

Reducción de datos CCD con IRAF y Python

Alexis Andrés

15 de agosto de 2024

Índice general

1. Introducción a Python	2
1.1. Nociones básicas	2
1.2. Python en diferentes sistemas operativos	2
1.2.1. Python en Windows	2
1.2.2. Python en Linux	3
1.2.3. Python en Mac OSX	3
1.3. Ejecutando Python	4
1.3.1. Creando un entorno virtual	5
1.3.2. El editor Geany	5
1.3.3. Jupyter Notebook	7
1.3.4. Visual Studio Code y PyCharm	8
1.4. Python como calculadora	9
1.4.1. Operaciones aritméticas	9
1.4.2. Operaciones matemáticas avanzadas	10

Clase 1 | Introducción a Python

1.1. Nociones básicas

Python es un lenguaje de programación interpretado de alto nivel cuyas características permiten su uso en diversas disciplinas. Los lenguajes de alto nivel en programación poseen una sintaxis que se asemeja al lenguaje humano, lo que facilita la claridad y legibilidad del código, haciendo su escritura y comprensión más accesibles. En contraste, los lenguajes de bajo nivel, como el *ensamblador*, utilizan una sintaxis más cercana al lenguaje máquina, lo que los hace más eficientes para la computadora pero más complejos de leer y escribir para los humanos.

Python, al ser un lenguaje interpretado, traduce y ejecuta el código línea por línea durante la ejecución, lo que puede resultar en tiempos de ejecución más largos en comparación con los lenguajes compilados, que se traducen en código máquina antes de ejecutarse. Sin embargo, esta diferencia en el tiempo de ejecución suele ser insignificante en muchos casos prácticos. Además, el tiempo requerido para desarrollar y mantener código en Python es considerablemente menor comparado con los lenguajes de bajo nivel, equilibrando así el tiempo total de desarrollo.

Adicionalmente, Python permite la integración de bibliotecas y códigos escritos en otros lenguajes de programación, aprovechando sus ventajas y capacidades para mejorar el rendimiento. Esta flexibilidad y facilidad de uso han contribuido a que Python se convierta en uno de los lenguajes de programación más populares en la actualidad, especialmente en los campos de ciencias e ingeniería.

En esta primera clase revisaremos algunos conceptos básicos de Python, cómo instalarlo en cualquier sistema operativo y además elegiremos un entorno de desarrollo integrado para escribir nuestros códigos Python.

1.2. Python en diferentes sistemas operativos

Python es un lenguaje de programación multiplataforma, esto significa que puede utilizarse en todos los sistemas operativos principales. Sin embargo, los métodos de instalación difieren dependiendo de cada sistema operativo. El objetivo de esta sección es que puedas ejecutar el famoso programa «¡Hola Mundo!» usando Python en tu sistema operativo. Ten en cuenta que trabajaremos exclusivamente con Python 3.

1.2.1. Python en Windows

Por lo general, Windows no cuenta con una versión de Python por defecto, por lo que será necesario instalarlo además de un editor de texto. Para verificar si tu versión de Windows

cuenta con Python, primero debes abrir el menú de inicio, escribir los caracteres **cmd** y hacer clic sobre la aplicación llamada *símbolo del sistema*. Esto abrirá una terminal de comandos en la que deberás escribir las palabras **python --version** (todo en minúsculas) y presiona la tecla Enter.

Si el resultado es algo similar a **3.x.x**, entonces Python ya está instalado en tu sistema. Sin embargo, si el resultado es un mensaje que dice que **python** no es un comando reconocido, entonces sigue las siguientes instrucciones para instalarlo.

