



## Python con Excel I Introducción

*Conceptos básicos de iniciación a Python en Excel. Porque de uso, principales librerías, familiarización con el editor, tipos de salidas, etc.*

Jose Ignacio González Gómez

Departamento de Economía, Contabilidad y Finanzas - Universidad de La Laguna

[www.jggomez.eu](http://www.jggomez.eu)

Ejercicio adaptado de: jggomez

**V. 2.5**

Archivos fuentes:

## Contenido

|                                                               |    |
|---------------------------------------------------------------|----|
| Presentacion .....                                            | 3  |
| Objetivo.....                                                 | 3  |
| ¿Qué es Python en Excel? .....                                | 3  |
| Porque Python en Excel .....                                  | 3  |
| Otras consideraciones.....                                    | 3  |
| Python se ejecuta en la nube .....                            | 3  |
| Copilot en Excel con Python.....                              | 4  |
| Principales librerías de Python .....                         | 4  |
| Librerías básicas.....                                        | 4  |
| Introducción .....                                            | 4  |
| Pandas. Manipulación y análisis de datos. ....                | 4  |
| Matplotlib. Visualización de datos con gráficos. ....         | 5  |
| NumPy. Operaciones matemáticas y manejo de arrays.....        | 5  |
| Seaborn. Gráficos estadísticos avanzados .....                | 6  |
| Statsmodels. Modelos estadísticos y econométricos .....       | 7  |
| Otras librerías en Anaconda .....                             | 8  |
| Importar librerías.....                                       | 8  |
| Librerías cargada por defecto e importación de ANACONDA ..... | 8  |
| Restricciones de las librerías con Python en Excel .....      | 9  |
| IA con Python en Excel.....                                   | 9  |
| Primeros pasos de Python en Excel.....                        | 9  |
| Empezar a usar Python.....                                    | 9  |
| Procesos básicos.....                                         | 9  |
| Carga de Librerías .....                                      | 9  |
| Definición de un DataFrame .....                              | 9  |
| Orden de cálculo y análisis de datos .....                    | 10 |
| Combinar Python con celdas y rangos de Excel .....            | 10 |
| Tipos de salidas .....                                        | 11 |

|                                                                            |    |
|----------------------------------------------------------------------------|----|
| Presentacion del editor de codigo Python del Excel .....                   | 12 |
| Entrar en modo Python.....                                                 | 12 |
| Acceso .....                                                               | 13 |
| Opciones de edición en el panel de tareas.....                             | 13 |
| Mostrar salida de Python en el panel de tareas .....                       | 13 |
| Salida de Python de un DataFrame. ....                                     | 14 |
| Tipos de datos como objetos de Python: DataFrames y tipos de salidas ..... | 14 |
| Orden de cálculo, recálculo y errores.....                                 | 16 |
| Orden de cálculo.....                                                      | 16 |
| Recálculo.....                                                             | 16 |
| Errores .....                                                              | 16 |
| Importar datos externos.....                                               | 17 |
| Introducción .....                                                         | 17 |
| Power Query (PQ) para importar datos para Python en Excel.....             | 17 |
| Casos de iniciación a Python con Excel. ....                               | 20 |
| Ventas de Coches .....                                                     | 20 |
| Presentacion. Configuración del DataFrame.....                             | 20 |
| Asignación de nombres a las variables.....                                 | 20 |
| Introducción al análisis de datos con Python .....                         | 20 |
| Usando la IA para obtener el codigo Python.....                            | 24 |

## Presentacion

### Objetivo

Este documento pretende ser una introducción a Python para Excel, en concreto trataremos cuestiones basicas como:

- Utilizar una biblioteca llamada Openpyxl para leer y escribir archivos Excel utilizando Python
- Crea operaciones aritméticas y fórmulas de Excel en Python
- Manipula hojas de cálculo de Excel con Python
- Construye visualizaciones en Python y guárdalas en un archivo Excel
- Modifica colores y estilos de las celdas de Excel con Python

### ¿Qué es Python en Excel?

Python en Excel aporta la potencia del análisis de Python en Excel. Úselo para procesar datos en Excel con código Python. Escribe Python directamente en una celda, los cálculos de Python se ejecutan en la nube de Microsoft y los resultados se devuelven a la hoja de cálculo.

### Porque Python en Excel

El 54 % de las empresas utilizan Excel para realizar operaciones aritméticas, analizar datos, crear visualizaciones y generar informes. Incluso es posible llevar a cabo tareas de modelado predictivo, como regresión y agrupamiento, directamente en Excel.

Sin embargo, a pesar de sus fortalezas, Excel presenta ciertas limitaciones que pueden dificultar su eficacia en tareas específicas, especialmente cuando se trabaja con grandes volúmenes de datos. Una de sus principales debilidades es su capacidad limitada para manejar conjuntos de datos extensos. Al realizar operaciones complejas con muchas entradas, es común experimentar problemas de rendimiento, especialmente si las fórmulas y macros no están optimizadas.

Además, Excel puede volverse ineficiente cuando se trata de tareas repetitivas. Por ejemplo, si cada semana necesitas realizar el mismo análisis en varios archivos, tendrías que abrirlos manualmente y copiar las fórmulas una y otra vez. De hecho, encuestas revelan que el 93 % de los usuarios de Excel consideran que consolidar hojas de cálculo consume demasiado tiempo, y los empleados dedican en promedio 12 horas al mes solo a combinar archivos.

Estas limitaciones pueden superarse mediante la automatización de flujos de trabajo con Python. Tareas como la consolidación de hojas, la limpieza de datos o el modelado predictivo pueden realizarse en minutos con un simple script de Python que interactúe con archivos de Excel. Además, es posible programar la ejecución automática de estos scripts en intervalos definidos, reduciendo significativamente la intervención manual.

Una de las grandes ventajas de Excel es su amplia adopción: desde usuarios sin conocimientos técnicos hasta empleados de nivel básico, todos están familiarizados con su interfaz. Por otro lado, Python es ideal para analizar grandes volúmenes de datos y construir modelos complejos, lo que permite automatizar procesos tediosos y mejorar la eficiencia organizacional.

La combinación de Excel y Python permite optimizar los flujos de trabajo empresariales, ahorrando horas de trabajo mientras se mantiene una interfaz accesible para todos los miembros de la organización.

### Otras consideraciones

#### Python se ejecuta en la nube

Los cálculos de Python en Excel se ejecutan en Microsoft Cloud con una versión estándar del lenguaje Python. Python en Excel viene con un conjunto principal de bibliotecas de Python proporcionadas por [Anaconda](#) a través de una distribución estándar y segura. Para obtener más información, consulte [Seguridad de datos y Python en Excel](#).

No necesita una versión local de Python para usar Python en Excel. Si tiene una versión local de Python instalada en el equipo, las personalizaciones que haya realizado en esa instalación de Python no se reflejarán en los cálculos de Python en Excel.

### Copilot en Excel con Python

Copilot en Excel con Python le permite utilizar AI (inteligencia artificial) y Python en Excel juntos, sin ser un experto de Python o Excel. Úselo para obtener información más profunda de sus datos aprovechando la potencia del análisis de Python con la ayuda de Copilot. Para obtener más información, consulte [Copilot en Excel con Python](#).

## Principales librerías de Python

Python en Excel viene con un conjunto principal de bibliotecas proporcionadas por Anaconda que puede ser usadas para simplificar el análisis de datos, buscar patrones y perspectivas ocultas, y visualizar los datos con trazados.

Mostramos a continuación una relación de las principales bibliotecas disponibles:

### Librerías básicas

#### Introducción

Por defecto, las bibliotecas que se importan cada vez que ejecutamos Python son las siguientes como vemos en la opción configuración.

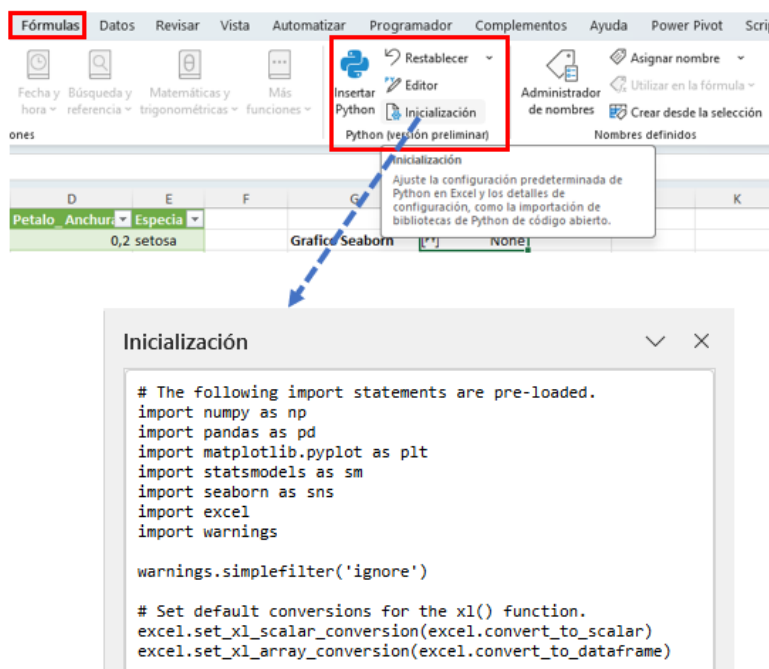


Ilustración 1

Las bibliotecas Matplotlib, NumPy, seaborn, statsmodels, Excel y Warning y pandas se importan de forma predeterminada cada vez que ejecutamos este complemento de Python en Excel. Veamos a continuación las principales características de cada una de ellas.

### Pandas. Manipulación y análisis de datos.

Esta es la principal biblioteca, ofrece muchas funcionalidades para analizar y comprender los datos. Entre sus características destacar las siguientes:

1. Estructuras de datos eficientes
  - DataFrame: tabla bidimensional con etiquetas en filas y columnas (similar a una hoja de Excel).

- Series: arreglo unidimensional con etiquetas (como una columna de Excel).
- 2. Lectura y escritura de datos. Soporta múltiples formatos: CSV, Excel, JSON, SQL, etc.
- 3. Manipulación de datos
  - Filtrado, ordenamiento, agrupamiento (groupby), combinación (merge, join), eliminación de duplicados, manejo de valores nulos.
  - Transformaciones con funciones como apply, map, replace.
- 4. Estadísticas y resumen. Métodos como mean(), sum(), describe(), value\_counts() para obtener estadísticas rápidas.
- Integración con otras librerías. Funciona muy bien con numpy, matplotlib, seaborn, scikit-learn, etc. Ideal para preparar datos antes de visualizarlos o modelarlos.
- Manejo de fechas y tiempos, Soporte robusto para series temporales, conversión de fechas, resampling, y operaciones con fechas.
- Alto rendimiento, optimizado para trabajar con grandes volúmenes de datos.

