PRÁCTICAS DE MECÁNICA

INTRODUCCIÓN

2º Curso de Ingeniería de Organización Industrial

CENTRO UNIVERSITARIO DE LA DEFENSA DE ZARAGOZA

1.- Introducción.

Las prácticas de la asignatura de "Mecánica" han sido rehechas como resultado del Proyecto de Innovación Docente *Prácticas de Mecánica con software libre* desarrollado dentro de la II convocatoria de "Proyectos de Innovación Docente" del "Centro Universitario de la Defensa de Zaragoza".

Se mantienen los sistemas estudiados en otros años, por lo que quienes tengan aprobadas las prácticas de otros cursos no se verán obligados a repetirlas y se conservará la nota que hubiesen obtenido previamente. Sin embargo, se ha modificado el desarrollo de las mismas.

El principal cambio es que las prácticas se han organizado en torno a *cuadernos* escritos sobre la aplicación *Jupyter Lab*, haciendo uso del lenguaje de programación *Python*.

Se sigue manteniendo una práctica de "Estática", otra de "Cinemática" y una tercera de "Dinámica". Asimismo, las prácticas se realizarán en grupos pequeños, de dos o tres alumnos, cuya composición habrá que notificar al profesor.

Conforme se termine el temario teórico-práctico correspondiente a cada una de las prácticas, se liberará en el curso virtual de *Moodle* de la asignatura una versión incompleta del cuaderno necesario para realizar la práctica correspondiente junto con una primera parte del guion de la misma. En esa primera parte del guion, se da una introducción teórica al sistema estudiado y se pide que se complete el *cuaderno* de modo correcto, de acuerdo a una serie de instrucciones. Este trabajo deberá ser realizado por los miembros de cada grupo de prácticas de modo autónomo, aunque con supervisión por parte del profesor en caso necesario, con antelación a la fecha de realización de la segunda parte de la práctica, que se hará en un aula de Informática. Es importante que la primera parte se haga cuanto antes tras su liberación en *Moodle* ya que es cuando los conceptos teóricos están más frescos en la mente y cuando de mayor ayuda será repasar el funcionamiento del sistema y su traducción a un programa informático de simulación numérica del mismo.

De acuerdo al calendario del curso, hay una serie de sesiones de prácticas en las "Aulas de Informática". Habrá un total de tres sesiones correspondientes a cada una de las prácticas previamente liberadas. A dichas sesiones habrá que acudir con el correspondiente *cuaderno de Jupyter* completo y funcional (se puede llevar en un USB, o tenerlo cargado en un disco compartido de google al que se tenga acceso, o tenerlo cargado en la página de la *Universidad de Zaragoza* http://jupyter.unizar.es/ , ...). En las sesiones presenciales, se hará llegar a los alumnos una segunda parte del guion así como un cuestionario que deberán rellenar. Se tratará de modificar parámetros de funcionamiento del sistema para estudiar cómo responde ante dichos cambios. El cuestionario será la herramienta empleada para evaluar y, puesto que se trata de modificar parámetros sobre un sistema previo, se entiende la necesidad de acudir a las sesiones con un *cuaderno* correcto y que sea capaz de reproducir el funcionamiento del sistema base sobre el cual se aplicarán las modificaciones que se indiquen en la segunda parte del guion. Se aconseja acudir a las sesiones con la primera parte del guion impresa, para poder consultarla, ya que en ella se contiene la explicación del sistema estudiado que se dará por sabida durante el desarrollo de la sesión presencial.

2.- El lenguaje de programación *Python*.

Se trata de un lenguaje de programación muy utilizado por las áreas de Tecnología en los Institutos de ESO y Bachillerato, lo que ha motivado su elección ya que habrá alumnos que estén familiarizados con el mismo.

Su principal característica es que se diseñó pensando en que su código fuente fuera fácilmente legible por un humano, las reglas sintácticas del lenguaje son muy simples, y ésta es la razón principal de su popularidad.

Por lo demás tiene una serie de propiedades interesantes:

1. Es multiparadigma. Permite programación imperativa (se dan una serie de órdenes que hay que ejecutar sucesivamente), objetiva (se diseñan unos tipos especiales llamados objetos a los que se aplican de modo repetitivo una serie de operaciones llamadas métodos; un objeto bien diseñado y libre de fallos se puede emplear como "ladrillo" en la construcción de un programa más complejo) y funcional (el código se organiza en torno a funciones que producen un resultado a partir de unas variables y que, a su vez, pueden ser variables o resultados de otras funciones, incluso de modo recursivo); aunque su soporte para ésta última no es tan bueno como el de otros lenguajes.

