PROPEDÉUTICO DE MATEMÁTICAS Y PROGRAMACIÓN · FLACSO ECUADOR

Tu propio repositorio en GitHub

Manual de la sesión — crearás tu repositorio desde cero, guardarás tus primeros cambios con `git add` *y* `git commit`*, y los publicarás con* `git push`*.*

Modalidad: práctica guiada en clase **Fecha:** miércoles, 15 de julio de 2026

Trae: tu computadora con todo lo de la sección «Antes de la clase»

Antes de la clase — requisitos, no tarea

Esta sesión continúa exactamente donde terminó la anterior: hoy *tú* produces y publicas. **No repetiremos los pasos de instalación ni los de la clase 2**; verifica esta lista y, si te falta algo, resuélvelo **antes** de la clase por el canal de ayuda.

⚠️ **Verifica que ya tienes:**

- **Las tres herramientas funcionando**
  — VS Code, Python y Git, con tu identidad configurada en Git (tu nombre y tu correo). Todo está en el
  Manual de instalación
  . Comprobación de 10 segundos: `git config --global --list` debe mostrar tu `user.name` y tu `user.email` ; si no aparecen, resuélvelo con la sección G antes de venir.
- **La práctica de la clase 2 completa**
  — el repositorio del curso clonado en `Documentos/propedeutico` y tu entorno `.venv` creado. Si faltaste o algo quedó a medias, sigue el
  Manual de la clase 2
  .
- **Tu cuenta de GitHub con el correo verificado**
  — hoy publicarás en ella. Comprueba que puedes iniciar sesión en github.com/login.

Tres ideas antes de tocar el teclado

Commit

Una fotografía de tu proyecto en un momento dado: qué archivos había, qué decían, quién tomó la foto y cuándo, con un mensaje que explica el cambio. El historial de un repositorio es un álbum de commits.

Preparación (add)

Antes de la foto, decides quién sale en ella: `git add` coloca tus cambios en la «bandeja de preparación». Solo lo que está en la bandeja entra en el próximo commit.

Publicar (push)

Tus commits nacen en tu computadora. `git push` los sube a GitHub, donde quedan respaldados y visibles. Es el gesto inverso al `git pull` con que bajas los materiales del curso.

EN LA CLASE · LO HAREMOS JUNTOS

Práctica guiada: crear, guardar y publicar

1 Crea una carpeta vacía para tu repositorio

1. Abre el explorador de archivos (Windows) o el Finder (Mac) y entra a la carpeta `propedeutico` de tus Documentos — la carpeta madre de la clase 2, donde ya vive `propedeutico-flacso`.
2. Ahí dentro crea una carpeta nueva y nómbrala exactamente `mi-propedeutico` — minúsculas, sin espacios, tildes ni eñes: la regla de oro de la clase 2 también aplica aquí.

CÓMO DEBE QUEDAR

```
Documentos/propedeutico/
├── propedeutico-flacso/   ← el del curso: solo lees y haces git pull
└── mi-propedeutico/       ← el tuyo: vacío por ahora — hoy lo llenas
```

> ⚠️ **Un repositorio nunca va dentro de otro**
>
> Crea la carpeta directamente en `propedeutico`, como hermana de `propedeutico-flacso` — nunca adentro. Si quedó anidada, revisa P6 para moverla.

Ahora ábrela en VS Code: **File → Open Folder** (Archivo → Abrir carpeta) → `mi-propedeutico` → confía en los autores (eres tú). El explorador de VS Code se verá vacío — correcto: todavía no hay nada.

2 Crea tu repositorio en GitHub — vacío, esa es la clave

1. Entra a github.com con tu cuenta y pulsa el botón `+` (arriba a la derecha) → **New repository**.
2. En **Repository name** escribe exactamente `mi-propedeutico` — el mismo nombre que la carpeta del paso 1.
3. En **Description** (opcional) puedes poner: `Bitácora del propedéutico — FLACSO Ecuador`.
4. Deja la visibilidad en **Public**: este repositorio es el inicio de tu portafolio.
5. **No marques nada** en la sección **Initialize this repository with**: sin **README**, sin **.gitignore**, sin licencia. El repositorio debe nacer vacío, porque todo su contenido nacerá en tu computadora y viajará a GitHub con tu primer push.
6. Pulsa **Create repository**.

