**Temas a desarrollar:**

-**Estructuras secuenciales y condicionales.**

-Trabajo colaborativo en programacion.

-Bucles y estructuras repetitivas.

-Bucles y estructuras repetitivas.

-Funciones y reutilización de codigo.

-Estructuras de datos y su manejo.

-Control de errores y depuración.

-Test unitarios y buenas prácticas.

-Análisis de algoritmos.

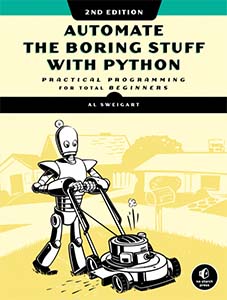
-Algoritmos de búsqueda y ordenamiento.

-Recursividad y optimización.

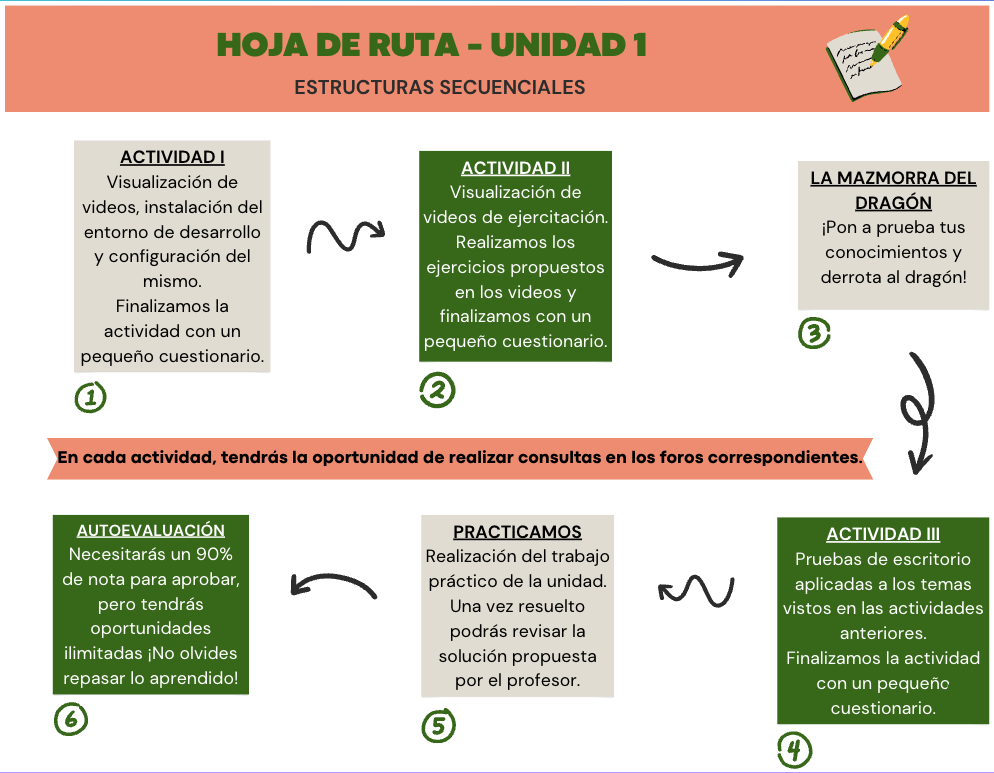
-Introducción a estructuras de datos avanzadas.

BIEN PARA CONTINUAR LA BIBLIOGRAFIA SERIA LA SIGUIENTE:

<https://automatetheboringstuff.com/#toc>



Bueno dicho de donde se sacara la info que nos formara en esta materia, empecemos viendo el recorrido que vamos a transitar cada semana en unidades:



Vamos con el principio de todo…

**- ¿QUE ES PROGRAMAR?**

En primera instancia es indicarle a la computadora los pasos que debe realizar para resolver un determinado problema mediante un codigo escrito en lenguaje de programacion. Tenemos que tener en cuenta que estos pasos ya fueron perfectamente analizados.

¿Como logramos comunicarnos con una máquina que ni siquiera habla nuestro mismo idioma? Para ello están los lenguajes de programacion los cuales existe una gran diversidad que ya estuvimos mencionando en clase. Lo importante es que lo que termina haciendo el programador no es solamente escribir codigo, ya que esa es la parte mas facil por asi decirlo, lo que importa es que codigo escribir y para que, no solo a la hora de aplicarlo sino tambien para formarnos. Aun asi sabiendo como y donde escribir codigo las fallas van a seguir estando, es decir que vamos a lidiar muchisimo con ellos y hay que mentalizarse en que estos son parte fundamental de nuestro aprendizaje. Para solucionar estos errores lo primero que hacemos es acudir a lo que nostros sabemos e ir repasándolo, luego puede tocar averiguar sobre el mismo en la web e incluso charlarlo con algun compañero, porque es parte del proceso de aprendizaje.