### Matplotlib. Visualización de datos con gráficos.

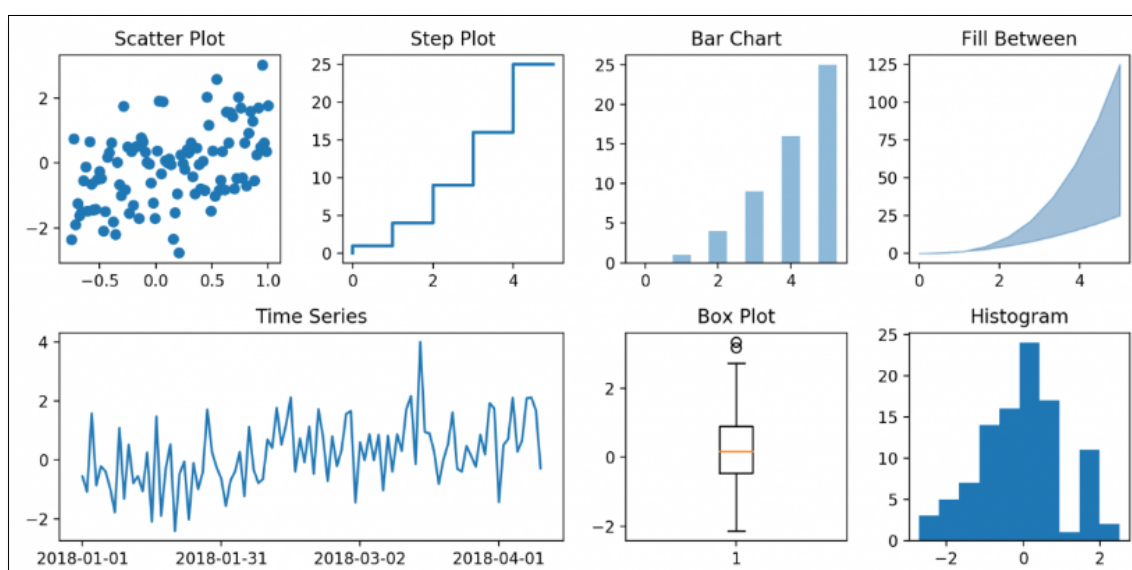


Ilustración 2

1. Amplia variedad de gráficos
  - Permite crear gráficos de líneas, barras, histogramas, dispersión (scatter), pastel (pie), áreas, mapas de calor, entre otros.
  - Ejemplo: plt.plot(), plt.bar(), plt.scatter(), plt.pie()
2. Control total sobre el diseño
  - Puedes personalizar cada aspecto del gráfico: colores, estilos de línea, etiquetas, leyendas, títulos, ejes, tamaños, fuentes, etc.
  - Ideal para presentaciones profesionales o publicaciones científicas.
3. Interfaz similar a MATLAB. Su sintaxis es familiar para quienes han trabajado con MATLAB, lo que facilita la transición.
4. Integración con otras librerías. Funciona muy bien con pandas y numpy, permitiendo graficar directamente desde estructuras como DataFrame o Series.

### NumPy. Operaciones matemáticas y manejo de arrays

La librería NumPy (abreviación de Numerical Python) es fundamental en el ecosistema de Python para el cálculo numérico y científico. Algunas de sus principales características son:

- Arreglos multidimensionales (ndarray)

- Estructura central que permite almacenar y manipular datos en múltiples dimensiones (1D, 2D, 3D, etc.).
- Funciones matemáticas avanzadas. Incluye funciones para álgebra lineal, estadísticas, transformadas de Fourier, funciones trigonométricas, exponenciales, logarítmicas, etc.
- Manejo eficiente de memoria. Usa estructuras compactas y optimizadas para almacenar datos, lo que permite trabajar con grandes volúmenes de información.
- Compatibilidad con otras librerías. Es la base para muchas otras librerías como pandas, scikit-learn, matplotlib, SciPy, etc.
- Generación de datos. Funciones para crear arreglos con valores aleatorios, secuencias numéricas (arange, linspace), matrices identidad, etc

### Seaborn. Gráficos estadísticos avanzados

La librería **Seaborn** (no "Seanborn", aunque es una confusión común) es una biblioteca de visualización de datos basada en matplotlib, diseñada para hacer gráficos estadísticos más atractivos y fáciles de crear. Entre sus características principales, destacar:

1. Gráficos estadísticos de alto nivel
  - Permite crear gráficos complejos con una sola línea de código.
  - Ejemplos: gráficos de dispersión con regresión (sns.regplot), histogramas (sns.histplot), mapas de calor (sns.heatmap), diagramas de violín (sns.violinplot), boxplots (sns.boxplot), etc.
2. Integración con pandas
  - Trabaja directamente con DataFrame, lo que facilita la visualización de datos tabulares.
  - Usa nombres de columnas como variables en los gráficos.
3. Estética mejorada por defecto
  - Los gráficos tienen un diseño más limpio y profesional que los de matplotlib por defecto.
  - Incluye estilos predefinidos como darkgrid, whitegrid, ticks, etc.
4. Automatización de análisis estadístico
  - Puede incluir automáticamente líneas de regresión, estimaciones de densidad, y otras funciones estadísticas en los gráficos.
5. Facilidad para comparar categorías
  - Ideal para visualizar relaciones entre variables categóricas y numéricas.
  - Ejemplo: comparar distribuciones por grupo con `sns.boxplot(x="grupo", y="valor", data=df)`
6. Paletas de colores personalizables
  - Ofrece muchas paletas de colores (deep, muted, bright, pastel, dark, colorblind) y permite crear tus propias combinaciones.
7. Facilidad para crear gráficos múltiples
  - Funciones como FacetGrid y pairplot permiten crear múltiples gráficos en una sola figura, segmentados por variables.

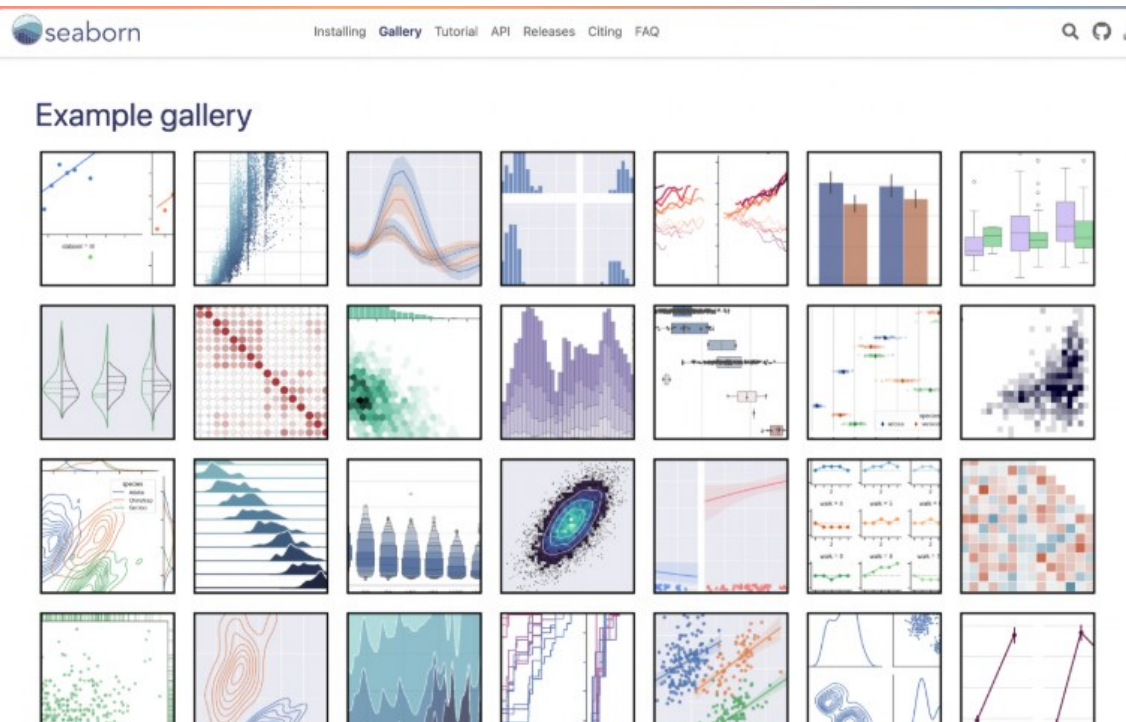


Ilustración 3

### Statsmodels. Modelos estadísticos y econométricos

La librería **statsmodels** es una herramienta poderosa en Python para realizar análisis estadísticos, especialmente útil en econometría, modelado estadístico y pruebas de hipótesis. Aquí tienes sus características principales:

#### 1. Modelos estadísticos clásicos

Soporta una amplia gama de modelos como:

- Regresión lineal (OLS)
- Regresión logística
- Modelos de series temporales (AR, ARMA, ARIMA, SARIMA)
- Modelos de efectos mixtos
- Modelos de elección discreta (probit, logit)
- Modelos de varianza constante y heterocedástica (GLS, WLS)

#### 2. Análisis de series temporales

- Herramientas para descomposición, predicción, y ajuste de modelos temporales.
- Compatible con fechas y frecuencias de pandas.

#### 3. Pruebas estadísticas

Pruebas de hipótesis como:

- t-test, ANOVA, chi-cuadrado
- Pruebas de normalidad, autocorrelación, heterocedasticidad
- Pruebas de cointegración y estacionariedad (ADF, KPSS)

#### 4. Estimación y resumen detallado

- Los modelos generan resúmenes completos con coeficientes, errores estándar, valores p,  $R^2$ , y más.
- Ejemplo: `model.summary()` muestra una tabla detallada del ajuste.

#### 5. Interfaz con fórmulas estilo R

- Usa fórmulas tipo  $y \sim x_1 + x_2$  para definir modelos, lo que facilita la especificación.
  - Ejemplo: `smf.ols('ventas ~ precio + publicidad', data=df)`
6. Visualización de resultados
- Incluye gráficos de diagnóstico como:
- Gráficos de residuos
  - QQ plots
  - Gráficos de influencia y leverage
7. Integración con pandas y numpy
- Trabaja directamente con DataFrame y Series, lo que facilita el análisis de datos tabulares.

## Otras librerías en Anaconda

Existen otras librerías adicionales para importar manualmente disponibles en el entorno de Anaconda y puedes importarla según tus necesidades:

- Astropy. Una biblioteca de Python para astronomía.
- BeautifulSoup4. Una biblioteca de Python diseñada para screen scraping.
- Faker. Genera datos falsos.
- NLTK – Procesamiento de lenguaje natural.
- plotnine – Visualización estilo ggplot.
- PyTables, PyWavelets, Prince, mlxtend, entre muchas otras
- scikit-learn – Aprendizaje automático.
- SciPy – Cálculos científicos.
- SymPy – Matemáticas simbólicas.
- tabulate – Formateo de tablas.
- wordcloud – Generación de nubes de palabras.

## Importar librerías

### Librerías cargada por defecto e importación de ANACONDA

Si quisiéramos cargar librerías o confirmar librerías cargadas tendríamos las siguientes opciones mostradas en la Ilustración 4.

Además de las bibliotecas principales, puede importar otras bibliotecas disponibles a través de Anaconda. Importar bibliotecas de Python en Excel mediante una instrucción de importación de Python en una celda de Python en Excel, como **import numpy as np**. Esta instrucción importa la biblioteca NumPy y le asigna el alias **np**. Después de escribir esta instrucción import en una celda de Python, puede hacer referencia a la biblioteca NumPy como **np** en las fórmulas de Python de ese libro.

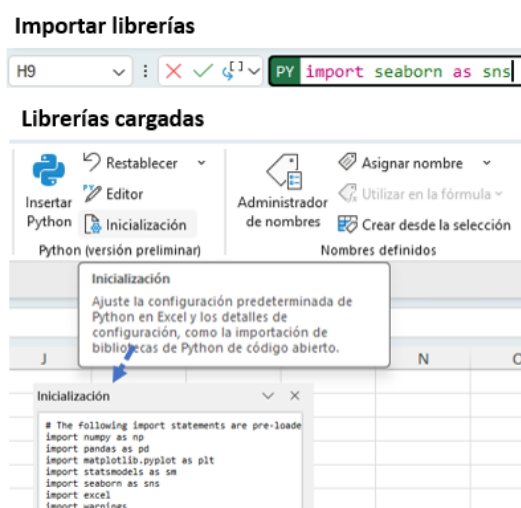


Ilustración 4

**Sugerencia:** Para asegurarse de que las bibliotecas se importan antes de que se ejecuten las fórmulas de Python, escriba las instrucciones import y cualquier configuración en la primera hoja de cálculo del libro. Si lo desea, puede reservar la primera hoja de cálculo específicamente para las instrucciones de importación y la configuración.

