

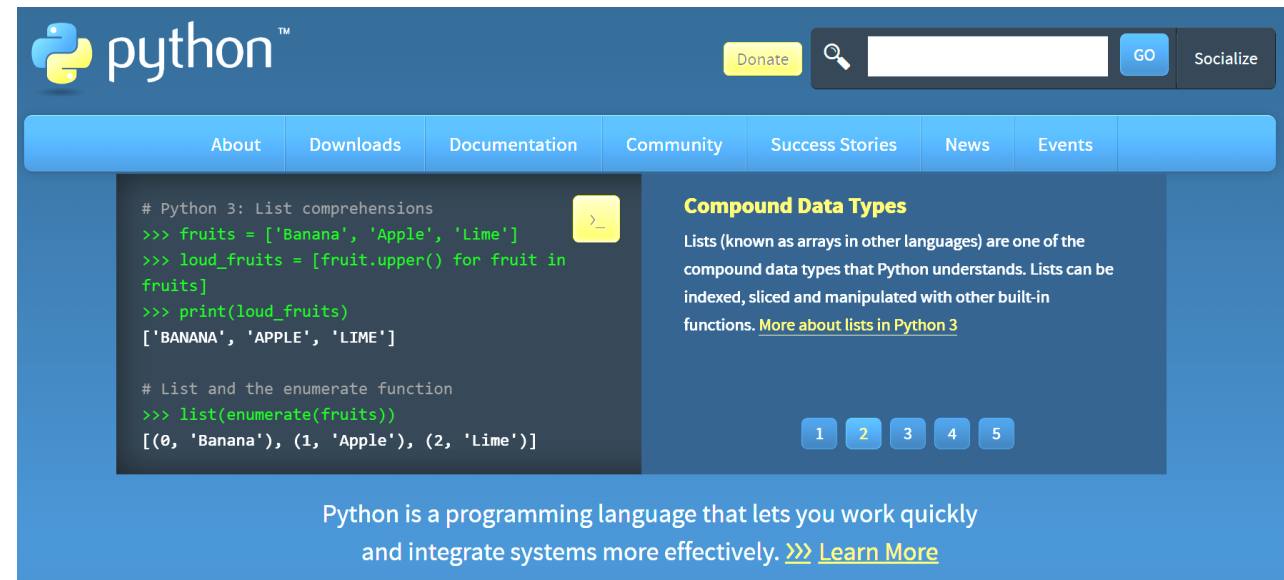
Fundamentos teóricos de programación

Joshua Martínez Domínguez

Python

Python es un lenguaje de programación de alto nivel, interpretado, e interactivo. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y programación funcional.

Administrado por Python Software Foundation, posee una licencia de código abierto.



Lenguaje de programación

Un lenguaje de programación es cualquier sistema de notación que permite expresar programas.

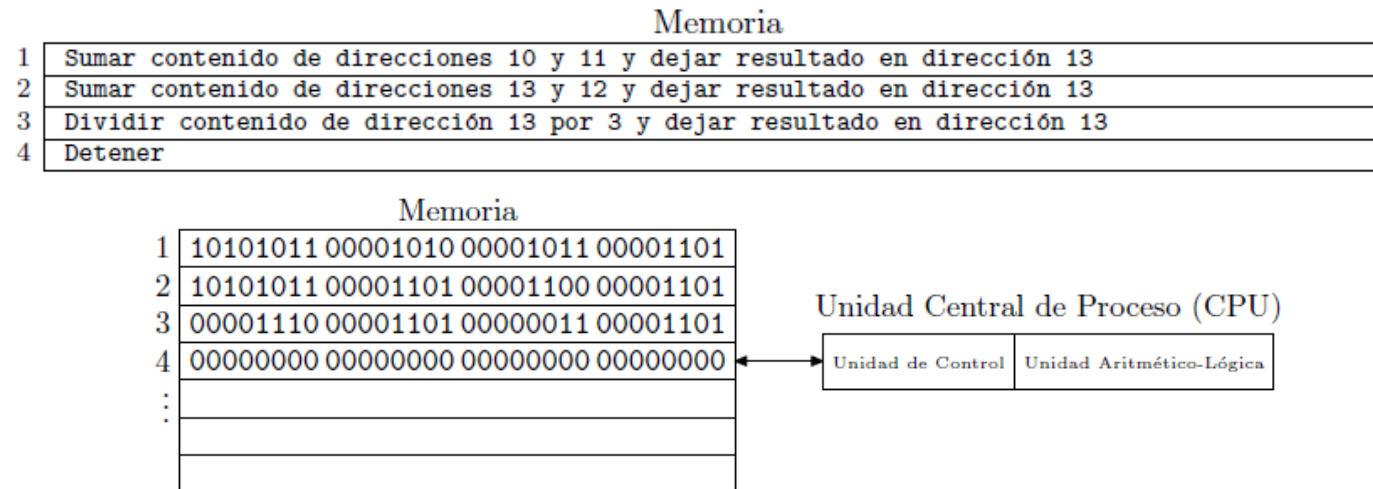
Programa

Es la secuencia de instrucciones que un ordenador o computador puede ejecutar.