Todo este proceso puede parecer intimidante al principio debido a la incertidumbre, pero cuando nuestras creaciones funcionan créanme que no hay mejor sensación que esa. Ademas si el software que nos interesa existe lo usamos, si es mejorable lo van a poder mejorar y si no existe lo van a poder crear.

Para poder empezar a aprender todo esto, es necesario desarrollar una habilidad fundamental en nuestra mente y es el **pensamiento computacional.**

**-El PENSAMIENTO COMPUTACIONAL**

El termino Pensamiento computacional proviene del pensador Seymour Papert (1928-2016) fue un científico de la computacion y pionero en el campo de la IA pero uno de sus mas grandes aportes sin duda fue en el ambito de la psicología ya que a principio de los años 60´ trabajo con Piaget del cual tomo sus ideales para desarrollar su propia idea de método de aprendizaje. Fue creador del primer lenguaje de programacion para niños llamado Logo que consistía en que estos pudieran programar a una tortuga en base a diferentes instrucciones pre armadas y que esta de forma grafica se mueva para ver si se solucionaba el problema o no lo que tambien permitía que los chicos pudieran autocorregirse a través de estas instrucciones y resultados gráficos.

Si bien las ideas planteadas por Papert marcaron un antes y despues, quien desarrolla el pensamiento computacional y lo promociona en realidad fue **Jeannette Wing(1956)** una científica de la computacion y que actualmente es directora en el instituto de ciencias de datos de la universidad de Columbia.

Para ella el pensamiento computacional ***“Implica resolver problemas, diseñar sistemas y comprender el comportamiento humano, basándose en los conceptos fundamentales de la ciencia de la computacion”*** y además menciona que “***representa una actitud y unas habilidades universales que todos los individuos, no solo los científicos computacionales, deberían aprender y usar”.*** Entonces partiendo de las palabras estas, no necesitamos volvernos ingenieros o hackers para desarrollar nuestro pensamiento lógico, sino que van a poder aplicarlo en cualquier aspecto de su vida.

Como menciona Jeannette el pensamiento computacional implica resolver problemas, pero ¿que entendemos por **problema?**

Es una situación incierta que provoca la necesidad de hallar una solución y para poder hacer esto se utiliza tanto para problemas simples como para problemas complejos un proceso el cual llamaremos:

**Proceso de resolución de problemas**

**1)ANALISIS**

En esta primera fase vamos a comprender e interpretar el enunciado del problema, identificar elementos explícitos de este, desambiguación (significa realizar alguna acción para que el discurso sea más facil de entender) del sentido de las palabras, entender lo que se pide y dejar de lado lo que no es importante. Tenemos que tener bien claro cual es nuestro objetivo.

**2)DISEÑO**

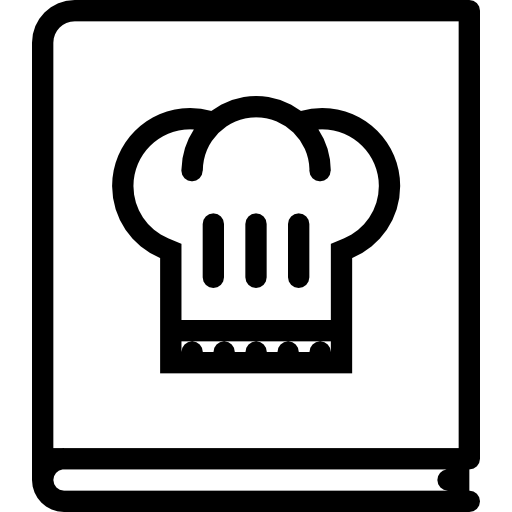
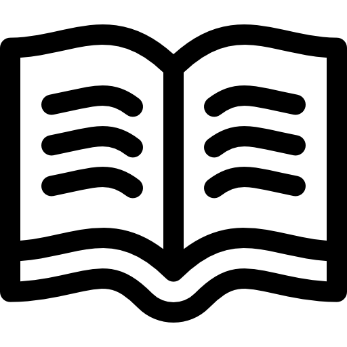
Este es el momento de elaborar un plan de resolución el cual consiste en primera instancia en dividir el problema en pequeñas partes que lo componen y que son mas simples de resolver de a uno. Por otra parte hay que reconocer patrones es decir tenemos que identificar que cosa ya hemos visto en otros problemas y podria proponerse una o mas estrategias para abordar por analogía, en otras palabras es relacionarlo con lo que ya sé de otros problemas. En esta fase se necesitan **datos de entrada** para resolver ese problema.

**3)ALGORITMO**

Luego de las anteriores fases estamos listos para establecer cuales son los pasos que tiene que realizar la maquina para lograr llegar al resultado final esperado. Pero la palabra algoritmo es mucho mas compleja asique como vimos en clase se las voy a desarrollar a continuación.

**ALGORITMO**

Se llama asi al conjunto de pasos o instrucciones con un orden y una terminación que conduce a la solución de un determinado problema. Para que una maquina como una computadora pueda llevar a cabo una tarea o resuelva un problema es preciso diseñar y representar un algoritmo en una forma que sea compatible con la máquina.

