

3

ANTES DE EMPEZAR

Antes de poder trabajar con Python necesitamos algunas herramientas. La programación puede ser una labor muy compleja que requiera muchas herramientas pero, para comenzar a trabajar, nos bastará con el propio intérprete de Python y un editor de texto adecuado.

3.1. INSTALAR PYTHON

Como hemos visto, Python es un lenguaje interpretado por lo que, para poder ejecutar un programa escrito en este lenguaje, hace falta instalar un *intérprete*.

Instalar el intérprete de Python es muy simple y, en general y dependiendo del sistema operativo que estemos usando, se hace igual que cualquier otro programa.








3.1.1. Windows

El primer paso, naturalmente, es obtener el programa instalador. La forma más habitual para ello es descargarlo de Internet. Como ya hemos visto, Python es software libre, lo que significa que podemos descargarlo gratuitamente y usarlo sin tasas ni licencias.

La página oficial de descargas de la Python Software Foundation, donde se encuentra la última versión, es <https://www.python.org/downloads/>.



Looking for a specific release?
Python releases by version number:

Release version	Release date		Click for more
Python 3.13.2	Feb. 4, 2025	 Download	Release Notes
Python 3.12.9	Feb. 4, 2025	 Download	Release Notes
Python 3.12.8	Dec. 3, 2024	 Download	Release Notes
Python 3.13.1	Dec. 3, 2024	 Download	Release Notes
Python 3.11.11	Dec. 3, 2024	 Download	Release Notes
Python 3.10.16	Dec. 3, 2024	 Download	Release Notes
Python 3.9.11	Dec. 3, 2024	 Download	Release Notes

[View older releases](#)

Esta página detecta el sistema operativo que estamos usando, y nos muestra en un lugar destacado el enlace a la descarga adecuada para nuestro sistema. Si por alguna razón este enlace no se muestra, o si queremos descargar una versión distinta (una versión anterior o para otro sistema operativo), esa misma página dispone de un listado de enlaces a las descargas de otras versiones (ordenadas por fecha).

Si hemos decidido elegir nosotros mismos la versión del instalador, el enlace del listado nos dirigirá a la página de descargas, donde se encuentran las versiones para los distintos sistemas operativos. Los dos enlaces fundamentales para Windows son **Windows installer** para ordenadores de 32 bits, y **Windows installer** para máquinas más modernas de 64 bits.

Una vez descargado el instalador (que ocupa unos 20 MB) solo es necesario ejecutarlo y seguir las instrucciones, como cualquier otro programa de instalación.

El instalador nos permite configurar, si así lo deseamos, el directorio de instalación y los paquetes adicionales. Las opciones que da por defecto son perfectamente válidas y no es necesario cambiar nada.

NOTA

Para solucionar problemas o encontrar más información de la instalación y el uso de Python en Windows puede ser útil la documentación oficial:

<https://docs.python.org/3/using/windows.html>

3.1.2. Mac OS

A partir de **macOS 12.3 (Monterey, marzo de 2022)**, **Python 3 viene instalado por defecto**, ya que Apple eliminó la versión de Python 2 que anteriormente estaba preinstalada en macOS. Las opciones que da por defecto son perfectamente válidas y no es necesario cambiar nada.

NOTA

Para solucionar problemas o encontrar más información de la instalación y el uso de Python en Mac puede ser útil la documentación oficial:

<https://docs.python.org/3/using/mac.html>

3.1.3. Linux

Es importante destacar que, aunque muchas distribuciones modernas de Linux incluyen Python 3 por defecto, la versión específica puede variar. Además, algunas distribuciones pueden no establecer Python 3 como la versión predeterminada para el comando python. Por lo tanto, es recomendable verificar la versión instalada en su sistema y, si es necesario, instalar o actualizar a la versión deseada de Python 3.

3.2. UNA VEZ INSTALADO

El intérprete no es lo único que necesitaremos. Para escribir programas hace falta un editor de texto. Es importante distinguir un editor de texto, que es un programa para crear o modificar *texto plano* (esto es, sin formato ni características especiales) de un *procesador de textos* que genera documentos complejos, con diversos tipos de letra, títulos, negritas, etc.

Windows dispone del Bloc de notas, al igual que Mac tiene el TextEdit. Ambos permiten guardar documentos en texto plano (también llamado a veces *texto simple* o *texto normal*), que es lo que necesitamos para nuestros *scripts* en Python.