## Restricciones de las librerías con Python en Excel

- No se permite el acceso a archivos locales ni a la red por razones de seguridad.
- Las bibliotecas deben estar incluidas en la distribución segura de Anaconda.
- No puedes instalar paquetes personalizados con pip.

## IA con Python en Excel

Si no sabemos que comando utilizar para hacer determinados cálculos con Python en Excel podemos hacer nuestras preguntas a la IA, Copilot, ChatGpt, Geminis, etc.. y el resultado copiarlo. Veremos un ejemplo más adelante, “Usando la IA para obtener el código Python”

## Primeros pasos de Python en Excel

### Empezar a usar Python

Para empezar a usar Python en Excel, seleccione una celda y, en la pestaña **Fórmulas**, seleccione **Insertar Python**. Esto indica a Excel que desea escribir una fórmula Python en la celda seleccionada.

O use la función **=PY** en una celda para habilitar Python. Después de escribir **=PY** en la celda, elija PY en el menú Función Autocompletar con las teclas de flecha abajo y tab, o agregue un paréntesis de apertura a la función: **=PY(**. Ahora, puede escribir código Python directamente en la celda.

La siguiente captura de pantalla muestra el menú Autocompletar con la función PY seleccionada.

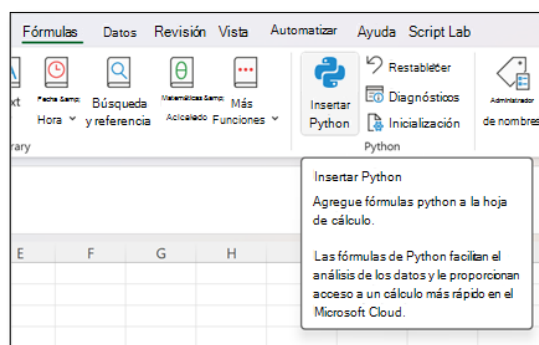


Ilustración 5

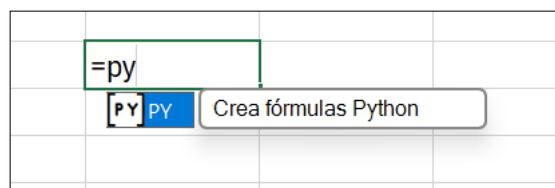


Ilustración 6

Una vez que Python está habilitado en una celda, esa celda muestra un icono **PY**. La barra de fórmulas muestra el mismo icono PY cuando se selecciona la celda Python. Consulta la siguiente captura de pantalla para obtener un ejemplo.

## Procesos básicos

### Carga de Librerías

El primer paso es cargar las librerías necesarias.

- **Ejemplo:** Para importar Pandas y Seaborn, se escribiría: `import pandas` y `import seaborn`.
- **Visualización:** Para evitar que la celda muestre cero (ya que no hay nada que mostrar de la importación), se recomienda añadir una cadena de texto descriptiva, por ejemplo: "Librerías cargadas".
- **Objeto Python:** Si la celda no muestra el icono de objeto, se puede cambiar a "Objeto Python" a través del menú desplegable de la celda.

### Definición de un DataFrame

Un paso crucial para el análisis de datos es definir un DataFrame a partir de los datos de Excel.

- **Nomenclatura:** Un nombre común para un DataFrame es DF.

- **Creación:** Para definir un DataFrame, se escribe =PY, se nombra el DataFrame (ej. DF), se selecciona el rango de datos en Excel y se indica si los datos tienen encabezados (ej. DF = xl("A1:G10", headers=True)).
- **Salida por Defecto:** Inicialmente, el resultado puede "derramar" los datos en celdas adyacentes. Se recomienda cambiar la salida a "Objeto Python" para que se colapse en una sola celda, facilitando su referencia.
- **Buena Práctica:** Es aconsejable definir las librerías y el DataFrame en celdas separadas del código Python principal para que estén disponibles para otras fórmulas Python en el libro.

### Orden de cálculo y análisis de datos

Una vez que el DataFrame está definido, se pueden realizar análisis de datos.

- **Orden de Cálculo:** Las celdas Python se calculan en orden de fila principal (de izquierda a derecha, de arriba abajo). Por lo tanto, el DataFrame debe estar definido en una celda *superior* a cualquier fórmula Python que lo referencie.
- **Estadísticas Descriptivas:** Para generar estadísticas descriptivas de un DataFrame completo, se usa DF.describe(). Esto devolverá estadísticas para todas las columnas numéricas del conjunto de datos.

### Combinar Python con celdas y rangos de Excel

Para hacer referencia a objetos de Excel en una celda de Python, asegúrese de que la celda de Python está en modo de edición y, a continuación, seleccione la celda o el rango que desea incluir en la fórmula de Python. Esto rellena automáticamente la celda Python con la dirección de la celda o rango que ha seleccionado.

Python en Excel utiliza la función personalizada de Python xl() para interfaz entre Excel y Python. La función xl() acepta objetos de Excel como rangos, tablas, consultas y nombres.

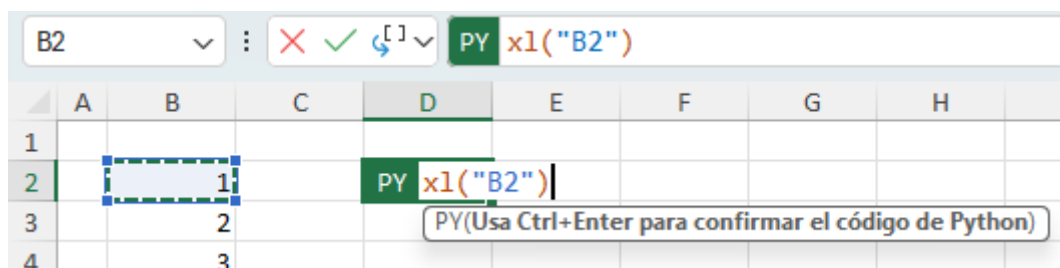


Ilustración 7

También puede escribir referencias directamente en una celda de Python con la función xl() . Por ejemplo, para hacer referencia a la celda **B** use xl("B2") y para el rango **B1:C4** use xl("B1:C4"). Para una tabla con encabezados denominados **TablaEdad**, use xl("TablaEdad").

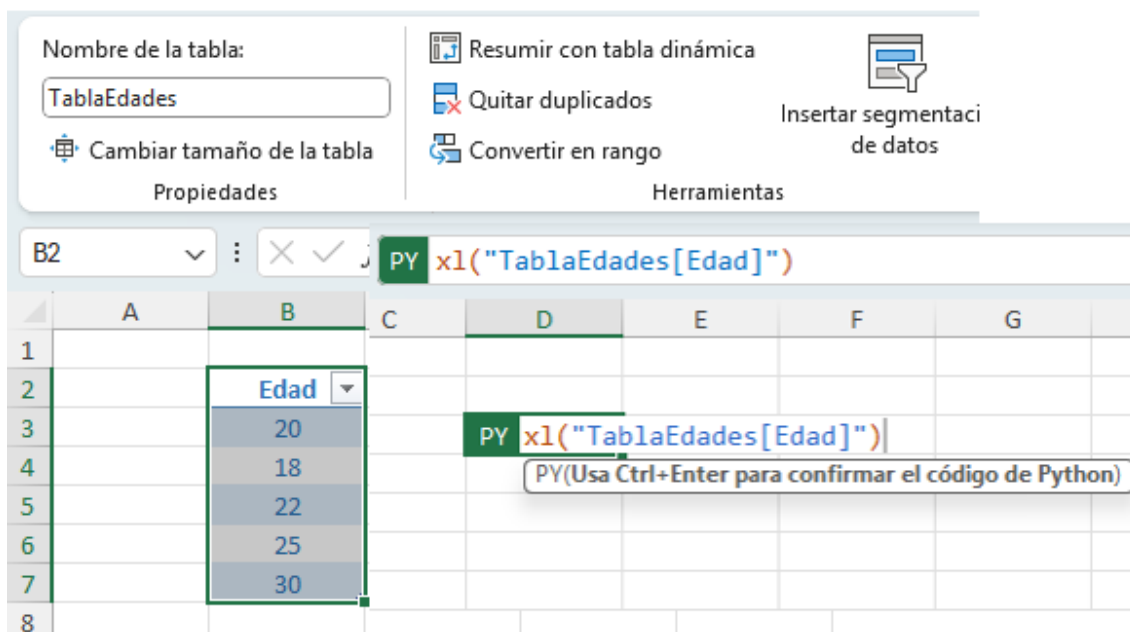


Ilustración 8

La imagen siguiente muestra un cálculo de Python en Excel que agrega los valores de la celda A1 y B1, con el resultado de Python devuelto en la celda C1.

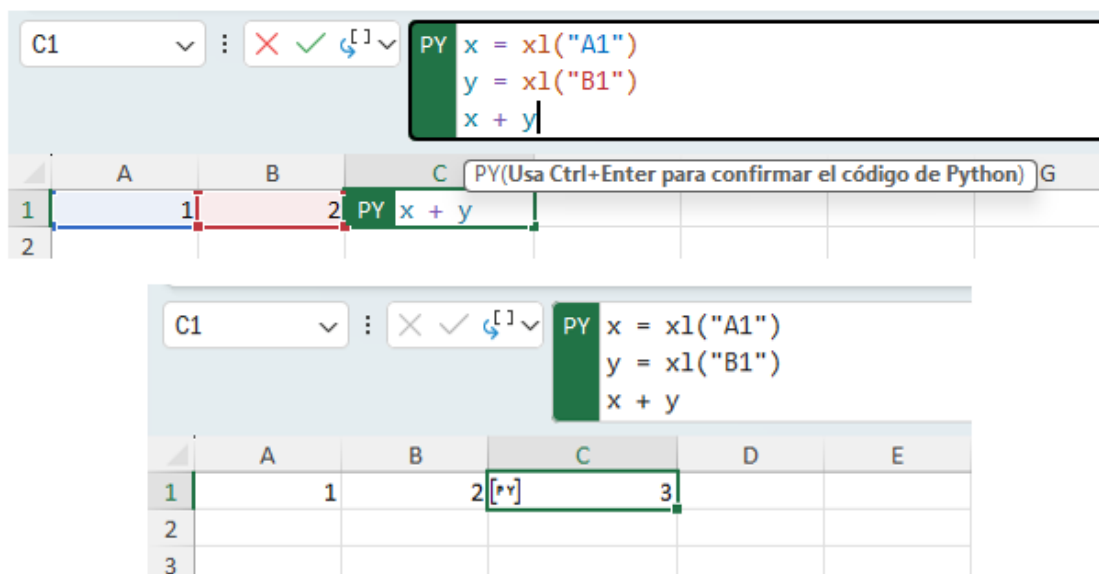


Ilustración 9

## Tipos de salidas

Use el menú de salida de Python en la barra de fórmulas para controlar cómo se devuelven los cálculos de Python. Devolver cálculos como objetos python o convertir cálculos a valores de Excel y enviarlos directamente a una celda.