RECETA MANUAL DE INESTRUCCIONES  ATARSE LOS CORDONES

Estos son claros ejemplos de algoritmos que tenemos en nuestro día a día y están pensados para que los ejecute un ser humano. Ahora cuando queremos que esos algoritmos los ejecute una maquina debemos cumplir con ciertos requisitos para que sean satisfactorias la comunicación.

|  |  |
| --- | --- |
| FINITO | Osea que debe tener un final, no puede ser infinito. |
| ORDENADO | Debe seguir un orden lógico para poder llegar al resultado que se espera. |
| PRECISO | No debe dar pie a ambigüedades es decir que se pueda interpretar de distintas formas. |
| DEFINIDO o EXACTO | Es decir, si ejecutamos muchas veces el mismo algoritmo el resultado siempre será el mismo. |

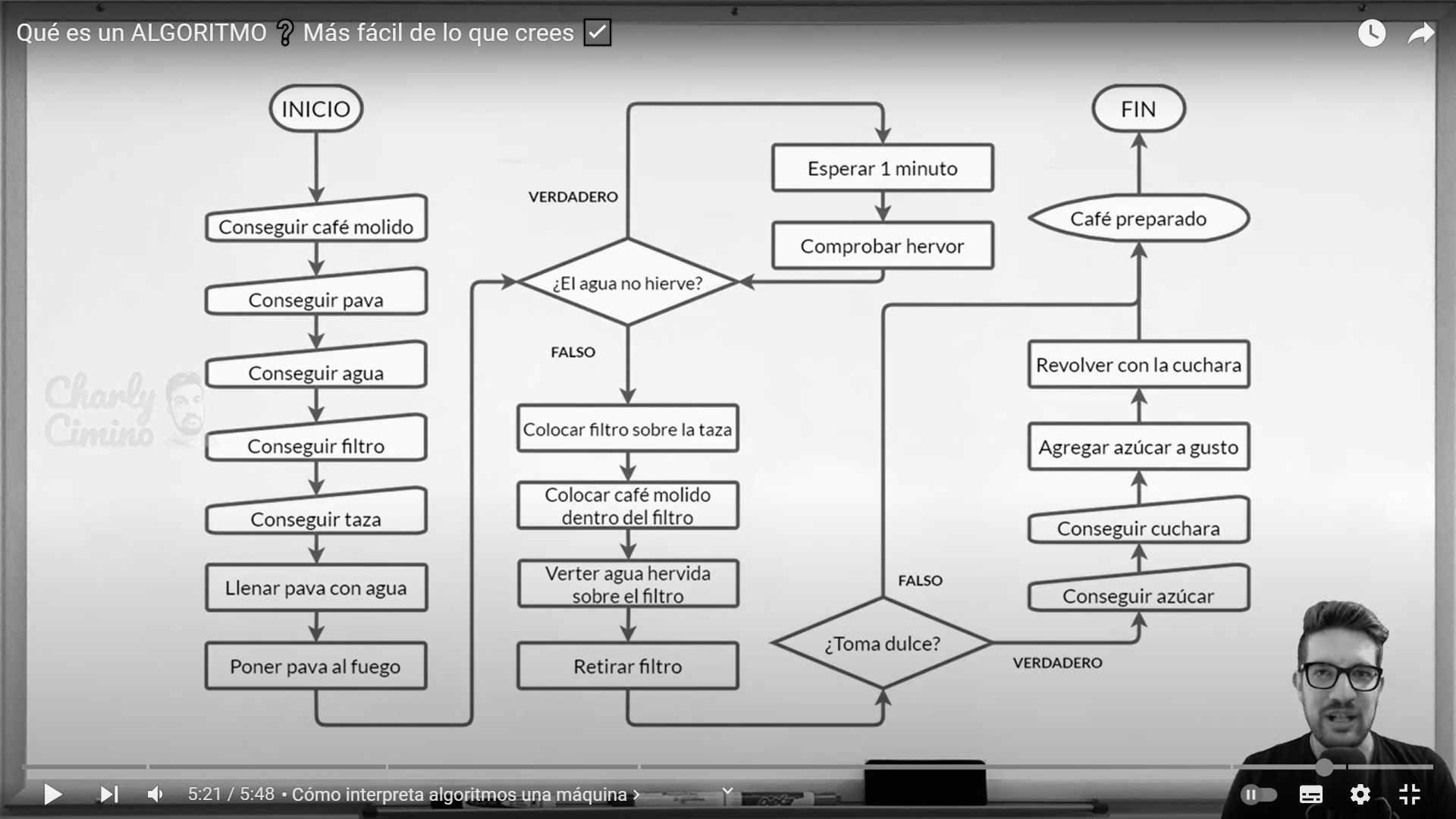
Si lo viéramos como un resumen muy global nuestro algoritmo debería estar representado asi:

SALIDA

PROCESO

ENTRADA

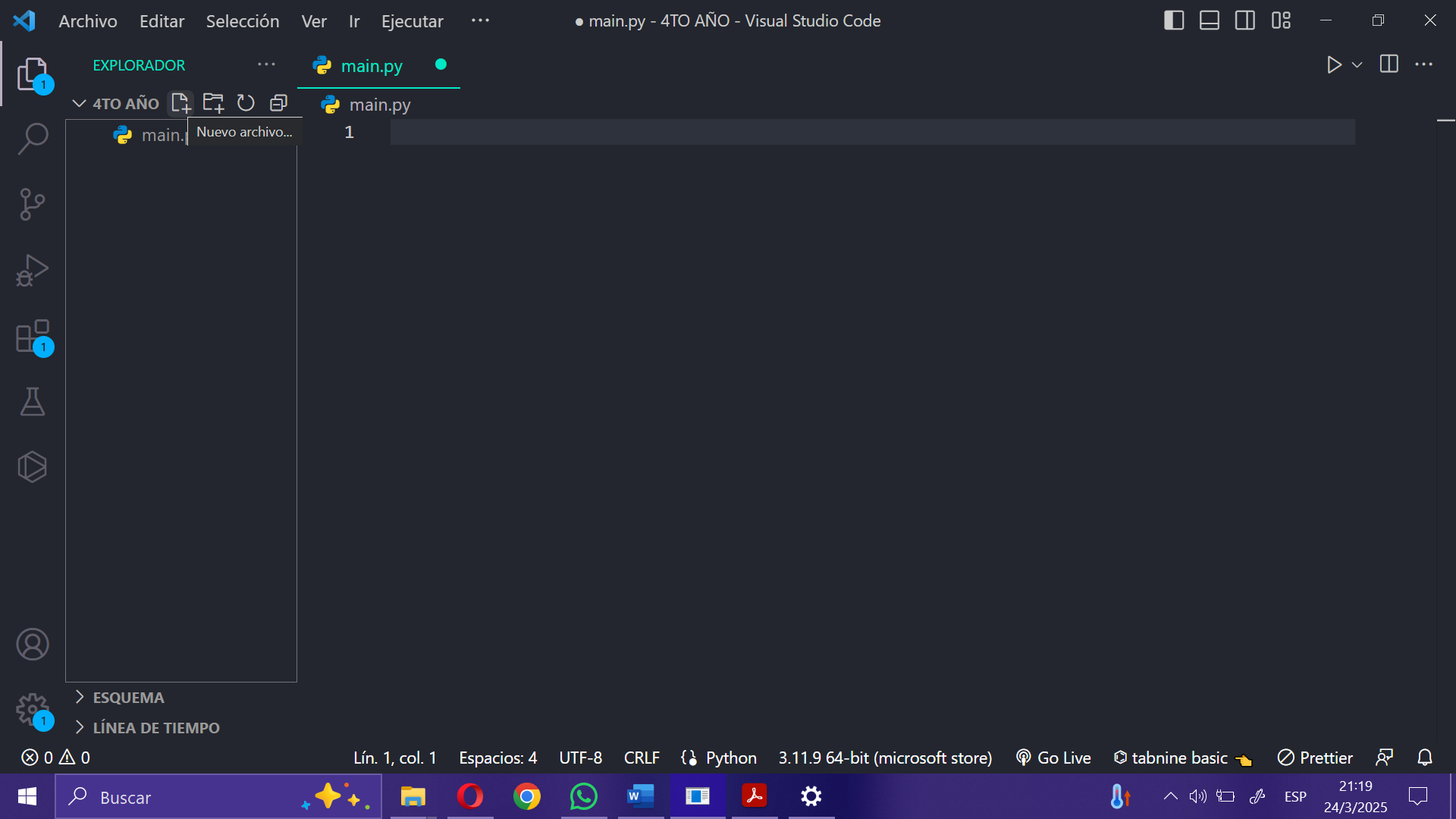
Vamos con un ejemplo de algoritmo representado en diagrama de flujo para “**hacerse una infusión de café para la merienda**”.



|  |
| --- |
| ATENCION:  -Antes de continuar debes ver el archivo **Diagramas de flujo y algoritmos.**  -Antes de continuar debes ver el archivo **Requisitos para Empezar.** |

Luego de haber instalado lo que es Python, Visual Studio Code con sus respectivas extensiones y git, procedemos a iniciar nuestra primera interacción con el terminal de nuestro interprete.

Para comenzar tenemos que crear un nuevo archivo de extensión **.py** en el cual vamos a escribir, entonces hacemos **click** en **nuevo archivo**: “*seria en la hoja con el +”*



Como les eh comentado en algunas ocasiones los archivos tienen extensión que definen o al menos nos dan un indicio el tipo de datos que contiene, por lo tanto asi como existe el **.mp3** o el **.mp4** y **.txt** . Tambien existe el tipo de extensión para Python, **.py** antepuesto por algun nombre de dicho archivo, en el caso de la imagen de ejemplo es main.

