



Argentina  
programa  
4.0

# Argentina programa 4.0

## Introducción – Programación en Python

Del 13 de Febrero al 19 de Abril.  
Lunes y Miércoles de 18 a 20 hs.

# Plan de Estudios – Programa

- El entorno y las variables: Diferentes entornos de programación Python (consola, IDE, notebooks). Sintaxis del lenguaje. Tipos de datos básicos. Funciones y su documentación.
- Estructuras de control: Condicionales. Iteraciones. Comprensión de listas. Recursión.
- Estructuras de datos: Diccionarios, listas, tuplas, vectores, matrices y árboles.
- Programación orientada a objetos: Concepto de objeto. Métodos. Herencia.
- Python para el análisis de datos: Archivos de entrada/salida. Cómputo de estadísticos. Regresión lineal.
- Visualización de datos. Aplicaciones con Numpy, SciPy y Matplotlib.
- Testeo y Debuggeo de programas: Diseño de experimentos. Manejos de excepciones. Control de flujos.
- Introducción a la complejidad de algoritmos: Concepto de complejidad. Algoritmos de búsqueda. Algoritmos de ordenamiento.
- Aplicaciones de la programación a diversos ámbitos: Negocios, finanzas, seguros, ciencia.

Más informaciones en <https://argentinaprograma.unsam.edu.ar/>

# Etapas de la programación

- El objetivo final de la programación es producir un programa. Para que el programa ejecute correctamente lo que desea, es necesario seguir algunos pasos:
- Comprender bien el problema, cuáles son los datos de entrada y qué se quiere obtener como resultado (salida).
- Desarrollar una forma de resolver el problema – Algoritmo. Esta forma debe ser lógica, clara y enlazada.
- Probar el algoritmo "en el papel", pensando en las diversas posibilidades de los datos de entrada y el flujo del programa.
- Transcribir el algoritmo a un lenguaje de programación, obedeciendo las reglas de sintaxis.
- Probar el programa hasta el agotamiento, con diferentes valores de entrada. Debuggear.

# Algoritmos

- Es una estrategia o secuencia de procedimientos que sirve para resolver un problema
- La resolución de problemas complicados hizo con que el estudio de los algoritmos se transforme en un campo importante de las matemáticas.
- En el fondo siempre creamos y seguimos un algoritmo para resolver problemas: presentarlo de forma explícita hace que sea mucho más fácil verificar si la solución es correcta.
- El concepto de algoritmo es fundamental para la computación.
- Las computadoras siguen órdenes y procesan operaciones lógicas.
- Las computadoras necesitan instrucciones claras y precisas.
- Puede existir más de un algoritmo posible para resolver un mismo problema
- Los algoritmos se pueden expresar en distintos lenguajes

# ¿Qué es un programa?

Un programa es un archivo de texto plano (.txt)

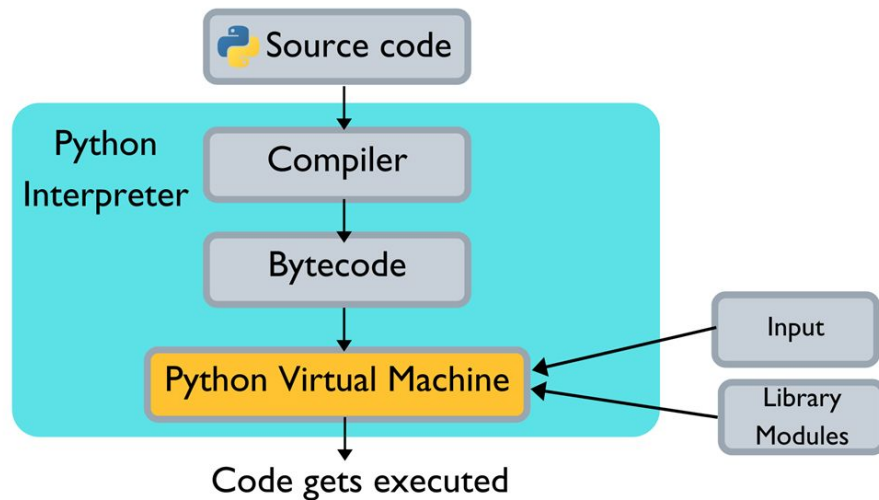
Que contiene instrucciones (código) en algún lenguaje (“**source code**”)

El código es **interpretado** y **ejecutado** por alguien que entiende el lenguaje.