Lo malo es que son editores muy limitados.

Para qué un editor de texto sea una herramienta de programación cómoda, lo mínimo que necesitas son características como mostrar el número de línea, iluminar el código (poner automáticamente en distintos colores los diferentes elementos del lenguaje), utilidades para autocompletar el código mientras se escribe o para pegar *snippets* (pequeños pedazos de código predefinidos), etc.

En el wiki oficial de Python hay un extenso listado de editores para todas las plataformas y sus características (<https://wiki.python.org/moin/PythonEditors>).

Los editores más especializados, conocidos como IDE (*Integrated Development Environments*), disponen además de herramientas más avanzadas como control de versiones, explorador de archivos, *debugger* (para buscar y gestionar errores), gestor de proyectos, consola virtual, etc.

Aunque todo ello es a costa de ser más complejos de instalar, configurar y usar, y no se recomiendan para el usuario menos avanzado.

Actualmente, los dos entornos de desarrollo integrado (IDE) más conocidos y utilizados son:

1. Visual Studio Code (VS Code)

- **Descripción:** Un editor de código fuente ligero y altamente extensible desarrollado por Microsoft.
- **Características destacadas:**
 - Compatibilidad con múltiples lenguajes de programación mediante extensiones.
 - Integración con sistemas de control de versiones como Git.
 - Depuración integrada y terminal incorporada.
 - Amplia biblioteca de extensiones para personalizar y ampliar sus funcionalidades.
- **URL de descarga:** <https://code.visualstudio.com/>
- **Compatible con Windows, macOS y Linux**

2. IntelliJ IDEA

- **Descripción:** Un IDE robusto y completo desarrollado por JetBrains, especialmente popular entre desarrolladores de Java.
- **Características destacadas:**
 - Soporte avanzado para lenguajes de la JVM como Java, Kotlin y Scala.
 - Herramientas integradas para refactorización, navegación de código y análisis estático.
 - Integración con sistemas de construcción y control de versiones.
 - Soporte para desarrollo web y empresarial con una amplia gama de plugins.
- **URL de descarga:** <https://www.jetbrains.com/idea/download/>
- **Disponible en versión gratuita (Community) y de pago (Ultimate)**
- **Compatible con Windows, macOS y Linux.**

3.2.1. EL IDLE

Si no tenemos especial preferencia por ningún editor de texto, es posible que queramos usar el IDLE (<https://docs.python.org/3/library/idle.html>).

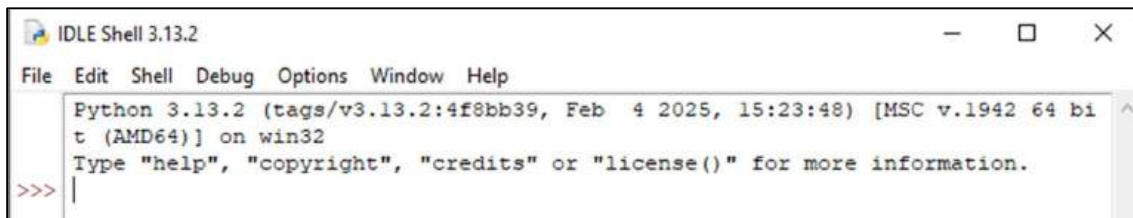
En Windows y Mac, al instalar Python, se añade automáticamente una aplicación llamada IDLE.

El IDLE (*Integrated Development and Learning Environment*) es un programa con interfaz gráfica (y escrito en Python) que contiene varias utilidades para Python, como un intérprete en tiempo real (en el que podemos escribir instrucciones de Python y ver sus resultados sobre la marcha), un editor de texto sencillo y algunas otras herramientas; y es probablemente la mejor manera de comenzar a usar Python.

Para iniciar el IDLE en sistemas Windows existe un icono en el menú inicio para ejecutar el IDLE.



Al iniciar el IDLE por primera vez se nos mostrará una ventana con un *shell* interactivo de Python (esto puede cambiarse en la configuración del programa, por medio de la opción **Configure IDLE** del menú **Options**).



En esta ventana se muestra información de nuestra versión de Python y se nos presenta un *prompt* consistente en tres símbolos de “mayor que” (>>>) en el que podemos escribir instrucciones de Python y ejecutarlas una a una.