Código de máquina

El código de máquina codifica las secuencias de instrucciones como sucesiones de unos y ceros que siguen ciertas reglas.

Cada familia de ordenadores dispone de su propio repertorio de instrucciones, es decir, de su propio código de máquina.



Lenguaje de bajo nivel

En los primeros tiempos de la informática los programas se introducían directamente en código de máquina, indicando uno por uno el valor de los bits de cada una de las posiciones de la memoria.

Pronto se diseñaron notaciones que simplificaban la programación: cada instrucción de código de máquina se representaba mediante un código mnemotécnico, es decir, una abreviatura fácilmente identificable con el propósito de la instrucción.

Lenguaje de bajo nivel

```
SUM #10, #11, #13  
SUM #13, #12, #13  
DIV #13, 3, #13  
FIN
```

Ejemplo de código de máquina: En este lenguaje la palabra SUM representa la instrucción de sumar, DIV la de dividir y FIN representa la instrucción que indica que debe finalizar la ejecución del programa.

La almohadilla # delante de un número indica que deseamos acceder al contenido de la posición de memoria cuya dirección es dicho número

Lenguaje de bajo nivel

El ensamblador es un programa traductor que lee el contenido de las direcciones de memoria en las que hemos almacenado códigos mnemotécnicos y escribe en otras posiciones de memoria sus instrucciones asociadas en código de máquina.

Los lenguajes ensambladores y los códigos de máquina se denominan lenguajes de programación de bajo nivel.

Lenguaje de alto nivel

Son lenguajes de programación que, sin ser tan potentes y expresivos como los lenguajes naturales, eliminan buena parte de la complejidad propia de los lenguajes ensambladores y están bien adaptados al tipo de problemas que podemos resolver con las computadoras.

El calificativo “de alto nivel” señala su independencia de un ordenador concreto.

Lenguaje de alto nivel

```
a = 5  
b = 10  
c = 6  
media = (a + b + c) / 3
```

Ejemplo de lenguaje de alto nivel, Python: Las primeras tres líneas definen valores y la cuarta calcula la media. Como puedes ver, resulta mucho mas legible que un programa en código de máquina o en un lenguaje ensamblador.

Para cada lenguaje de alto nivel y para cada CPU se puede escribir un programa que se encargue de traducir las instrucciones del lenguaje de alto nivel a instrucciones de código de máquina, con lo que se consigue la deseada independencia de los programas con respecto a los diferentes sistemas computadores.

Hay dos tipos de traductores dependiendo su funcionamiento: compiladores e intérpretes

Compiladores

Un compilador lee completamente un programa en un lenguaje de alto nivel y lo traduce en su integridad a un programa de código de máquina equivalente.

El programa de código de máquina resultante se puede ejecutar cuantas veces se desee, sin necesidad de volver a traducir el programa original.

Intérpretes

Un intérprete lee un programa escrito en lenguaje de alto nivel instrucción a instrucción y, para cada una de ellas, efectúa una traducción a las instrucciones de código de máquina equivalentes y las ejecuta inmediatamente.

No hay un proceso de traducción separado por completo del de ejecución.

Cada vez que ejecutamos el programa con un intérprete, se repite el proceso de traducción y ejecución, ya que ambos son simultáneos.

Compiladores e interpretes

Por regla general, los intérpretes ejecutarán los programas más lentamente. Un programa interpretado suele ser mucho más lento que otro equivalente que haya compilado, entre 2 y 100 veces mas lento.

Los intérpretes permiten una mayor flexibilidad que los compiladores y ciertos lenguajes de programación han sido diseñados para explotar esa flexibilidad (Python, Java). Otros lenguajes sacrifican esa flexibilidad en aras de una mayor velocidad de ejecución (C, C++, Fortran).

