

# Programación para Física y Astronomía

Departamento de Física.

---

Corodinadora: C Loyola

Profesores C Femenías / F Bugini / D Basantes

Primer Semestre 2025

Universidad Andrés Bello

Departamento de Física y Astronomía



Contexto del curso e Introducción

Herramientas Principales

Fundamentos de Python

Actividad Práctica

Curiosidades y Recursos

Conclusiones

## Contexto del curso 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.**

# Contexto del curso e Introducción $\ni$ ¿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.

- **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.



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.

## Paso 1: Acceso inicial a Google Colab

Te damos la bienvenida a Colab

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda

Q Comandos + Código + Texto Ejecutar todo Copiar en Drive

Compartir Gemini

RAM Disco

**Índice**

- Te damos la bienvenida a Colab
- Introducción**
- Ciencia de datos
- Aprendizaje automático
- Más recursos
- Ejemplos destacados
- + Sección

**Te damos la bienvenida a Colab**

**Explora la API de Gemini**

La API de Gemini te brinda acceso a los modelos de Gemini creados por Google DeepMind. Los modelos de Gemini se desarrollan desde un principio para ser multimodales, por lo que puedes razonar sin problemas en texto, imágenes, código y audio.

**Cómo comenzar**

- Ve a [Google AI Studio](#) y accede con tu Cuenta de Google.
- [Crea una clave de API](#)
- Usa una guía de inicio rápido de [Python](#) o llama a la API de REST con [curl](#).

**Descubre las funciones avanzadas de Gemini**

- Juega con los [resultados multimodales](#) de Gemini combinando texto e imágenes de forma iterativa.
- Descubre la [API de multimodal Live](#) (demostración [aquí](#)).
- Aprende a [analizar imágenes y detectar elementos en tus fotos](#) con Gemini (también hay una [versión en 3D](#)).
- Aprovecha el poder del [modelo de pensamiento de Gemini](#), capaz de resolver tareas complejas con sus pensamientos internos.

**Explora casos de uso complejos**

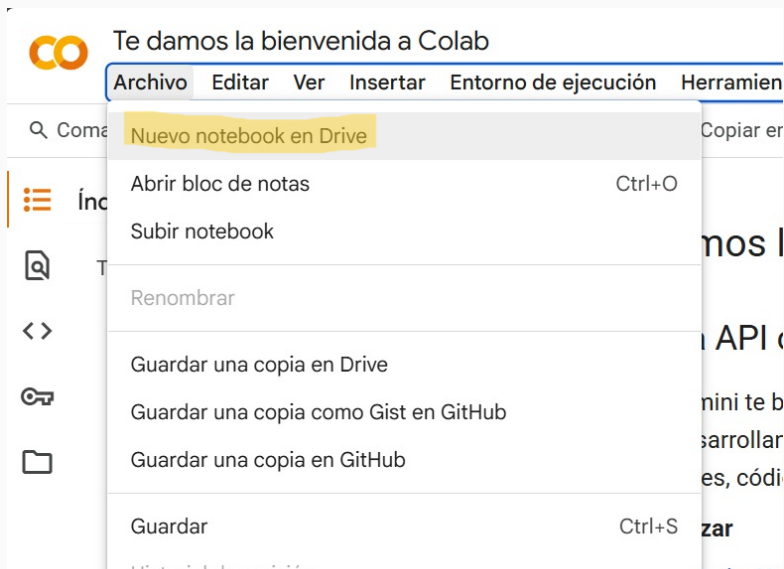
- Usa las [funciones de fundamentación de Gemini](#) para crear un informe sobre una empresa basado en lo que el modelo encuentra en Internet.
- Extrae [facturas y datos de formularios de archivos PDF](#) de forma estructurada.
- Usa Imagen y la ventana de contexto grande de Gemini para crear [ilustraciones basadas en un libro completo](#).

Para obtener más información, consulta la [guía de soluciones de Gemini](#) o visita la [documentación de la API de Gemini](#).

