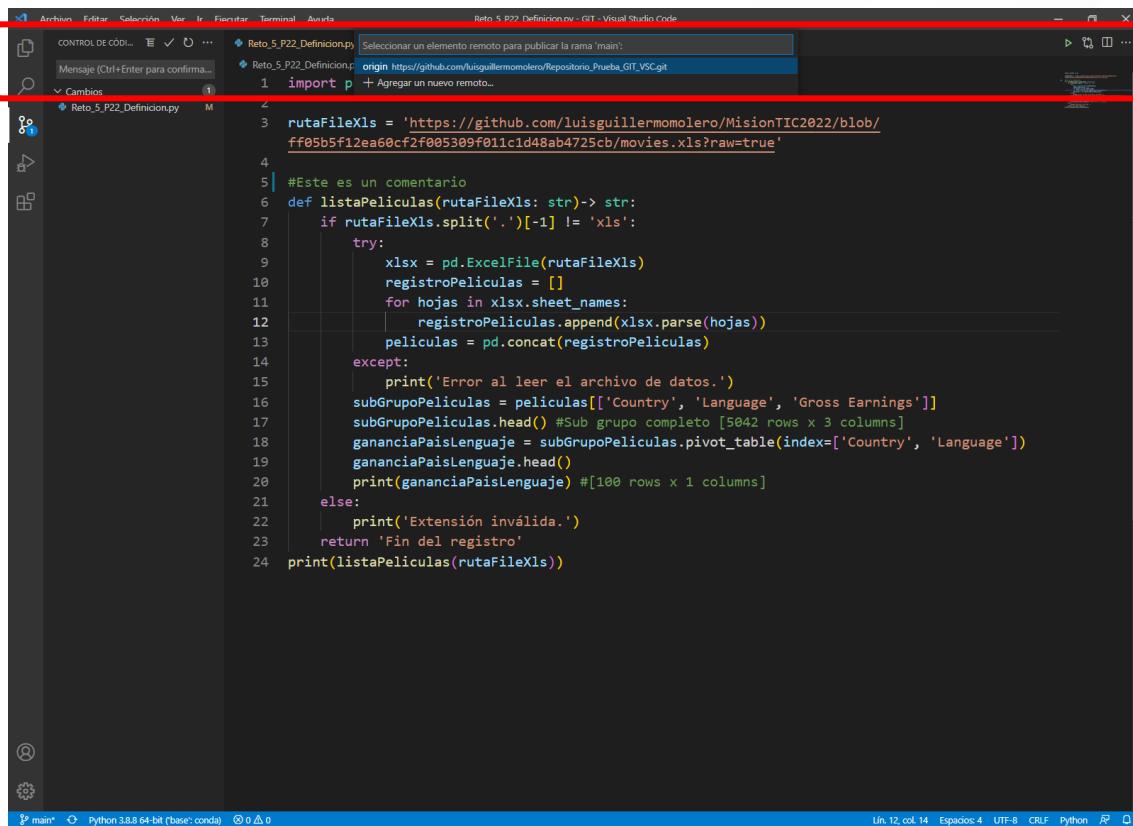
```
luisg@DESKTOP-4CBB4GB MINGW64 ~/OneDrive/Escritorio/GIT (master)
$ echo "# Repositorio_Prueba_GIT_VSC" >> README.md
luisg@DESKTOP-4CBB4GB MINGW64 ~/OneDrive/Escritorio/GIT (master)
$ git init
Reinitialized existing Git repository in C:/Users/luisg/OneDrive/Escritorio/GIT/.git/
luisg@DESKTOP-4CBB4GB MINGW64 ~/OneDrive/Escritorio/GIT (master)
$ git add README.md
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory
luisg@DESKTOP-4CBB4GB MINGW64 ~/OneDrive/Escritorio/GIT (master)
$ git commit -m "First commit"
[master 949fd67] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
luisg@DESKTOP-4CBB4GB MINGW64 ~/OneDrive/Escritorio/GIT (main)
$ git branch -M main
luisg@DESKTOP-4CBB4GB MINGW64 ~/OneDrive/Escritorio/GIT (main)
$ git remote add origin https://github.com/luisguillermomolero/Repositorio_Prueba_GIT_VSC.git
luisg@DESKTOP-4CBB4GB MINGW64 ~/OneDrive/Escritorio/GIT (main)
$ git push -u origin main
Enumerating objects: 100%, done.
Counting objects: 100% (9/9), done.
Delta compressing: 100% (0/0), done.
Writing objects: 100% (9/9), 1.2k KiB | 638.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/luisguillermomolero/Repositorio_Prueba_GIT_VSC.git
 * [new branch]  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
luisg@DESKTOP-4CBB4GB MINGW64 ~/OneDrive/Escritorio/GIT (main)
```