**main.py**

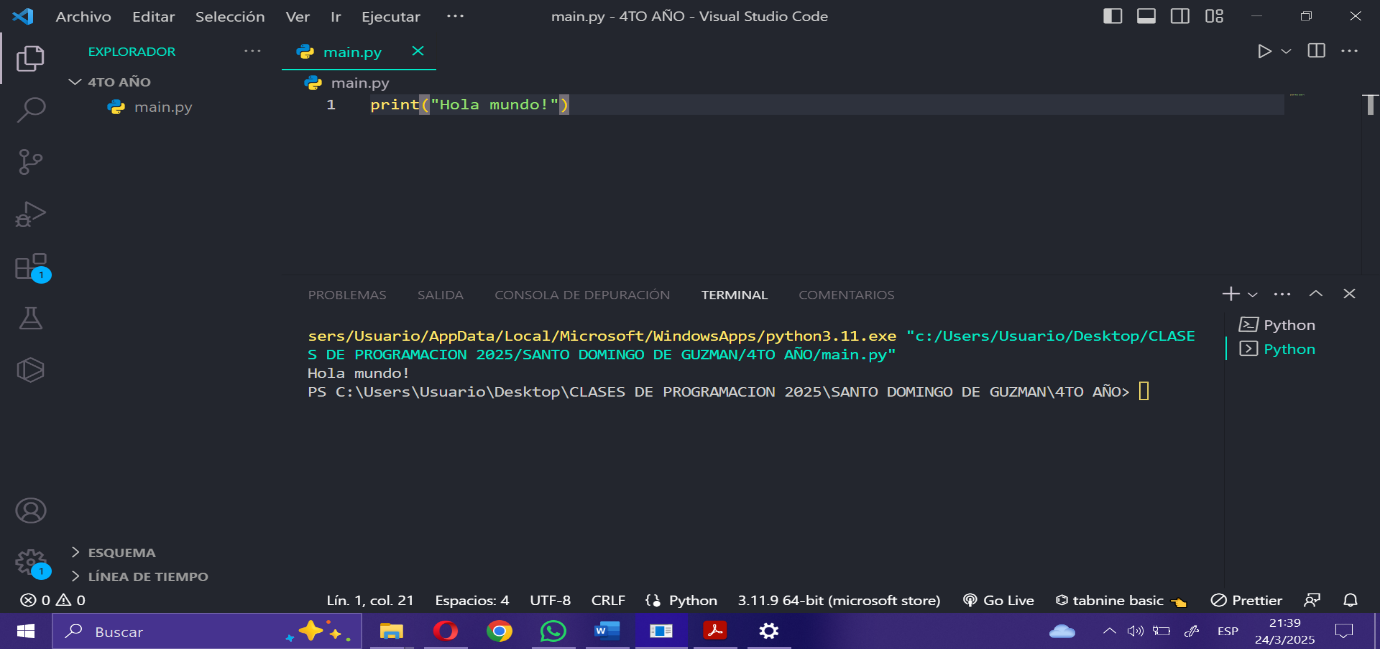
¡Dicho esto, ya podemos escribir nuestro mensaje!

1-Poner la palabra **print** seguida de paréntesis.

2-Dentro de los paréntesis vamos a escribir el mensaje **“hola mundo”** entre comillas.

