

1.2. Introducción a Python en GoogleColab para el diseño gráfico.

La integración de **Python en Google Colab** como herramienta para explorar la **generación computacional de geometría en Ingeniería Gráfica** proporciona un enfoque didáctico, contemporáneo y visual que favorece la motivación y el aprendizaje del alumnado. Para poder comenzar es necesario seguir los siguientes pasos iniciales.

Las ventajas de trabajar en este entorno es que, al trabajar en la nube, además que no necesitamos instalar el programa Python, siempre tendremos disponible el trabajo y ampliando así la accesibilidad a éstos.

1.2.1. Tener una cuenta de Google.

Si ya usas Gmail, YouTube o Drive, ya la tienes. Si no, crea una:

<https://accounts.google.com>

1.2.2. Acceder a Google Colab.

Ve a: <https://colab.research.google.com>

1.2.3. Crear un nuevo notebook.

- Haz clic en "Archivo" → "Nuevo cuaderno" (o "New notebook").
- Se abrirá una nueva pestaña con una celda lista para escribir código Python.

1.2.4. Escribir y ejecutar código.

Para comenzar a escribir código el Google Colab, por defecto aparece un cuaderno en blanco. Podríamos abrir un cuaderno que ya tengamos en nuestro Drive, crear un nuevo cuaderno o subirlo desde nuestro PC si no necesitamos (Archivo → Cuaderno nuevo en Drive o Abrir cuaderno o Subir cuaderno).

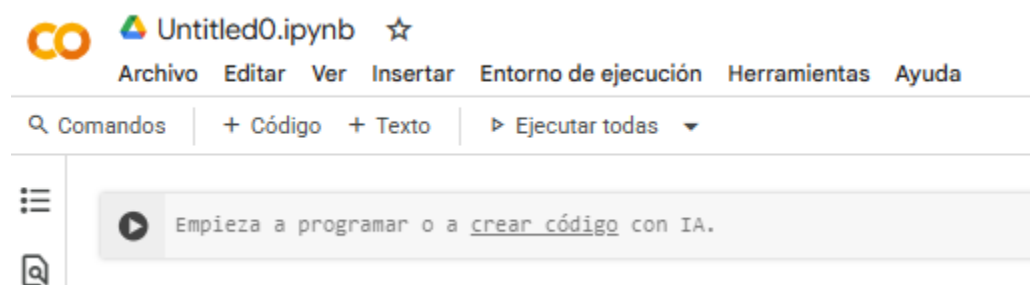


Figura 1. Cuaderno por defecto.

Para escribir los scripts, podemos ir agregando Código en **+ Código**:

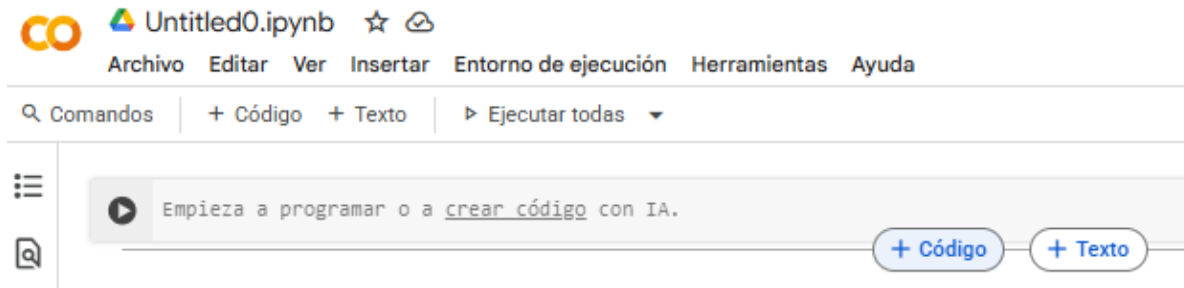


Figura 2. Agregamos código en + Código.

Prueba:

- Escribe código Python en la celda, por ejemplo 1.4.1 y 1.4.2:
- Pulsa **Shift + Enter** o haz *clic* en el botón  para ejecutar.

Ejercicio Resuelto 1.2.4.1. Salida por pantalla en Python.

```
print("Hola, Mundo")
```

Hola, Mundo

Ejercicio Resuelto 1.2.4.2. Salida por pantalla en Python con salto de línea \n.

```
print("salida", "de", "datos", "en", "Python", sep="\n") # si no se indica sep="\n"
se usa espacio en blanco como separador.
```

```
salida
de
datos
en
Python
```

Se recomienda tener la sesión iniciada en Google Drive para poder trabajar en ambos entornos, cargar archivos desde el drive a Google Colab.

Cómo cargar archivos desde Google Drive a Google Colab

1. Monta tu Google Drive en Colab.
2. Ejecuta esta celda al principio: Pulsa **Shift + Enter** o haz *clic* en el botón  para ejecutar.

