AWAKELAB

# BASECAMP

## Ciencia de Datos

**Módulo : Fundamentos de Programación en Python**

## Aprendizaje Esperado

- Distinguir las características principales del lenguaje Python y su utilización para resolver distintas problemáticas.

## Lenguaje Python


**Python es un lenguaje de programación** de código abierto, orientado a objetos, muy simple y fácil de entender. Tiene una sintaxis sencilla que cuenta con una vasta biblioteca de herramientas, que hacen de Python un lenguaje de programación único.

# Reseña lenguaje Python

Python es el tercer lenguaje de programación más usado en el mundo. Pero, ¿cómo ha llegado hasta este punto? Para dar respuesta a esto, vamos a hacer un repaso por toda la **historia de Python.** Desde la primera versión, lanzada a finales de los años 80 hasta la última, aparecida este mismo año.

**Guido Van Rossum** es el creador y responsable de que Python exista. Se trata de un informático de origen holandés que fue el encargado de diseñar Python y de pensar y definir todas las vías posibles de evolución de este popular lenguaje de programación

En las navidades de 1989 Van Rossum, mientras trabajaba en un centro de investigación holandés (CWI), decidió empezar un nuevo proyecto como pasatiempo personal. **Pensó en darle continuidad a ABC**, un lenguaje de programación que se desarrolló en el mismo centro en el que estaba trabajando.

Sin embargo, el proyecto no llegó mucho más lejos por las limitaciones del hardware de la época, así que **Van Rossum decidió darle una segunda vida a su idea** y partiendo de la base que tenía, **empezó a trabajar en Python.**

En 1994 se formó **comp.lang.python**, foro de discusión principal de Python, marcando un hito en el crecimiento del grupo de usuarios de este lenguaje.

Pero es en el año 2000 cuando se produce un importante hito. El equipo principal de desarrolladores cambia su dominio a BeOpen.com para formar **BeOpen PythonLabs** coincidiendo con la liberación de la versión 1.6 y 2.0 casi al mismo tiempo. Después que Python 2.0 fuera publicado por BeOpen.com, Guido van Rossum y los otros desarrolladores de PythonLabs se unieron a Digital Creations para continuar el desarrollo de las diferentes versiones hasta nuestros días.

## Para que se utiliza Python

Python es un lenguaje de programación de propósito general, que es otra forma de decir que puede ser usado para casi todo. Lo más importante es que se trata de un lenguaje interpretado, lo que significa que el código escrito no se traduce realmente a un formato legible por el computador en tiempo de ejecución.

Este tipo de lenguaje también se conoce como «lenguaje de scripting» porque inicialmente fue pensado para ser usado en proyectos sencillos.

El concepto de «lenguaje de scripting» ha cambiado considerablemente desde su creación, porque ahora se utiliza Python para programar grandes aplicaciones de estilo comercial, en lugar de sólo las simples aplicaciones comunes.

Una encuesta realizada en 2019 entre los usuarios de Python indicó que los usos más populares eran para el desarrollo web y el análisis de datos. Sólo alrededor del 6 % de los encuestados lo utilizaron para el desarrollo de juegos o el desarrollo de aplicaciones.

Esta dependencia de Python ha crecido aún más a medida que Internet se ha hecho más popular. Una gran mayoría de las aplicaciones y plataformas web dependen de su lenguaje, incluido el motor de búsqueda de Google, YouTube, y el sistema de transacciones orientado a la web de la Bolsa de Nueva York (NYSE).

En definitiva, sabes que el lenguaje debe ser realmente importante cuando se encarga de impulsar un sistema bursátil. De hecho, la NASA lo utiliza cuando programan sus equipos y maquinaria especial.

Existen muchas aplicaciones comerciales para la programación en Python, pero el lenguaje también se ha afianzado en los círculos académicos, especialmente entre los que trabajan con **grandes cantidades de datos.**

También, puede ser usado para procesar texto, mostrar números o imágenes, resolver ecuaciones científicas y guardar datos.

En resumen, se utiliza entre bastidores para procesar un montón de elementos que podrías necesitar o encontrar en tu(s) dispositivo(s), incluido el móvil.