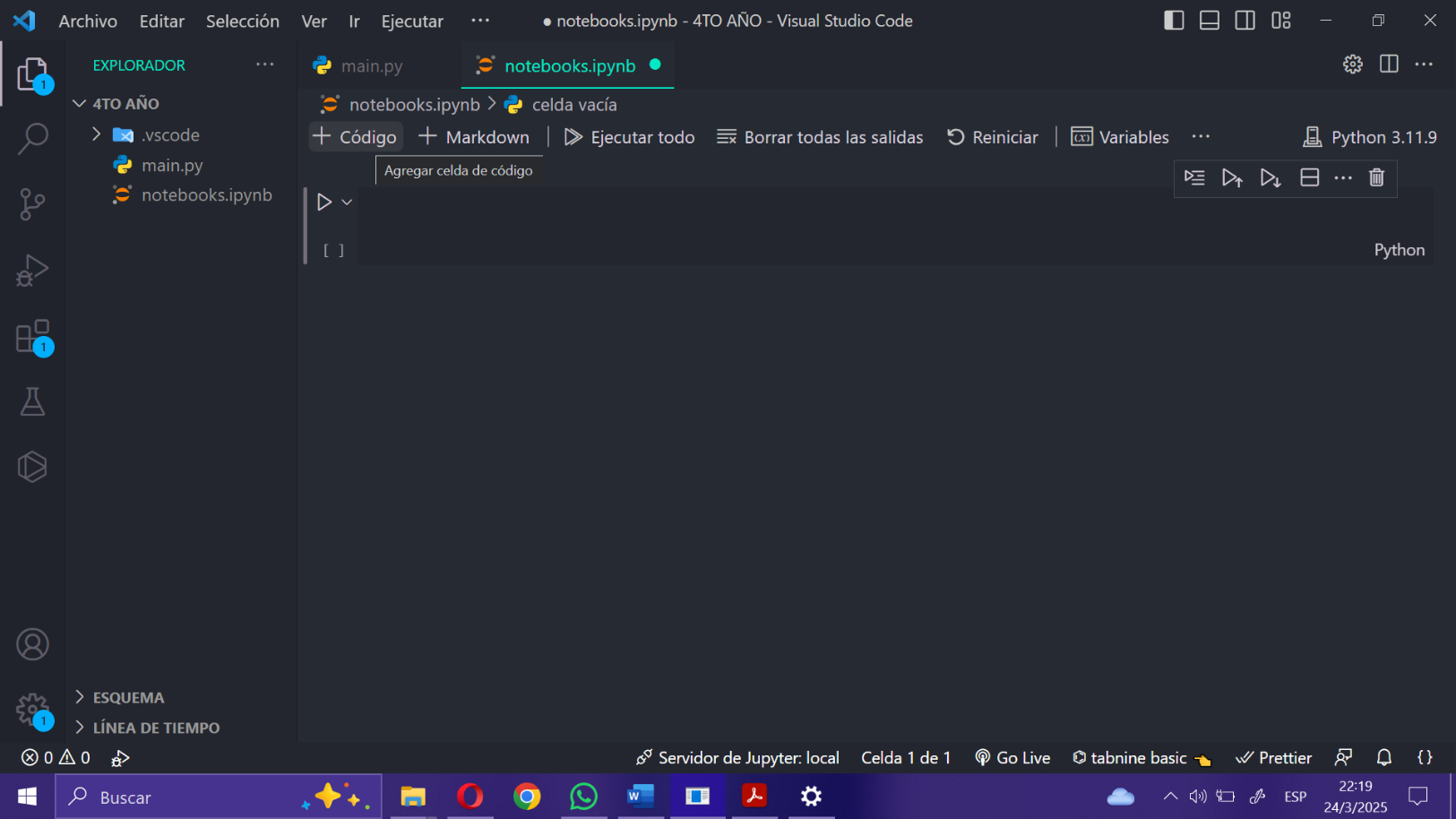
3-Para ejecutar nuestro primer programa y ver este mensaje seleccionamos el botón de play en la parte superior derecha.

3-Sino otra opcion es **click derecho** y seleccionamos **Run Python File in Terminal** o **ejecuta el archivo de Python en terminal** ydeberíamos ver dicho mensaje en el terminal.