Ejercicio 1.2.4.3. Monta tu Google Drive en Colab.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
from google.colab import drive
drive.mount('/content/drive')
```

Figura 3. Interfaz de Google Cola. Ejecuta esta celda al principio haciendo *clik* en el botón .

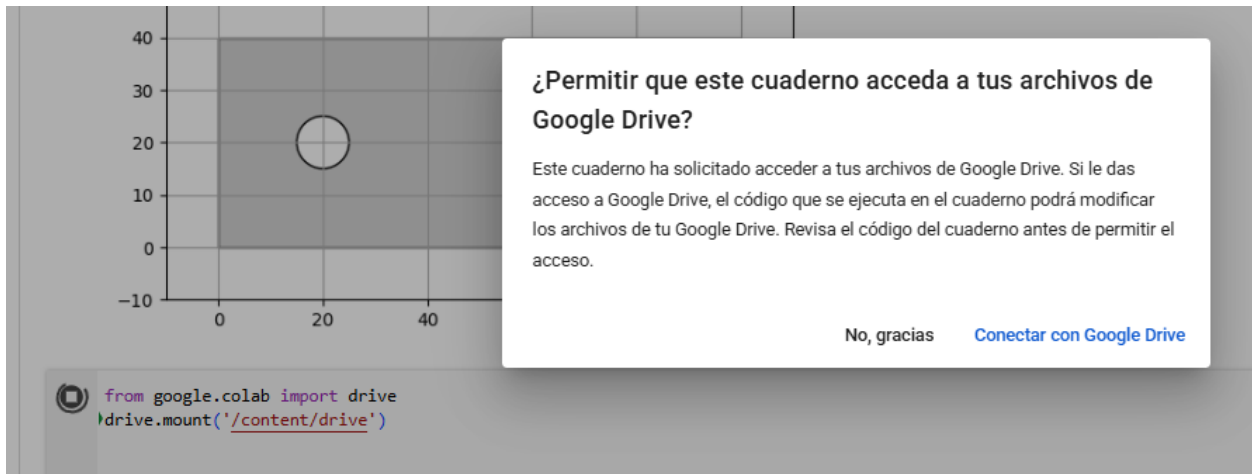


Figura 4. Permitir Conectar con Google Drive.

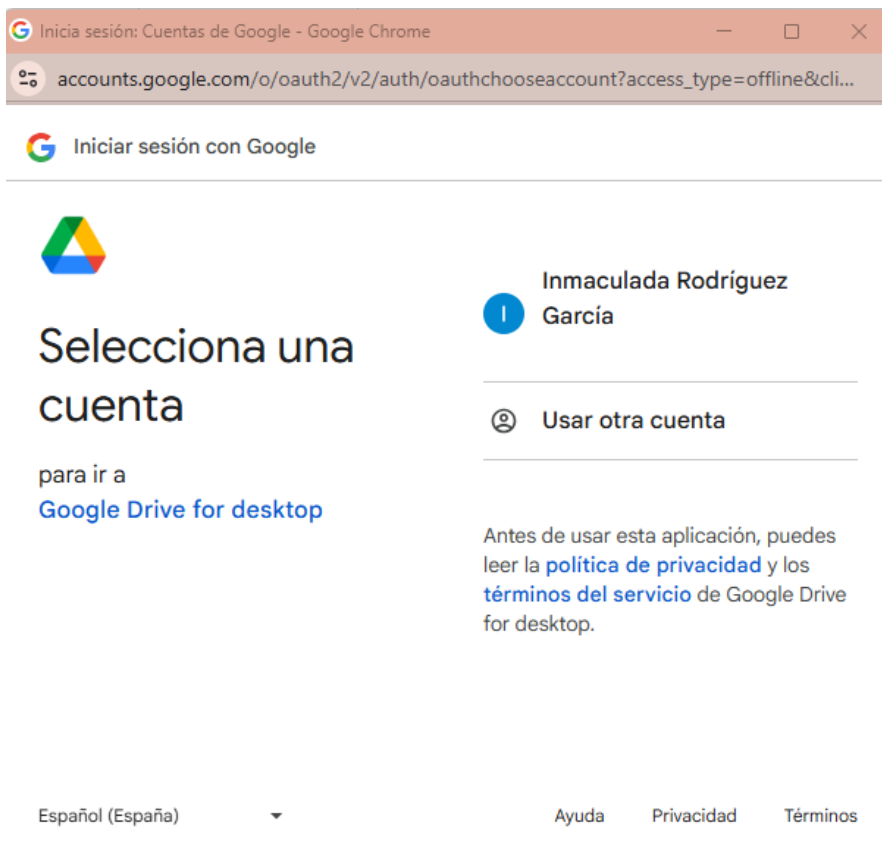


Figura 5. Seleccionar cuenta. **Continuar** → **continuar**

Accede a tus archivos

Tu Drive estará montado en:

`/content/drive/MyDrive/`

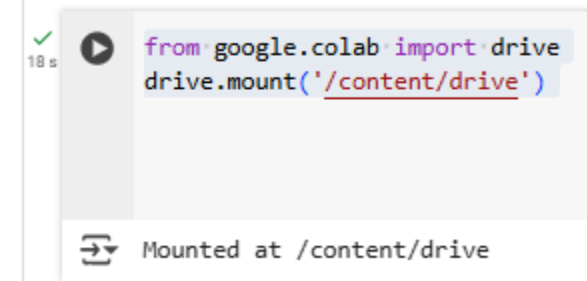


Figura 6. Drive montado en `/content/drive/MyDrive/`

Al ejecutarla:

- Se abrirá un enlace.
- Haz clic, autoriza con tu cuenta de Google (elígela, continuar, continuar). Puede que necesites un código de autorización, en ese caso pégalo en el cuadro que aparece en Colab.

Para acceder a tus archivos, por ejemplo, si tienes un archivo llamado **datos.csv** en la raíz de tu Drive necesitarás la siguiente instrucción:

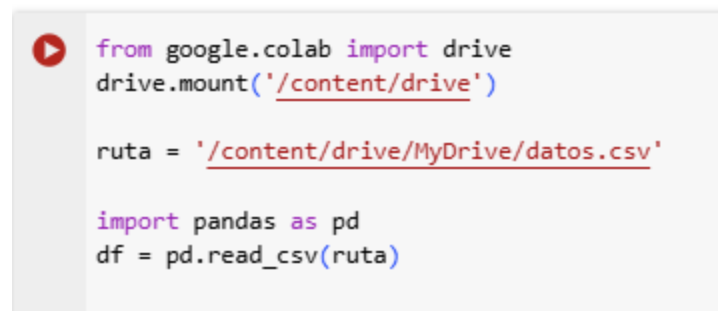