**¿Cómo funciona Python?**

El lenguaje de programación Python utiliza módulos de código que son intercambiables en lugar de una larga lista de instrucciones que era estándar para los lenguajes de programación funcional.

La implementación estándar de Python se llama «cpython«. En definitiva, no convierte su código en lenguaje de máquina o código máquina, algo que el hardware puede entender.

En realidad, lo convierte en algo llamado código de byte. Este código de bytes no puede ser entendido por la CPU. Así que necesitamos un intérprete llamado Máquina Virtual Python (PVM) que ejecuta los códigos de bytes.

El intérprete de Python realiza las siguientes tareas para ejecutar un programa:

- **Paso 1** : El intérprete lee un código o instrucción Python. Luego verifica que la instrucción esté bien formateada, es decir, comprueba la sintaxis de cada línea. Si encuentra algún error, detiene inmediatamente la traducción y muestra un mensaje de error.

- **Paso 2** : Si no hay ningún error, es decir, si la instrucción o el código Python está bien formateado, el intérprete lo traduce a su forma equivalente en un lenguaje intermedio llamado «código Byte». Así, después de la ejecución exitosa de la escritura o el código Python, se traduce completamente en código Byte.

- **Paso 3**: El código del byte se envía a la Máquina Virtual Python, donde de nuevo se ejecuta el código del byte en PVM. Si se produce un error durante esta ejecución, ésta se detiene con un mensaje de error.


*imagen: https://www.cursosaula21.com/*

**Algunas razones del por qué aprender Python:**

Python es uno de los lenguajes de programación más queridos por los desarrolladores, **científicos de datos**, ingenieros de software e incluso hackers debido a su versatilidad, flexibilidad y características orientadas a los objetos.

Muchas de las aplicaciones web y móviles que disfrutamos hoy en día se deben a las abundantes librerías de Python, a sus diversos frameworks, a sus extensas colecciones de módulos y a sus extensiones de archivos. No sólo eso, también es excelente para construir servicios web de microproyectos a macroempresas, así como para soportar otros tipos de lenguajes de programación.

Aunque es un lenguaje de alto nivel y puede hacer tareas complejas, es fácil de aprender y tiene una sintaxis limpia. Por lo tanto, es recomendable tanto para principiantes como para programadores experimentados. Además, hay muchas maneras de aprender Python; de forma autodidacta

De hecho, hay muchas razones por las que deberías aprender Python. Y para entender más profundamente su alcance y beneficios.

1. *Buena Oferta laboral y bien remunerada.*

Solo tienes que pasarte por cualquier buscador de empleo online y escribir «Python» para descubrir las numerosas ofertas relacionadas que aparecen. Dado que la programación en Python se puede utilizar de muchas maneras, hay puestos de trabajo que se ajustan a todos los niveles de experiencia e interés laboral, incluyendo puestos

de ingeniero de control de calidad, puestos de ingeniero de software de nivel básico y puestos de alto nivel como **ingenieros de machine learning e inteligencia artificial**.

2. *De uso frecuente en startups*(organizaciones de gran capacidad de cambio).

Las startups necesitan funcionar con presupuestos ajustados para sobrevivir, y eso significa que cuando están construyendo sus productos digitales (ya sean sitios web, aplicaciones para móviles o programas de software) esos productos deben completarse dentro del presupuesto y antes de lo previsto.

Debido a su capacidad para ayudar a cumplir con ambos objetivos, Python es un lenguaje de programación adorado en el mundo de las startups.

La eficiencia y la facilidad de uso significan menos tiempo de desarrollo, un proceso de control de calidad y depuración racionalizado, y un mayor retorno de la inversión en general que las alternativas de codificación más difíciles de navegar. Las startups son un gran lugar para conseguir un primer trabajo y empezar a ganar experiencia en la industria, y conocer tu camino.

3. *Se tarda poco tiempo en aprender.*

Por lo eficiente y versátil que es un lenguaje como Python, se podría pensar que se necesitan años para aprender a programarlo. ¡No es así! Los profesionales de la industria dicen que los fundamentos (cosas como la sintaxis, las palabras clave y los tipos de datos) se pueden aprender en tan sólo 6-8 semanas si tienes experiencia previa con lenguajes de codificación.

