

# Programación para Física y Astronomía

Departamento de Física.

---

Corodinadora: C Loyola

Profesoras/es C Loyola / C Femenías / Y Navarrete / C Ruiz / F Bugini

Primer Semestre 2025

Universidad Andrés Bello

Departamento de Física y Astronomía



Contexto e Introducción

Herramientas Principales

Fundamentos de Python

Actividad Práctica

Curiosidades y Recursos

Conclusiones

# Contexto e Introducción

---

- **Asignatura:** Programación para la Física y Astronomía.
- **Período:** 1er semestre (de acuerdo al Syllabus).
- **Objetivo principal de esta sesión:**
  - Presentación de Syllabus Oficial
  - Introducir las herramientas fundamentales del curso.
  - Familiarizarnos con el entorno de programación (Python, Google Colab).
- Corresponde a la **Sesión 1, Semana 1.**

# ¿Por qué Programar en Física y Astronomía?

- Muchas áreas de la Física y Astronomía requieren simulaciones y análisis de grandes volúmenes de datos.
- Python se ha vuelto esencial para:
  - Resolver problemas numéricos complejos.
  - Procesar y visualizar datos (observacionales o experimentales).
  - Facilitar la reproducibilidad de la investigación.
- Además, tiene una enorme comunidad científica activa.

# Breve Vista al Syllabus

- **Unidad I:** Elementos Básicos (GNU/Linux, Google Colab, Python).
- **Unidad II:** Programación en Python (tipos, aritmética, funciones).
- **Unidad III:** Controladores y arreglos (if, while, for, listas, slicing).
- **Unidad IV:** Gráficas con Matplotlib.
- **Unidad V:** Manejo de datos (clases, estadística, NumPy/Pandas).
- **Unidad VI:** Algoritmos y Performance (sorting, recursividad, hilos, etc.).

## Syllabus 2025

Vamos entonces a revisar el detalle del Syllabus de este período.

# Herramientas Principales

---

- Plataforma online gratuita de Google para programar en Python.
- **Ventajas:**
  - No requiere instalación local.
  - Integrada con Google Drive (colaboración sencilla).
  - Ejecución en la nube (libera recursos locales).
- Sólo necesitas una cuenta de Google.

**Nota:** Otras alternativas incluyen Jupyter, VSCode, etc.

## IMPORTANTE!!!

El que nuestro curso sea de programación, es **altamente recomendado** que cada estudiante tenga su propio cuaderno, en donde vamos a anotar cosas importante a medida que las clases avancen.



The screenshot shows the Google Colaboratory web interface. At the top, there's a header with the Colab logo, the text 'Welcome To Colaboratory', and a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the right of the header are buttons for 'Share', 'Settings', and a user profile icon. Below the header, a 'Table of contents' sidebar is visible on the left, listing links like 'Getting started', 'Data science', 'Machine learning', 'More Resources', 'Machine Learning Examples', and 'Section'. The main content area has a title 'What is Colaboratory?' with the Colab logo. Below the title, it explains that Colaboratory (or 'Colab') allows writing and executing Python in the browser. It lists three benefits: 'Zero configuration required', 'Free access to GPUs', and 'Easy sharing'. A paragraph follows, stating that Colab can make work easier for students, data scientists, and AI researchers, with a link to 'Introduction to Colab'. A section titled 'Getting started' explains that the document is an interactive 'Colab notebook' for writing and executing code. It provides an example of a 'code cell' containing a short Python script that calculates the number of seconds in a day. The code is shown in a light gray box with line numbers. Below the code, the output '86400' is displayed. A final paragraph explains how to execute the code by clicking a play button or using a keyboard shortcut, and how to edit the code by clicking the cell.

Welcome To Colaboratory

File Edit View Insert Runtime Tools Help

Share Settings User Profile

Table of contents

- Getting started
- Data science
- Machine learning
- More Resources
- Machine Learning Examples
- Section

## What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!

### Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] 1 seconds_in_a_day = 24 * 60 * 60
    2 seconds_in_a_day
```

86400

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

# Creación de un Notebook en Colab

1. Visita: <https://colab.research.google.com>
2. Inicia sesión con tu cuenta de Google.
3. Crea un nuevo cuaderno (*New Notebook*).
4. Almacena el archivo en tu Google Drive.

**Sugerencia:** Organiza tus carpetas en Drive para mantener un buen orden.

# Nuestra Primera Ejecución en Colab

```
1 print("¡Hola, mundo de la Física y Astronomía!")
```

- Presiona **Shift+Enter** o haz clic en el “play” para ejecutar.
- Observa el resultado inmediatamente.

# Fundamentos de Python

---

- **int** (enteros), **float** (reales), **str** (cadenas), **bool** (True/False).
- En Python, basta con asignar para crear una variable:

`x = 10` (int)

`saludo = "Hola"` (str)

- **No se requiere declaración previa** de tipo. Otros lenguajes sí, por ejemplo C, o C++.

- Usar nombres descriptivos (`masa_objeto`, `velocidad_inicial`, etc.).
- No comenzar con dígitos (`2variable` es inválido, `variable2` es válido).
- **Distinción de mayúsculas/minúsculas:** `Radio` vs `radio`.
- Evitar palabras reservadas (`if`, `else`, `class`, etc.).

# Operaciones Básicas con Python

```
1 a = 5
2 b = 2
3 suma      = a + b      # 7
4 resta     = a - b      # 3
5 producto  = a * b      # 10
6 division  = a / b      # 2.5 (float)
7 exponente = a ** b     # 25 (5^2)
8 modulo    = a % b      # 1 (resto de la división)
```

- / produce un resultado float.
- // realiza *división entera*.

Acá si ejecutamos no veremos los resultados, ya que para mostrar estos resultados en la pantalla, debemos utilizar la función **print** de python.

- `input()` para capturar información del usuario (devuelve `str`).
- `print()` para mostrar resultados en pantalla.