Cuando se devuelve una fórmula como un objeto Python, la celda muestra un icono de tarjeta que contiene información adicional dentro de la celda Ilustración 10.

Para ver la información adicional, abra la tarjeta seleccionando el icono de la tarjeta. La información que se muestra en la tarjeta es una

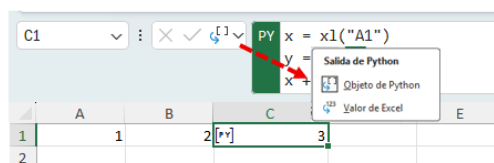


Ilustración 10

vista previa del objeto, que es útil al procesar objetos grandes (Ilustración 11).

Los resultados de la fórmula devueltos a los valores de Excel se traducen al equivalente de Excel más cercano. Si planea reutilizar el resultado en un cálculo futuro de Python, se recomienda devolver el resultado como un objeto Python. Devolver un resultado como valores de Excel le permite ejecutar análisis de Excel, como gráficos de Excel, fórmulas y formato condicional, en el valor.

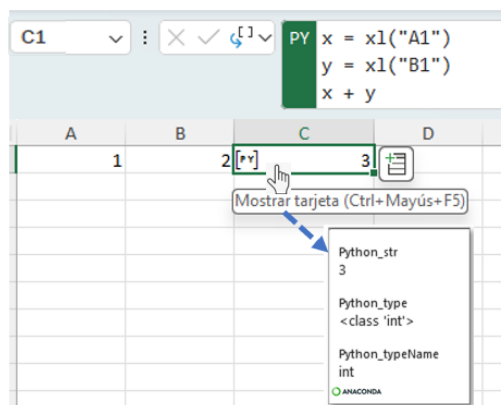


Ilustración 11

Python en Excel puede devolver muchos tipos de datos como objetos de Python. Una python útil en el tipo de datos de Excel es un DataFrame objeto.

## Presentacion del editor de codigo Python del Excel

### Entrar en modo Python

Para entrar en modo de edición de Python disponemos de tres formas,

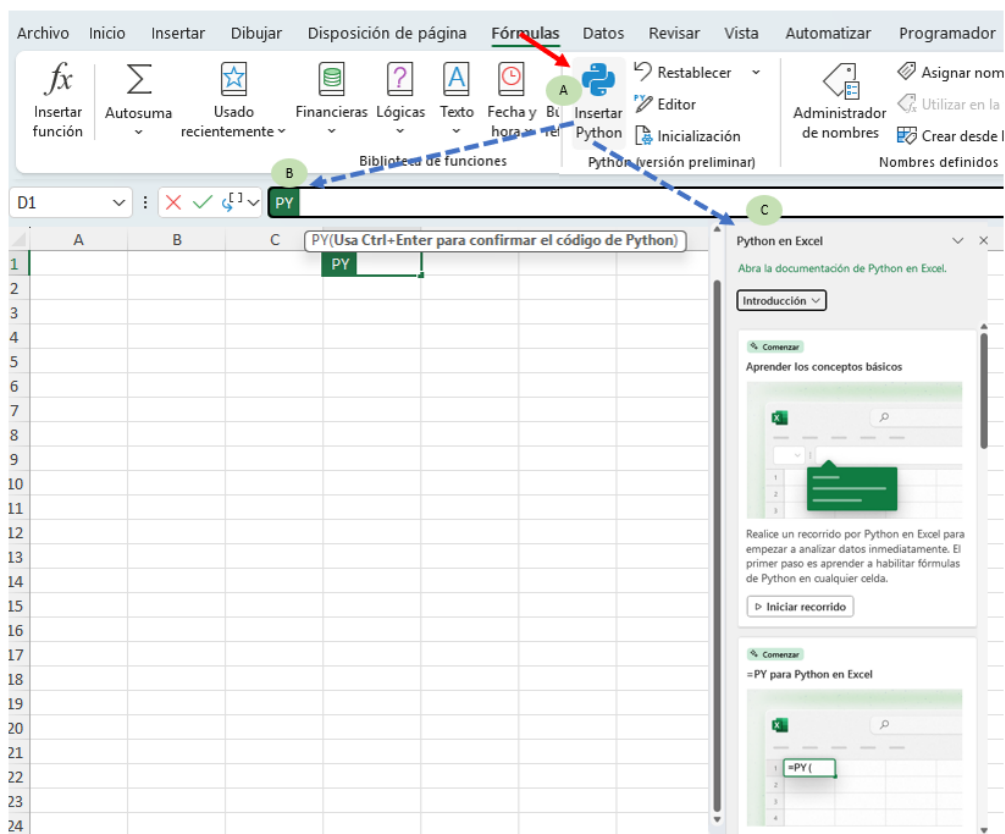


Ilustración 12

- Accediendo a el menú Formulas y seleccionar Insertar Python con lo cual se nos activa automáticamente el area B y C, desde donde podemos escribir los script deseados
- Otra opción directa es acudir a la barra de fórmulas y comenzar con =PY( (y ya podemos definir la instrucción, es decir escribiendo =PY y abriendo la barra de paréntesis, entramos así el modo Python.
- O acceder directamente al panel de edición Python en Excel.

## Acceso

Para obtener un espacio de edición de código aún mayor disponemos de un editor de código de Python en Excel. Este se muestra como un panel de tareas que proporciona un espacio de edición más amplio. Muestra todo el código Python en el libro, enumerado por hoja de cálculo y número de celda. El editor de código también incluye características como IntelliSense y colorización de texto para admitir la edición de grandes bloques de código. Para abrir el editor seleccionamos una celda de Python en el libro y, accedemos a **Fórmulas > Editor**.

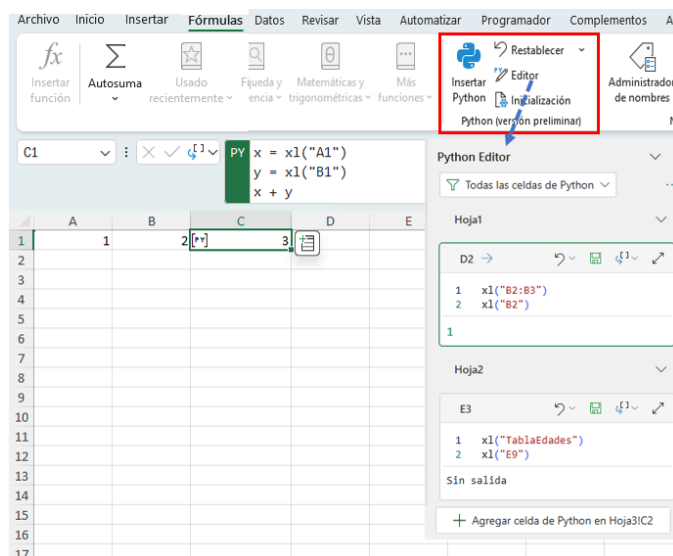


Ilustración 13

Se abrirá el panel de tareas y se mostrarán todas las celdas de Python en el libro.

## Opciones de edición en el panel de tareas

Dentro del panel de tareas del editor de código, puede editar cada celda python del libro. Use el menú situado encima de cada celda del panel de tareas para guardar y ejecutar una celda de Python, cambiar entre tipos de salida (de objeto de Python a valor de Excel), deshacer cambios o expandir una celda a un área de edición de código más grande.

Destacar también que podemos hacer uso del filtro en la parte superior del panel de tareas para restringir el contenido que se muestra en el panel de tareas del editor de código, como mostrar solo las celdas de Python en la hoja de cálculo actual o mostrar solo errores de Python.

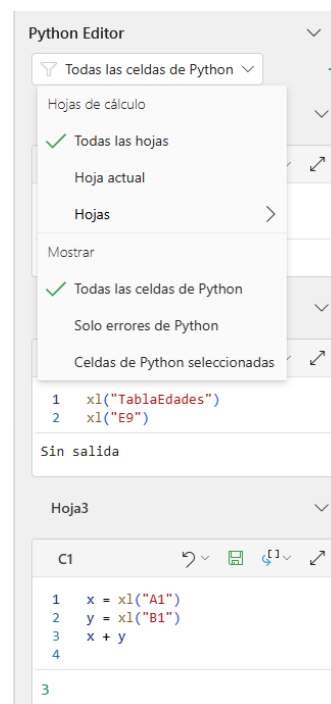


Ilustración 14

## Mostrar salida de Python en el panel de tareas

La salida de cada celda python en el panel de tareas del editor de código se muestra en el libro de Excel. También puede ver la salida en el panel de tareas del editor de código expandiendo el tipo de salida en la parte inferior de cada bloque de código

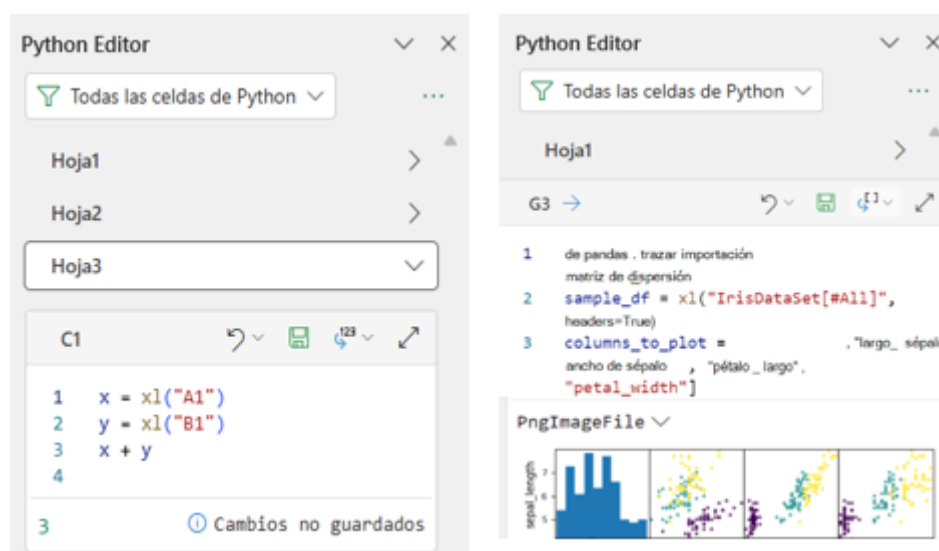


Ilustración 15

### Salida de Python de un DataFrame.

Pero estos valores los podemos ver como objeto de Python o como Valor de Excel, tal y como se muestra en la siguiente ilustración.