Como el repositorio está vacío, GitHub no te muestra archivos sino una página de **Quick setup** con instrucciones. Localiza el bloque titulado **…or create a new repository on the command line**: son los comandos que usarás en el siguiente paso, ya con **tu** nombre de usuario en la dirección. **Deja esta pestaña abierta.**

Fíjate en la dirección del navegador: `github.com/tu-usuario/mi-propedeutico`. Ese es **tu** espacio público; el del curso era el del docente.

---

3 En la terminal: párate en tu carpeta y pega los comandos de GitHub

1. Vuelve a VS Code y abre una terminal (**Terminal → New Terminal**). Como tu carpeta de trabajo es `mi-propedeutico`, la terminal nace «parada» exactamente ahí. **Compruébalo:** la ruta antes del cursor debe terminar en `mi-propedeutico`.
2. En la pestaña de GitHub, copia el bloque **…or create a new repository on the command line** completo, con el icono de copiar que aparece a su derecha. **Cópialo de tu página de GitHub, no de este manual:** el de GitHub ya trae tu nombre de usuario en la dirección.
3. Pégalo en la terminal (clic derecho → Pegar, o **Ctrl+V** / **⌘V**). Si VS Code pregunta si de verdad quieres pegar varias líneas, acepta. Al final presiona Enter: la última línea queda escrita pero aún sin ejecutar.

> 💡 **`cd`: así se mueve uno por las carpetas**
>
> La terminal siempre está parada en una carpeta, y los comandos actúan ahí. Para moverte se usa `cd` (*change directory*): escribe `cd`, un espacio y la ruta de la carpeta destino. Un truco que no falla: escribe `cd ` y **arrastra la carpeta** desde el explorador o el Finder hasta la terminal — la ruta se escribe sola; presiona Enter y confirma dónde quedaste con `pwd`. Hoy VS Code te ahorra el viaje: al abrir `mi-propedeutico` como carpeta de trabajo, su terminal ya nace parada ahí.

EL BLOQUE QUE COPIARÁS DE GITHUB (DONDE DICE TU-USUARIO, EL TUYO DIRÁ TU NOMBRE)

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

Siete líneas que hacen todo el trabajo de fundación. Léelas antes de ejecutarlas — cada una tiene un oficio:

Comando	Su oficio
`echo "# mi-propedeutico" >> README.md`	Crea el archivo `README.md` con una línea de título — la portada de tu repositorio.
`git init`	Convierte la carpeta en un repositorio Git: nace el álbum donde se guardarán las fotografías.
`git add README.md`	Prepara el archivo para la primera fotografía.
`git commit -m "first commit"`	La toma: el primer commit de tu historial.
`git branch -M main`	Nombra `main` a la rama principal del historial (y de paso resuelve el aviso `hint:` que quizá imprimió `git init`).
`git remote add origin …`	Anota la dirección de tu repositorio en GitHub bajo el alias `origin` — por eso esta línea lleva **tu** usuario.
`git push -u origin main`	Publica: sube el commit a GitHub. El `-u` deja la ruta memorizada; desde ahora bastará `git push`.

> ⚠️ **En el último comando, GitHub te pedirá identificarte**

> Es normal y ocurre una sola vez — y por eso el push se hace **siempre desde la terminal integrada de VS Code**, no desde la Terminal del sistema. `WIN` Aparece una ventanita de *Git Credential Manager* (**Connect to GitHub**): pulsa **Sign in with your browser** y entonces se abre el navegador — inicia sesión con tu cuenta de GitHub y pulsa **Authorize**. `MAC` Primero VS Code muestra un pequeño aviso pidiendo permiso — acéptalo (**Allow**) y después se abre el navegador para autorizar. Tu computadora quedará autorizada para los siguientes push. Si la terminal se queda esperando o falla, revisa P2 y P3.