Figura 7. Creamos la ruta hacia el archivo (**datos.csv**) que queremos leer. En este caso no hemos creado una carpeta donde trabajar, sino que estamos trabajando en la carpeta general del Drive. Esto realmente no se aconseja, hay que crear una carpeta de trabajo.

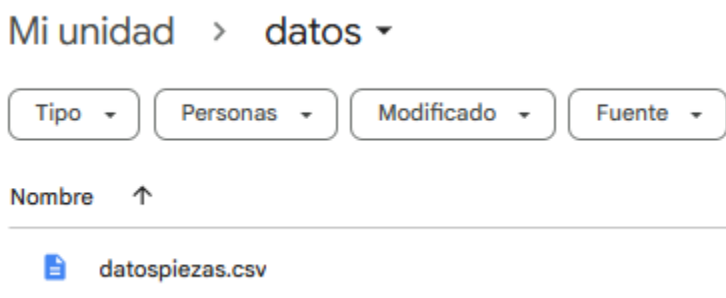


Figura 8. Añadimos archivo con los datos a nuestro Google Drive para poder leerlo. Conviene crear una carpeta dentro de nuestra unidad donde trabajaremos. En este caso la he llamado “datos” (es un *folder*) y dentro de ésta tendremos el archivo de **datospiezas.csv**.

Consejo:

Puedes navegar por tus carpetas con el explorador de archivos de Colab (icono de carpeta a la izquierda) y hacer clic derecho en cualquier archivo → Copiar ruta para pegarla en el código.

1.2.5. Guardar tu notebook.

- Google lo guarda automáticamente en tu Google Drive.
- Puedes cambiarle el nombre arriba donde dice *Untitled0.ipynb*.

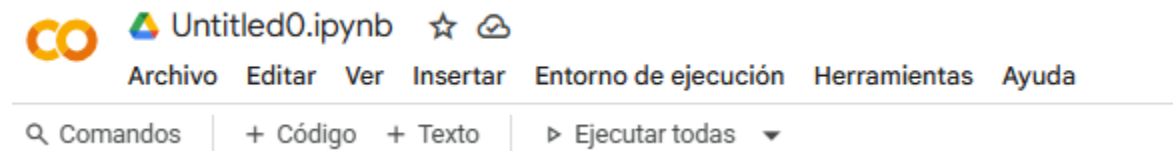


Figura 9. Nombre de archivo Python por defecto.

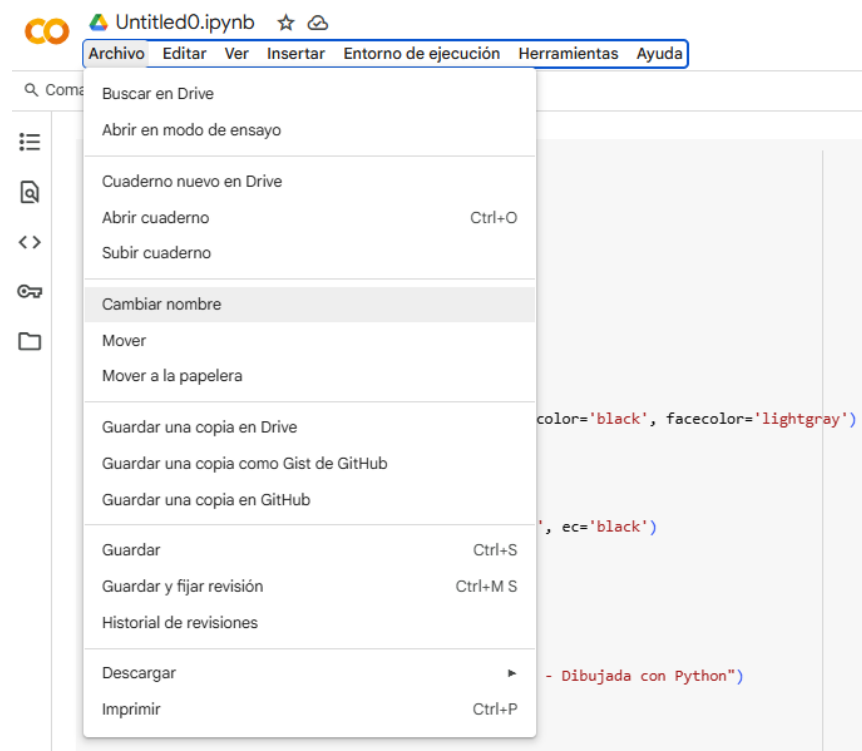


Figura 10. Cambiar nombre al archivo.