Si el lenguaje es Python, el intérprete de Python (python.exe, python3, etc.) es el que se encarga de leer, interpretar (entender) y ejecutar (correr) el código.

El intérprete es un programa que debe estar ya instalado en la máquina (ej python3)

Para escribir un programa es necesario aprender las reglas, la sintaxis de cada lenguaje, para que las instrucciones que se pasan a la computadora sean precisas.



# ¿Qué es Python?

Python es un lenguaje de programación moderno, de propósito general, orientado a objetos y de alto nivel.

- Lenguaje limpio y simple: código intuitivo y fácil de leer, sintaxis minimalista fácil de aprender, la facilidad de mantenimiento se adapta bien al tamaño de los proyectos.
- Lenguaje expresivo: Menos líneas de código, menos errores, más fácil de mantener.

## Ventajas técnicas:

- No es necesario definir el tipo de variables, argumentos de función o tipos de devolución.
- No es necesario asignar y desasignar memoria explícitamente para variables y matrices de datos. No hay errores de pérdida de memoria.
- No es necesario compilar el código. El intérprete de Python lee y ejecuta el código de Python directamente.

# ¿Qué es Python?

Python no es solo un lenguaje de programación, sino que también se refiere a la implementación estándar del intérprete que realmente ejecuta el código de Python en una computadora.

También hay muchos entornos diferentes a través de los cuales se puede usar el intérprete de python. Cada entorno tiene diferentes ventajas y es adecuado para diferentes flujos de trabajo.

## Entornos:

- **Intérprete de Python:**

La forma estándar de usar el lenguaje de programación Python es usar el intérprete de Python para ejecutar el código de Python. El intérprete de python es un programa que lee y ejecuta el código de python en los archivos que se le pasan como argumentos. En el símbolo del sistema, el comando python se usa para invocar al intérprete de Python.

- **IPython**

Es un shell interactivo que aborda la limitación del intérprete estándar de python y es un caballo de batalla para el uso científico de python. Proporciona un indicador interactivo para el intérprete de python con una facilidad de uso muy mejorada.

- **Notebook**

Es un entorno de cuaderno basado en HTML para Python. Proporciona un entorno basado en celdas con gran interactividad, donde los cálculos se pueden organizar y documentar de forma estructurada. En nuestro curso vamos a utilizar este tipo de entorno, en particular Google Colab (<https://colab.research.google.com>)

# Entornos de programación Python

## Editor de Texto + Consola

Escribimos código en el editor (Notepad++, Sublime, Atom, etc)

programa.py

```
1 nombre = "Fernan"
2 apellido = "Agüero"
3 edad = 50
4 altura = 1.67
5
6 print(nombre, " ", apellido, " tiene ", edad, " años y mide ", altura, "\n")
```

Y ejecutamos programa.py en la consola (PowerShell, Terminal, etc)

```
[fernán@backuplex] python3 programa.py
Fernan  Agüero  tiene  50  años y mide  1.67
```