2. Es multiplataforma. Se puede ejecutar en cualquier ordenador y sistema operativo que disponga de un intérprete de *Python*. Esto incluye *Windows, Apple* y *Linux.* Asimismo, existen intérpretes en linea capaces de ejecutar el código que se les pase a través de un navegador web.

3. El tipo de las variables es dinámico (se decide en tiempo de ejecución) y fuerte (si se intenta emplear una variable de tipo erróneo, sin haber realizado una conversión de tipos explícita, se produce un error que hay que corregir). Esto facilita el diseño de los intérpretes de *Python,* pero obliga a ser cuidadoso con el manejo de los tipos (tipos fuertes) y puede penalizar la velocidad de ejecución (tipos dinámicos).

4. Es interpretado. No se compila a código máquina sino que un intérprete se encarga de ejecutar el código de modo interactivo. Esto facilita el desarrollo rápido de una aplicación a través del sistema de "prueba y error", aunque penaliza el rendimiento cuando haya que trabajar con grandes conjuntos de datos y operaciones.

5. Dispone de un gran número de bibliotecas de funciones y aplicaciones que extienden la funcionalidad básica del lenguaje haciendo uso de código escrito en lenguajes compilados (y por lo tanto muy rápido en su ejecución) con el se pueden intercambiar los datos y variables resultado de los cálculos a través de una interfaz que oculta al usuario final los detalles del código compilado que invocan las funciones de *Python* manejadas por el usuario y cuya explicación figura en los manuales correspondientes. Esto sirve de ayuda para minimizar los problemas causados por el carácter interpretado de *Python*.

3.- El entorno de programación *Jupyter*.

Jupyter es un entorno de computación interactiva desarrollado a partir del concepto de *cuaderno*: a través de un navegador web se puede acceder a un entorno organizado en celdas de entrada y salida en las que se pueden ir ejecutando bloques de código, escribir texto, representar gráficas y añadir materiales interactivos. Mediante *Jupyter* es posible analizar datos, desarrollar algoritmos y crear modelos o aplicaciones. Aunque su ejecución no resulta tan rápida como la de un programa compilado, *Jupyter* resulta muy útil para el desarrollo de prototipos.

Aunque soporta varios lenguajes de programación (núcleos de acuerdo a la terminología de *Jupyter*) cuyos resultados se pueden ir evaluando de modo interactivo en las celdas correspondientes, en esta práctica se trabajará con el núcleo que depende del lenguaje *Python* que fue el primero de los existentes y el más extendido.

Jupyter tiene el aspecto que se muestra en la figura 1. La lista de archivos disponibles (los cuadernos tienen extensión .ipynb) para ser cargados se encuentra en la columna de la izquierda.