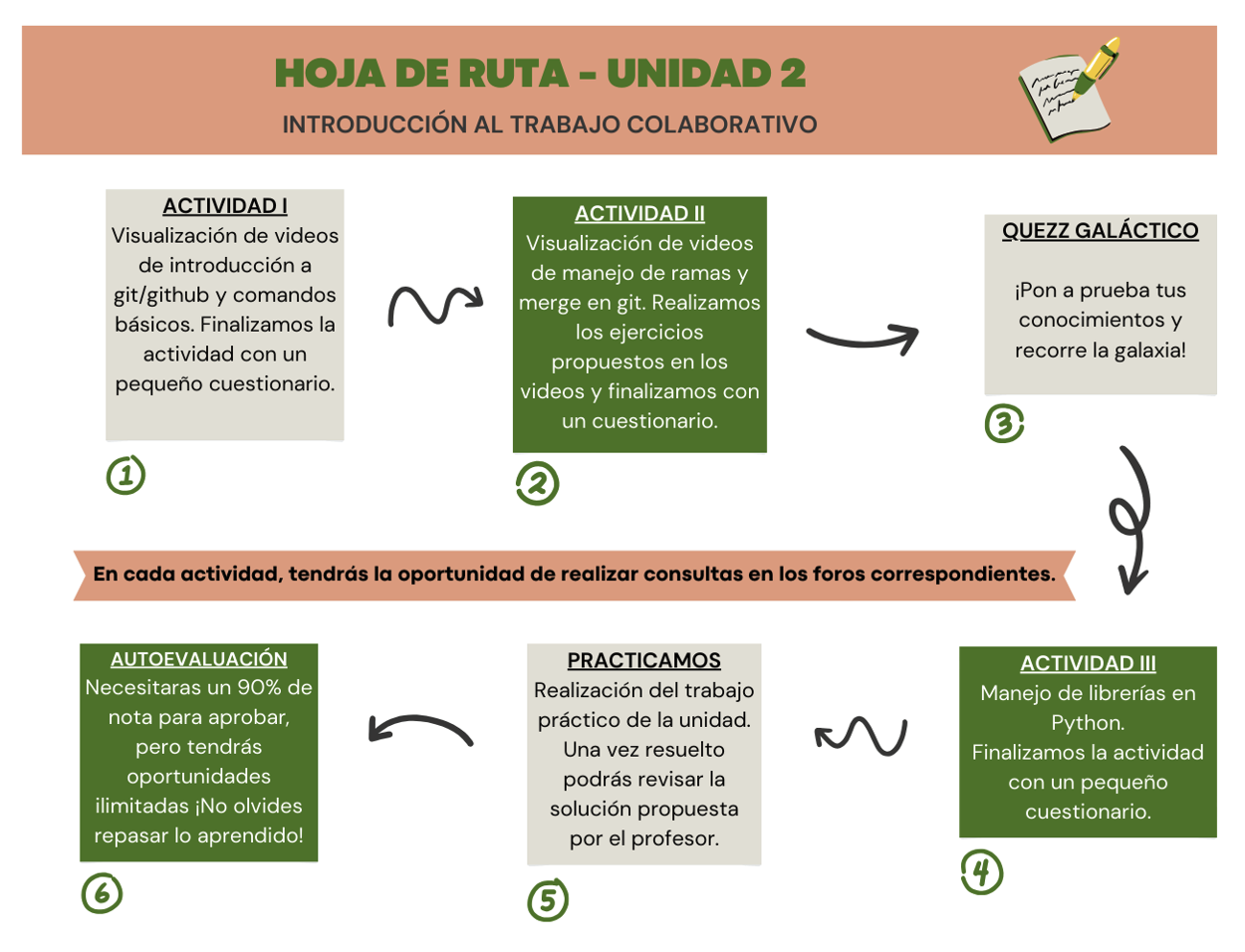


**Jubyter notebooks**

Es un entorno de darrollo dinámico que nos permite escribir secciones de codigo y tambien secciones de documentación para ir anexandolo a lo largo del archivo y que esto nos sirva como herramienta la cual permite la posibilidad de trabajar ejecutando individualmente nuestro codigo y eso es creando un archivo con extensión **.ipynb**



Tienen la opcion **+codigo** de agregar un cuadrado de codigo que podrán ejecutar directamente con el **PLAY** del costado y tambien donde dice **+ Markdown** pueden agregar espacio para alguna nota de texto. En fin es buena herramienta para trabajar tomando notas.



**Unidad 2-Introduccion al trabajo colaborativo**

**NOMBRA A: CVS, Subversion, Mercurial, Monotone, pero el mas importante es …**

**Git** es un sistema de control de versiones distribuidos, usado principalmente en el desarrollo de software. Permite:

**-Guardar diferentes estados de un proyecto.**

**-Volver atrás si se comete un error.**

**-Trabajar colaborativamente sin sobrescribir archivos ajenos.**

Creado por Linus en 2005 para gestionar el desarrollo del núcleo de Linux, se destaca por su velocidad, eficiencia y seguridad.