Luego de que eventualmente se realice algún cambio en nuestro código nos vamos a “Insertar en”

The screenshot shows the Visual Studio Code interface with the 'Git' context menu open over a Python file named 'Retorno\_P22\_Definicion.py'. The 'Insertar en...' option is highlighted with a red box.



Y en la “paleta de comandos” nos listara los repositorios donde podremos actualizar nuestro proyecto.



The screenshot shows a Visual Studio Code interface with a Python script named 'Reto\_5\_P22\_Definicion.py' open. A red box highlights the top right corner of the window, which contains a GitHub repository selection dialog. The dialog shows the URL 'origin https://github.com/luisguillermomolero/Repositorio\_Prueba\_GIT\_VSC.git' and a message 'Seleccionar un elemento remoto para publicar la rama 'main''. The main code editor shows a function 'listaPelículas' that reads an Excel file and prints its contents. The status bar at the bottom indicates the file is 'main', the language is 'Python 3.8.8 64-bit (base; conda)', and the encoding is 'UTF-8'.

```
import pandas as pd

rutaFileXls = 'https://github.com/luisguillermomolero/MisionTIC2022/blob/ff05b5f12ea60cf2f005309f011c1d48ab4725cb/movies.xls?raw=true'

#Este es un comentario
def listaPelículas(rutaFileXls: str)-> str:
    if rutaFileXls.split('.')[1] != 'xls':
        try:
            xls = pd.ExcelFile(rutaFileXls)
            registroPelículas = []
            for hojas in xls.sheet_names:
                registroPelículas.append(xls.parse(hojas))
            películas = pd.concat(registroPelículas)
        except:
            print('Error al leer el archivo de datos.')
            subGrupoPelículas = películas[['Country', 'Language', 'Gross Earnings']]
            subGrupoPelículas.head() #Sub grupo completo [5042 rows x 3 columns]
            gananciaPaisLenguaje = subGrupoPelículas.pivot_table(index=['Country', 'Language'])
            gananciaPaisLenguaje.head()
            print(gananciaPaisLenguaje) #[100 rows x 1 columns]
        else:
            print('Extensión inválida.')
    return 'Fin del registro'
print(listaPelículas(rutaFileXls))
```



Seleccionamos nuestro repositorio y listo, nos aparecerá en nuestro repositorio GitHub

The screenshot shows a GitHub repository page. At the top, there's a navigation bar with links for Search or jump to..., Pull requests, Issues, Marketplace, and Explore. Below the header, the repository name 'luisguillermomolero / Repositorio\_Prueba\_GIT\_VSC' is displayed, along with a star count of 0 and a fork count of 0. The main content area shows a single commit by 'luisguillermomolero' with the message 'first commit'. This commit was made 3 minutes ago and has 3 commits. Below the commit, there are two files listed: 'README.md' and 'Reto\_5\_P22\_Definicion.py'. The 'About' section contains the text: 'Repository de prueba para configurar control de cambios en VSC'. Other sections like Releases, Packages, and Insights are also visible.

### IMPORTANTE:

Los pasos de “Almacenar todos los cambios”, “Confirmar todo” e “Insertar en” deben llevarse a cabo luego de una modificación en el código.