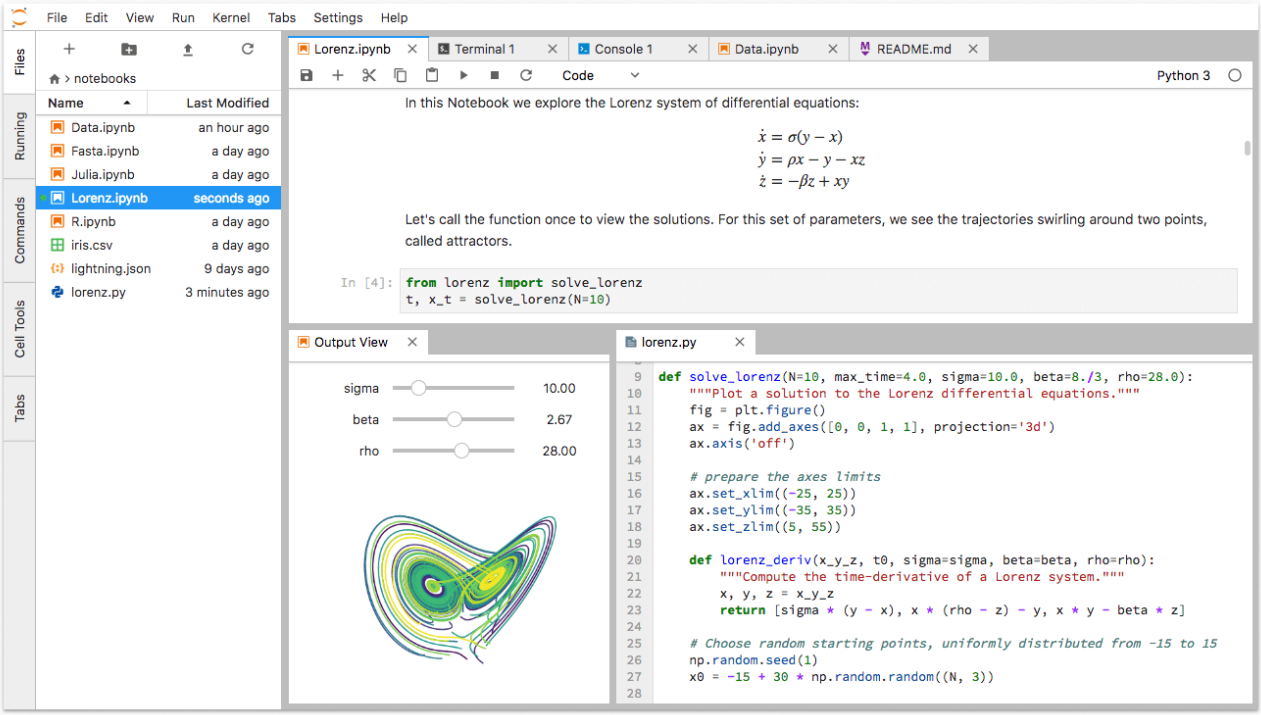

Figura 1: Aspecto de una sesión de *Jupyterlab* en ejecución.

A través de las distintas ventanas en que se divide el espacio de trabajo es posible editar archivos de texto, representar datos, disponer de una terminal de comandos del sistema y, por supuesto, ejecutar instrucciones de modo interactivo sobre un cuaderno.

Un cuaderno se estructura en celdas, bloques compactos que se ejecutan a la vez. Hay tres tipos de celdas: *code*, *markdown* y *raw*. El tipo *code* contiene una serie de instrucciones en el lenguaje del núcleo que se esté usando (*Python* en este curso) que **se ejecutarán cuando se seleccione la celda en cuestión y se pulse el símbolo de la barra superior con forma de triángulo negro**. El tipo *markdown* sirve para escribir texto formateado con unas marcas de formato muy similares a las del html empleado en la escritura de páginas web; si se ejecuta una celda de tipo *markdown*, el texto en cuestión quedará con un formato similar al que se mostraría en una página web. El tipo *raw* sirve para exportar un fichero fuente de *Jupyter* a otros formatos y no se usará en este curso.

3.1- Empleo de *Jupyter*.

Existen tres formas posibles de acceder a un entorno *Jupyter-lab.* En todos los casos, la sesión se desarrolla a través de un navegador web.

1.- Ejecutándolo en modo local desde una computadora. Si no está activo el navegador por defecto del sistema, se lanzará una sesión del mismo con una ventana que sirva de interfaz a *Jupyterlab.* Si ya estaba activo, simplemente se abrirá una pestaña nueva como interfaz.