**BRANCH(Rama)**

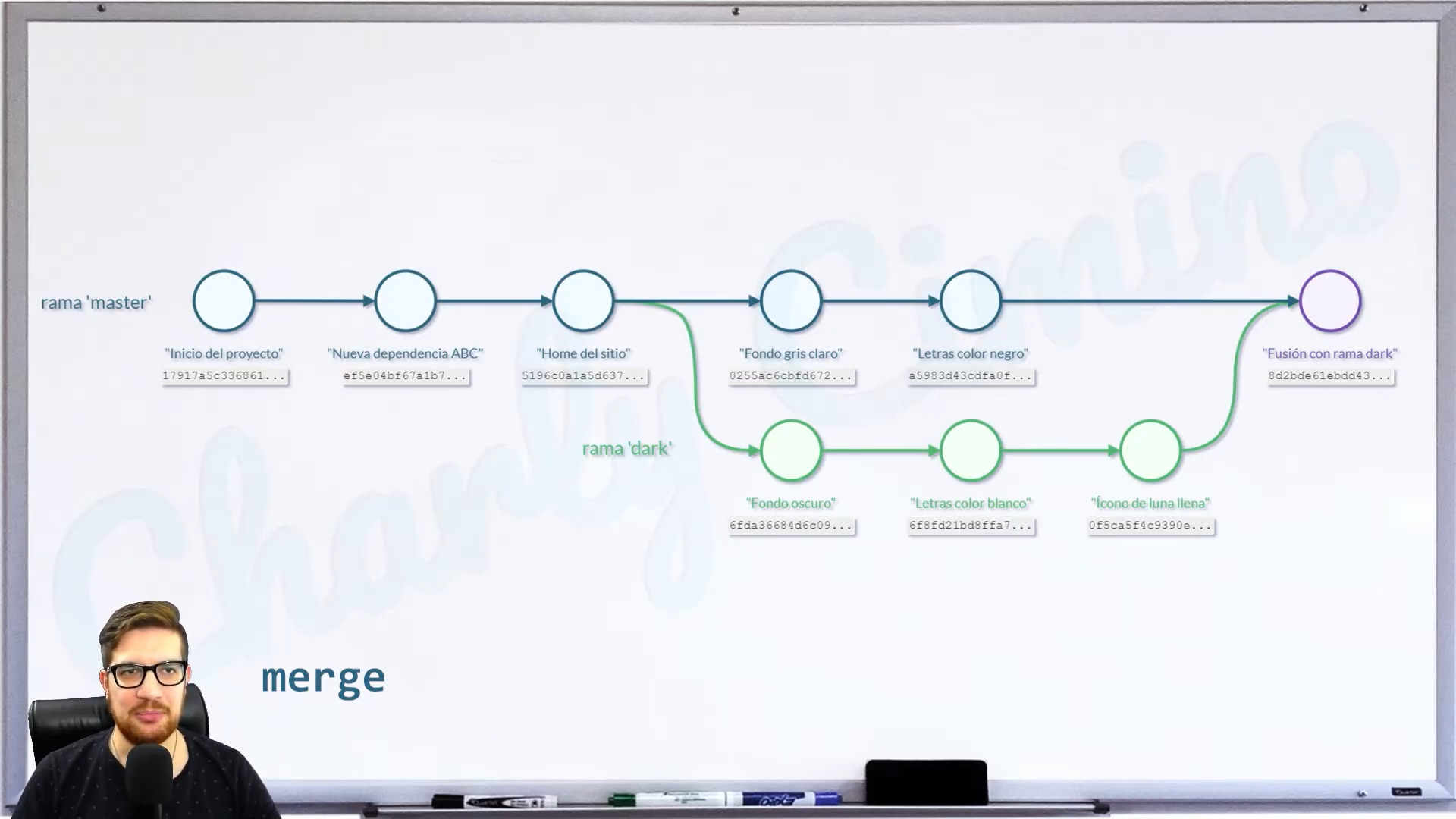
Es como una cronología o línea del tiempo de desarrollo independiente. Es como hacer una copia temporal del proyecto para trabajar sin afectar la versión principal. Se va a poder plasmar los distintos puntos en donde hubo cambios.

La rama principal suele llamarse **MAIN o MASTER.**

Podes crear una rama para desarrollar una nueva función, y despues, unirla a MAIN.

Podemos desarrollar el sitio por ejemplo en 2 versiones abriendo una nueva rama, en donde uno sea color claro y el otro sea color oscuro, tener dos ramas implica que tengamos 2 versiones del proyecto, donde se puede mantener los cambios en paralelo, cuando saco una rama secundaria se denomina **RAMA DARK.**

Si por alguna razon esa rama dividida en 2 se tiene que volver a unir en alguna parte a eso se lo denomina **MERGE (union)**



**COMMIT**

Es como una foto instantánea del proyecto, es un “punto de guardado” dentro de git. Cada vez que se hace un cambio importante, se guarda con un mensaje explicativo.

Incluye:

**Que archivos cambiaron.**

**Quien hizo el cambio.**

**Cuando se hizo.**

**Mensaje descriptivo.**

Tambien se dice que es mas conveniente tener varios commit a que tener solo uno que abarque todos los cambios realizados.

Todo esto que mencionamos sobre GIT funciona a nivel local en una computadora, pero si quisiéramos trabajar en equipo con compañeros de la oficina o codigo abierto con gente de cualquier parte del mundo deberíamos trabajar con:

**REPOSITORIOS REMOTOS**

Puede lograrse mediante una red privada, o desde internet. Para ello necesitamos un lugar donde alojar nuestro repositorio como Gitlab O Sourceforge, pero nosotros vamos a trabajar con **GIT-HUB.** Aca es donde generalmente deberíamos cargar nuestrosproyectos para presentar a las empresas, importa muchisimo mas incluso que el curriculum.

Creamos un repositorio en github, luego lo asociamos mediante comandos con los repositorios locales nuestros y estaremos en condiciones de hacer lo que se conoce como PUSH que es tener una copia de todo lo que codeamos en nuestra pc, para los servidores de github.

Luego en otra computadora en la que estemos y querramos continuar laburando vamos a instalar el cliente de git para realizar lo que se conoce como un **CLONE**, es decir que descargamos una copia exacta de lo que tenemos subido para poder editarlo, hacerle commit, Branch, etc y para mantenerlo sincronizado una vez que aplicamos cambio, vamos a trabajar con los comandos **PULL|PUSH** que actualiza constantemente los cambios. Por otro lado tenemos el **PULL REQUEST** lo cual es básicamente proponerle cambios a un repositorio de github que nos haya gustado y nos descargamos para trabajarlo, pero es de otra persona a la cual no conocemos y queremos mejorar su codigo o aportarle de alguna forma.

Este seria básicamente lo que hacen los programadores de todas partes del mundo cuando laburan en remoto.