Cada vez que se ejecuta una orden (pulsando la tecla **ENTER** tras escribirla) Python nos muestra el *retorno* de esta instrucción, el resultado de su ejecución.

El *shell* de Python del IDLE puede ser muy útil para ejecutar comandos y ver su resultado, pero lo que más usaremos será el editor de texto.

Para comenzar a escribir un nuevo programa, tenemos la opción **New File** del menú **File**. Esto nos abrirá la ventana del editor vacía. También podemos abrir una ventana conteniendo un programa ya existente con la opción **Open...**


```
Ejercicio_8_4.py - C:\Users\Admin\Desktop\Python\Pythonista\Tema_8_Listas\Ejercicio_8_4\Ejer...
File Edit Format Run Options Window Help

# Ejercicio 8.4

# PASOS:
# 1. Abre el archivo romeo.txt
# 2. Léelo línea por línea.
# 3. Para cada línea, divídala en una lista de palabras utilizando el
#    método split().
# 4. Para cada palabra de cada línea, verifica si la palabra ya está
#    en la lista y, si no, añádela a la lista.
# 5. Cuando el programa se complete, ordena e imprime las palabras resultantes
#    en el orden sort() de Python como se muestra en la salida deseada.

# El programa debería generar una lista de palabras.
# Resultado deseado:
# ['Arise', 'But', 'It', 'Juliet', 'Who', 'already', 'and', 'breaks', 'east',
# 'envious', 'fair', 'grief', 'is', 'kill', 'light', 'moon', 'pale', 'sick',
# 'soft', 'sun', 'the', 'through', 'what', 'window', 'with', 'yonder']

# Usa el archivo romeo.txt
fname = input("Introduce el nombre del archivo: ")
# Abro el archivo (no lee)
fh = open(fname)
# Inicio una lista vacía
lst = list()
# Leo cada una de las líneas del archivo
for line in fh:
    # Genero una lista con las palabras de cada línea
    lstline = line.rstrip().split()
    # Paso por cada palabra de la lista lstline
    for i in range(len(lstline)):
        # Si la palabra no está en la lst la añado al final
        if lstline[i] not in lst:
            lst.append(lstline[i])

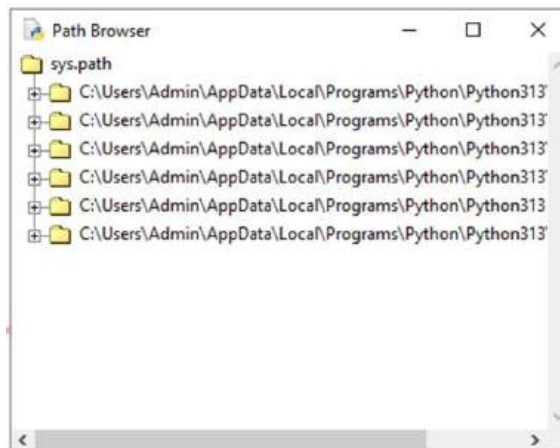
#Ordeno la lista con el método sort()
lst.sort()
# Imprimo la lista lst
print(lst)
```

La ventana del editor es muy similar a la de *shell*, aunque algunos menús de la barra superior cambian entre una y otra.

El primero de estos menús, común ambas ventanas, es **File**.

Este menú, como hemos visto, contiene opciones como **Open...**, que sirve para abrir un nuevo fichero (lo que, como veremos un poco más adelante, inicia el editor de textos del IDLE) y **Open Module...**, que abre un archivo ya existente. La opción **Recent Files**, por su parte, abre un desplegable con los últimos archivos que se han editado (la primera vez, lógicamente, no muestra nada) y también permite abrirlos.

El navegador de módulos **Module Browser** es una herramienta muy útil, ya que dispone de un buscador que nos permite abrir cualquier módulo de Python conociendo su nombre y nos lo muestra acompañado de una ventana con sus clases, funciones, etc. La opción **Path Browser** es similar, pero explorando un árbol de directorios en lugar de un buscador.



Los siguientes tres elementos del menú, **Save**, **Save As** y **Save Copy As**, permiten guardar el trabajo realizado, guardarlo asignándole un nombre y guardar una copia con un nombre distinto al que tiene.

Print imprimirá el contenido de la ventana y, por último **Close** y **Exit** cerrarán la ventana con lo que estamos trabajando y todas las ventanas abiertas, respectivamente.