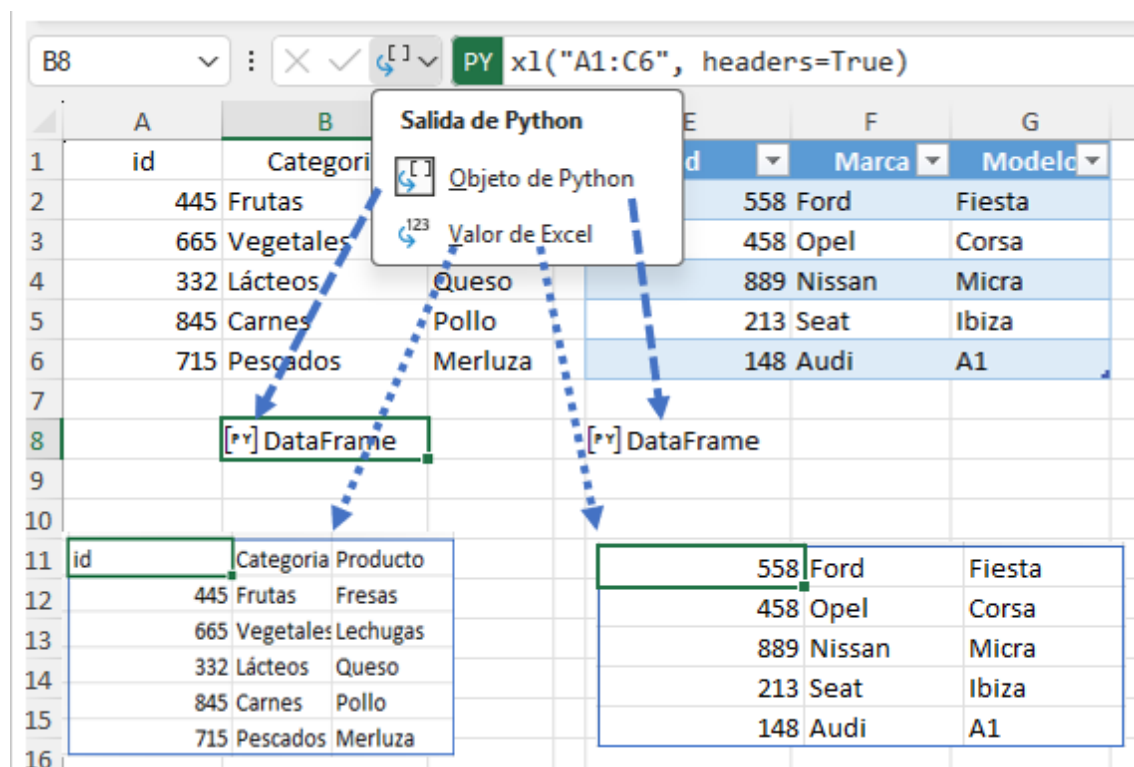


Ilustración 16

### Tipos de datos como objetos de Python: DataFrames y tipos de salidas

Como hemos comentado, Python en Excel puede devolver muchos tipos de datos como objetos de Python. Una python útil en el tipo de datos de Excel es un DataFrame objeto. Para obtener más información sobre los DataFrames de Python, consulte [Python en Excel DataFrames](#).

Un DataFrame es una estructura de datos bidimensional en lenguajes de programación de computadoras, similar a una tabla de Excel. En Python, un DataFrame es un objeto en la biblioteca [pandas](#). La biblioteca pandas es una biblioteca principal utilizada por Python en Excel, y los objetos DataFrame son una estructura clave para analizar datos con Python en Excel.

**Nota:** Python en Excel usa un DataFrame como objeto predeterminado para rangos bidimensionales.

Python en Excel puede generar un DataFrame de dos maneras: como un objeto de Python o convertido en valores de Excel. Cuando se devuelve un DataFrame como un objeto de Python, la celda muestra el texto "DataFrame", precedido por un icono de tarjeta. En la Ilustración 17 se muestra un DataFrame como un objeto de Python en una celda de Excel.

Para ver la información dentro de un DataFrame, seleccione el icono de tarjeta en la celda o use **Ctrl+Mayús+F5**. En la captura de pantalla siguiente se muestra un objeto DataFrame con la vista de tarjeta abierta. Este DataFrame de ejemplo repite los datos de las celdas **B3:D5** de la hoja de cálculo.

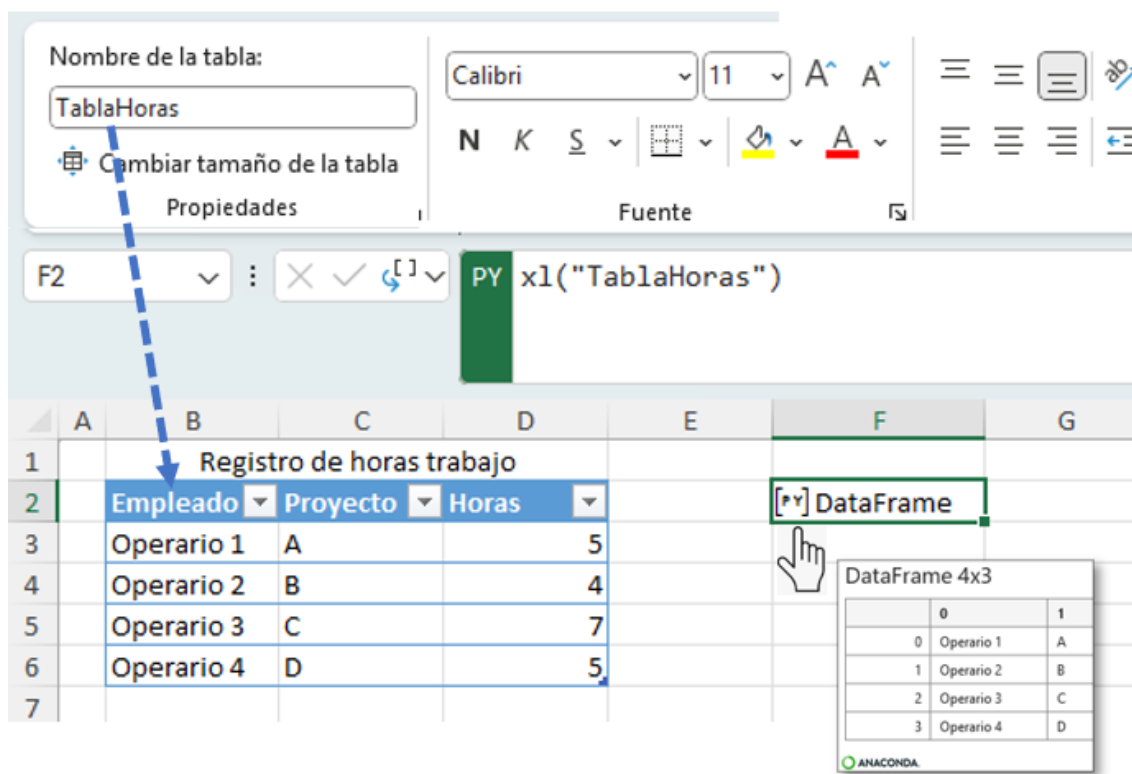


Ilustración 17

También puede importar datos externos a Excel y luego procesar esos datos con Python en Excel. Para importar datos externos, use Power Query.

#### Cambiar el tipo de salida de Python

Los datos dentro de un DataFrame se pueden devolver como valores de Excel en lugar de como un objeto de Python. Genere un DataFrame como valores de Excel para incorporar otros análisis basados en Excel, como gráficos, fórmulas de Excel y formato condicional.

Use el menú de salida de Python en la barra de fórmulas para controlar cómo se devuelven los cálculos de Python. En la captura de pantalla siguiente se muestra el menú de salida de Python en la barra de fórmula

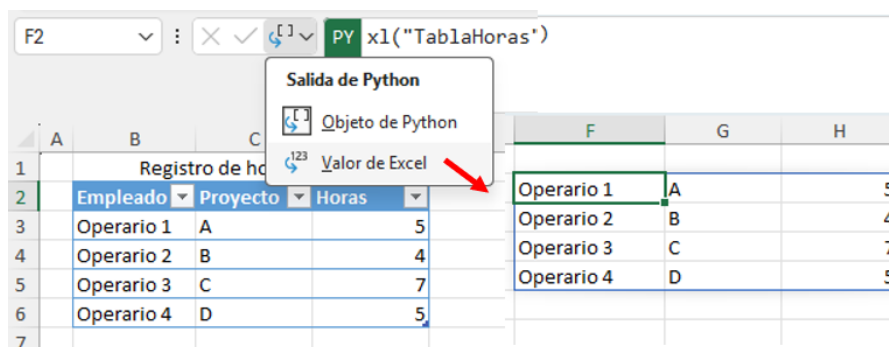


Ilustración 18

## Orden de cálculo, recálculo y errores

### Orden de cálculo

Las instrucciones de Python tradicionales se calculan de arriba a abajo. En una celda de Python en Excel, las instrucciones de Python hacen lo mismo: calculan de arriba a abajo. Pero en una hoja de cálculo de Python en Excel, las celdas de Python se calculan en orden principal de fila. Los cálculos de celdas se ejecutan a través de una fila (de la columna A a la columna XFD) y, a continuación, a través de cada fila siguiente hacia abajo en la hoja de cálculo.

Las instrucciones python están ordenadas, por lo que cada instrucción Python tiene una dependencia implícita en la instrucción python que precede inmediatamente en el orden de cálculo.

*El orden de cálculo es importante al definir variables y hacer referencia a ellas en una hoja de cálculo, porque debe definir variables antes de poder hacer referencia a ellas.*

**Importante:** El orden de cálculo de fila principal también se aplica en todas las hojas de cálculo de un libro y se basa en el orden de las hojas de cálculo del libro. Si usa varias hojas de cálculo para analizar datos con Python, asegúrese de incluir datos y cualquier variable que almacene datos en celdas y hojas de cálculo anteriores a las celdas y hojas de cálculo que analizan esos datos.

### Recálculo

Cuando cambia un valor dependiente de una celda de Python, todas las fórmulas de Python se vuelven a calcular secuencialmente. Para suspender los recálculos de Python y mejorar el rendimiento, utilice el **modo cálculo parcial** o el **modo de cálculo manual**. Estos modos te permiten desencadenar un cálculo cuando estés listo. Para cambiar esta configuración, vaya a la cinta de opciones, seleccione **Fórmulas** y, después, abra **Opciones de cálculo**. A continuación, seleccione el modo de cálculo deseado. Los modos **cálculo parcial** y **cálculo manual** suspenden el recálculo automático para las tablas de datos y Python.

*Deshabilitar el recálculo automático en un libro durante el desarrollo de Python puede mejorar el rendimiento y las velocidades individuales de cálculo de las celdas de Python. Sin embargo, debe volver a calcular manualmente el libro para garantizar la precisión en cada celda de Python.* Hay tres formas de volver a calcular manualmente un libro en el **modo Cálculo parcial** o **Cálculo manual**.

1. Use el método abreviado de teclado **F9**.
2. Vaya a **Fórmulas** > **Calcular ahora** en la cinta de opciones.
3. Vaya a una celda con un valor obsoleto, mostrado con formato de tachado, y seleccione el símbolo de error junto a esa celda. Después, seleccione **Calcular ahora** en el menú.

### Errores

Los cálculos de Python en Excel pueden devolver errores como **#PYTHON!**, **#BUSY!** y **#CONNECT!** a las celdas de Python. Para obtener más información, consulte [Solucionar problemas de Python en errores de Excel](#).

## Importar datos externos

### Introducción

Todos los datos que procesa con Python en Excel deben proceder de la hoja de cálculo o de Power Query. Para importar datos externos, use la característica Obtener transformación de & en Excel para acceder a [Power Query](#). Para obtener más información,

**Importante:** Para proteger su seguridad, las funciones de datos externos comunes en Python, como `pandas.read_csv` y `pandas.read_excel`, no son compatibles con Python en Excel. Para obtener más información, consulte [Seguridad de datos y Python en Excel](#).

### Power Query (PQ) para importar datos para Python en Excel

#### Introducción y consideraciones

Use Power Query con Python en Excel para importar datos externos en Excel y luego analizar esos datos con Python.

**Importante:** Power Query es la única manera de importar datos externos para su uso con Python en Excel.

**Nota:** La importación de datos externos con Power Query usar con Python en Excel no está disponible para Excel en la Web.

#### Pasos a seguir

Siga estos pasos para aprender a importar un conjunto de datos externos de ejemplo y a usarlo con Python en Excel.

Suponiendo que ya tenemos creado la conexión a nuestro origen de datos con PQ y disponible en el libro Excel de trabajo, vamos a presentar los pasos a seguir para que este este operativa con Python.