4. *Comunidad de usuarios muy activa.*

Python es un lenguaje de código abierto, lo que significa que es libre de usar y cualquiera puede modificar o crear extensiones para este lenguaje.

El hecho de ser de código abierto es lo que permite a los lenguajes tener bibliotecas, marcos de trabajo (frameworks) y otras herramientas que mantienen al lenguaje relevante y adaptable a lo largo del tiempo. Por ello, el código abierto sólo está a la altura de su potencial si hay una comunidad de usuarios comprometidos con el lenguaje.

La Python Software Foundation tiene una página comunitaria (foro) en su web que enlaza con varios grupos comunitarios y foros donde se pueden encontrar compañeros para recibir consejos, tutorías, inspiración o simplemente para charlar sobre la brillantez de Python.

5. *Lenguaje de programación popular.*

Según The Economist (2018), Python va camino de convertirse en el lenguaje de codificación más popular del mundo. Mientras que lenguajes como **Fortran** y **Lisp** han experimentado un precipitado declive, y lenguajes como **C y C++** permanecen estables, lenguajes como **Python** y **JavaScript** están en alza.

6. *Lenguaje Versátil*

Ser un lenguaje de programación de propósito general significa que sus procedimientos, instrucciones y estructuras de datos están diseñados para resolver cualquier problema. Es por ello, que los grandes éxitos de la tecnología como Google, Facebook e Instagram utilizan la programación Python para construir partes de sus paquetes tecnológicos. Pero también puede ser usado para construir programas básicos y proyectos en todos los puntos intermedios.

La versatilidad de Python indica que, como desarrollador, tendrás una amplia gama de opciones de trabajo. Ya sea que quieras trabajar para un gigante de la tecnología, construir tus propios programas de software a menor escala, o trabajar como un desarrollador web.

7. *Numerosos complementos para los se necesite.*

Si necesitas algo más personalizable que la configuración de Python lista para usar, ¡no hay problema! Al igual que los lenguajes como JavaScript, dispones de librerías y frameworks para Python que se ajustan a tus necesidades de codificación específicas.

Los frameworks populares de Python como Django están diseñados para hacer que sea más efectivo en la creación de aplicaciones web, mientras que PyQt es un framework que permite a Python construir Interfaces Gráficas de Usuario (GUI) - interfaces de usuario que implican el uso de iconos en pantalla y gráficos para procesar comandos de usuario.

8. *Automatización de tareas y procesos.*

Una de las partes más difíciles de trabajar en la tecnología (independientemente de su función) es la gestión de todas esas tareas repetitivas, que consumen mucho tiempo, relacionadas con la tecnología. Pequeñas cosas como copiar archivos, arrastrar carpetas y renombrarlas, subir activos a los servidores… todo esto se traduce en mucho tiempo a largo plazo.

La automatización es otra área por la que vale la pena aprender Python. La capacidad de este lenguaje para escribir scripts de sistema origina que puedes crear programas Python sencillos para automatizar tareas monótonas que disminuyen tu productividad.

El tiempo que te ahorrarás en saber cómo automatizar procesos con Python es un gran argumento para aprender este lenguaje.

9. *Te da herramientas para trabajar en cualquier tecnología.*

**Aprender el código Python** no hace más que prepararte para el desarrollo de Internet, te preparará para el futuro de los trabajos tecnológicos, porque se utiliza para algo más que el desarrollo tradicional. De hecho, es importante para los campos emergentes de la **ciencia de los datos** como:

- Análisis de datos (Big Data)
- Inteligencia Artificial
- Machine Learning

La ciencia de los datos es otra posibilidad tecnológica que se te abre si decides aprender Python.

# Python y sus características.

**Lenguaje de propósito general**

Eso significa que no está orientado a un fin concreto, como puede ser PHP, pensado sobre todo para hacer páginas de internet.

Con Python podrás crear páginas sin tener un alto conocimiento (con Javascript como un poderoso aliado), pero también hacer scripts o software para el sistema operativo Windows.