El siguiente menú, también presente tanto en el Shell como en el editor, es **Edit**. Sus dos primeros elementos, **Undo** y **Redo**, sirven para deshacer la última acción y para volver a hacerla, respectivamente.

Cut y **Copy** sirven, respectivamente, para cortar un texto seleccionado y para copiarlo, mientras **Paste** sirve para pegar un texto previamente copiado en la posición actual del cursor. **Select All** selecciona todo el contenido de la ventana.

Find abre un menú de diálogo que nos permite buscar el texto que deseemos en la ventana. **Find Again** repite la última búsqueda realizada. **Find Selection** busca el texto que tengamos seleccionado en ese momento, y **Find in Files** permite buscar en múltiples archivos. **Replace** funciona igual que **Find**, pero permite reemplazar el texto buscado por otro.

Go to Line mueve automáticamente el cursor a la línea indicada.

Show completion muestra una lista con posibles opciones de funciones y elementos del lenguaje, para terminar el texto que se está escribiendo, para poder autocompletar este y agilizar la escritura de órdenes. Un efecto similar tiene pulsar la tecla **TAB** mientras está escribiendo, y es una de las herramientas más útiles que el programador puede encontrar en un editor de texto.

Expand Word tiene un efecto similar, pero trata de autocompletar a partir de palabras ya presentes en la ventana.

Show Call Tip muestra un pequeño texto contextual de ayuda que se despliega cuando el puntero está en los paréntesis de una función o método.

Show Surrounding Parens resalta el espacio entre paréntesis donde se encuentre el cursor.

El menú **Shell**, que solo se encuentra en la ventana del *shell* y no en la del editor, tiene dos opciones: **View Last Restart**, que hace *scroll* en la pantalla hacia retornar a la última vez que se reinició el *shell*, y **Restart Shell**, que reinicia el *shell* borrando las variables y estados que hubiese en memoria.

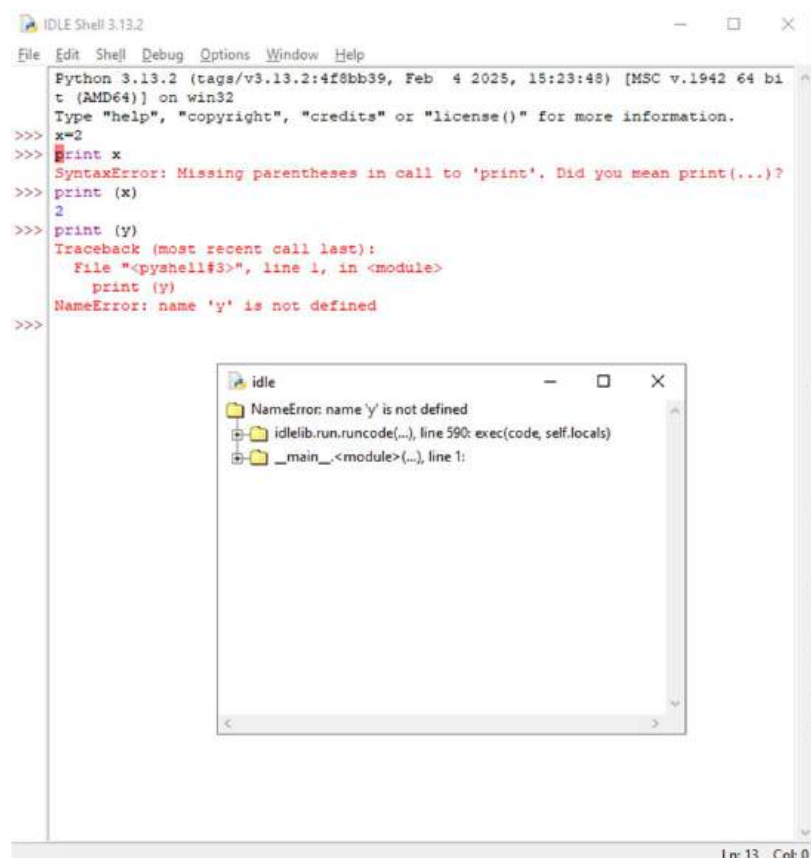
El menú **Debug** también es exclusivo de la ventana de *shell*, y permite controlar la ejecución de las instrucciones para depurar errores. El primer elemento, **Go to File/Line**, busca en torno al cursor un nombre de archivo y un número de línea y, si lo encuentra, abre una ventana del editor conteniendo ese archivo y mostrando resaltado el contenido de esa línea. El formato para indicar fichero y línea debe ser parecido a éste:

File "test.py", line 15

Este es el mismo formato en el que Python devuelve los mensajes de error, por lo que resulta muy cómodo para encontrar problemas y errores en nuestro código.