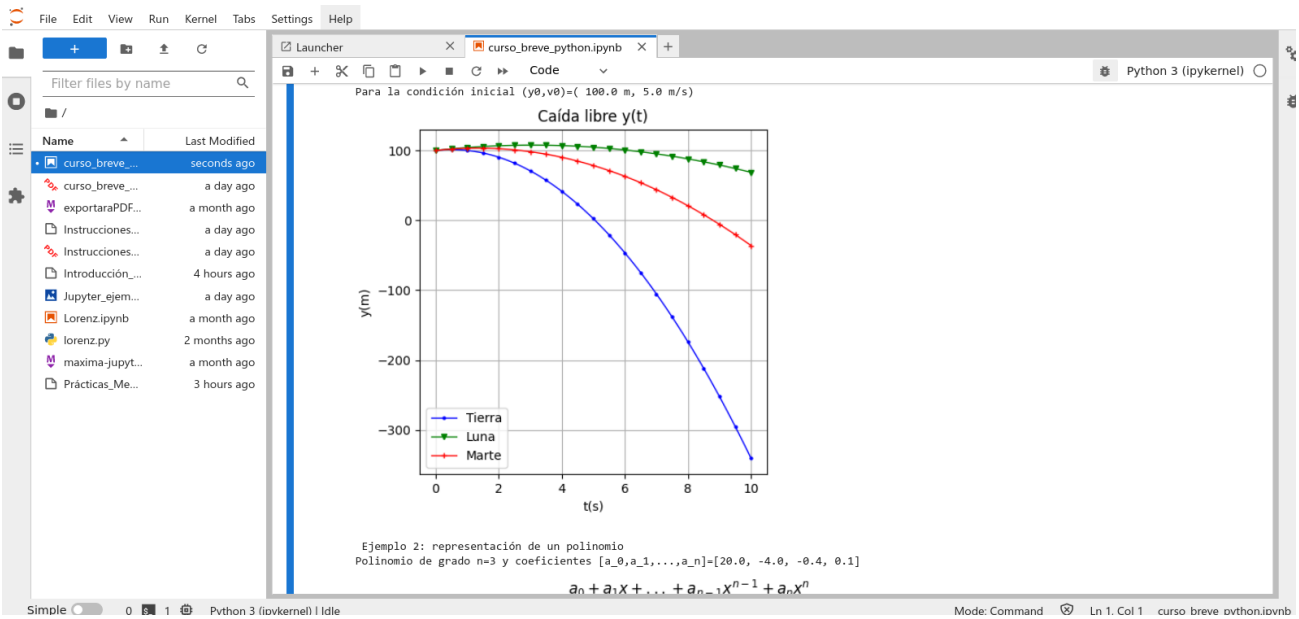

Figura 2: Sesión de *Jupyterlab* local.

2.- Ejecutándolo en línea a través de algún servidor que ofrezca esa posibilidad. Por ejemplo, accediendo a http://jupyter.unizar.es/ mediante el NIP y la contraseña administrativa que se haya recibido por ser alumno de la "Universidad de Zaragoza".

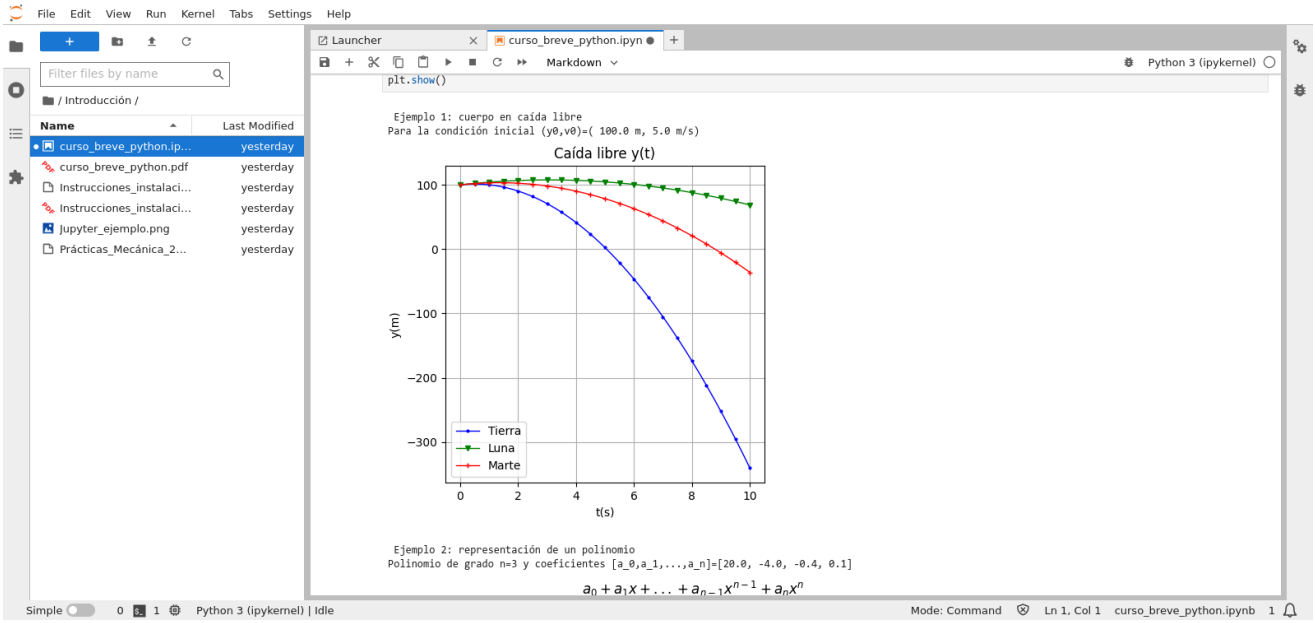

Figura 3: Sesión de *Jupyterlab* en línea en el servidor de la Universidad de Zaragoza.