Aún no hay nada destacado para dispositivos móviles, pero se puede usar Kivy para este propósito

**Es multiparadigma**

¿Y qué significa eso? ¿Multiparadigma?

Pues aunque su fuerte sea la programación orientada a objetos (es un lenguaje de alto nivel), existen otros paradigmas o estilos de programación para sus usuarios, como es

la programación imperativa (con sentencias de bucle) o la programación funcional (con módulos y funciones).
Así que si no sabes nada de objetos y sólo sabes escribir código mediante métodos, puedes usar Python perfectamente, cosa que en otros lenguajes hacer eso es imposible.

**Python es un lenguaje interpretado**
Cuando programamos en Python, no compilamos el código fuente a código máquina, sino que hay un intérprete que es el que ejecutará el programa basándose en el código directamente.
Aunque esta propiedad hace pensar que los programas puedan ser más lentos, que en lenguaje Python no suele ser así, eso facilita el desarrollo para la siguiente característica.

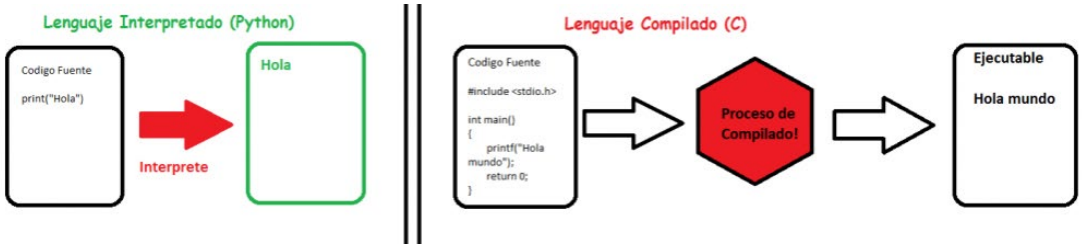



*imagen:https://pythones.net/*

**Es multiplataforma**

Al contrario que muchos lenguajes como **visual basic**, que principalmente solo puedes hacer cosas para Windows, con Python tienes la posibilidad de usarlo en muchos dispositivos y sistemas operativos, ya que se han creado intérpretes para Unix, Linux, Windows y sistemas Mac Os

**Es de tipado dinámico**

Cuando declaramos una variable, no es necesario decirle de qué tipos son los datos (si es **int**, **string**, **float**, etc.). La variable se adapta a lo que escribimos cuando se ejecuta el programa.

Antes esta característica siempre ha sido criticada en otros lenguajes, por la optimización de la memoria, errores a la hora de escribir código, etc. pero con Python el objetivo es que el lenguaje ayude a la creación de software, no tener que lidiar con peculiaridades propias del lenguaje.
Igualmente, Python es fuertemente tipado, por ejemplo, no podrás sumar números y texto (una variable del tipo int con una de tipos cadenas) porque daría error.

```
import random

def buscarElemento(lista, elemento):
    for i in range(0, len(lista)):
        if (lista[i] == elemento):
            return i

def imprimirLista(lista, nombre):
    for i in range(0, len(lista)):
        print(nombre + "[" + str(i) + "]" + str([i]))

def leerLista():
    lista=[]

    i = 0
    while i < 10:
        lista.append(int(random.randint(0,10)))
        i = i+1
    return lista

A = leerLista()
imprimirLista(A, "A")
cn = int(input("Numero a buscar: "))
print("A[" + str(buscarElemento(A,cn)) + "]")
```

**Es orientado a objetos**

Ya hemos dicho que podemos aplicar otro estilo de programación, hacer software orientado a objetos conlleva una serie de ventajas estándar, sobre todo a la hora de reutilizar los componentes gracias a la herencia y sus funciones de polimorfismo.

**Otras propiedades de Python**
Vamos a nombrar brevemente otras funciones o elementos propios de Python, como pueden ser:

- De *libre distribución*.
- Gracias a su popularidad, existen una gran cantidad de *librerías y funciones* ya hechas, que podemos utilizar gracias a su extensa biblioteca.
- Tiene soporte para múltiple variedad de *bases de datos*.
- Tiene un gran *soporte* gracias a su comunidad. Por ejemplo, la última versión de Visual Studio te permite desarrollar en Python, o la comunidad de la página oficial de Python, donde vemos todas las actividades que hacen en el mundo.