1. **Descarga el instalador de Python.** Abre tu navegador y dirígete a la página <https://www.python.org/downloads/>. Haz clic en el botón que dice «Download Python 3.x.x» (donde «3.x.x» es la versión más reciente).
2. **Ejecuta el instalador.** Encuentra el archivo del instalador que acabas de descargar (normalmente estará en tu carpeta de descargas) y haz doble clic para ejecutarlo. En la primera ventana del instalador, asegúrate de marcar la casilla que dice «Add Python 3.x to PATH». Esto es importante para que puedas usar Python desde terminal de comandos sin problemas. Finalmente, haz clic en «Install Now» para comenzar la instalación.
3. **Verifica la instalación.** Una vez que finalice la instalación, abre una nueva terminal de comandos y escribe las palabras **python --version**. Esto debería mostrar la versión de Python que has instalado.

1.2.2. Python en Linux

Cualquier sistema operativo Linux cuenta con una versión de Python instalada por defecto. Por lo tanto, solo necesitas verificar cuál versión tienes instalada en tu sistema. Para lograrlo, abre una terminal de comandos con la combinación de teclas **Ctrl+Alt+T** y escribe las palabras **python --version** (todo en minúsculas) y presiona la tecla Enter. Si el resultado es un mensaje similar a **2.x.x** o si en cambio te aparece un mensaje que dice que el comando **python** no fue encontrado, entonces intenta escribiendo **python3 --version** y presiona Enter. El resultado debería ser algo como **3.x.x**.

1.2.3. Python en Mac OSX

La mayoría de los sistemas Mac OSX cuenta con al menos una versión de Python instalada por defecto. Esto significa que es posible que tengas con dos versiones, para verificarlo, abre una terminal de comandos haciendo clic en **Aplicaciones > Utilidades > Terminal**. Ahora escribe las palabras **python --version** (todo en minúsculas) y presiona la tecla Enter.

Al igual que con los sistemas Linux, es posible que el resultado sea un error o un mensaje con algo similar a **2.x.x**, lo que significa que la versión instalada es Python 2. Si cualquiera de las dos opciones anteriores es cierta, intenta escribiendo en la terminal **python3 --version** y presiona Enter. Si el resultado es algo similar a **3.x.x**, entonces cuentas con una versión de Python 3.

Si por algún motivo al escribir **python3 --version** obtienes un error, entonces necesitas instalar Python 3 en tu sistema. Para eso, sigue los siguientes pasos.

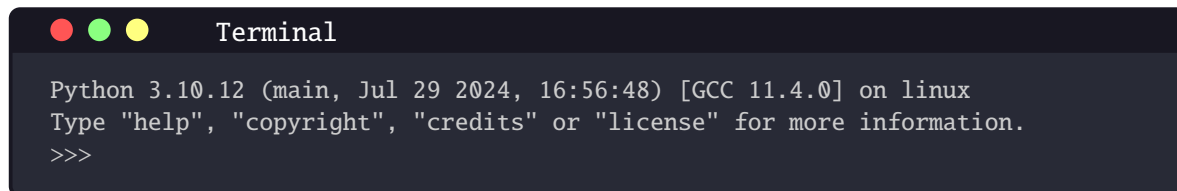
1. **Descarga el instalador.** Dirígete a la página <https://www.python.org/>. Coloca el cursor sobre el botón «Downloads» y selecciona «macOS». En la página de descargas

deberías ver un botón llamado **macOS 64-bit universal2 installer** debajo del título «Stable Releases». Haz clic sobre ese botón para descargar el instalador con extensión **.pkg**

2. **Ejecuta el instalador.** Una vez que el archivo **pkg** se haya descargado, haz doble clic en él para iniciar el proceso de instalación. Aparecerá una ventana del asistente de instalación. Sigue las instrucciones en pantalla. Generalmente, solo necesitas hacer clic en «Continuar» y luego en «Instalar». El instalador pedirá tu contraseña de administrador para proceder. Una vez que se complete el proceso, haz clic en «Cerrar» para terminar.
3. **Verifica la instalación.** Abre una nueva terminal de comandos y escribe las palabras **python3 --version**. El resultado debería ser la versión de Python que acabas de instalar.

