1. Breve introducción a Colab

Lo que estás leyendo es un ***Colab*** (*Colaboratory*, de Google). Un Colab es un entorno gratuito de ***Jupyter Notebook*** en el que podés almacenar, editar y ejecutar códigos completamente en la ***nube***. El lenguaje de programación elegido para este entorno es ***Python***.

El objetivo de esta breve introducción es que puedas empezar a utilizar Colab y sus herramientas. **No se pretende que aprendas exhaustivamente a programar**, pero sí que puedas reutilizar estos códigos como una herramienta para analizar tus resultados. Si te interesa introducirte en la programación en Python para laboratorios un excelente recurso creado por Marcelo Luda es https://marceluda.github.io/python-para-fisicos/tuto/labo2/.

1.1. Accediendo a Google Colab

Lo primero que necesitás para empezar es acceder a tu cuenta Google Colab a través de https://colab.research.google.com/. Si ya tenés una cuenta con Google, te pedirá autorización para acceder; si no tenés una, primero vas a necesitar crearte una.

Una vez que hayas accedido a tu cuenta de Google Colab vas a poder abrir, crear y editar archivos de extensión `.ipynb`. Estos archivos son Jupyter Notebooks, aunque para simplificar los podés llamar simplemente Colabs, y se almacenan en *Google Drive*.

Si abrís un Colab compartido por otra persona, **es importante que primero crees tu *copia del archivo en Drive***, ya que en caso contrario vas a estar haciendo modificaciones sobre el archivo original.

1.2. Editando un Colab

La estructura de tu Colab se dividirá en ***bloques de texto*** y ***bloques de código***. Los bloques de texto te permitirán editar texto, tanto tradicional como *LaTex*. Los bloques de código te permitirán crear, editar y ejecutar individualmente scripts de Python.

Un bloque de texto o código puede ser creado a través de `Insert` en el menú principal; o mismo yendo a la parte inferior de cualquier bloque y seleccionar `+ Code` o `+ Text`. Te sugerimos que explores un poco cómo crear y editar bloques.

Dentro de los bloques de texto vas a poder escribir en lenguaje matemático en formato LaTex. Para ello, se debe abrir y cerrar el texto en LaTex con `$` (o con `$$` si es en línea aparte). Por ejemplo, se puede escribir la ecuación $E = mc^2$ en una misma línea; u otra ecuación en línea aparte como

$$f(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i \omega x}.$$

A continuación ya tenés creado un *bloque de código*. El mismo contiene una única línea con la función `print()` y como *input* (entrada) una cadena de texto `'¡Hola, soy una cadena de texto!'`. Probá ejecutar el bloque y ver si se reproduce en el *output* (salida) el texto contenido.

```
print('¡Hola, soy una cadena de texto!')
```

Es importante que recuerdes que **la ejecución de bloques de códigos es secuencial**: si modificás un bloque de código con instrucciones que influyen más adelante, recordá ejecutar el mismo y los subsiguientes para que los cambios se reflejen.

1.3. Creando y ejecutando un código simple

Crear un código desde cero puede, en general, convertirse en una tarea muy complicada. Una forma de simplificar esto es ***importando*** una ***biblioteca*** (en inglés, *library*). Las bibliotecas son códigos ya desarrollados y testeados que te permiten implementar funcionalidades a tu código, por ejemplo: graficar, integrar numéricamente, u ordenar datos.

Dos de las bibliotecas más utilizadas son `NumPy` (funciones matemáticas) y `Matplotlib` (generación de gráficos). En el siguiente bloque se ejemplifica cómo importar estas dos bibliotecas: recordá que es necesario ejecutar el bloque para que esto impacte.

```
import numpy as np
import matplotlib.pyplot as plt
```

Para ejemplificar un código simple, suponé que querés computar y graficar la función $f(x) = \sin(\pi x^2 + \pi/2)$. Para ello, podés definir tu función y verificar (por ejemplo) cuánto vale para $x = 2$, del siguiente modo:

```
f = lambda x : np.sin(np.pi*x**2+np.pi/2)
print(f(2))
```

Siguiendo con el ejemplo, si quisieras graficar esta función, primero tendrías que definir un intervalo y una partición (discreta) de valores de x. Esto se debe a que el cómputo de tu función debe ser discreto, es decir, podés computar y graficar finitos valores de $\big(x, f(x)\big)$.

El bloque de código a continuación crea una partición de 1000 elementos para $x \in [0, 4]$ y grafica en dicho intervalo $\big(x, f(x)\big)$.

```
x = np.linspace(0, 4, 1000)
```

```
plt.plot(x,f(x))
```

1.4. Cómo continuar (o empezar)

Formar conocimientos de programación, aunque básicos, es fundamental para la educación en ciencias. En especial, en las ciencias naturales, gran parte del trabajo científico requiere tener una formación en programación. Esta formación no suele ser abordada a través de la educación formal con la solidez y actualidad necesarias. Por esto, es importante que, aunque en pequeños pasos, puedas involucrarte en aprender a programar en cada oportunidad que tengas.

Esta introducción no te va a enseñar a programar, ni menos aun pretende eso: es simplemente un puntapié inicial y sugestivo para que puedas conocer una herramienta que te permita abordar el estudio de las ciencia desde la computación.

En este camino, la mayor sugerencia es: **¡experimentá y jugá!** Hoy en día hay infinidad de recursos para que puedas aprender a programar y, por tanto, tu mayor aliado van a ser los **buscadores**. Ya sea que quieras encontrar algún código afín, conocer más sobre programación o buscar el mensaje de error de tu código; los buscadores van a ser tu *navaja suiza*.

¡Éxitos en tu camino!