Programa de python

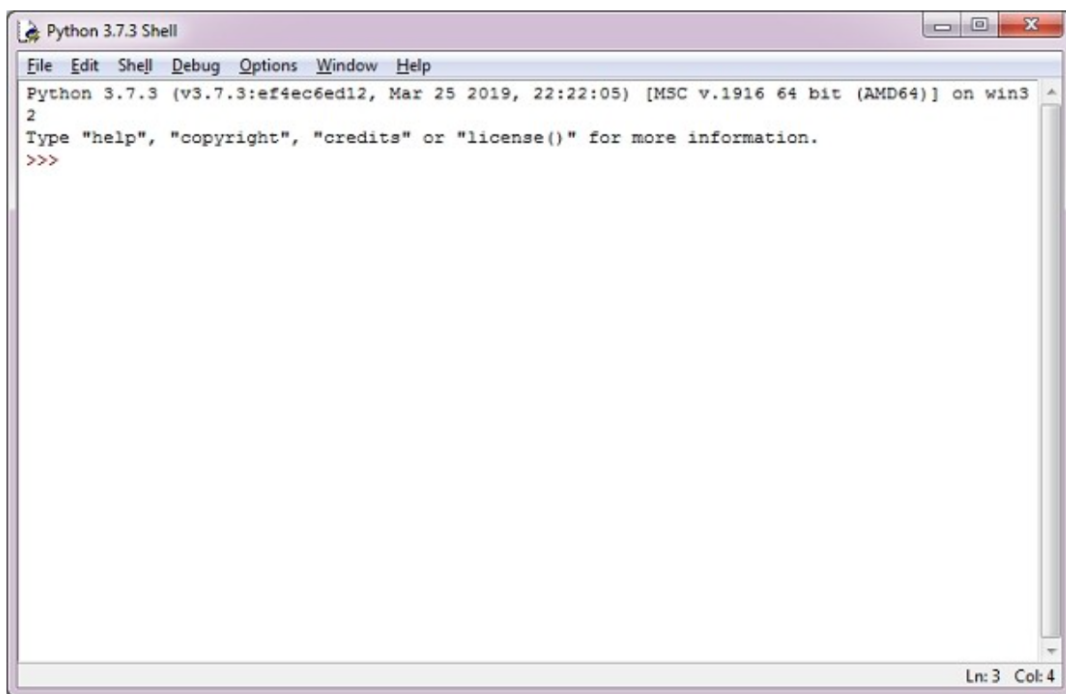
Es una secuencia de definiciones y comandos que son ejecutados por un intérprete de Python en el Shell.

El Shell es la capa más externa del sistema operativo.

IDE

Escribir programas directamente en el Shell es inconveniente. La mayoría de los programadores prefieren usar una clase de editor de texto que es parte de un entorno de desarrollo integrado (integrated development environment, IDE).

IDLE

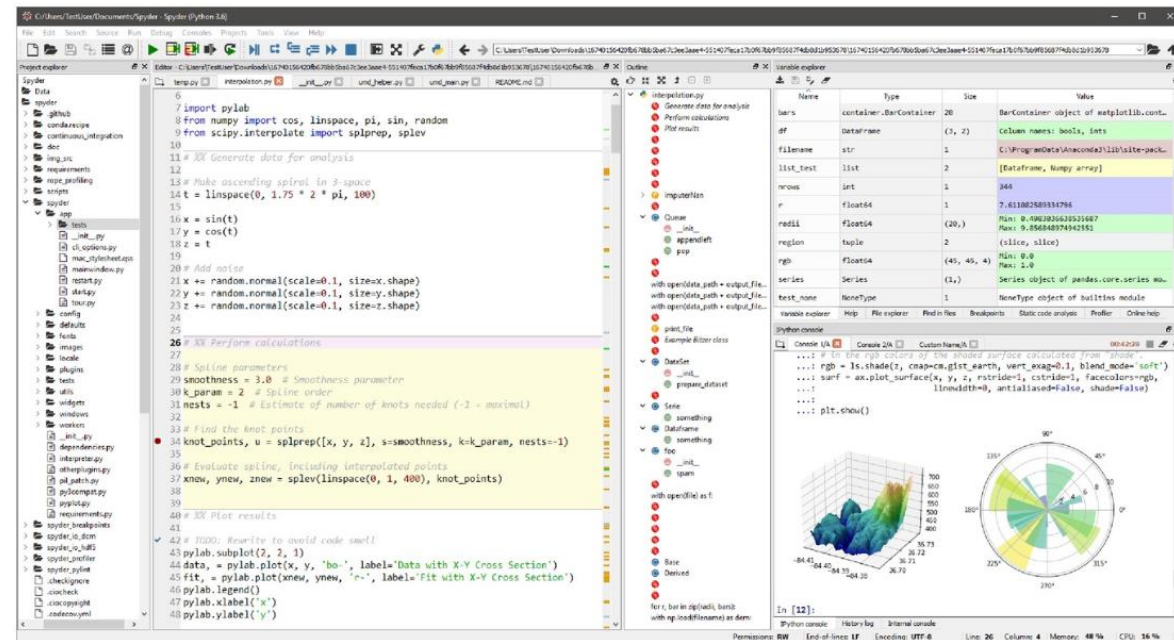


IDLE es un IDE de código abierto multiplataforma **que viene por defecto al descargar Python**. Está completamente escrito en Python, y su nombre es un homenaje a Eric Idle, uno de los miembros fundadores de Monty Python. Se trata de una herramienta algo más básica que otras que verás en la lista, y está pensada para los usuarios primerizos que quieren empezar a aprender Python.