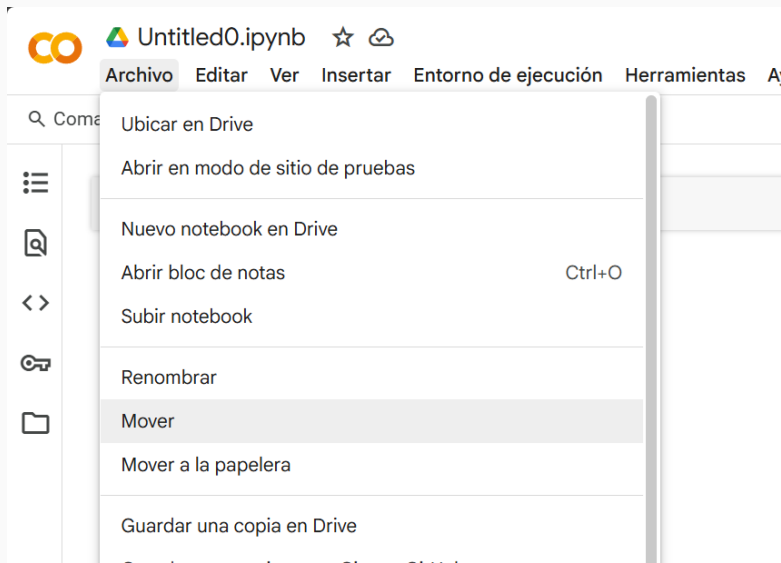
Ahora, Colab tiene funciones basadas en IA con la tecnología de [Gemini](#). En el siguiente video, se muestra cómo usar estas funciones, ya sea que estés dando tus primeros pasos con Python o tengas bastante experiencia.

Variables Terminal Python 3

## Paso 2: Creación de nuevo notebook



## Paso 3: Donde guardar el notebook



## Paso 4: Empezar a crear carpetas

### Selecciona una carpeta

← Colab Notebooks 

No hay resultados



[Cancelar](#) [Seleccionar carpeta](#)

## Paso 5: Crear carpeta con nombre del curso

### Selecciona una carpeta



No hay resultados



Cancelar

Seleccionar carpeta

## Paso 6: Seleccionar la carpeta creada

### Selecciona una carpeta

← Curso Programación 1 

No hay resultados



Cancelar

Seleccionar carpeta

## Paso 7: Darle nombre al notebook





---

```
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.).

```
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 division_entera = a // b # 2 (int)
8 exponente = a ** b     # 25 (5^2)
9 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.

## Fundamentos de Python $\ni$ 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`.





```
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

---

# Actividad Práctica $\ni$ Problema 1:

## Conversión de Unidades de Distancia



### Enunciado

- Pedir una distancia en metros al usuario.
- Convertir a kilómetros, centímetros y milímetros.
- Mostrar todos los resultados con sus unidades correspondientes.

### Pistas:

- Recordar las conversiones:  $1 \text{ m} = 0.001 \text{ km}$ ,  $1 \text{ m} = 100 \text{ cm}$ ,  $1 \text{ m} = 1000 \text{ mm}$ .
- Usar **f-strings** para mostrar resultados con formato claro.
- ¿Qué pasa si el usuario ingresa un número muy grande o muy pequeño?

**Ejemplo esperado:** Si ingresa 5 metros, debería mostrar:  $5 \text{ m} = 0.005 \text{ km} = 500 \text{ cm} = 5000 \text{ mm}$ .

# Actividad Práctica $\ni$ Problema 2:

## Calculadora de Velocidad



### Enunciado

- Pedir distancia recorrida (en metros) y tiempo empleado (en segundos).
- Calcular la velocidad usando  $v = \frac{d}{t}$ .
- Mostrar el resultado en m/s y como **bonus** convertir a km/h.

### Pistas:

- Para convertir de m/s a km/h, multiplicar por 3.6.
- Considerar qué pasa si el tiempo es cero (¡división por cero!).
- Incluir unidades en la salida para claridad.

**Física relevante:** Esta es una de las ecuaciones cinemáticas más fundamentales.

# Actividad Práctica $\ni$ Problema 3:

## Calculadora de Índice de Masa Corporal



### Enunciado

- Pedir peso (en kg) y altura (en metros) al usuario.
- Calcular el IMC usando la fórmula:  $IMC = \frac{\text{peso}}{\text{altura}^2}$ .
- Mostrar el resultado con interpretación básica.

### Pistas:

- Usar el operador **\*\*** para elevar al cuadrado.
- $IMC < 18.5$ : bajo peso,  $18.5-24.9$ : normal,  $25-29.9$ : sobrepeso,  $\geq 30$ : obesidad.
- ¿Cómo mostrar el resultado con 2 decimales?

**Aplicación:** Aunque no es estrictamente física, involucra cálculos matemáticos relevantes para ciencias de la salud.

## Indicaciones

- Trabajen en equipos de 2-3 personas.
- Cada equipo debe intentar resolver al menos los Problemas 1 y 2.
- El Problema 3 es opcional (para equipos que terminen rápido).
- Prueben sus programas con diferentes valores de entrada.
- Anoten cualquier error o comportamiento inesperado que encuentren.

**Tiempo sugerido:** 20-25 minutos para programar + 10 minutos para discusión grupal.

## Importante

¡No se preocupen por hacer el código "perfecto"! El objetivo es practicar lo aprendido y familiarizarse con la programación.

- ¿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** - Documentación oficial.
- **Google Colab** - Entorno en la nube.
- **Real Python** (sitio con tutoriales y guías).
- **Stack Overflow** (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.