1.3. Ejecutando Python

Para comenzar a usar Python, solo tienes que abrir una terminal de comandos, escribir la palabra **python3** y presionar la tecla Enter (Nota: este documento fue escrito en un sistema Linux, si en cambio estás usando Windows, para ejecutar Python solo debes escribir **python** en la terminal de comandos, sin agregar el número 3. Ten esto en cuenta en todo el documento a partir de este momento). El resultado debe ser algo similar a lo siguiente:



```
Python 3.10.12 (main, Jul 29 2024, 16:56:48) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Este mensaje nos muestra información sobre la versión de Python instalada, el sistema operativo en el que se instaló e información general. Inmediatamente después, aparece una línea con el símbolo **>>>**, el cual es conocido como el «Intérprete de Python» (en inglés: *Python prompt*). El intérprete de Python nos indica que podemos ingresar nuestros comandos desde ahí y serán ejecutados al presionar la tecla Enter.

Para nuestro primer ejemplo usaremos la función **print** definida por defecto en Python. Escribe el comando **print("¡Hola mundo!")** en tu sesión de Python y presionamos Enter. El resultado debería ser el siguiente:



```
>>> print("¡Hola mundo!")
¡Hola mundo!
>>>
```

Como puedes ver, la instrucción que se utilizó para mostrar el mensaje «¡Hola mundo!» es muy similar al lenguaje humano. Además, en la línea siguiente aparece de nuevo el intérprete de Python. Esto nos indica que Python está listo para seguir recibiendo instrucciones.

Para salir de Python y volver a una terminal de comandos normal, debes utilizar la función **exit** también definida por defecto. Específicamente, debes escribir el comando **exit()** y presionar Enter.

Esta es la forma más simple de ejecutar comandos de Python. Afortunadamente, no es la única. Para los propósitos del curso, necesitaremos de un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés).

1.3.1. Creando un entorno virtual

Antes de continuar e independientemente de tu sistema operativo, es conveniente crear un entorno virtual de Python para evitar cualquier problema de compatibilidad con el sistema operativo y que las cosas funcionen correctamente. Primero debes abrir una nueva terminal de comandos e instalar el paquete **virtualenv**, ejecutando la instrucción:

```
pip install virtualenv
```

Cuando termine, escribe la siguiente instrucción para crear un entorno virtual llamado «**omm**» (puedes cambiar la palabra **omm** a cualquier otro nombre que prefieras, pero debe ser una única palabra, sin espacios):

```
python3 -m venv omm
```

El comando anterior creó un directorio llamado «**omm**» (o el nombre que hayas elegido) en el directorio en el que abriste la terminal de comandos. Por defecto, la terminal de comandos se abre en el directorio «C:\Users\user» en Windows, y en el directorio «/home/user/» en Linux/MacOSX (donde **user** es tu nombre de usuario en tu computadora, independientemente de tu sistema operativo).

Para activar el entorno virtual en Windows, debes ejecutar el siguiente comando:

```
C:\Users\user\omm\Scripts\activate
```

Ten en cuenta que debes cambiar la palabra **user** por tu nombre de usuario.

En cambio, para activarlo en Linux y Mac OSX, se hace con el comando:

```
source ~/omm/bin/activate
```

Como resultado, verás que a la izquierda de tu terminal de comandos aparece el nombre de tu entorno virtual encerrado en paréntesis, esto indica que está activado. Deberás ejecutar este comando para utilizar el entorno virtual de Python cada vez que abras una nueva terminal. Para desactivarlo, debes escribir el comando **deactivate** y ejecutarlo.

1.3.2. El editor Geany

Un programa de Python es simplemente un archivo de texto con extensión **.py** que puede escribirse en cualquier editor de texto. Existen muchos editores de texto y puedes usar el que prefieras. Recomendando usar Geany porque es bastante simple, es ligero y fácil de instalar. Además permite ejecutar los programas directamente desde el editor, también utiliza el resaltado de sintaxis y permite usar una terminal integrada para ejecutar los códigos si así lo prefieres.

Geany en Windows

Para instalar Geany en Windows, dirígete a la página <https://www.geany.org/download/releases/> y descarga el instalador para Windows. Una vez que se haya descargado el archivo `.exe`, haz doble clic en él para iniciar el proceso de instalación. Sigue las instrucciones del asistente de instalación. Puedes dejar las opciones predeterminadas, a menos que desees personalizar la instalación.

Configuración de Geany en Windows

Una vez instalado, abre Geany desde el menú de aplicaciones. En la ventana del editor, escribe la instrucción `print("¡Hola mundo!")` y utiliza la combinación de teclas **Ctrl+S** para guardar el archivo. Puedes guardar el archivo con el nombre `hello_world.py` en tu carpeta de trabajo. Asegúrate de colocar la extensión `.py` en el nombre de tu archivo al momento de guardarlo.

Para que Geany funcione correctamente con el entorno virtual, se debe hacer una configuración sencilla. Dentro de la ventana de Geany, dirígete a **Build -> Set Build Commands**. Ahora haz lo siguiente:

1. En el campo llamado «**Compile**», edita su contenido para que el comando tenga lo siguiente: `C:\Users\user\omm\Scripts\python -m py_compile "%f"`
2. En el campo llamado «**Execute**», edita su contenido para que el comando tenga lo siguiente: `C:\Users\user\omm\Scripts\python "%f"`
3. Haz clic en OK para guardar los cambios.

Usa las opciones de **Build** o los atajos de teclado (por ejemplo, **F8** para compilar y **F5** para ejecutar) para compilar y ejecutar el archivo Python. Verifica que el intérprete de Python utilizado sea el del entorno virtual y no el global.

El resultado debería ser un mensaje que dice «¡Hola mundo!», tal como cuando se ejecutó el comando desde una terminal.

Geany en Linux y Mac OSX

Para instalar Geany en Linux, es suficiente con que ejecutes los siguientes comandos en una terminal:

```
sudo apt update
sudo apt install geany
```

En Mac OSX, puedes usar **Homebrew** para instalar Geany desde una terminal con el comando:

```
brew install --cask geany
```

La configuración de Geany en Linux y Mac OSX es la misma. Para hacerlo, sigue los pasos descritos en la sección «Configuración de Geany en Windows» con la única diferencia que el campo «**Compile**» debe contener lo siguiente: `/home/user/omm/bin/python -m py_compile "%f"` y el campo «**Execute**» debe tener: `/home/user/omm/bin/python "%f"`.

Ten en cuenta que tanto en Windows como en Linux y Mac OSX, debes reemplazar la palabra «**user**» por tu nombre de usuario. Utilizaremos Geany principalmente para escribir y editar algunas funciones que utilizaremos. Ahora revisaremos algunos IDEs un poco más avanzados y que permiten hacer más cosas.

1.3.3. Jupyter Notebook

Jupyter Notebook es un IDE de Python basado en navegadores web, que además de ejecutar códigos de Python, permite incluir texto con formato, ecuaciones, imágenes y mucho más. Esencialmente, es un cuaderno donde es posible escribir código y tomar apuntes. Para instalarlo debes abrir una nueva terminal, activar el entorno virtual que creaste anteriormente y ejecutar la instrucción **pip install notebook**. En una terminal se vería así:

```
source ~/omm/bin/activate
pip install notebook
```

Recuerda utilizar los comandos adecuados si usas Windows. Una vez que termine la instalación, puedes usar Jupyter Notebook escribiendo la instrucción **jupyter notebook** (separado y en minúsculas) en la misma terminal de comandos. Esto abrirá una nueva pestaña en tu navegador predeterminado y te mostrará un explorador de archivos con el contenido de la carpeta donde ejecutaste la instrucción anterior, como se muestra en la Figura 1.1

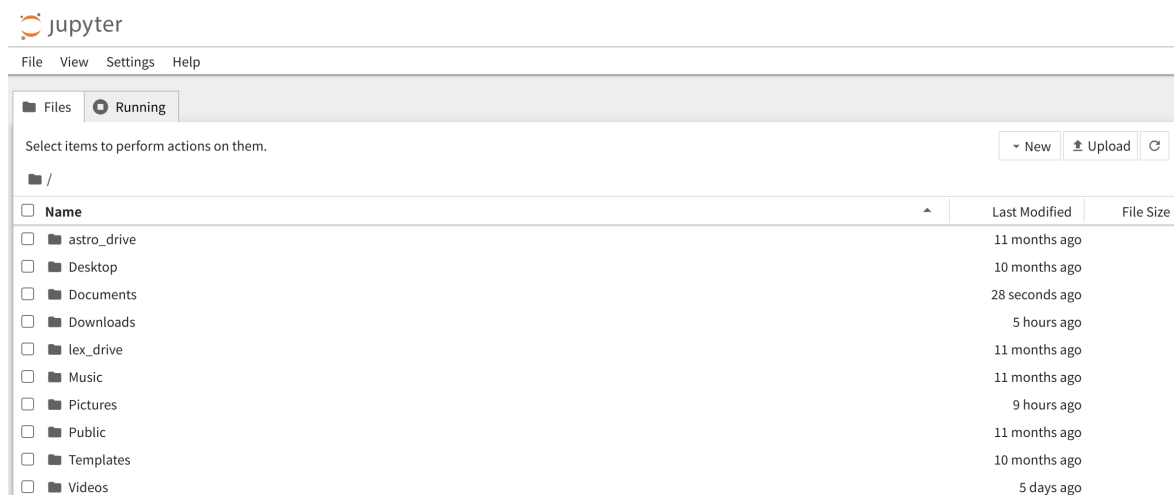


Figura 1.1: Interfaz gráfica de Jupyter Notebook

Desde la interfaz de Jupyter dirígete a tu carpeta de trabajo y crea un nuevo Notebook. Para lograrlo, haz click en el botón «**New**» en la esquina superior derecha y selecciona la opción «**Notebook**», luego elige el *kernel* llamado «**Python 3 (ipykernel)**» y esto abrirá una nueva pestaña en tu navegador.

En esa nueva pestaña, verás que aparecen los caracteres `[]:` (encerrado con un círculo rojo en el panel izquierdo de la Figura 1.2) y a la derecha aparece una celda vacía donde puedes escribir los comandos de Python. Los caracteres `[]:` en Jupyter Notebook son el equivalente a los caracteres `>>>` en la terminal.

En la parte superior del Notebook hay una barra de menú, que a su vez contiene un botón con la palabra «**Code**» (encerrada con un rectángulo rojo en el panel izquierdo de la Figura

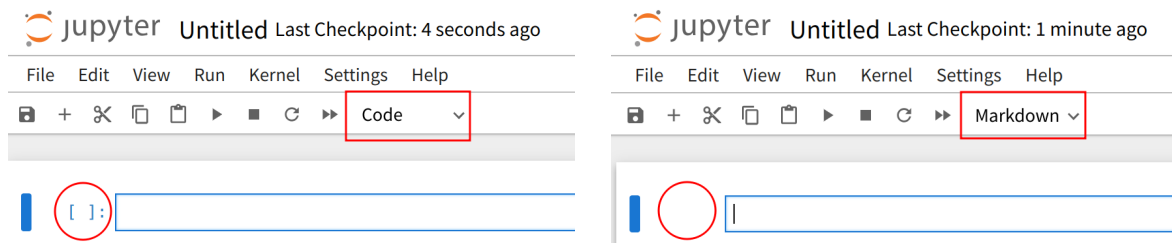


Figura 1.2: Opciones disponibles para escribir en Jupyter

1.2). Al darle clic se muestra un menú desplegable cuyas opciones son «Code» y «Markdown». Este último es un tipo de *lenguaje de marcas* bastante popular para escribir texto con formato. Si seleccionas «Markdown», los símbolos `[]:` de la celda desaparecen (como se muestra en el panel derecho de la Figura 1.2) y ahora puedes escribir texto, ecuaciones, imágenes y más. Puedes revisar algunas nociones básicas sobre la escritura de texto Markdown en este enlace de [Github Docs](#).

Cuando vas a escribir código, siempre debes verificar que la celda está en modo «Code» o en su defecto, que a la izquierda de la celda aparezcan los símbolos `[]:`, de lo contrario, tus comandos no se ejecutarán. Por ejemplo, da clic en la celda vacía y asegúrate que se encuentre en modo «Code» y escribe el comando `print("¡Hola mundo!")`. Para ejecutar el código de una celda, debes dar click a la celda y usar la combinación de teclas **Shift+Enter**. El resultado debería ser algo similar a lo siguiente:

```
[1]: print("¡Hola mundo!")
¡Hola mundo!
```

Para cerrar correctamente Jupyter Notebook, dirige tu cursor hacia la esquina superior izquierda del Notebook y selecciona **File -> Shutdown**. Una vez que lo hagas, podrás cerrar esa pestaña del navegador. Debes repetir este procedimiento para todas las pestañas del Notebook abiertas. Como último dato sobre Jupyter Notebook, debes saber que aunque necesite de un navegador web para funcionar, en realidad no necesita de una conexión a internet para ejecutar los comandos. El navegador web es únicamente para la interfaz gráfica.

1.3.4. Visual Studio Code y PyCharm

Visual Studio Code (VS Code) y PyCharm son otros IDEs muy populares que también trabajan con Notebooks de Python y también archivos de texto plano con extensión `.py`. Su interfaz es similar a la de Jupyter Notebook y funcionan de la misma manera. Puedes revisar las instrucciones de instalación y configuración de VS Code y PyCharm en los sitios web oficiales <https://code.visualstudio.com/> y <https://www.jetbrains.com/es-es/pycharm/download/>, respectivamente.

Como última opción, si no te es posible instalar Python en tu computadora, puedes utilizar la opción en línea llamada Google Colaboratory (o simplemente Colab), a la que puedes acceder desde tu cuenta de google drive. La ventaja de usar Colab es que no necesitas instalar nada en tu computadora y puedes usar todos los paquetes de Python que requieras. Sin embargo, sí necesitarás de una conexión a internet para poder utilizarlo.

PyCharm, VS Code, Colab y Jupyter Notebook son totalmente equivalentes entre sí, teniendo diferencias mínimas y por lo tanto puedes utilizar el IDE que sea de tu preferencia.

Teniendo esto en cuenta, podemos iniciar con algunas nociones sobre la escritura de código Python usando Notebooks.

1.4. Python como calculadora

1.4.1. Operaciones aritméticas

Una de las formas elementales en las que Python puede ser utilizado es como una calculadora. En este sentido, la Tabla 1.1 muestra las operaciones aritméticas que pueden ejecutarse con Python. Quizá no estés muy familiarizado con las operaciones de división entera (representada con los caracteres `//`) y división modular (representada con el carácter `%`), pero las revisaremos con detalle más adelante.

Tabla 1.1: Operadores aritméticos	
Símbolo/Operador	Significado/Operación
<code>+</code>	Suma
<code>-</code>	Resta
<code>*</code>	Multiplicación
<code>/</code>	División
<code>**</code>	Exponenciación
<code>//</code>	División entera
<code>%</code>	División modular

Abre una nueva terminal de comandos, activa tu entorno virtual de Python e inicia una nueva sesión de Jupyter Notebook. Puedes abrir el mismo Notebook que creaste anteriormente o crear uno nuevo. Asegúrate que las celdas estén en modo «Code» e intenta ejecutar los siguientes comandos. El resultado de cada comando se mostrará al lado de los caracteres `[]:` (en color naranja).

Veamos cómo funcionan los operadores aritméticos con unos ejemplos sencillos. Comenzamos con una simple, para calcular el valor de la operación $5 + 9$:

```
[2]: 5 + 9
```

```
[2]: 14
```

La siguiente expresión es equivalente a $40 - 5 \times 4$:

```
[3]: 40 - 5*4
```

```
[3]: 20
```

Y también podemos agrupar términos utilizando los paréntesis. Por ejemplo, podemos calcular el valor de la expresión $(40 - 5 \times 4)/4$ usando la siguiente sintaxis:

```
[4]: (40 - 5 * 4) / 4
```

```
[4]: 5.0
```

Y para calcular el valor de $3^2 + 4^2$ se usa la sintaxis:

```
[5]: 4**2 + 3**2
```

```
[5]: 25
```

Ahora revisemos con un poco de detalle el operador `//`, que en Python es el operador de división entera. Este operador divide dos números y redondea el resultado hacia abajo al número entero más cercano. Por ejemplo, sabemos que $7/3 = 2.333$. Si aplicamos el operador de división entera a esos mismos números se obtiene:

```
[6]: 7 // 3
```

```
[6]: 2
```

En este caso, `7 // 3` es igual a 2 porque la parte fraccionaria (0.333...) se descarta. Ahora veamos cómo se comporta este operador con números negativos:

```
[7]: -7 // 3
```

```
[7]: -3
```

En este caso, `-7 // 3` es igual a -3 porque el resultado se redondea hacia abajo (es decir, al entero más negativo).

Por otro lado, el operador `%` en Python es el operador de división modular (también conocido como resto o residuo). Este operador devuelve el resto de la división entre dos números. Es útil para determinar si un número es divisible por otro, o para obtener la parte sobrante después de dividir un número por otro. Veamos cómo funciona con ejemplos:

```
[8]: 7 % 3
```

```
[8]: 1
```

En este caso, `7 % 3` es igual a 1 porque 7 dividido entre 3 es igual a 2 con un residuo de 1.

1.4.2. Operaciones matemáticas avanzadas

Para realizar operaciones matemáticas más avanzadas, necesitaremos utilizar un *módulo* de Python llamado **math**. Un módulo es un archivo que contiene definiciones y declaraciones de Python, como funciones, variables y clases. Los módulos permiten organizar y reutilizar código.

Para usar el módulo **math**, debemos importarlo en nuestro código. La forma más sencilla de hacerlo es usando la palabra clave «**import**». Dentro del módulo **math** están definidas muchas funciones matemáticas como la raíz cuadrada, logaritmos, funciones trigonométricas y mucho más. Para usarlas, debemos hacerlo con el operador de acceso, que en Python es un punto. La siguiente celda de código muestra cómo importar el módulo **math** y cómo acceder a la función raíz cuadrada, definida como **sqrt** dentro de dicho módulo:

```
[9]: import math
```

```
print(math.sqrt(16))
```

```
4.0
```

También podemos hacer importaciones específicas de las funciones dentro del módulo. Para esto usamos la palabra «**from**». El siguiente ejemplo muestra cómo importar la función **sqrt** definida dentro del módulo **math**:

```
[10]: from math import sqrt  
  
print(sqrt(16))  
  
4.0
```

En este último caso, no fue necesario escribir **math.sqrt** para calcular la raíz cuadrada, porque importamos directamente la función **sqrt**.

Adicionalmente, puedes importar módulos o funciones específicas y renombrarlas a tu gusto usando la palabra «**as**». Los siguientes dos ejemplos muestran cómo funciona esta sintaxis.

```
[11]: import math as m  
  
print(m.sqrt(16))  
  
4.0
```

```
[12]: from math import sqrt as raiz_cuadrada  
  
print(raiz_cuadrada(16))  
  
4.0
```

Como normal general, se recomienda hacer todas las importaciones al inicio del programa y utilizar alias claros y fácilmente identificables.