Entre sus características, está la de tener un editor de texto de varias ventanas con consejos, sangría inteligente, función de deshacer y de colorear. Tiene un potente depurador con puntos de interrupción continuos, vista global y espacios locales. También permite hacer búsquedas en cualquier ventana.

- Enlace: docs.python.org/3/library/idle.html

Spyder



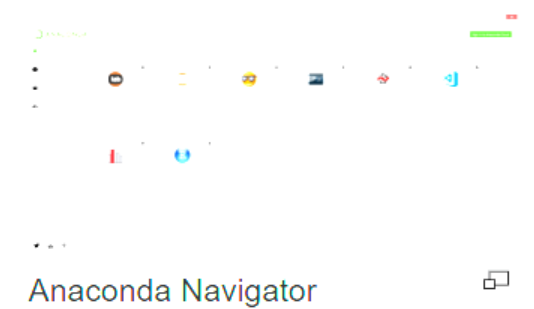
Otro IDE de **código abierto y totalmente gratuito**. Fue desarrollado principalmente para científicos e ingenieros, con la idea de que tuvieran un entorno científico capaz para Python. Ofrece un nivel avanzado de edición, depuración, y funciones de explorador de datos. Tiene también cabida para complementos y API.

Entre sus mejores funciones está el resaltado de sintaxis, **la finalización automática de código**, o la posibilidad de explorar y editar variables desde la propia GUI. Tiene una buena integración con la consola ipython, e interactúa y modifica las variables sobre la marcha.

- Enlace: spyder-ide.org

Anaconda Navigator [\[edit\]](#)

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage Conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them.^[38] It is available for Windows, macOS and Linux.^{[18][48]}



The following applications are available by default in Navigator:^[48]

- JupyterLab
- Jupyter Notebook
- QtConsole^[49]
- Spyder
- Glue
- Orange
- RStudio
- Visual Studio Code

Conda [edit]

Main article: [Conda \(package manager\)](#)

Conda is an open source,^[16] cross-platform,^[17] language-agnostic^[18] package manager and environment management system^{[19][20][50]} that installs, runs, and updates packages and their dependencies.^[16] It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects.^[18] The Conda package and environment manager is included in all versions of Anaconda, Miniconda,^[51] and Anaconda Repository.^[45]

Products / [Anaconda Navigator](#)

Launch data science applications from your desktop with Anaconda Navigator

The Desktop Portal to Data Science.

[Download Now](#) >

Terminal Window Not Required

Anaconda Navigator is a graphical user interface (GUI) that enables you to work with packages and environments without needing to type conda commands in a terminal window. Find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator. Anaconda Navigator is a desktop GUI included in Anaconda Distribution.

HERRAMIENTAS

Jupyter Notebook: Herramienta esencial para científicos de datos

¿Te has parado alguna vez a preguntarte cómo los científicos de datos obtienen resultados tan rápidos y eficientes en su trabajo diario? Es gracias al uso de Jupyter Notebook, una herramienta diseñada para facilitar tanto el desarrollo, como el estudio y presentación de los datos de investigación. En este artículo descubrirás cómo esta herramienta también puede ayudarte en tu trabajo diario.

Herramientas

Data Science



Candela García Fernández

Lectura 11 minutos

Publicado el 19 de junio de 2024

Compartir   



Tabla de contenidos

Introducción

Qué es Jupyter Notebook

Componentes clave

Arquitectura técnica

Funcionalidades principales

Extensiones y plugins

Tutorial inicial de Jupyter Notebook

Conclusiones

Introducción

En el campo de la ciencia de datos, contar con herramientas eficientes y versátiles es crucial para gestionar y analizar grandes volúmenes de datos. Una de las herramientas más destacadas en este ámbito es **Jupyter Notebook**.

Esta plataforma **se ha convertido en un estándar entre los científicos de datos** debido a su capacidad para combinar código, texto, y visualizaciones en un solo documento interactivo.

Su popularidad ha crecido exponencialmente gracias a su flexibilidad y a la facilidad con la que puede integrarse con diversas tecnologías y lenguajes de programación, incluyendo Python, R, Julia, y muchos otros.

Jupyter Notebook permite a los usuarios experimentar diversas ideas a través de bloques de código sin necesidad de montar un entorno personalizado previo.