RESULTADO ESPERADO, RESUMIDO (EL CÓDIGO DEL COMMIT VARÍA)

```
~/Documentos/propedeutico/mi-propedeutico> (pegas el bloque y presionas Er
Initialized empty Git repository in .../mi-propedeutico/.git/
[master (root-commit) f7a8b9c] first commit  ← dirá master (o main); la lí
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
… (aquí te identificas con GitHub — ver aviso arriba) …
Writing objects: 100% (3/3), 230 bytes | 230.00 KiB/s, done.
To https://github.com/tu-usuario/mi-propedeutico.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

¿Alguna línea falló a medio camino? Identifica cuál por su mensaje: «Please tell me who you are» → P1; la autorización no llega o falla → P2 y P3; «remote origin already exists» → P6; «rejected … fetch first» → P8. Resuelto el problema, repite desde la línea que falló (normalmente `git commit -m "first commit"` y `git push -u origin main`).

---

4 Abre tu repositorio en GitHub

Vuelve a la pestaña de GitHub y actualiza la página (`F5` / `⌘R`). La página de **Quick setup** desapareció: en su lugar está tu repositorio de verdad, con `README.md` mostrando el título **mi-propedeutico** y el contador marcando **1 commit** — tu «first commit», con tu nombre y tu fecha.

Detente un segundo en lo que acaba de pasar: la carpeta de tu computadora y esta página son **el mismo repositorio**, conectados. Todo lo que fotografíes con `commit` y empujes con `push` aparecerá aquí. Vamos a comprobarlo creando tu primer archivo de verdad.

---

5 Crea tu primer archivo: la bitácora del curso

1. En el explorador de VS Code (barra izquierda), haz clic derecho en el área vacía bajo la lista de archivos → **New File** (Nuevo archivo) y nómbralo `bitacora.md` — el nombre sin tildes; el *contenido* puede

llevarlas con total normalidad. Debe quedar al mismo nivel que `README.md` (el archivo que creó la línea `echo` del paso 3).

2. Escribe algo como esto (o tu propia versión) y **guarda** con `Ctrl+S` / `⌘S`:

```
# Bitácora del propedéutico

## Clase 3 — 15 de julio de 2026

Hoy creé mi primer repositorio y aprendí el ciclo
editar → add → commit → push.
```

> 💡 **El punto ● en la pestaña significa «sin guardar»**
>
> Mira la pestaña de `bitacora.md` en VS Code: mientras muestre un punto ● en lugar de la ×, lo que escribiste vive solo en la pantalla — en el disco todavía no existe, y para Git «no pasó nada». Al guardar con `Ctrl+S` / `⌘S` el punto se convierte en ×: recién entonces Git puede ver tu cambio. Hazlo reflejo: **terminaste de escribir → Ctrl+S**.

Los archivos `.md` (Markdown) son texto plano con formato sencillo: `#` marca un título, `##` un subtítulo. GitHub los muestra ya formateados — por eso el README de cualquier repositorio se ve «bonito».

---

6 `git status` — pregunta a Git qué ve

Vuelve a la terminal del paso 3 (si la cerraste, abre otra: `Terminal → New Terminal` — nace parada en tu repositorio). `git status` no cambia nada: solo informa — úsalo todo el tiempo, antes y después de cada paso:

```
git status
```

RESULTADO ESPERADO

```
~/Documentos/propedeutico/mi-propedeutico> git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        bitacora.md
```

```
nothing added to commit but untracked files present (use "git add" to trac
```

Cómo leerlo: *untracked* (en rojo) significa que Git ve el archivo nuevo pero todavía no lo vigila ni lo incluirá en ninguna fotografía. La propia salida te sopla el siguiente paso: `git add`.

---

7 `git add` — prepara la fotografía

Coloca el archivo en la bandeja de preparación:

```
git add bitacora.md
```

No imprime nada si todo va bien. Comprueba con `git status`: el archivo ahora aparece en verde, listo para el commit:

RESULTADO ESPERADO

```
~/Documentos/propedeutico/mi-propedeutico> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   bitacora.md
```

💡 Más adelante usarás `git add .` (con punto) para preparar *todos* los cambios pendientes de una vez.

---

8 `git commit` — toma la fotografía

El commit guarda lo preparado en el historial, con un mensaje entre comillas que explica el cambio:

```
git commit -m "Crea la bitacora del curso"
```

RESULTADO ESPERADO (EL CÓDIGO VARÍA)