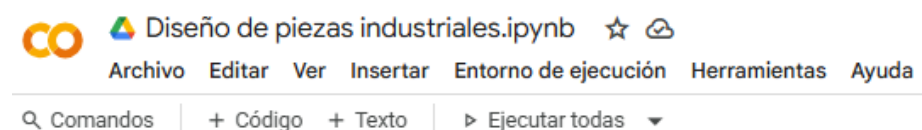


Figura 11. Nombre de archivo “Diseño de piezas industriales.ipynb”.

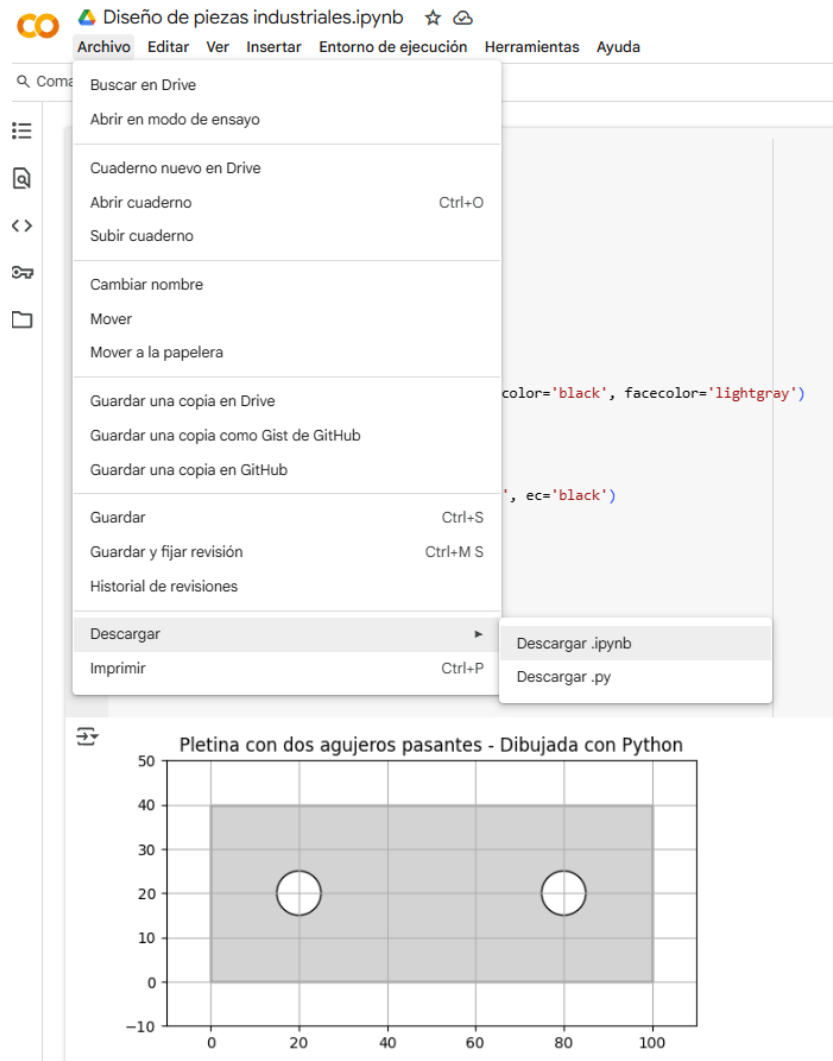


Figura 12. Descargar el archivo Python al PC personal (extensión *.ipnb*).

Guardar un archivo.

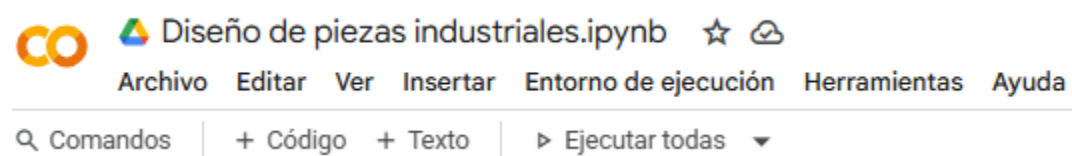


Figura 13. Cuando aparece el símbolo de la nube con un tick dentro, indica que el archivo ha sido guardado correctamente en la nube.

Para evitar perder el trabajo, se recomienda descargar el archivo en formato *.ipynb* para tener una copia de seguridad, incluso incorporarlo a nuestra carpeta de trabajo y cargarlo desde ahí si se nos hubiera cerrado.

Mi unidad > datos ▾

Tipo ▾
 Personas ▾
 Modificado ▾
 Fuente ▾

Nombre ↑	Propietario	Última modifi...	Tamaño de a	
 datospiezas (1).csv	 yo	18:19	218 bytes	⋮
 datospiezas.csv	 yo	18:05	186 bytes	⋮
 Diseño_de_piezas_industriales.ipynb	 yo	19:47	194 kB	⋮

Figura 14. Archivos dentro de la carpeta *datos* de trabajo.

Cuando tardamos mucho en trabajar sobre el cuaderno de GoogleColab éste se desconectará y nos pedirá volver a conectar.