Qué es Jupyter Notebook

Jupyter Notebook es una aplicación web de código abierto que permite a los usuarios crear y compartir documentos que contienen código en vivo, ecuaciones, visualizaciones y texto narrativo. Esta herramienta es especialmente popular entre los científicos de datos, analistas y desarrolladores por su capacidad para facilitar el desarrollo, la documentación y la presentación de proyectos tanto de aprendizaje como profesionales.

Características clave

Una de las principales características de Jupyter Notebook es **su capacidad para soportar múltiples lenguajes de programación** a través de sus "kernels". Aunque Python es el lenguaje más comúnmente utilizado, Jupyter Notebook también puede ejecutar código en otros lenguajes como R, Julia, Ruby, Matlab, y Perl, entre otros. Esta flexibilidad permite a los usuarios trabajar con el lenguaje que mejor se adapte a sus necesidades y al proyecto en cuestión.

Otra característica destacada es la posibilidad de incluir celdas de código y celdas de texto en un mismo documento. Las celdas de código pueden ejecutarse de manera interactiva, permitiendo ver los resultados inmediatamente, mientras que las celdas de texto pueden contener explicaciones detalladas, ecuaciones en formato LaTeX, e incluso imágenes y gráficos.

Esta combinación facilita la creación de documentos completos y comprensibles, a modo de informe, que integran el código con sus respectivos resultados y explicaciones.

Diferencias con herramientas similares

A diferencia de otras herramientas de desarrollo de software y análisis de datos, Jupyter Notebook se distingue por su interfaz web interactiva y su capacidad de presentar los resultados de manera visualmente atractiva y fácilmente comprensible.

Mientras que los entornos de desarrollo integrados (IDEs) tradicionales, como PyCharm o Visual Studio Code, están diseñados principalmente para la escritura y depuración de código, **Jupyter Notebook se centra en la presentación interactiva y la documentación del trabajo realizado.**

Otra diferencia significativa es la facilidad con la que Jupyter Notebook permite compartir y colaborar en proyectos. Los documentos de Jupyter, conocidos como “notebooks”, pueden guardarse en formato JSON, lo que facilita su distribución y colaboración a través de plataformas como GitHub o JupyterHub.

Además, existen servicios en línea como Google Colab o Databricks, que permiten trabajar en notebooks directamente desde el navegador sin necesidad de instalar software adicional, haciendo que la colaboración en tiempo real sea más accesible y sencilla.

En resumen, Jupyter Notebook es una herramienta esencial para cualquier profesional que necesite combinar código, datos y explicaciones en un entorno interactivo y colaborativo. Su flexibilidad y características únicas lo hacen una elección preferida en la ciencia de datos y otros campos relacionados.

Componentes clave

Jupyter Notebook se compone de varios elementos que trabajan en conjunto para proporcionar un entorno de desarrollo interactivo y flexible. A continuación, describimos los componentes clave que hacen de esta herramienta un poderoso recurso para científicos de datos y desarrolladores.

Interfaz de usuario

La interfaz de usuario de Jupyter Notebook se ejecuta en un navegador web, lo que la hace accesible desde cualquier dispositivo con conexión a internet. Esta interfaz permite a los usuarios crear y gestionar notebooks fácilmente, ofreciendo una experiencia intuitiva y amigable.

Los elementos principales de la interfaz incluyen:

- **Celdas:** Los notebooks están divididos en celdas, que pueden ser de código o de texto. Las celdas de código permiten la ejecución interactiva de fragmentos de código, mientras que las de texto, formateadas en Markdown, permiten agregar explicaciones, ecuaciones y visualizaciones.
- **Menús y barra de herramientas:** La interfaz cuenta con menús desplegables y una barra de herramientas que facilitan el acceso a diversas funcionalidades, como la ejecución de celdas, el guardado de notebooks, y la inserción de nuevas celdas.
- **Panel lateral:** Ofrece acceso rápido a la estructura del notebook, permitiendo a los usuarios navegar fácilmente entre las diferentes secciones de su documento.

Servidor de Jupyter

El servidor de Jupyter es el componente que permite ejecutar y gestionar notebooks. Al iniciar el script, se lanza un servidor local que proporciona la interfaz web accesible desde el navegador. Este servidor maneja la comunicación entre el navegador y el kernel, gestionando la ejecución del código y el almacenamiento de los notebooks. Además, facilita la integración con otros servicios y recursos, como bases de datos y sistemas de archivos.

Archivos y almacenamiento

Los notebooks de Jupyter se guardan en archivos con extensión `.ipynb`, que **contienen tanto el código como los resultados de su ejecución y el texto explicativo** en formato JSON.