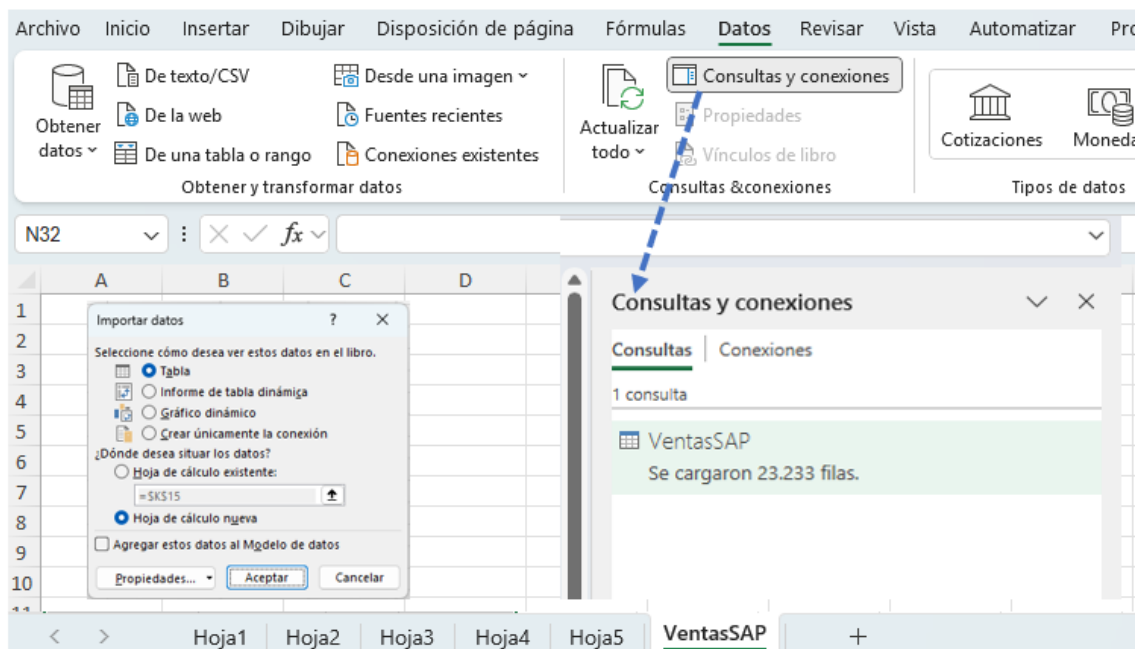


Ilustración 19

En este caso solo tenemos la conexión realizada a la fuente.

1. Para trabajar con datos externos con Python en Excel, habilite Python en una celda especificando la función `=PY`. Después, use python en Excel `xl()` fórmula para hacer referencia a elementos de Excel, como una consulta de Power Query. Para este ejemplo, escriba `xl("TablasVentasSAP")` en python en la celda de Excel.

- La celda Python ahora muestra los datos dentro de un DataFrame.

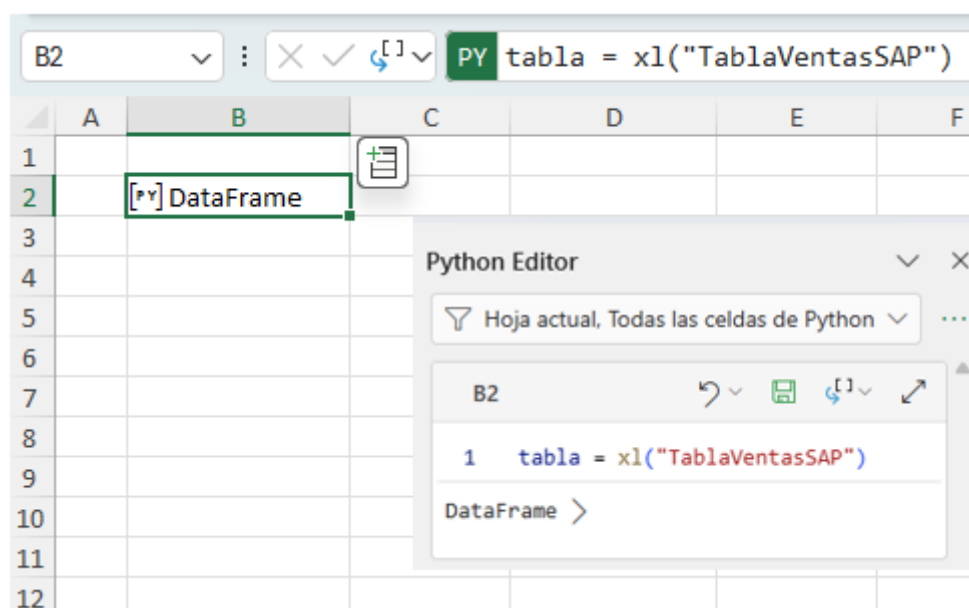


Ilustración 20

Creamos la variable que llamamos `tabla` definiendo `xl()` como una **función personalizada** para acceder a objetos de Excel, como tablas, rangos o nombres definidos, en este caso a la tabla **TablaVentasSAP**.

¿Qué hace `xl("TablaVentasSAP")`?

- Busca en el libro de Excel** un objeto con el nombre "TablaVentasSAP".
- Devuelve un rango o una tabla** que puedes manipular desde Python.
- Es parte de una **abstracción** que simplifica el acceso a elementos de Excel sin tener que usar directamente `xlwings.Book`, `Sheet`, `Range`, etc.

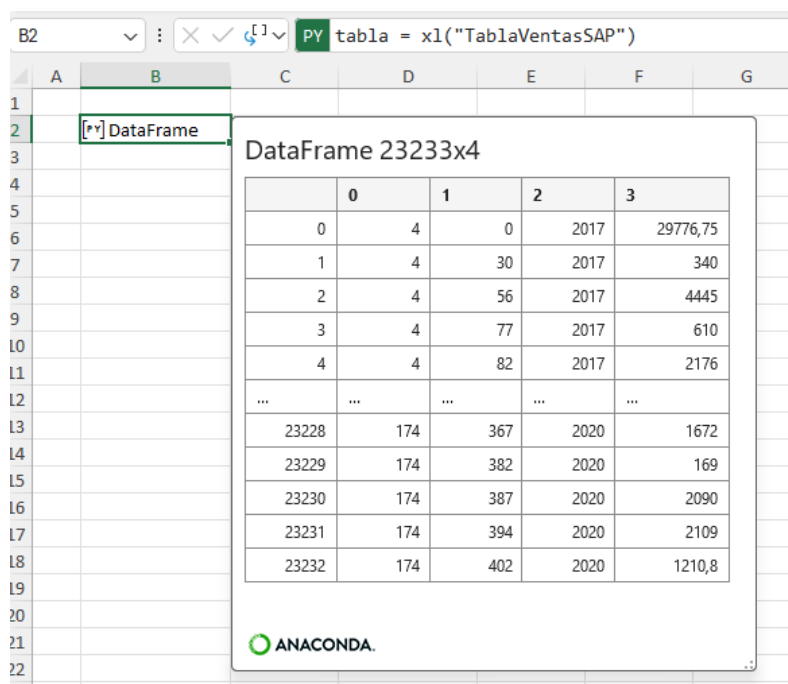


Ilustración 21

**Sugerencia:** Cambie el tamaño del cuadro de diálogo `DataFrame` con el icono de la esquina inferior derecha.

Convierta los datos del DataFrame en valores de Excel, que devuelve los datos a la cuadrícula de Excel. Para convertir el DataFrame en valores de Excel, seleccione la celda que contiene el DataFrame y, a continuación, seleccione el icono **Insertar datos**. La siguiente captura de pantalla muestra un ejemplo de este icono junto al DataFra

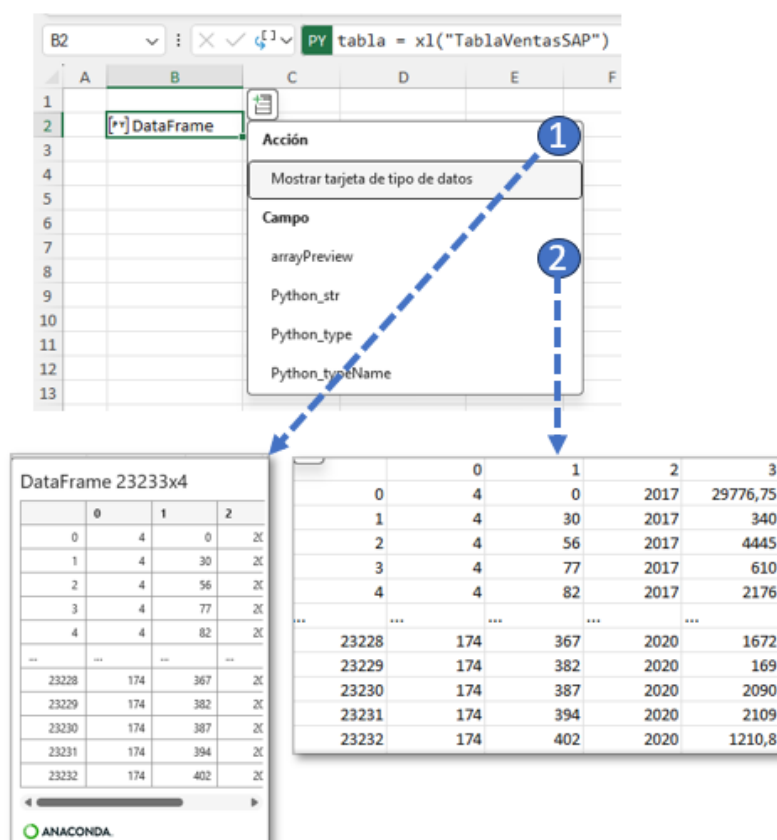


Ilustración 22

Ahora que ha usado Power Query para importar datos externos y procesado esos datos con Python en Excel, está listo para empezar a analizar datos con Python en Excel.

## Casos de iniciación a Python con Excel.

### Ventas de Coches

Adaptado de [Python en Excel \[¡Ahora Integrado!\] - YouTube](#)

#### Presentación. Configuración del DataFrame

Disponemos de la siguiente tabla de ventas de coches de segunda mano y comenzamos creando el DataFrame en Python tal, para lo cual seleccionamos todo incluidos los encabezados.

|   | A   | B      | C      |          |
|---|-----|--------|--------|----------|
| 1 |     |        |        |          |
| 2 | id  | Marca  | Modelo | Cantidad |
| 3 | 558 | Ford   | Fiesta | 50       |
| 4 | 458 | Opel   | Corsa  | 40       |
| 5 | 889 | Nissan | Micra  | 20       |
| 6 | 213 | Seat   | Ibiza  | 15       |
| 7 | 148 | Audi   | A1     | 10       |
| 8 |     |        |        |          |

Ilustración 23

Podríamos crear una variable para que el Dataframe se llamara df y de esta forma quedaría mejor asignado.