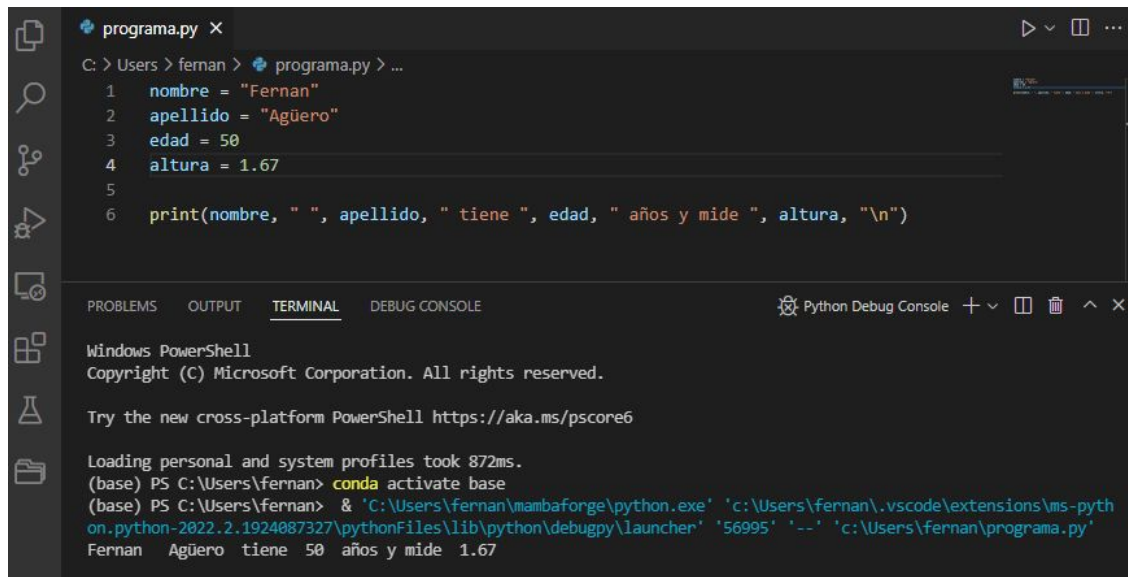


# Entornos de programación Python

## IDE (Integrated Development Environment)

- Varios disponibles: Microsoft Visual Studio Code (VSCode), PyCharm/JetBrains, etc.
  - <https://code.visualstudio.com/>
  - <https://www.jetbrains.com/es-es/pycharm/>

Escribimos código en el editor  
integrado dentro del IDE →



```
programa.py X
C: > Users > ferman > programa.py > ...
1 nombre = "Fernan"
2 apellido = "Agüero"
3 edad = 50
4 altura = 1.67
5
6 print(nombre, " ", apellido, " tiene ", edad, " años y mide ", altura, "\n")

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Python Debug Console + - □ ×
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

Loading personal and system profiles took 872ms.
(base) PS C:\Users\fernan> conda activate base
(base) PS C:\Users\fernan> & 'C:\Users\fernan\mambaforge\python.exe' 'c:\Users\fernan\.vscode\extensions\ms-python.python-2022.2.1924087327\pythonFiles\lib\python\debugpy\launcher' '56995' '--' 'c:\Users\fernan\programa.py'
Fernan Agüero tiene 50 años y mide 1.67
```

Ejecutamos el código en la consola  
integrada dentro del IDE →

# Entornos de programación Python

## Notebooks

- Varios disponibles: Jupyter Lab, Google Colab
  - En el curso vamos a usar Google Colab!
  - <https://research.google.com/colaboratory/>

Escribimos código en una celda dentro de la notebook



Ejecutamos el código de la celda

The screenshot shows the Google Colab interface. At the top, there's a header with the Colab logo, the name 'programa.ipynb', and a star icon. Below this is a menu bar with options: Archivo, Editar, Ver, Insertar, Entorno de ejecución, Herramientas, Ayuda, and a status message 'Se han guardado todos los...'. The main area has a sidebar on the left with icons for file explorer, search, and execution. The central pane shows a code cell with the following Python code:

```
nombre = "Fernan"
apellido = "Agüero"
edad = 50
altura = 1.67

print(nombre, " ", apellido, " tiene ", edad, " años y mide ", altura, "\n")
```

Below the code, the output is displayed: 'Fernan Agüero tiene 50 años y mide 1.67'.

# Sintaxis en Python

## Breve y rápido listado de conceptos

- Los comentarios se marcan con `#`
- El final de línea termina una instrucción
- Opcionalmente una instrucción puede terminarse con `;`
- Indentación: el espacio importa!
  - Delimita **bloques** de código
  - Está precedido de `:`
- Pero: dentro de una línea de código, el espacio no cuenta
- `x=1+2`
- `x = 1 + 2`
- Los paréntesis se usan para agrupar y para hacer llamadas

```
1  # fijar el medio
2  medio = 5
3
4  # hacer dos listas vacias
5  bajos = []; altos = []
6
7  # partir los numeros en bajos y altos
8  for i in range(10):
9      if (i < medio):
10         bajos.append(i)
11     else:
12         altos.append(i)
13
14  print("bajos:", bajos)
15  print("altos:", altos)
```