3.- Ejecutándolo en línea a través de un disco de google. Si se dispone de una cuenta de google con un disco (drive) disponible (todos los miembros de la "Universidad de Zaragoza" disponen de una, consulte en caso necesario con el serivicio informático); cada vez que se intente abrir un archivo con la extensión .ipynb (un cuaderno de *Jupyter*) se lanza de modo inmediato la aplicación *colab* que permite la ejecución de dicho cuaderno.

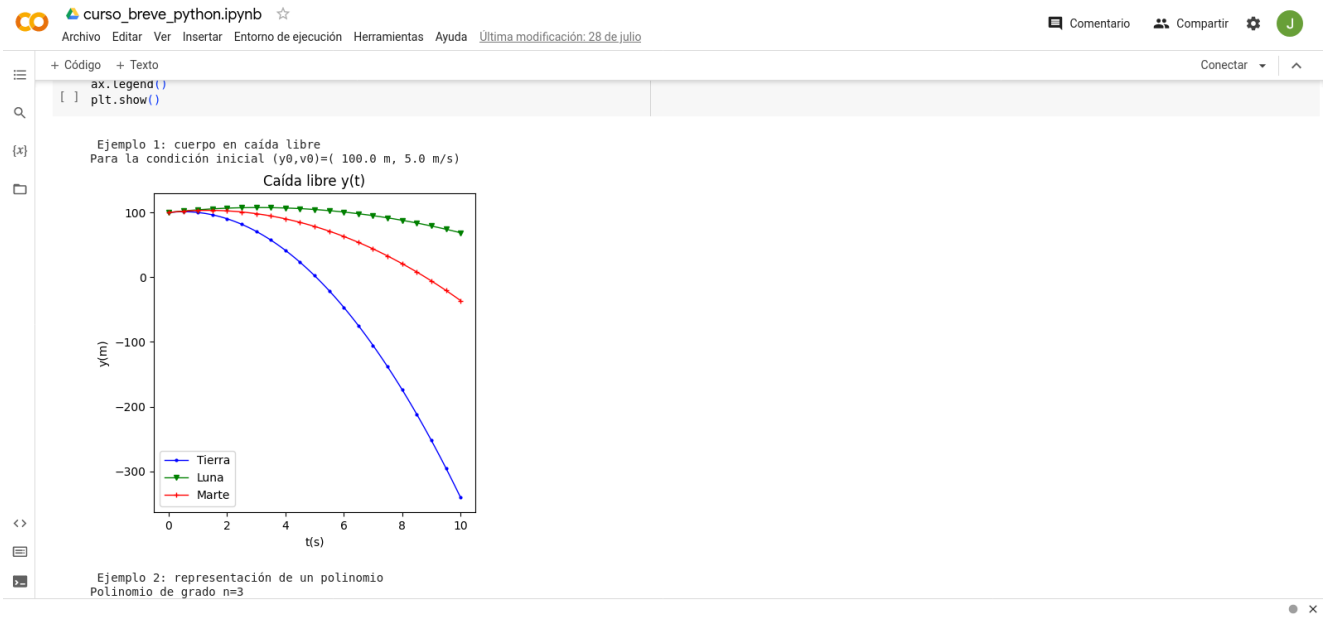

Figura 4: Sesión de *jupyterlab* en línea a través de la aplicación *Colab.*

La ejecución en modo local requiere que se instalen una serie de paquetes informáticos de acuerdo a las instrucciones que se pueden consultar en el **Anexo A**. La ejecución en línea sólo requiere de la presencia de un navegador web. La principal ventaja de la ejecución local es la posibilidad de trabajar aunque no se disponga de una conexión a internet y de que el procedimiento sea más rápido y fluido en situaciones en las que haya peligro de saturación de la red por un exceso de usuarios simultáneos; pero tiene el inconveniente de la necesidad de que el usuario descargue e instales una serie de paquetes de software.