```
~/Documentos/propedeutico/mi-propedeutico> git commit -m "Crea la bitacora
[main a1b2c3d] Crea la bitacora del curso
 1 file changed, 6 insertions(+)
 create mode 100644 bitacora.md
```

> 💡 **Mensajes de commit que sirven**
>
> Escribe qué hace el cambio, en presente y de forma concreta: «Crea la bitacora del curso», «Anade la entrada de la clase 4». Dentro de unas semanas, `git log` será tu memoria — un historial de mensajes tipo «cambios» o «avance» no le dice nada a nadie, ni a ti. ¿Y por qué «Anade», sin tilde ni eñe? Git las acepta sin problema, pero muchos proyectos las evitan en los mensajes porque algunas terminales antiguas las muestran mal; sigue la costumbre que prefieras — en tus *archivos*, escribe siempre con ortografía normal.

El `-m` es importante — la m de *mensaje*. Sin él, o si la letra sale mal (el typo clásico es `-n`), Git abre un editor para pedirte el mensaje (si te pasó, revisa P5).

---

9 `git push` — publica en GitHub

Hasta aquí, el commit existe solo en tu computadora. Súbelo:

```
git push
```

> 💡 **Esta vez sin trámites**
>
> GitHub ya no te pedirá identificarte: tu computadora quedó autorizada con el push del paso 3. Y como aquel primer push llevó `-u`, ahora basta `git push` a secas — Git ya sabe la ruta. Si aun así te pide algo, revisa P2 y P3.

RESULTADO ESPERADO (LOS NÚMEROS VARÍAN)

```
~/Documentos/propedeutico/mi-propedeutico> git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 428 bytes | 428.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (0/0), done.
To https://github.com/tu-usuario/mi-propedeutico.git
   f7a8b9c..a1b2c3d  main -> main
```

Ahora ve al navegador, entra a `github.com/tu-usuario/mi-propedeutico` y actualiza la página: ahí está `bitacora.md`, y el contador dice **2 commits** (el «first commit» del paso 3 y el de tu bitácora). El ciclo

completo — editar, `add`, `commit`, `push` — acaba de funcionar de principio a fin, hecho por ti.

10 El ciclo completo, una vez más

Lo que convierte esto en hábito es repetirlo. Edita `bitacora.md` — añade al final una línea nueva, por ejemplo *«Publiqué mi primer repositorio.»* — guarda con `Ctrl+S` / `⌘S`, y recorre el ciclo entero:

```
git status
git add .
git commit -m "Anade la primera entrada de la bitacora"
git push
```

Refresca tu repositorio en el navegador: la página muestra **3 commits**. Pulsa sobre ese contador para ver tu historial — cada commit, con tu nombre, fecha y mensaje.

Este es el ciclo que repetirás toda la maestría: editar → `git add` → `git commit` → `git push`. Tu bitácora crecerá una entrada por clase; ese será tu ejercicio permanente.

✅ Salida de la clase

Antes de cerrar la laptop, verifica que puedes marcar todo:

- [ ] Mi repositorio `mi-propedeutico` existe en mi cuenta de GitHub y es público.
- [ ] Su carpeta vive en mi computadora junto a la del curso (hermanas, no anidadas), conectada con GitHub y abierta en VS Code.
- [ ] Hice al menos **tres commits propios**: el «first commit» y dos más con mensajes descriptivos.
- [ ] Mis cambios se ven en la página de mi repositorio en github.com, desde el navegador.
- [ ] Puedo recitar el ciclo sin mirar: editar → `add` → `commit` → `push`.
- [ ] Distingo los dos repositorios: en el del curso solo hago `git pull`; en el mío, todo lo demás.

El ciclo de Git en seis comandos

Tu tabla de consulta rápida de esta sesión. Los cuatro primeros son el ciclo diario; los dos últimos, tus aliados de consulta:

Comando	Qué hace
`git status`	Informa qué ve Git: archivos nuevos, modificados o preparados. No cambia nada.
`git add archivo` · `git add .`	Prepara un archivo (o todos, con el punto) para el próximo commit.
`git commit -m "mensaje"`	Guarda la fotografía en el historial, con tu nombre, la fecha y el mensaje.
`git push`	Sube tus commits a GitHub. Hasta entonces, solo existen en tu computadora.
`git log --oneline`	Muestra el historial resumido: un commit por línea. Sal con la tecla `q` si ocupa más de una pantalla.
`git pull`	Baja las novedades desde GitHub — lo que haces en el repositorio del curso cada semana.

Y, aparte, los cuatro comandos **de fundación** que usaste una sola vez en el paso 3. No hace falta memorizarlos: solo se ejecutan al crear un repositorio, y GitHub te los recuerda en su página de **Quick setup** cada vez que fundes uno nuevo:

Comando	Qué hace
`git init`	Convierte la carpeta actual en un repositorio Git.
`git branch -M main`	Nombra `main` a la rama principal del historial.
`git remote add origin dirección`	Conecta el repositorio local con su dirección en GitHub, bajo el alias `origin`.
`git push -u origin main`	El primer push: publica y deja la ruta memorizada para que después baste `git push`.

Si algo se atasca

▼ **P1 · El commit responde "Please tell me who you are"**

Git necesita saber quién toma la fotografía y falta tu identidad (sección G del Manual de instalación). Configúrala una sola vez, sustituyendo `Nombre Apellido` y `tu-correo@ejemplo.com` por tus datos reales:

```
git config --global user.name "Nombre Apellido"
git config --global user.email "tu-correo@ejemplo.com"
```

Usa el mismo correo de tu cuenta de GitHub y repite el `git commit`.

▼ **P2 · Al hacer push se abre el navegador pidiendo autorización (o no pasa nada)**

Es el camino correcto: inicia sesión y pulsa **Authorize**. `WIN` Antes del navegador aparece una ventanita de *Git Credential Manager* — pulsa ahí **Sign in with your browser**. Si la terminal se queda esperando y no ves nada, mira la barra de tareas: a veces esa ventanita (o el navegador) se abre detrás de VS Code. `MAC` Busca el pequeño aviso de VS Code pidiendo permiso y pulsa **Allow** — el navegador se abre después. Ocurre solo la primera vez; después, `git push` pasa directo.

▼ **P3 · `git push` falla con "Authentication failed" o pide contraseña en la terminal**

GitHub ya no acepta tu contraseña escrita en la terminal. Si te la pidió, cancela con **Ctrl+C** y asegúrate de estar en la **terminal integrada de VS Code** (no en la Terminal del sistema). Repite el push que falló (en el paso 3 es `git push -u origin main`; en adelante, `git push` a secas) y esta vez acepta todo lo que aparezca: el aviso de VS Code pidiendo permiso (**Allow**) y la ventana del navegador (**Authorize**). Si sigue fallando, avisa al docente — se resuelve en un minuto y no es culpa tuya.

▼ **P4 · "nothing to commit, working tree clean" — pero yo sí cambié el archivo**

Casi siempre el archivo no está **guardado**: mira la pestaña en VS Code — un punto ● en lugar de la ✕ indica cambios sin guardar. Guarda con **Ctrl+S** / **⌘S** y repite `git status`. Si ya estaba guardado, puede que estés en la carpeta equivocada: `pwd` debe terminar en `mi-propedeutico`.

▼ **P5 · Olvidé el `-m` (o escribí `-n`) y la terminal se convirtió en algo extraño**

Git abrió un editor para pedirte el mensaje. Dos escenarios:

1. **Pantalla oscura dentro de la terminal (Vim):** presiona **Esc**, escribe `:q!` y presiona Enter. Sales sin guardar y el commit se cancela.
2. **Se abrió una pestaña en VS Code (COMMIT_EDITMSG):** escribe el mensaje en la primera línea, guarda y cierra la pestaña — el commit se completa.

En cualquier caso, la próxima vez: `git commit -m "mensaje"`.

▼ **P6 · Mi carpeta quedó dentro del repositorio del curso (o pegué los comandos en la carpeta equivocada)**

Dos variantes del mismo despiste:

1. **La carpeta `mi-propedeutico` quedó dentro de `propedeutico-flacso`.** Git se confunde con repositorios anidados, así que muévela: cierra VS Code y, con el explorador de archivos o el Finder, arrastra la carpeta completa hacia afuera, a `Documentos/propedeutico`, junto a la del curso. No se rompe nada — incluso si ya habías ejecutado los comandos del paso 3. Vuelve a abrirla desde VS Code.
2. **Pegaste los comandos con la terminal parada en otra carpeta.** La señal típica: `git remote add origin` responde `error: remote origin already exists` — estabas dentro del repositorio del curso. Para dejarlo como estaba, ejecuta ahí mismo `git reset --hard origin/main` (devuelve el repositorio del curso exactamente a su estado publicado; ahí no debes tener cambios propios). Después abre tu carpeta correcta (**File → Open Folder** → `mi-propedeutico`) y repite el paso 3. Si dudas, pide ayuda al docente o usa la plantilla de IA de más abajo.

▼ **P7 · Hice commit pero en github.com no aparece nada nuevo**

El commit vive en tu computadora hasta que lo empujas: ejecuta `git push` y refresca la página. Si push responde `Everything up-to-date`, entonces no hay commits pendientes de subir — revisa `git status`: probablemente el cambio no pasó por `add` y `commit`.

▼ **P8 · El primer push responde "rejected" o "failed to push some refs" y sugiere "fetch first"**

Casi seguro marcaste **Add a README file** (o una licencia) al crear el repositorio: GitHub hizo un commit propio que tu historial local no conoce, y los dos historiales chocan. Como ese commit de GitHub es solo el README automático — nada tuyo —, lo más simple es sobrescribirlo:

```
git push -u origin main --force
```

`--force` le dice a Git «lo mío manda» y reemplaza lo que había en GitHub. Úsalo solo aquí, en tu repositorio recién nacido y tuyo — **nunca** en un repositorio compartido con otras personas.

> 💡 **¿Tu problema no está en la lista? Pídele ayuda a una IA**
>
> Un asistente de IA (ChatGPT, Claude, Gemini o Copilot) resuelve muy bien este tipo de atascos **si le das contexto suficiente**. Copia esta plantilla, rellena los huecos [entre corchetes] y pégala en el chat:
>
> ```
> Estoy siguiendo un manual del curso propedéutico de FLACSO
> y me atasqué en [paso o sección del manual donde estabas].
> Mi sistema operativo es [Windows o Mac].
> Escribí este comando o código:
> [pega aquí exactamente lo que escribiste]
> Y obtuve este mensaje de error o problema:
> [pega aquí el mensaje completo, copiado desde la terminal]
> Por favor: primero explícame por qué ocurre el problema,
> en lenguaje sencillo (no tengo experiencia en programación),
> y después dame los pasos para resolverlo, uno por uno.
> ```
>
> Tres consejos para que la respuesta sirva:
>
> - **Copia el error completo como texto** — selecciónalo en la terminal y cópialo — mejor que una captura de pantalla o un fragmento suelto.
> - **Lee cada comando que la IA te sugiera antes de ejecutarlo**; si no entiendes qué hace, pregúntale. Nunca ejecutes a ciegas comandos que borren archivos o carpetas.
> - Si la primera respuesta no funciona, **cuéntale qué pasó al intentarlo**: la conversación de ida y vuelta es parte normal del proceso.

Qué sigue

Ya dominas los dos sentidos del tráfico: *recibir* con `git pull` (clase 2) y *publicar* con `add` → `commit` → `push` (hoy) — y además fundaste un repositorio desde cero, algo que mucha gente aprende bastante después. En la próxima sesión volvemos a los notebooks — algoritmos, tipos de variables y operadores en Python — y tu bitácora empieza a trabajar: al final de cada clase, escribe una entrada nueva y publícala. Ese pequeño ritual es tu respaldo, tu registro de avance y el primer contenido de tu portafolio público.

Y para consolidar la vista de conjunto, lee el **Mapa de las herramientas**: qué hace (y qué no hace) cada pieza del taller, el semáforo de señales de VS Code y los errores de siempre con su remedio. Es el manual de consulta del resto del curso.

Propedéutico de Matemáticas y Programación · FLACSO Ecuador — Manual de sesión: tu propio repositorio, versión julio 2026.

Continúa el **Manual de la clase 2** (GitHub y tu primer entorno) y complementa al **Manual de instalación** (VS Code, Python y Git). Manual de consulta: el **Mapa de las herramientas**.