```
1 nombre = input("¿Cuál es tu nombre?: ")
2 print("Hola", nombre, "bienvenido/a al curso!")
```

Esta forma de la función `print` es hoy día mayoritariamente utilizada con *f-string*, lo que sería:

```
1 nombre = input("¿Cuál es tu nombre?: ")
2 print(f'Hola {nombre} bienvenido/a al curso!')
```

Para el curso, trataremos de utilizar *f-string*, aunque otras alternativas no están obsoletas, ni prohibidas.



## Ejemplo 1: Cálculo de Área de un Círculo

```
1 import math
2
3 r_str = input("Ingresa el radio del círculo: ")
4 r = float(r_str)
5 area = math.pi * (r**2)
6 print(f'El área del círculo es: {area}')
```

- `import math` habilita funciones y constantes matemáticas (ej. `math.pi`).
- **Observación:** Si se introduce un valor no numérico, el programa fallará (manejo de errores).

### Ejercicio

- Escribe un programa que pida la masa (kg) y la aceleración ( $\text{m/s}^2$ ).
- Calcula la fuerza resultante ( $F = m \times a$ ).
- Muestra el resultado en pantalla.

**Tip:** Recuerda convertir la cadena de `input()` a `float`.

# Solución Propuesta

```
1 m_str = input("Ingresa la masa (kg): ")
2 a_str = input("Ingresa la aceleración (m/s^2): ")
3
4 m = float(m_str)
5 a = float(a_str)
6 F = m * a
7
8 print(f'La fuerza resultante es: {F} N')
```

- Añadir unidades en la salida (ej.: “N” para Newtons).
- Discutir manejo de errores, validación de datos, etc.

## Actividad Práctica

---

# Problema 1: Suma de Enteros Consecutivos

## Enunciado

- Pedir un número entero  $n$ .
- Calcular  $\sum_{k=1}^n k$ .
- Mostrar el resultado final.

## Pistas:

- Usar un bucle o la fórmula  $\frac{n(n+1)}{2}$ .
- ¿Cambios si  $n$  es muy grande?

Claramente, no hemos revisado bucles, ni cosas similares ya que es nuestra primera clase, pero anímese e investigue online sobre distintas formas de atacar este problema. Más adelante vamos a ir conociendo más sobre python.

## Problema 2: Cálculo de Energía Cinética

### Enunciado

- Dada masa  $m$  y velocidad  $v$ , calcular  $E_c = \frac{1}{2}mv^2$ .
- Pedir  $m$  y  $v$  repetidamente.
- Detener cuando  $m = 0$ .

### Discusión:

- ¿Por qué usar **while**?
- ¿Cómo terminar el bucle de forma limpia?

## Indicaciones

- Organizarse en grupos de 2-3 personas.
- Revisen las soluciones de todos.
- Anoten dificultades o errores surgidos.
- Elaboren pequeñas conclusiones o dudas para la clase siguiente.

- ¿Problemas encontrados?
- ¿Qué fue lo más intuitivo / confuso?
- ¿Dudas para la próxima clase?

Mantenga estos datos en su cuaderno, le servirá para estudiar.



## Curiosidades y Recursos

---

- Creado por Guido van Rossum a finales de los 80.
- Nombre inspirado en el grupo de comedia *Monty Python*.
- Filosofía: legibilidad, sencillez y productividad.

- [python.org](https://python.org) - Documentación oficial.
- [Google Colab](https://colab.research.google.com/) - Entorno en la nube.
- [Real Python](https://realpython.com/) (sitio con tutoriales y guías).
- [Stack Overflow](https://stackoverflow.com/) (para consultas y dudas).

- **Stack Overflow:** millones de preguntas y respuestas.
- **Reddit /r/learnpython:** foros de principiantes.
- **GitHub:** proyectos y ejemplos de ciencia y Python.

**Tip:** Buscar soluciones o inspiración es parte del desarrollo como programador.

# Conclusiones

---

- Configuramos Google Colab y ejecutamos nuestro primer código en Python.
- Conocimos los tipos de datos y operaciones aritméticas básicas.
- Hicimos ejercicios de entrada/salida y ejemplos físicos sencillos.
- Exploramos recursos para continuar aprendiendo.

- **Unidad II:** Estructuras de control (`if`, `for`, `while`) y funciones en Python.
- **Tarea Sugerida:**
  - Practicar más ejercicios con `input()` y conversión de tipos.
  - Explorar `math`, `random`, etc.

# ¡Gracias y hasta la próxima sesión!

- Revisen la plataforma (Colab) para ejercicios adicionales.
- Traigan sus dudas la próxima clase.