Comparando las figuras 2, 3 y 4; se puede ver que tanto la ejecución local como la ejecución a través del servidor de la "Universidad de Zaragoza" tienen, prácticamente, el mismo aspecto. La principal diferencia se encuentra en la pestaña de nombre *Launcher* (cuyos contenidos no aparecen en las figuras) donde aparecen todos los kernels (lenguajes de programación que se pueden ejecutar) que el sistema tiene instalados. En el caso del servidor universitario, además de *Python;* también se puede trabajar con *gnuplot, maxima, octave, R, SageMath* y *VisualStudio CodeServer.* En el caso de una instalación local, sería el propio usuario quien debería instalar los paquetes de software necesarios para disponer de kernels adicionales, si así lo necesitara. La ejecución a través de *Colab* sólo permite trabajar con *Python* y de allí que no aparezca la pestaña *Launcher* (no tiene sentido ya que sólo hay un kernel posible); además, debido a esto, cambia el número de *Menús* disponibles en la parte superior de la aplicación y éstos han sido traducidos al español.

En todo caso, se puede trabajar con los cuadernos con cualquiera de estas tres opciones con resultados muy similares.

4. Curso de *Python*.

Como parte del Proyecto de Innovación Docente por el que se han modificado las prácticas de "Mecánica", también se ha creado una pequeña introducción al lenguaje de programación *Python* con el fin de que los alumnos sean capaces de entender y manejar los códigos que se emplearán en las propias prácticas.

El primer paso para realizar este curso de carácter introductorio será descargar el archivo *curso_breve_python.ipynb* o bien desde el curso virtual en *Moodle* o bien desde el enlace https://github.com/universidad-zaragoza/IOI-CUD-Mec , a través de la carpeta *0-Introducción.*

Si se ha realizado una instalación local de *Jupyterlab*, habrá que guardar el archivo en la carpeta en la que el enlace de acceso directo a *jupyter-lab* busca, por defecto, los cuadernos que cargará. Se puede controlar la localización de esta carpeta a través de las "Propiedades" del enlace directo (que se muestran al pulsar con el botón derecho[1] del ratón sobre el mismo) y, dentro de éstas, seleccionando "Acceso directo" e "Iniciar en:". Una vez que se esté seguro de que el archivo se encuentra en la carpeta en la que *Jupyterlab* lo buscará, se puede lanzar la aplicación pulsando con el botón izquierdo del ratón sobre el enlace directo lo que creará una pestaña en un Navegador web que contendrá en la columna de la izquierda un listado con todos los archivos accesibles por parte de *Jupyterlab* y en el resto del espacio una pestaña de *Launcher* con los kernels disponibles. Si todo ha ido bien, en la la columna de la izquierda se podrá ver el archivo *curso_breve_python.ipynb* y pulsando con el botón izquierdo ratón sobre el mismo, sus contenidos se abrirán en una nueva subpestaña, disponibles para su edición y ejecución.

Una vez abierto el cuaderno conviene tener en cuenta las siguientes consideraciones: las líneas que comienzan por '#' están comentadas y no se ejecutan. Para comentar o descomentar un conjunto de líneas (sin necesidad de ir una por una, añadiendo o quitando el carácter '#') se pueden seleccionar las líneas en cuestión con el ratón y, a continuación pulsar simultáneamente las teclas 'control' + '/'. Esta última tecla, '/', se pulsa en la zona numérica del teclado a la derecha del mismo.

Lea con cuidado cada celda de texto (*markdown*) y, a continuación, la de código (*code*) que le sigue, ejecute la celda de código (triángulo negro en la parte superior de la subpestaña donde se haya abierto el cuaderno) y compruebe que el resultado se corresponde con las explicaciones previas y el código escrito.

Puede jugar con el cuaderno y hacer las pruebas que estime oportunas para afianzar los conceptos. Si acaba llegando a una situación en la que se produce un error que no sabe como corregir, siempre puede volver a descargarse la versión original.

Si *Python* se queda colgado, puede intentar los siguientes pasos:

1. - Interrumpir el kernel (cuadrado negro en la parte superior de la subpestaña que contiene el cuaderno) y volver a darle a ejecutar (triángulo negro).