Este formato permite una fácil compartición y colaboración, ya que los notebooks pueden ser versionados y almacenados en sistemas de control de versiones como Git. Además, es posible exportar los notebooks a otros formatos, como HTML, PDF, y LaTeX, lo que facilita la creación de informes y presentaciones profesionales.

Arquitectura técnica

La arquitectura técnica de Jupyter Notebook está diseñada para proporcionar un entorno de trabajo interactivo, flexible y escalable, capaz de soportar múltiples lenguajes de programación y diferentes tipos de análisis de datos. A continuación, se describen los componentes principales que conforman esta arquitectura:

Cliente-Servidor

Como ya comentábamos previamente, Jupyter Notebook sigue un modelo de arquitectura cliente-servidor. El cliente es la interfaz de usuario que se ejecuta en un navegador web, mientras que el servidor es una aplicación que se ejecuta en un entorno local o remoto.

Esta separación permite que los usuarios interactúen con los notebooks desde cualquier dispositivo con acceso a internet, proporcionando flexibilidad y accesibilidad.

- **Cliente (Front-End):** La interfaz de usuario del cliente está desarrollada en HTML, CSS y JavaScript. Utiliza el marco de trabajo Bootstrap para el diseño de la interfaz y CodeMirror para la edición de código. Esta interfaz permite a los usuarios crear, editar y ejecutar notebooks, así como visualizar resultados y gráficos de manera interactiva.
- **Servidor (Back-End):** El servidor de Jupyter Notebook está escrito en Python y se encarga de gestionar la comunicación entre el cliente y el kernel. Maneja peticiones HTTP/S, gestiona archivos y directorios, y coordina la ejecución del código en el kernel correspondiente.



 Get [GitHub Copilot Free](#) in VS Code!

Overview

SETUP

GETTING STARTED

USER GUIDE

SOURCE CONTROL

TERMINAL

GITHUB COPILOT

LANGUAGES

NODE.JS /
JAVASCRIPT

TYPESCRIPT

PYTHON

JAVA

C++

C#

DOCKER

DATA SCIENCE

INTELLIGENT APPS

AZURE

REMOTE

DEV CONTAINERS

REFERENCE

Visual Studio Code FAQ

Edit

Our docs contain a **Common questions** section as needed for specific topics. We've captured items here that don't fit in the other topics.

If you don't see an answer to your question here, check our previously [reported issues on GitHub](#) and our [release notes](#).

What is the difference between Visual Studio Code and Visual Studio IDE?

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as [Visual Studio IDE](#).

Which OSs are supported?

VS Code runs on macOS, Linux, and Windows. See the [Requirements documentation](#) for the supported versions. You can find more platform specific details in the [Setup overview](#).

Is VS Code free?

Yes, VS Code is free for private or commercial use. See the [product license](#) for details.

Markdown

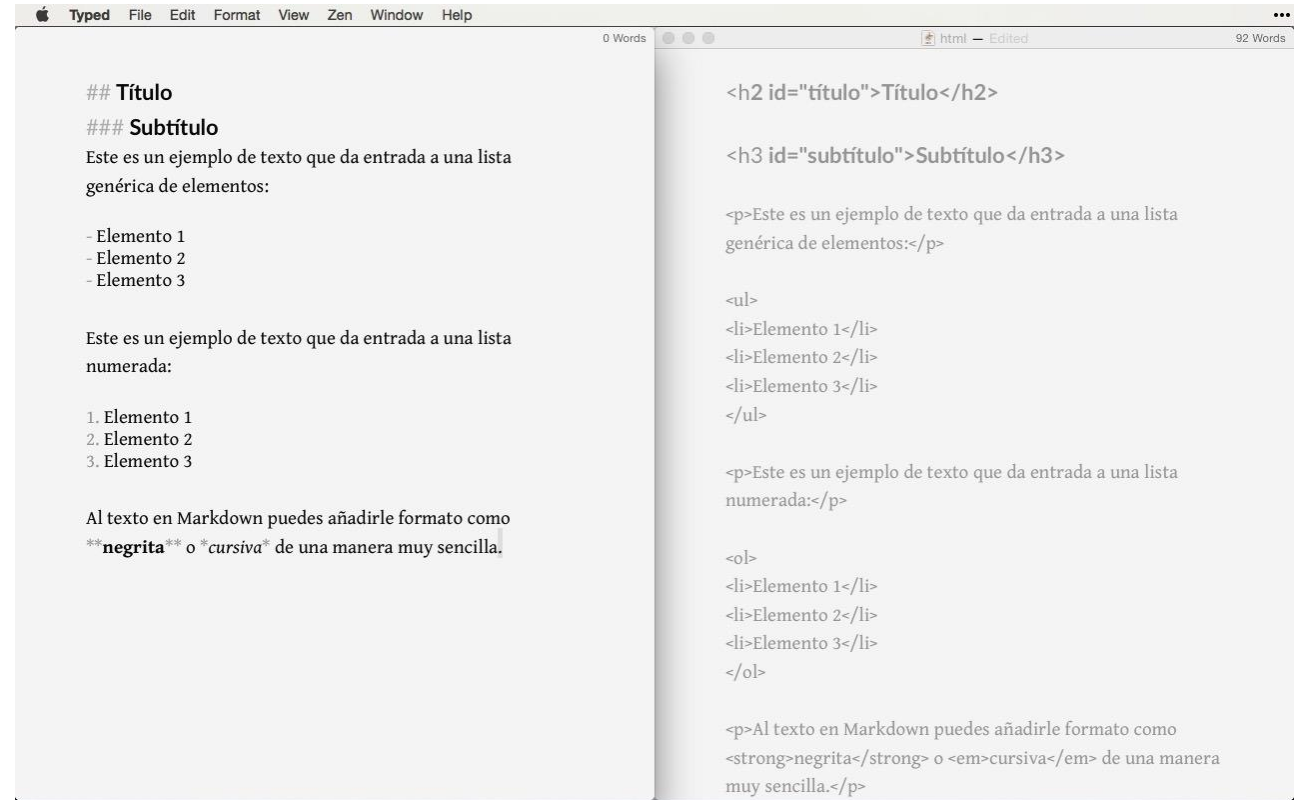
Markdown nació como herramienta de conversión de texto plano a HTML.