```

# Importar librerías
import sklearn
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Datos simulados
# X = años de experiencia
X = np.array([[1], [2], [3], [4], [5]])
# y = ingreso en miles de dólares
y = np.array([30, 35, 40, 45, 50]) # regresión lineal perfecta
y = np.array([32, 38, 46, 49, 52])

# Crear el modelo de regresión lineal
modelo = LinearRegression()

# Entrenar el modelo
modelo.fit(X, y)

# Predecir con el modelo
y_pred = modelo.predict(X)

# Mostrar la línea ajustada
plt.scatter(X, y, color='blue', label='Datos reales')
plt.plot(X, y_pred, color='red', label='Línea de regresión')
plt.xlabel('Años de experiencia')
plt.ylabel('Ingreso (mil USD)')
plt.title('Regresión lineal con scikit-learn')
plt.legend()
plt.grid(True)
plt.show()

# Mostrar coeficientes
print(f"Pendiente: {modelo.coef_[0]}")
print(f"Intersección: {modelo.intercept_}")

```

Entorno de ejecución desconectado

El entorno de ejecución se ha desconectado debido a inactividad o porque ha alcanzado la duración máxima posible. [Más información](#)

Si te interesa disfrutar de tiempos de ejecución más largos y tiempos de inactividad más laxos, echa un vistazo a [Colab Pro](#).

Cerrar [Volver a conectar](#)



Figura 15. Entorno desconectado. Volver a conectar.

1.2.6. Importar librerías.

Puedes usar muchas librerías como *matplotlib*, *numpy*, *pandas*, *scikit-learn*, etc. directamente:

```

import numpy as np
import matplotlib.pyplot as plt

```

Si necesitas instalar una nueva librería en Jupyter o Google Colab:

```
!pip install nombre_librería
```

Ejemplo:

```
!pip install trimesh
```

```
!pip install sklearn
```

Si necesitas instalar una nueva librería en Python:

```
pip install ezdxf
```

```

Collecting ezdxf
  Downloading ezdxf-1.4.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (9.8 kB)
Requirement already satisfied: pyparsing>=2.0.1 in /usr/local/lib/python3.11/dist-packages (from ezdxf) (3.2.3)
Requirement already satisfied: typing_extensions>=4.6.0 in /usr/local/lib/python3.11/dist-packages (from ezdxf) (4.14.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from ezdxf) (2.0.2)
Requirement already satisfied: fonttools in /usr/local/lib/python3.11/dist-packages (from ezdxf) (4.59.0)
Downloading ezdxf-1.4.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (5.8 MB)
5.8/5.8 MB 46.8 MB/s eta 0:00:00
Installing collected packages: ezdxf
Successfully installed ezdxf-1.4.2

```

Figura 16. Al instalar la librería *ezdxf* aparecerá este mensaje.

Puedes instalar una nueva librería la primera vez, al inicio del cuaderno, y ya funcionará para el resto del código tan sólo con importarla (`import ezdxf`).

Ejercicio Resuelto 1.2.6.1. Importar librerías. Ejemplos

```

# Importar librerías
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import ast
import math
import sklearn

!pip install trimesh

```

¿Para qué se usa cada una de las librerías?

Librerías importadas

- ◆ *pandas* `as pd`
 - **¿Para qué se usa?:** Para leer, organizar y analizar datos en forma de tablas, como los archivos `.csv`.
 - **Ejemplo típico:** leer un fichero con dimensiones de piezas y convertirlo en una tabla de trabajo.

