










Clase introducción a Curso Python para profesores 2026

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general creado por Guido van Rossum y lanzado en 1991.

Tiene una sintaxis clara y legible, una gran comunidad de usuarios, multiplataforma y una extensa librería de paquetes.

Está considerado ideal para iniciarse en la programación, pero pese a su facilidad de aprendizaje es muy utilizado en ciencia de datos, web, automatización, IA...

En cuanto a salidas profesionales, y si tenemos en cuenta el índice tiobe, python es el lenguaje de programación más popular (si tenemos en cuenta el número de ingenieros cualificados que lo utilizan, los cursos que se realizan). La estadística de julio de 2025:

Jun 2025	Jun 2024	Change	Programming Language		Ratings	Change
1	1			Python	25.87%	+10.48%
2	2			C++	10.68%	+0.65%
3	3			C	9.47%	+0.24%
4	4			Java	8.84%	+0.44%
5	5			C#	4.69%	-1.96%
6	6			JavaScript	3.21%	-0.11%
7	7			Go	2.28%	+0.35%
8	9	^		Visual Basic	2.20%	+0.54%
9	11	^		Delphi/Object Pascal	2.15%	+0.62%
10	10			Fortran	1.86%	+0.33%

¿Por qué es tan “amado”?

Bien, veamos algunos ejemplos de hola mundo en otros lenguajes de programación. Vamos a ir de “bajo nivel” a “alto nivel”

En el más bajo nivel nos encontramos con ensamblador. Este lenguaje nos permite controlar prácticamente todo lo que ocurre en nuestro programa (algunas excepciones como las llamadas al sistema y demás pero que no vienen al caso). Veamos un simple hola mundo para ensamblador de un intel de 64 bits (el que casi seguro tenéis todos en vuestras casas):

```

section .data
msg db "Hola Mundo", 0xA
len equ $ - msg

section .text
global _start

_start:
mov rax, 1 ;
mov rdi, 1 ;
mov rsi, msg ; dirección del mensaje
mov rdx, len ; longitud del mensaje
syscall ; llamada al sistema, en este caso para escribir en pantalla

mov rax, 60 ;
xor rdi, rdi ; código de salida: 0
syscall ; llamada al sistema para salir del programa

```

Para que este código funcione hay que pasarlo primero a código máquina, lo que se denomina compilación y enlazado, mediante los siguientes pasos, en una terminal:

```

nasm -f elf64 holamundoi686.s -o hola.o
ld hola.o -o hola

```

Esto generará un fichero hola que es ejecutable, pero únicamente en máquinas i686 y con sistema operativo Linux. Puedes descargar el fichero “hola” de la página de github del curso y ejecutarlo.

Pero si mi máquina es una Raspberry Pi que también funciona con Linux. Pues no funcionaría, ya que este miniordenador tiene otro tipo de procesadores, de la familia ARM, y el código es diferente:

```

.global _start

.section .data
msg:
.asciz "Hola Mundo\n"

.section .text
_start:
ldr r0, =1 @ stdout
ldr r1, =msg @ mensaje
ldr r2, =11 @ longitud
mov r7, #4 @ llamada al sistema para escribir en pantalla
svc #0

mov r7, #1 @ llamada al sistema para salir del programa
mov r0, #0
svc #0

```

Bien, os he convencido y queréis seguir trabajando con i686 pero esta vez en Windows. Pues mala suerte, este código no os va a servir y tiene que ser diferente.

Si lo intentáis con cualquier otro lenguaje “compilado” es fácil que tengáis que realizar cambios en vuestro código para que funcione en diferentes sistemas operativos y en hardware diferente.

Pero ¿y si lo hacemos con python?

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

Y si el fichero en el que has guardado el código se llama holamundo.py:

```
python3 holamundo.py
```

Y si quiero ejecutarlo en una Raspberry Pi:

```
python3 holamundo.py
```

¿Y en Windows?

```
python3 holamundo.py
```

¿Cuál es la trampa? La trampa es que python es un lenguaje interpretado, lo que quiere decir que el intérprete siempre tiene que ir con tu programa. Te va a bastar instalar python en el ordenador, o dispositivo que quieras, para que baste con ejecutar ese python3 holamundo.py o similar.

Es ese intérprete el que se encarga de hacer todo el trabajo sucio que antes hacíamos en ensamblador.

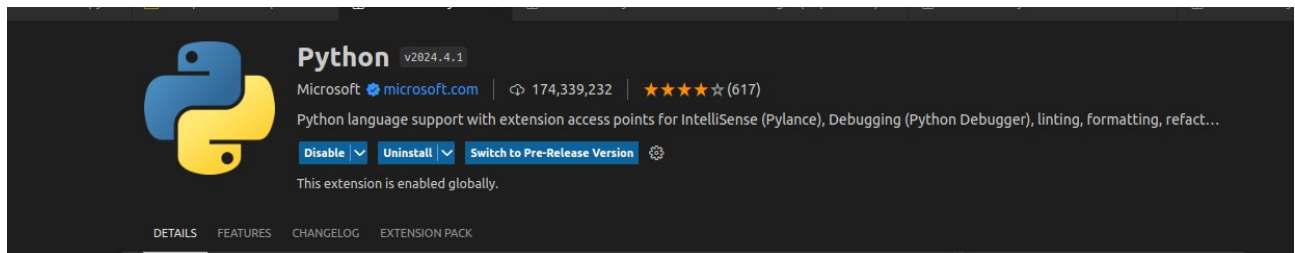
¿Cuál es el precio? Tiempo de ejecución. Podemos hacer comparativas con el comando time para observar la diferencia en tiempo de ejecución de los dos programas.