2.- Si el método anterior no funciona, reiniciar el kernel (símbolo con forma de giro horario, levógiro, en la parte superior de la subpestaña) y volver al inicio del cuaderno e ir ejecutando desde la primera celda, una por una.

1 Si el ratón está configurado para su uso por un zurdo, hay que modificar cualquier referencia que haya en el texto al botón izquierdo o derecho del mismo por el botón derecho o izquierdo, respectivamente. Sería más apropiado hablar de botón controlado por el dedo índice (izquierdo para diestros y derecho para zurdos) o por el dedo corazón (derecho para diestros e izquierdo para zurdos).

3.- Si no funcionan los dos métodos anteriores, salir de *Jupyterlab* (en el Menú de la pestaña principal, seleccionar *File* y allí dentro *Shutdown*) y volver a ejecutar desde el inicio (a través del enlace directo) el programa *jupyter-lab*.

ANEXO A. INSTALACIÓN LOCAL DE *JUPYTERLAB*.

En Windows.

1) Instalar la última versión de python: https://ww.python.org/downloads
- Seleccionar la “latest version for windows”. A fecha 10-VII-2023 es la 3.11.4. Se descargaría desde el enlace “Download Python 3.11.4”
- Se ejecuta el .exe descargado.

2) Instalar ffmpeg: https://github.com/BtbN/FFmpeg-Builds/releases
- Seleccionar “ffmpeg-master-latest-win64-gpl.zip”
- Extraer los contenidos de ese fichero .zip que irán a una carpeta que se llamará: “ffmpeg-master-latest-win64-gpl”
- Renombrar esa carpeta como “ffmpeg”
- ATENCIÓN PUEDE SURGIR UNA CARPETA “ffmpeg-master-latest-win64-gpl” dentro de otra carpeta con el mismo nombre al descomprimir. La que interesa es la más interna de las dos, dentro de la cual hay una carpeta “bin”.
- Desplazar la carpeta “ffmpeg” a la raíz del disco C:
- Ejecutar la línea de comandos, cmd, como administrador.
- Añadir una variable de entorno para ffmpeg con el comando:
C:\Windows\system32> setx /m PATH “C:\ffmpeg\bin;%PATH%”
- Salir de la línea de comandos y reiniciar el ordenador.
- Tras el reinicio, desde una línea de comandos como usuario, el comando:
C:> ffmpeg -version
debería dar información sobre el paquete instalado.

3) Instalar paquetes de python
Abrir una línea de comandos en windows y ejecutar:
C:> py -m pip install numpy
C:> py -m pip install sympy
C:> py -m pip install matplotlib
C:> py -m pip install ffmpeg-python

3) Instalar jupyterlab.
Desde la línea de comandos en windows, ejecutar:
C:> py -m pip install jupyterlab

En Apple.

1) Hay que instalar primero Python3 si no lo está ya, siguiendo las instrucciones que se pueden consultar en

https://programwithus.com/learn/python/install-python3-mac

Con la salvedad de que están pensadas para la versión 3.6 y ahora se dispone de un versión más moderna (numeración distinta, pero empezando por 3.).

2) Instalar *pip* siguiendo las instrucciones que se pueden encontrar en

https://phoenixnap.com/kb/install-pip-mac

es importante que sea la versión 3 de pip, correspondiente a Python3.

3) Instalar ffmpeg

Descargar el paquete correspondiente a Apple desde

http://ffmpeg.org/download.html

e instalarlo como cualquier otro paquete.

4) Instalar paquetes de python

Desde la línea de comandos ejecutar:

pip3 install numpy

pip3 install matplotlib

pip3 install ffmpeg-python

pip3 instsll jupyter-lab

En Linux.

1) A través del gestor de paquetes de la distribución que se tenga instalar python3, python-pip (o python3-pip, los nombres cambian según las distribuciones), ffmpeg, numpy, matplotlib, ffmpeg-python (en algunas distribuciones se llama ffmpeg-python-ffmpeg, en otras python3-av). Todo este software se instalará globalmente para su uso por todos los usuarios del sistema y la instalación se debe hacer con privilegios de administrador.

2) Instalar paquetes de python para su uso por el usuario que realiza la instalación.

python3 -m pip install --user jupyterlab

Esta instalación sólo es válida para un usuario y no requiere de privilegios de administrador.