```
df = pd.read_csv('datospiezas.csv')
```

- ◆ `matplotlib.pyplot` as `plt`
 - **¿Para qué se usa?:** Para crear gráficos y dibujos en 2D, como el contorno de una pieza industrial.
 - **Ejemplo típico:** dibujar una pletina, un círculo (agujero), o una tuerca con `plt.Circle` y `plt.Rectangle`.
- ◆ `numpy` as `np`
 - **¿Para qué se usa?:** Para operaciones matemáticas y geométricas más avanzadas. Es muy útil para trabajar con ángulos, arrays y coordenadas.
 - **Ejemplo típico:** calcular los vértices de un octágono con `np.cos` y `np.sin`.
- ◆ `ast` (Abstract Syntax Trees)
 - **¿Para qué se usa?:** Para convertir cadenas de texto en estructuras de datos reales, como listas o diccionarios.
 - **Ejemplo típico:** leer de un archivo CSV una lista de coordenadas en formato texto `"(20, 10), (30, 20)"` y transformarla a `[(20, 10), (30, 20)]` con:

```
coords = ast.literal_eval("(20, 10), (30, 20)")
```

- ◆ `math`
 - **¿Para qué se usa?:** La librería `math` en Python se usa para realizar operaciones matemáticas avanzadas que no están disponibles con las operaciones básicas (+, -, *, /).
 - **Ejemplo típico:** Función usada: `math.sqrt()`

```
c = math.sqrt(a**2 - b**2)
```

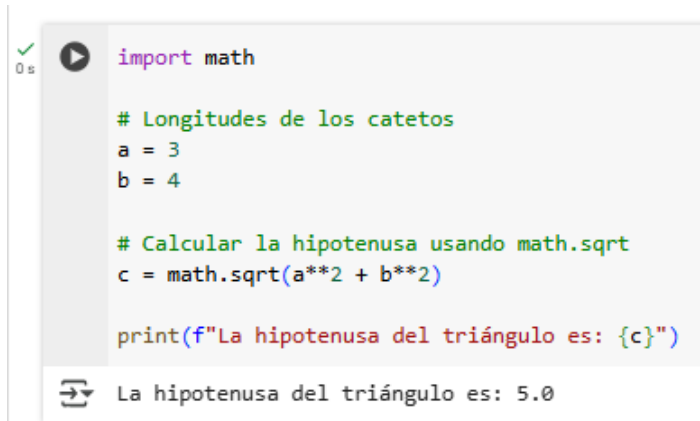
Ejercicio Resuelto 1.2.6.2. Ejemplo de uso librerías, primero se importa, después se puede usar.

```
import math

# Longitudes de los catetos
a = 3
b = 4

# Calcular la hipotenusa usando math.sqrt
c = math.sqrt(a**2 + b**2)

print(f"La hipotenusa del triángulo es: {c}")
```



```

import math

# Longitudes de los catetos
a = 3
b = 4

# Calcular la hipotenusa usando math.sqrt
c = math.sqrt(a**2 + b**2)

print(f"La hipotenusa del triángulo es: {c}")

```

La hipotenusa del triángulo es: 5.0

Figura 17. Código para importar librería *math* y salida esperada: “La hipotenusa del triángulo es: 5.0”.

◆ scikit-learn

- **¿Para qué se usa?:** La librería scikit-learn (también llamada sklearn para su importación) es una de las bibliotecas más populares en Python para *machine learning* (aprendizaje automático).
- **Ejemplo típico:** leer

Ejercicio Resuelto 1.2.6.3. Importar librerías. Ejemplo de uso scikit-learn

```

# Importar librerías
import sklearn
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Datos simulados
# X = años de experiencia
X = np.array([[1], [2], [3], [4], [5]])
# y = ingreso en miles de dólares
y = np.array([30, 35, 40, 45, 50]) # regresión lineal perfecta
y = np.array([32, 38, 46, 49, 52])

# Crear el modelo de regresión lineal
modelo = LinearRegression()

# Entrenar el modelo
modelo.fit(X, y)

# Predecir con el modelo
y_pred = modelo.predict(X)

```

```
# Mostrar la línea ajustada
plt.scatter(X, y, color='blue', label='Datos reales')
plt.plot(X, y_pred, color='red', label='Línea de regresión')
plt.xlabel('Años de experiencia')
plt.ylabel('Ingreso (mil USD)')
plt.title('Regresión lineal con scikit-learn')
plt.legend()
plt.grid(True)
plt.show()