#### Asignación de nombres a las variables

Podríamos crear una variable para que el Dataframe se llamara MisDatos y de esta forma quedaría mejor asignado, facilitaría la interpretación así como la referencia al DataFrame

|   | A   | B      | C      | D        | E              | F | G             | H |
|---|-----|--------|--------|----------|----------------|---|---------------|---|
| 1 |     |        |        |          |                |   |               |   |
| 2 | id  | Marca  | Modelo | Cantidad | Ingresos Total |   | [*] DataFrame |   |
| 3 | 558 | Ford   | Fiesta | 50       | 7.200          |   |               |   |
| 4 | 458 | Opel   | Corsa  | 40       | 6.800          |   |               |   |
| 5 | 889 | Nissan | Micra  | 20       | 6.100          |   |               |   |
| 6 | 213 | Seat   | Ibiza  | 15       | 7.500          |   |               |   |
| 7 | 148 | Audi   | A1     | 10       | 10.200         |   |               |   |
| 8 |     |        |        |          |                |   |               |   |

Ilustración 24

Igualmente podremos hacer para el resto celdas anteponiendo el nombre con un igual

#### Introducción al análisis de datos con Python

Vamos a realizar una breve introducción al análisis de datos con Python con los métodos básicos.

#### *Estadísticas principales de nuestros datos, metodo describe ( )*

Para ello seleccionamos la celda donde está el DataFrame, la superior y escribimos la sentencia que nos permite realizar un análisis básico

G3 : X ✓ PY x1("G2").describe()

|    | A | B   | C      | D      | E        | F              | G              | H | I |
|----|---|-----|--------|--------|----------|----------------|----------------|---|---|
| 1  |   |     |        |        |          |                |                |   |   |
| 2  |   | id  | Marca  | Modelo | Cantidad | Ingresos Total | [*v] DataFrame |   |   |
| 3  |   | 558 | Ford   | Fiesta | 50       | 7.200          | [*v] DataFrame |   |   |
| 4  |   | 458 | Opel   | Corsa  | 40       | 6.800          |                |   |   |
| 5  |   | 889 | Nissan | Micra  | 20       | 6.100          |                |   |   |
| 6  |   | 213 | Seat   | Ibiza  | 15       | 7.500          |                |   |   |
| 7  |   | 148 | Audi   | A1     | 10       | 10.200         |                |   |   |
| 8  |   |     |        |        |          |                |                |   |   |
| 9  |   |     |        |        |          |                |                |   |   |
| 10 |   |     |        |        |          |                |                |   |   |
| 11 |   |     |        |        |          |                |                |   |   |
| 12 |   |     |        |        |          |                |                |   |   |
| 13 |   |     |        |        |          |                |                |   |   |

DataFrame 8x3

|       | 0           | 3           | 4           |
|-------|-------------|-------------|-------------|
| count | 5           | 5           | 5           |
| mean  | 453,2       | 27          | 7560        |
| std   | 296,5513446 | 17,17556404 | 1566,205606 |
| min   | 148         | 10          | 6100        |
| 25%   | 213         | 15          | 6800        |
| 50%   | 458         | 20          | 7200        |
| 75%   | 558         | 40          | 7500        |
| max   | 889         | 50          | 10200       |

Ilustración 25

Si lo cambiamos a que nos muestre valor de Excel, este sería el resultado.

G3 : X ✓ PY x1("G2").describe()

|    | A | B   | C      | D      | E              | F         | G              | H |
|----|---|-----|--------|--------|----------------|-----------|----------------|---|
| 1  |   |     |        |        |                |           |                |   |
| 2  |   | id  | Marca  | Modelo | Ingresos Total |           | [*v] DataFrame |   |
| 3  |   | 558 | Ford   | Fiesta | 7.200          |           | [*v] DataFrame |   |
| 4  |   | 458 | Opel   | Corsa  | 6.800          |           |                |   |
| 5  |   | 889 | Nissan | Micra  | 6.100          |           |                |   |
| 6  |   | 213 | Seat   | Ibiza  | 7.500          |           |                |   |
| 7  |   | 148 | Audi   | A1     | 10.200         |           |                |   |
| 8  |   |     |        |        |                |           |                |   |
| 9  |   |     |        | 0      | 3              | 4         |                |   |
| 10 |   |     |        | count  | 5              | 5         |                |   |
| 11 |   |     |        | mean   | 453,2          | 27        |                |   |
| 12 |   |     |        | std    | 296,5513       | 17,175564 |                |   |
| 13 |   |     |        | min    | 148            | 10        |                |   |
| 14 |   |     |        | 25%    | 213            | 15        |                |   |
| 15 |   |     |        | 50%    | 458            | 20        |                |   |
| 16 |   |     |        | 75%    | 558            | 40        |                |   |
| 17 |   |     |        | max    | 889            | 50        |                |   |
| 18 |   |     |        |        |                |           |                |   |

Ilustración 26

En este caso para aquellas columnas que contiene valores numéricos (columna 0-3-4) se le aplica el análisis de los principales estadísticos de nuestros datos.

Si quisiéramos solo para una columna, por ejemplo Cantidades, tendríamos que modificar el código anterior incorporando el nombre de la columna como vemos en la siguiente ilustración.

G4 : `x1("G2")["Cantidades"].describe()`

|    | A   | B      | C      | D        | E              | F | G             | H |
|----|-----|--------|--------|----------|----------------|---|---------------|---|
| 1  |     |        |        |          |                |   |               |   |
| 2  | id  | Marca  | Modelo | Cantidad | Ingresos Total |   | [r] DataFrame |   |
| 3  | 558 | Ford   | Fiesta | 50       | 7.200          |   | [r] DataFrame |   |
| 4  | 458 | Opel   | Corsa  | 40       | 6.800          |   | [r] Series    |   |
| 5  | 889 | Nissan | Micra  | 20       | 6.100          |   |               |   |
| 6  | 213 | Seat   | Ibiza  | 15       | 7.500          |   |               |   |
| 7  | 148 | Audi   | A1     | 10       | 10.200         |   |               |   |
| 8  |     |        |        |          |                |   |               |   |
| 9  |     |        |        |          |                |   |               |   |
| 10 |     |        |        |          |                |   |               |   |
| 11 |     |        |        |          |                |   |               |   |
| 12 |     |        |        |          |                |   |               |   |
| 13 |     |        |        |          |                |   |               |   |
| 14 |     |        |        |          |                |   |               |   |
| 15 |     |        |        |          |                |   |               |   |

Series 8x1

|       | Cantidades  |
|-------|-------------|
| count | 5           |
| mean  | 27          |
| std   | 17,17556404 |
| min   | 10          |
| 25%   | 15          |
| 50%   | 20          |
| 75%   | 40          |
| max   | 50          |

Ilustración 27

### Promedio de la cantidad vendida. mean ()

Para mejorar el modelo primero cambiamos el nombre al DataFrame diseñado en la celda G2 (Ilustración 24) vamos a asignarle el nombre MisDatos y creamos la formula.

G5 : `MisDatos["Cantidades"].mean()`

|   | A   | B      | C      | D        | E              | F | G             |
|---|-----|--------|--------|----------|----------------|---|---------------|
| 1 |     |        |        |          |                |   |               |
| 2 | id  | Marca  | Modelo | Cantidad | Ingresos Total |   | [r] DataFrame |
| 3 | 558 | Ford   | Fiesta | 50       | 7.200          |   | [r] DataFrame |
| 4 | 458 | Opel   | Corsa  | 40       | 6.800          |   | [r] Series    |
| 5 | 889 | Nissan | Micra  | 20       | 6.100          |   | [r] 27        |
| 6 | 213 | Seat   | Ibiza  | 15       | 7.500          |   |               |
| 7 | 148 | Audi   | A1     | 10       | 10.200         |   |               |

Ilustración 28

### Ranking de las 2 mejores ventas. sort\_value ()

En este caso deseamos obtener los 3 mejores coches según cantidad ordenados descendientemente tal y como se muestra en la siguiente ilustración.

G6 : `Top_3=MisDatos.sort_values(by="Cantidades",ascending=False).head(3)`

|    | A   | B      | C      | D        | E              | F | G             | H | I | J |
|----|-----|--------|--------|----------|----------------|---|---------------|---|---|---|
| 1  |     |        |        |          |                |   |               |   |   |   |
| 2  | id  | Marca  | Modelo | Cantidad | Ingresos Total |   | [r] DataFrame |   |   |   |
| 3  | 558 | Ford   | Fiesta | 50       | 7.200          |   | [r] DataFrame |   |   |   |
| 4  | 458 | Opel   | Corsa  | 40       | 6.800          |   | [r] Series    |   |   |   |
| 5  | 889 | Nissan | Micra  | 20       | 6.100          |   | [r] 27        |   |   |   |
| 6  | 213 | Seat   | Ibiza  | 15       | 7.500          |   | [r] DataFrame |   |   |   |
| 7  | 148 | Audi   | A1     | 10       | 10.200         |   |               |   |   |   |
| 8  |     |        |        |          |                |   |               |   |   |   |
| 9  |     |        |        |          |                |   |               |   |   |   |
| 10 |     |        |        |          |                |   |               |   |   |   |
| 11 |     |        |        |          |                |   |               |   |   |   |
| 12 |     |        |        |          |                |   |               |   |   |   |
| 13 |     |        |        |          |                |   |               |   |   |   |
| 14 |     |        |        |          |                |   |               |   |   |   |
| 15 |     |        |        |          |                |   |               |   |   |   |

DataFrame 3x5

|   | id  | Marca  | Modelo | Cantidades | Ingresos Totales |
|---|-----|--------|--------|------------|------------------|
| 0 | 558 | Ford   | Fiesta | 50         | 7200             |
| 1 | 458 | Opel   | Corsa  | 40         | 6800             |
| 2 | 889 | Nissan | Micra  | 20         | 6100             |

Ilustración 29

El significado de la fórmula es el siguiente:

**Top\_3=MisDatos.sort\_values(by="Cantidades",ascending=False).head(3)**

Definimos la variable `Top_3`, y le decimos que partiendo de nuestro DataFrame (`MisDatos`), ordene los valores (`sort_value`) por la columna `Cantidades` y de forma descendente (`ascending=False`) y me devuelva los 3.

### Gráficos de datos con la librería seaborn. Gráfico de dispersión

En este caso nos interesa visualizar los datos para lo cual necesitamos disponer de la librería seaborn,

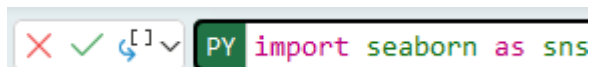


Ilustración 30

Si no está cargada por defecto, tendríamos que hacerlo con la instrucción de la Ilustración 30, pero como vimos en la Ilustración 4 esta se inicializa con Python para Excel.