Esta herramienta fue creada en 2004 por John Gruber, y se distribuye de manera gratuita bajo una licencia BSD.

Aunque en realidad Markdown también se considera un lenguaje que tiene la finalidad de permitir crear contenido de una manera sencilla de escribir, y que en todo momento mantenga un diseño legible, así que para simplificar puedes considerar Markdown como un método de escritura.

Markdown

Este método te permitirá añadir formatos tales como negritas, cursivas o enlaces, utilizando simplemente texto plano, lo que hará de tu escritura algo más simple y eficiente al evitar distracciones.



Observa la diferencia entre sintaxis Markdown (izquierda) y sintaxis HTML (derecha), ¿con cuál te quedas?

Conceptos básicos de Markdown

Las secciones siguientes describen los conceptos básicos de la creación en Markdown.

Encabezados

Para crear un encabezado, utilice el símbolo de almohadilla (#) al principio de una línea:

```
# This is level 1 (article title)
## This is level 2
### This is level 3
#### This is level 4
##### This is level 5
```

Texto básico

En Markdown, un párrafo no requiere sintaxis especial.

Para aplicar **negrita** al texto, se escribe entre dos asteriscos. Para aplicar *cursiva* al texto, se escribe entre un solo asterisco:

```
This text is **bold**.
```

```
This text is *italic*.
```

```
This text is both ***bold and italic***.
```

Listas numeradas y listas con viñetas

Para crear listas numeradas, empiece una línea con 1. or 1), pero no mezcle los formatos dentro de la misma lista. No es necesario especificar los números. GitHub lo hace por usted.

```
1. This is step 1.  
1. This is the next step.  
1. This is yet another step, the third.
```

Visualización:

1. This is step 1.
2. This is the next step.
3. This is yet another step, the third.

Para crear listas de viñetas, empiece una línea con *, - o +, pero no mezcle los formatos dentro de la misma lista. (No mezcle formatos de viñetas, como * y +, dentro del mismo documento).

```
* First item in an unordered list.  
* Another item.  
* Here we go again.
```

Visualización:

- First item in an unordered list.
- Another item.
- Y otro más.

También puede incrustar listas dentro de listas y añadir contenido entre elementos de la lista.

```
1. Set up your table and code blocks.  
1. Perform this step.  
  
    ![screen](https://experienceleague.adobe.com/docs/contributor/assets/adobe_standard_logo.png?lang=es)  
  
1. Make sure that your table looks like this:  
  
    | Hello | World |  
    |---|---|  
    | How | are you? |  
  
1. This is the fourth step.  
  
    >[!NOTE]  
    >  
    >This is note text.  
  
1. Do another step.
```

Visualización:

1. Set up your table and code blocks.
2. Perform this step.



3. Make sure that your table looks like this:

| | |
|-------|----------|
| Hello | World |
| How | are you? |

4. This is the fourth step.

① NOTE

Este es el texto de la nota.

5. Do another step.

[Home](#)[About](#)[Get](#)[LaTeX3](#)[Publications](#)[Help](#)[News](#)

LaTeX – A document preparation system

LaTeX is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation. LaTeX is the de facto standard for the communication and publication of scientific documents. LaTeX is available as [free software](#).