Software utilizado durante el curso

Vamos a utilizar Visual Studio Code como IDE (es opcional, puede utilizarse el editor que queramos)

La página de descarga del VisualStudioCode: <https://visualstudio.microsoft.com/es/>

Yo utilizo una extensión para python que facilita mucho la labor, sobretodo cuando el código empieza a tener un número importante de líneas:



Si estás trabajando con Linux no tienes que preocuparte por python3, ya está instalado.

Si vas a trabajar con Windows tendrás que descargar el intérprete de python desde la página python.org: <https://www.python.org/downloads/windows/>

Desde esa página te podrás descargar el instalador, no deja de ser un exe, seguir las instrucciones por defecto y, en principio, bastará.

Comprobamos el funcionamiento de nuestro entorno descargando el fichero primerprograma que encontrarás en github o en el aula virtual.

Ejercicio1:

Abrimos VisualStudioCode y abrimos una nueva carpeta. Llámala cómo quieras.

Ejercicio2:

Vamos a abrir un terminal, para eso tendremos que pulsar las teclas Ctrl-Shift-`

Observarás que se ha abierto un terminal justo debajo, si la configuración es la de por defecto, del fichero con el que estamos trabajando.

Ejercicio3:

En ese terminal teclea python3. Si todo ha ido bien, te mostrará algo similar a esto

```
Python 3.10.12 (main, May 27 2025, 17:12:29) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Justo después de los tres angulitos teclea 7 + 3 y pulsa enter
Para salir otra vez al terminal, teclea exit()

Ejercicio4:

En esa carpeta crea un fichero que se llame primerprograma.py (es importante que termine en .py ya que si esto no es así es fácil que te encuentres que la extensión de python no funcione)

Copia el código del fichero descargado y cópialo. (Ctrl-C y Ctrl-V)

Debes encontrar un código similar a este:

```
# Solicita el nombre al usuario
nombre = input("¿Cómo te llamas? ")

# Muestra un saludo personalizado
print(f"Hola, {nombre}!")

"""
if nombre == "Pedro"
    print("Pues el nombre que has introducido es Pedro")
    print("línea de más")
else:
    print("El nombre que has introducido no es Pedro")
    print("línea de más")
"""
```

Ejercicio5:

Vamos a guardar el fichero con los cambios realizados anteriormente, para eso pulsamos Ctrl-S

Ejercicio6:

Vamos a ejecutar el programa, para eso pinchamos en la terminal y tecleamos python3 primerprograma.py (o como hayas llamado al fichero)

Ejercicio7:

Elimina las líneas en las que aparecen las tres comillas, graba los cambios y vuelve a ejecutar el programa.

Ejercicio8:

En python las tabulaciones son muy importantes por lo que vamos a familiarizarnos con ellas.

- Añade un espacio justo delante de nombre. Graba los cambios y ejecuta el programa ¿qué ocurre?
- añade un carácter # delante de **else:**. Selecciona las dos líneas que hay justo debajo y mantén pulsadas las teclas Shift-Tabulador. Graba y ejecuta el programa
- Elimina el # de delante de else. Selecciona las dos líneas de abajo y pulsa la tecla Tabulador. Graba y ejecuta el programa
- En ocasiones nos va a ser interesante añadir, al inicio o al final de cada línea, los mismos caracteres. Para eso en VS Code podemos pulsar, a la vez, las teclas Alt+Shift, pinchamos el botón izquierdo del ratón para seleccionar las filas que queramos. Observaremos que el cursor aparece en todas esas líneas y si comenzamos a escribir, se repetirá el mismo texto en todas las líneas. Vamos a seleccionar las líneas que se encuentran debajo del else: y que comienzan con print para añadirles un # delante

e) Para eliminar esos caracteres basta con repetir la misma combinación de teclas pero ahora seleccionando todo el texto a eliminar.

Instalación de librerías de terceros

También vamos a instalar la librería (si hay dudas sobre lo que es una librería realizaremos previamente la ejecución del código de ejemplo sin la instalación) pygame. Para ello bastará con teclear desde un terminal **pip3 install pygame**

Para comprobar que se ha instalado correctamente abriremos un terminal, ejecutaremos python3 y teclearemos

```
import pygame
```

Aparecerá algo similar a esto:

```
>>> import pygame
pygame 2.6.1 (SDL 2.28.4, Python 3.10.12)
Hello from the pygame community. https://www.pygame.org/contribute.html
```