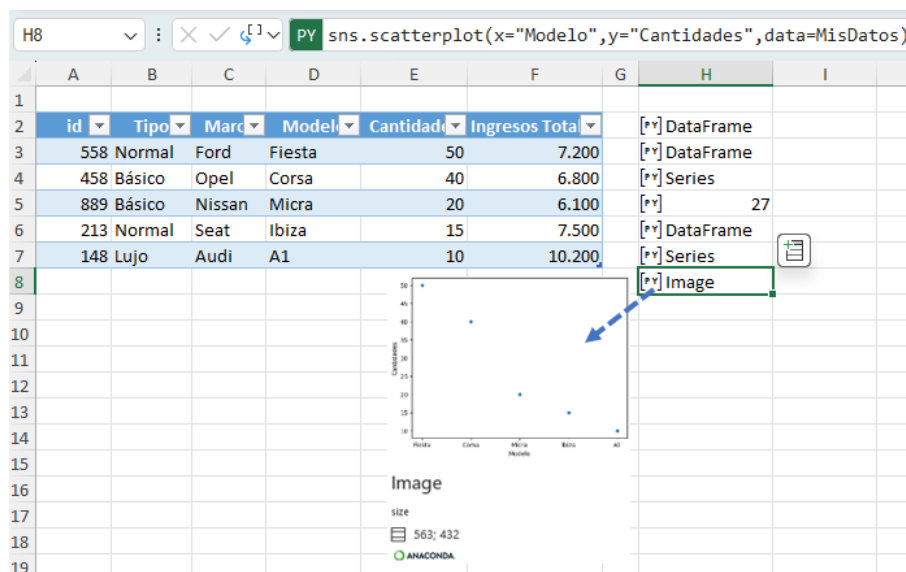


Ilustración 31

**`sns.scatterplot(x="Modelo",y="Cantidades",data=MisDatos)`**

Explicación del código

La indicación [Estoy trabajando con Python para Excel y tengo este... | Probar en Copilot Chat](#)

Esta instrucción en Python que utiliza la biblioteca **Seaborn** para crear un **gráfico de dispersión** (scatter plot). Vamos a desglosarlo:

- **¿Qué hace cada parte?**
- **`sns.scatterplot(...)`:** Llama a la función `scatterplot` de la biblioteca Seaborn (`sns` es el alias común para seaborn). Esta función se usa para crear gráficos de dispersión, que muestran la relación entre dos variables numéricas.
- **`x="Modelo"`:** Indica que la variable que se usará en el eje X del gráfico es la columna "Modelo" del DataFrame `MisDatos`.
- **`y="Cantidades"`:** Indica que la variable que se usará en el eje Y es la columna "Cantidades".
- **`data=MisDatos`:** Especifica el **DataFrame** que contiene los datos. En este caso, `MisDatos` debe ser un DataFrame de pandas que tiene al menos las columnas "Modelo" y "Cantidades".
- **¿Qué representa el gráfico?**

Este gráfico mostrará un punto por cada fila en `MisDatos`, ubicando cada punto según el valor de "Modelo" en el eje X y el valor de "Cantidades" en el eje Y. Es útil para visualizar cómo varían las cantidades según los modelos.

- ¿Qué podrías agregar?

Dependiendo de lo que quieras analizar, podrías añadir:

- **hue="Categoría"**: Para colorear los puntos según otra variable.
- **style="Tipo"**: Para cambiar el estilo de los puntos según una categoría.
- **size="Importancia"**: Para variar el tamaño de los puntos.

### Dimensionando y moviendo los elementos graficos

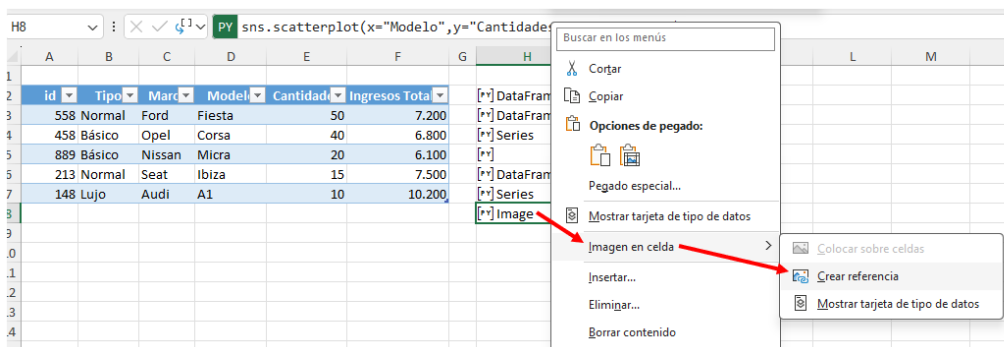


Ilustración 32

Sobre la celda que contiene el código de imagen y con el botón derecho accedemos a Imagen en Celda, Crear Referencia y nos da una imagen que podemos ajustar el tamaño y moverla.

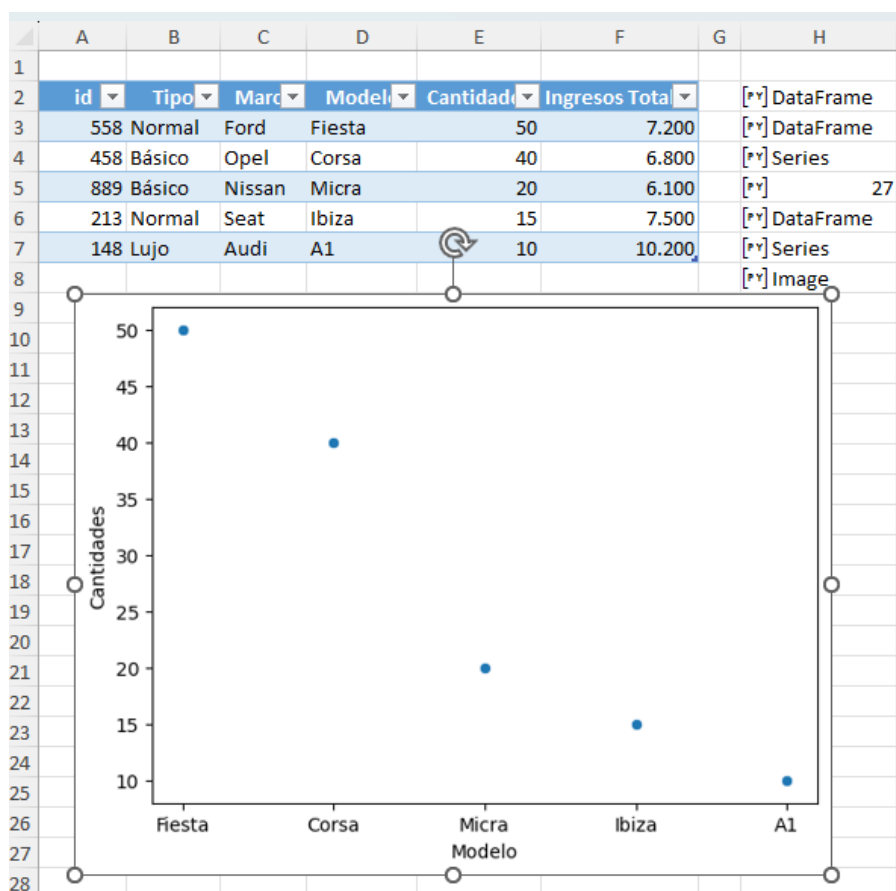


Ilustración 33

### Usando la IA para obtener el código Python

Para trabajar en este caso, hemos creado en nuestro DataFrame una nueva columna que hemos llamado Tipo, haciendo referencia al coche si es Básico, Normal o Lujo.

|   | A   | B      | C      | D      | E        | F                | G | H             |
|---|-----|--------|--------|--------|----------|------------------|---|---------------|
| 1 |     |        |        |        |          |                  |   |               |
| 2 | id  | Tipo   | Marca  | Modelo | Cantidad | Ingresos Totales |   | [v] DataFrame |
| 3 | 558 | Normal | Ford   | Fiesta | 50       | 7.200            |   | [v] DataFrame |
| 4 | 458 | Básico | Opel   | Corsa  | 40       | 6.800            |   | [v] Series    |
| 5 | 889 | Básico | Nissan | Micra  | 20       | 6.100            |   | [v] 27        |
| 6 | 213 | Normal | Seat   | Ibiza  | 15       | 7.500            |   | [v] DataFrame |
| 7 | 148 | Lujo   | Audi   | A1     | 10       | 10.200           |   |               |
| 8 |     |        |        |        |          |                  |   |               |

Ilustración 34

El objetivo es sumar las cantidades por tipo y ordenarlos de mayor a menor, para conocer que cantidad de coches vendo mas por categoría.

Vamos a preguntarlo a la IA para que nos oriente en el código.

En concreto preguntamos lo siguiente (Prompt):

*Dispongo en Excel de una tabla llamada MisDatos que contiene una columna Tipo y una Cantidades, como podría en una celda de Excel con Python obtener el total de las cantidades por Tipo y ordenarlos de mayor a menor. Necesito el código*

Vamos a ver la propuesta de la consulta a la IA

Mientras tanto, aquí tienes un ejemplo de código en Python que puedes usar con pandas para lograr lo que necesitas:

```

1 import pandas as pd
2
3 # Cargar el archivo Excel
4 df = pd.read_excel("ruta/al/archivo.xlsx", sheet_name="MisDatos")
5
6 # Agrupar por 'Tipo', sumar 'Cantidades' y ordenar de mayor a menor
7 resultado = df.groupby("Tipo")["Cantidades"].sum().sort_values(ascending=False)
8
9 # Mostrar el resultado
10 print(resultado)
11

```

Este código:

1. Carga el archivo Excel.
2. Agrupa los datos por la columna Tipo.
3. Suma las cantidades por cada tipo.
4. Ordena el resultado de mayor a menor.

¿Quieres que te ayude a adaptar este código a tu archivo específico? Puedes subirlo aquí.

Ilustración 35

*Fuente Copilot 365 La indicación [Dispongo en Excel de una tabla llamada MisDatos...](#) | [Probar en Copilot Chat](#)*

De todo lo anterior solo nos interesa la línea 7 porque el resto como importar las librerías y cargar archivo no lo necesitamos, por tanto, vamos a modificar y adaptar el fragmento de código propuesto.

```
resultado = df.groupby("Tipo")["Cantidades"].sum().sort_values(ascending=False)
```

*CantidadPorTipo = MisDatos.groupby("Tipo")["Cantidades"].sum().sort\_values(ascending=False)*

H7 : X ✓ [1] v py CantidadPorTipo= MisDatos.groupby("Tipo")["Cantidades"].sum().sort\_values(ascending=False)

|    | A   | B      | C      | D      | E        | F              | G | H             | I  | J | K | L | M |
|----|-----|--------|--------|--------|----------|----------------|---|---------------|----|---|---|---|---|
| 1  |     |        |        |        |          |                |   |               |    |   |   |   |   |
| 2  | id  | Tipo   | Marca  | Modelo | Cantidad | Ingresos Total |   | [*] DataFrame |    |   |   |   |   |
| 3  | 558 | Normal | Ford   | Fiesta | 50       | 7.200          |   | [*] DataFrame |    |   |   |   |   |
| 4  | 458 | Básico | Opel   | Corsa  | 40       | 6.800          |   | [*] Series    |    |   |   |   |   |
| 5  | 889 | Básico | Nissan | Micra  | 20       | 6.100          |   | [*]           | 27 |   |   |   |   |
| 6  | 213 | Normal | Seat   | Ibiza  | 15       | 7.500          |   | [*] DataFrame |    |   |   |   |   |
| 7  | 148 | Lujo   | Audi   | A1     | 10       | 10.200         |   | [*] Series    |    |   |   |   |   |
| 8  |     |        |        |        |          |                |   |               |    |   |   |   |   |
| 9  |     |        |        |        |          |                |   |               |    |   |   |   |   |
| 10 |     |        |        |        |          |                |   |               |    |   |   |   |   |
| 11 |     |        |        |        |          |                |   |               |    |   |   |   |   |
| 12 |     |        |        |        |          |                |   |               |    |   |   |   |   |
| 13 |     |        |        |        |          |                |   |               |    |   |   |   |   |
| 14 |     |        |        |        |          |                |   |               |    |   |   |   |   |
| 15 |     |        |        |        |          |                |   |               |    |   |   |   |   |
| 16 |     |        |        |        |          |                |   |               |    |   |   |   |   |
| 17 |     |        |        |        |          |                |   |               |    |   |   |   |   |

Series 3x1

| Tipo   | Cantidades |
|--------|------------|
| Normal | 65         |
| Básico | 60         |
| Lujo   | 10         |

ANACONDA

**Ilustración 36**

Así podemos observar como la aplicación de la IA en la codificación con Python es de gran ayuda para los iniciados en este lenguaje.