You don't have to pay for using LaTeX, i.e., there are no license fees, etc. But you are, of course, invited to support the maintenance and development efforts through a [donation to the TeX Users Group \(choose LaTeX Project contribution\)](#) if you are satisfied with LaTeX.

You can also sponsor the work of LaTeX team members through the [GitHub sponsor program](#) at the moment for [Frank](#), [David](#) and [Joseph](#). Your contribution goes without any reductions by GitHub to the developers in support of the project.

The volunteer efforts that provide you with LaTeX need financial support, so thanks for any contribution you are willing to make.

Recent News

29 January, 2025

[Interpreting a PDF Structure Tree as XML](#)

25 December, 2024

[Season's Greetings from the LaTeX Project!](#)

27 November, 2024

[Better multicolumn marks: Pre-release of LaTeX 2025-06-01 is available for testing](#)

1 November, 2024

[LaTeX 2024-11-01 released and distributed](#)

8 September, 2024


[TUG Conference 2024 in Prague: talks and accessibility workshop](#)

8 July, 2024

[How to make accessible PDF](#)

13 June, 2024

[LaTeX 2024-06-01 PL1 released and distributed](#)

[All News](#) · [Subscribe to our](#)  [RSS News Feed](#)

código

```

\documentclass[12pt]{article}
\usepackage[spanish]{babel}
\usepackage{amsmath}
\title{\LaTeX}
\date{}
% Este es un comentario, no será mostrado en el documento final.
\begin{document}
\maketitle

\LaTeX{} es un programa para preparar documentos con el sistema de
tipografías\footnote{%nota al pie de página
  Seg\un Wikipedia, la tipografía es el arte y t\ecnica del manejo
  y selecci\on de tipos, originalmente de plomo, para crear trabajos
  de impresi\on } %fin nota al pie de página
\TeX{}. \LaTeX{} fue desarrollado originalmente por Leslie Lamport en
1984 y se convirti\o en el m\etodo dominante para la manipulaci\on
de \TeX. La versi\on utilizada para generar este documento es \LaTeXe.
\newline
% El siguiente código muestra la calidad de la tipografía de LaTeX
\begin{align}
E &= mc^2 && \\\
m &= \frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}} \\
\end{align}
\end{document}

```

LaTeX

LaTeX es un programa para preparar documentos con el sistema de tipografías¹ TeX. LaTeX fue desarrollado originalmente por Leslie Lamport en 1984 y se convirtió en el método dominante para la manipulación de TeX. La versión utilizada para generar este documento es LaTeX 2_ε.

$$E = mc^2 \tag{1}$$

$$m = \frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}} \tag{2}$$

¹Según Wikipedia, la tipografía es el arte y técnica del manejo y selección de tipos, originalmente de plomo, para crear trabajos de impresión

[Features & Benefits ▼](#)[Plantillas](#)[Planes y precios](#)[Ayuda ▼](#)[Sign up](#)[Entrar](#)

`\begin{an}`

Write like a rocket scientist with Overleaf
—the collaborative, online LaTeX editor that *anyone* can use.

[Sign up with Google](#)[Sign up with ORCID](#)

OR

[Sign up for free](#)

Al registrarse, acepta nuestras [condiciones del servicio](#) y [notificación de privacidad](#)

Objetos

Los objetos son las principales cosas que manipulan los programas de Python.

Cada objeto tiene un “tipo (type)” que define la clase de cosas que los programas pueden hacer con ese tipo.

Los tipos pueden ser escalares o no escalares.

Objetos escalares

Los objetos escalares son indivisibles.

Python tiene 4 tipos de objetos escalares:

1. Enteros: int
2. Números reales: float
3. Valores booleanos: bool
4. None

Expresiones

Objetos y operadores pueden ser combinados para formar expresiones que denotan un objeto de algún tipo. Nos referiremos a esto como valor de la expresión.

Por ejemplo, la expresión $2 + 5$ denota el objeto 5 del tipo int y la expresión $3.0 + 2.0$ denota el objeto 5.0 del tipo float

Prompt

Es la serie de caracteres ((>>>)) que aparece en la última línea.
El prompt indica que el intérprete de Python espera que nosotros introduzcamos una orden utilizando el teclado.

```
$ python ↵  
Python 2.3 (#1, Aug 2 2003, 12:14:49)  
[GCC 3.3] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>>
```

Ejercicio 1

Instala anaconda navigator en tu computador

Abre un jupyter notebook

Escribe el titulo de tu tarea, tu nombre y un texto pequeño a elección libre que contenga ecuaciones matemáticas y una imagen

En una celda de código, usa la función print para escribir “¡Hola mundo!”

Guarda el archivo en html y entrégalo en el espacio correspondiente