La opción **Debugger** abre una ventana en la que podemos ver en tiempo real, mientras ejecutamos instrucciones, el estado de las variables, los módulos cargados y otros parámetros de programa en ejecución.

Stack Viewer muestra un diagrama en árbol con el estado de la pila de excepciones. Este diagrama permite acceder a las variables globales y locales del programa en ejecución. **Auto Open Stack Viewer** hace que se abra **Stack Viewer** se abra automáticamente cuando haya algún error.



El menú **Format** aparece solo en el editor, y tiene herramientas para formatear el código de una forma más cómoda.

Indent Region indenta el código seleccionando un nivel, justo lo contrario de lo que hace **Dedent Region**.

Comment Out Region *comenta* un bloque de código añadiéndole los símbolos **##** delante, mientras que **Uncomment Region** quita los signos de comentarios del área seleccionada.

Tabify Region convierte el indentado con espacios en tabulaciones, y **Untabify Region** hace lo contrario, mientras que **Toggle Tabs** cambia el indentado por defecto entre espacios y tabuladores.

En Python se recomienda olvidarse del carácter *tabulador* y usar en su lugar cuatro espacios para tabular el código, por lo que esa es la configuración por defecto del editor. De todos modos, en el siguiente elemento del menú **New Indent Width**, se puede cambiar el número de espacios que el editor usa cada vez que se pulsa la tecla **TAB**.

Format Paragraph reformatea el párrafo actual para que se adecue al ancho estándar predefinido (72 caracteres de ancho, por defecto).

Por último, **Strip Trailing Whitespace** elimina los espacios en blanco sobrantes al final de las líneas.

El menú **Run**, que también es exclusivo del editor, tiene solo tres opciones. La primera de ellas es **Python Shell**, que abre una ventana de *shell*. La segunda es **Check Module**, que comprueba la sintaxis de programa que haya en el editor en ese momento, avisando si hay algún error.

La última es **Run Module**, que ejecuta programa en un *shell*. Cuando estemos programando y queramos probar si nuestro programa funciona correctamente, esta opción nos permitirá ejecutar el programa sin tener que abrir un terminal ni buscar donde sea que lo tengamos guardado.

En menú **Options** tiene dos elementos, **Configure IDLE** y **Configure Extensions**, Que sirven para cambiar la configuración por defecto de los distintos componentes del IDLE, tales como qué ventana se abre al inicio, la tipografía o los colores, aspectos del autocompletado y un amplio etc.

El menú **Windows** permite aumentar el tamaño de la ventana y, cuando hay más de una ventana abierta, pasar de una a otra cómodamente.

Y, por último, el menú **Help** provee de información legal sobre el propio IDLE por medio del elemento del menú **About IDLE**, una pequeña ayuda de uso en **IDLE Help** y acceso a la documentación *online* de Python en **Python Docs**.

Además de lo visto, el editor del IDLE ilumina el código, destacando con distintos colores cada elemento del lenguaje (estos colores también pueden

configurarse). Probablemente la herramienta más útil del IDLE es su capacidad de autocompletado. Sí, mientras estamos escribiendo una sentencia a un elemento del lenguaje, pulsamos la tecla **TAB**, el IDLE intentará completar la palabra. Si hay varias opciones posibles, se mostrará una lista con ellas.

Casi con total seguridad, el IDLE no es el mejor ni el más completo editor de código para Python. Pero es una herramienta muy útil y fácil de usar, que nos va a servir de mucha ayuda para aprender Python.

3.3. ¿QUÉ HEMOS VISTO EN ESTE TEMA?

- ✓ Dónde puede obtenerse Python.
- ✓ La forma de instalarlo dependiendo de nuestro sistema operativo.
- ✓ Qué más programas necesitamos para poder trabajar cómodamente con Python.
- ✓ El IDLE.

3.3.1. Tareas sugeridas

- ✓ Instalar Python.
- ✓ Instalar un editor de texto que nos resulte cómodo (el IDLE es una buena idea para comenzar).
- ✓ Explorar nuestro editor de texto para familiarizarnos con él.