# Mostrar coeficientes
print(f"Pendiente: {modelo.coef_[0]}")
print(f"Intersección: {modelo.intercept_}")
```

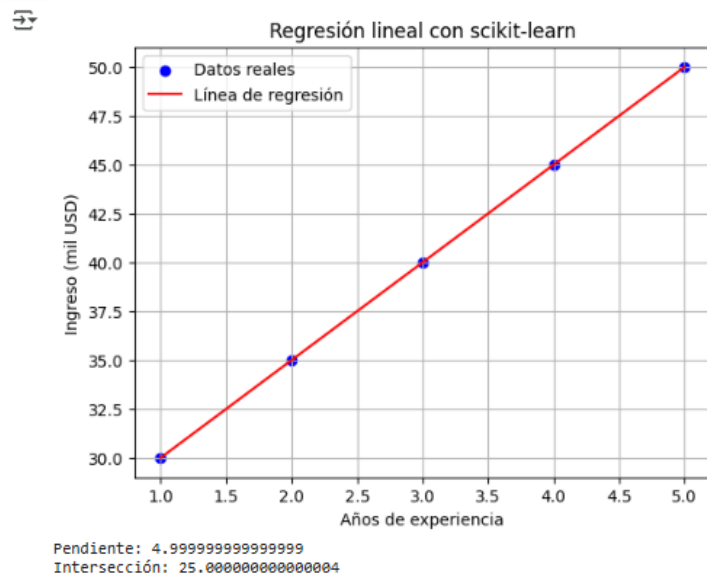


Figura 18. Usando librería scikit-learn. Nos ha dado una regresión lineal perfecta (en la realidad esto no suele suceder).

Ejercicio Propuesto 1.2.6.4.

- Cambiar los datos en las instrucciones para que la regresión lineal sea más realista (ejemplo en Figura 19).

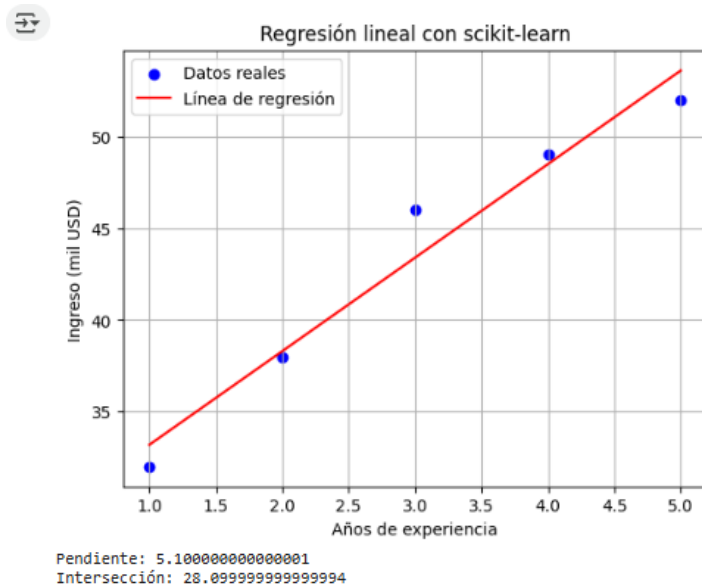


Figura 19. Usando librería scikit-learn. Aquí tenemos una salida un poco más realista.

Librería adicional: trimesh

`!pip install trimesh`

- ¿Para qué se usa? trimesh sirve para trabajar con geometría 3D, como mallas, modelos .STL o .OBJ, y escenas 3D.
- Importante en contextos de diseño industrial porque permite:
 - Crear y visualizar piezas en 3D
 - Exportar diseños a formatos imprimibles en 3D
 - Analizar volumen, área, normales y más propiedades geométricas

No se usa directamente para gráficos 2D, pero es esencial si se quiere llevar los diseños planos a modelos 3D o impresión 3D.

Ejemplo Básico:

```
import trimesh

# Cargar un modelo STL
mesh = trimesh.load('pieza.stl')

# Mostrar información de la malla
print(mesh)
print("Área superficial:", mesh.area)
print("Volumen:", mesh.volume)

# Visualización interactiva
mesh.show()
```

```
# Exportar a otro formato
mesh.export('pieza_convertida.obj')
```

1.2.7. Subir archivos.

Haz clic en el icono de carpeta (barra lateral izquierda) → botón "Subir".

1.2.8. Errores típicos.

Python incorpora una herramienta de IA que explica el error que aparece en pantalla.



```
from google.colab import drive
drive.mount('/content/drive')

ruta = '/content/drive/MyDrive/datos.csv'

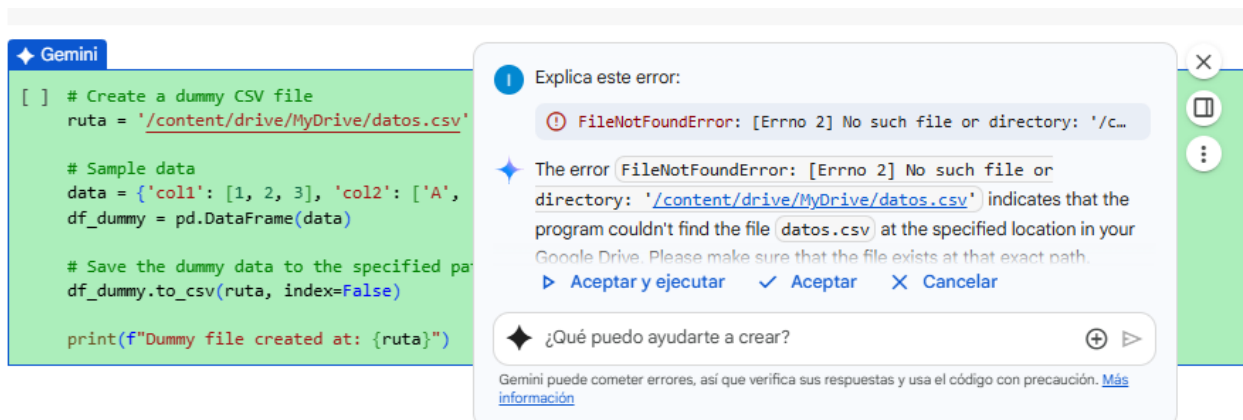
import pandas as pd
df = pd.read_csv(ruta)
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

FileNotFound: [Errno 2] No such file or directory: '/content/drive/MyDrive/datos.csv'

Pasos siguientes: [Explicar error](#)

Figura 20. Error debido a que aún no existe el archivo *datos.csv* en mi Drive.



```
[ ] # Create a dummy CSV file
    ruta = '/content/drive/MyDrive/datos.csv'

    # Sample data
    data = {'col1': [1, 2, 3], 'col2': ['A', 'B', 'C']}
    df_dummy = pd.DataFrame(data)

    # Save the dummy data to the specified path
    df_dummy.to_csv(ruta, index=False)

    print(f"Dummy file created at: {ruta}")
```

Explica este error:

FileNotFound: [Errno 2] No such file or directory: '/content/drive/MyDrive/datos.csv'

The error `FileNotFound: [Errno 2] No such file or directory: '/content/drive/MyDrive/datos.csv'` indicates that the program couldn't find the file `datos.csv` at the specified location in your Google Drive. Please make sure that the file exists at that exact path.

[Aceptar y ejecutar](#) [Aceptar](#) [Cancelar](#)

¿Qué puedo ayudarte a crear?

Gemini puede cometer errores, así que verifica sus respuestas y usa el código con precaución. [Más información](#)

Figura 21. Explicación del error. La IA te propone una solución para corregir el error.

Cómo poder corregir este error:

Paso 1: Guarda tu archivo en Google Drive

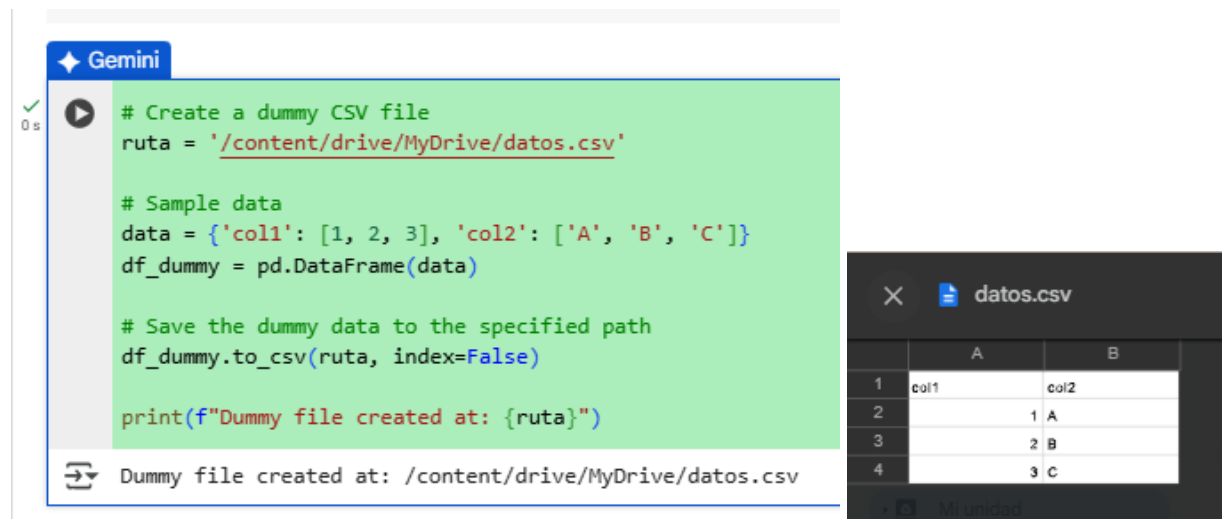
Primero, sube *datos.csv* a tu carpeta de Drive (por ejemplo, en la raíz o en una carpeta como Colab Notebooks).

Paso 2: Script completo en Colab

```
from google.colab import drive
drive.mount('/content/drive')

ruta = '/content/drive/MyDrive/datos.csv'

import pandas as pd
df = pd.read_csv(ruta)
```



The screenshot shows a Google Colab notebook interface. On the left, a Gemini AI chat window is open, displaying a suggestion to create a dummy CSV file. The suggested code is as follows:

```
# Create a dummy CSV file
ruta = '/content/drive/MyDrive/datos.csv'

# Sample data
data = {'col1': [1, 2, 3], 'col2': ['A', 'B', 'C']}
df_dummy = pd.DataFrame(data)

# Save the dummy data to the specified path
df_dummy.to_csv(ruta, index=False)

print(f"Dummy file created at: {ruta}")
```

Below the code, the AI indicates that the dummy file has been created at the specified path. On the right, a preview of the generated CSV file, named 'datos.csv', is shown. The file contains the following data:

	A	B
1	col1	col2
2		1 A
3		2 B
4		3 C

Figura 21. La IA te sugiere un cambio que si lo aceptas incluso te crea un archivo de muestra para corregir el error. Aparece en la carpeta “Mi unidad” de tu drive principal con el nombre que le indicaste en el *script*.

1.2.9. Compartir tu notebook.

Haz clic en el botón "Compartir" (arriba a la derecha), igual que en Google Docs. Puedes dar permisos de solo lectura o edición.