**Programas hechos con Python**

Ahora vamos a nombrar algunos programas famosos que están hechos con Python, como por ejemplo:

- **Calibre**: el mejor gestor de e-books para todos los usuarios.
- **GNU MailMan:** un programa para manejar listas de correo.
- **BitTorrent**: programa para compartir ficheros de tipo torrent estándar.
- **Odoo** (antes OpenERP): un ERP y mucho más para la gestión de empresas, de software libre.

Aunque hemos hablado de programas, Python se usa en webs conocidas, como Youtube y software hecho por Google.

# Futuro de Python

**Uso en Data Science, Big Data e Inteligencia Artificial (IA)**

A lo largo de los últimos 30 años Python ha disfrutado de una evolución constante además de el respaldo de la comunidad que le han hecho particularmente relevante en el desarrollo de aplicaciones en entorno servidor. Esto, junto con su sencillez, ha producido que pudiera situarse en la escena del Big Data y en especial del desarrollo de algoritmos de aprendizaje automatizado.

La generalización del Big Data en los últimos años, seguida de la explosión de la Inteligencia Artificial, Machine Learning, Deep Learning y la consolidación de la

ciencia de datos como un nuevo área de trabajo con especialistas propios, ha revolucionado el panorama.

Al estar Python muy presente en el ámbito educativo siendo usado como lenguaje de referencia en escuelas y universidades su presencia en el campo de la investigación es más que justificada. Lo que ha producido que muchas de las herramientas que han surgido en este sector han sido desarrolladas en este lenguaje y explotadas por los ingenieros de datos y los científicos de datos. Algunos ejemplos son PySpark (Big Data), Pandas, NumPy, Matplotlib o Jupyter (Data Science).

## Entorno de trabajo

Los entornos de desarrollo integrados, o como se denominan en inglés IDEs (Integrated Development Environment), son aplicaciones que ayudan a desarrollar proyectos de forma fácil y efectiva.

Cada IDE presenta ciertas características que lo diferencian de los demás aunque las características más útiles que se deben de buscar son:

- Capacidad de editar código.
- Soporte de ejecución de código.
- Interfaz intuitiva.
- Soporte de documentación del lenguaje.

La finalidad última de un IDE es hacer que los desarrolladores puedan desarrollar de forma efectiva, cómoda y rápidamente.

*imagen:https://elpythonista.com/*

**IDEs para programar en Python**

Cualquier editor de texto podría servir para programar en cualquier lenguaje, pero necesitaría programas externos para integrar funcionalidades específicas para Python como:

- Coloreado de sintaxis propia de Python.
- Documentación de módulos Python.
- Soporte de entornos de ejecución.
- Validadores de sintaxis.
- Validadores de estilos y seguimiento del PEP-8.
- Depurador de código Python.

Los IDEs para Python que se presentan a continuación se pueden separar en 3 grupos ordenados desde más específicos a más generalistas:

1. Entornos de desarrollo orientados a **desarrollo Python**.
2. Entornos que soportan **múltiples lenguajes** (generalistas) que usan complementos para dar soporte a Python
3. **Editores de texto potentes** que añadiendo muchos complementos manualmente soporta Python.

**IDEs exclusivamente para desarrollo Python**

La gran ventaja que presentan los entornos de desarrollo diseñados para desarrollar aplicación Python es que todas las herramientas están integradas y enfocadas a ser más productivo en Python.

En general ofrecen una mejor experiencia para los desarrolladores conociendo perfectamente el ecosistema de Python, dando soporte para ejecuciones, depuración, documentación, análisis de rendimiento y un largo etcétera.

**Pycharm**


Es quizás el IDE más utilizado por profesionales de desarrollan aplicaciones en Python y las características que lo destacan de los demás son las siguientes:

- Soporta **depuración** de código profesional e intuitiva.
- Ejecución de **test** soportando unittest, nosetests y pytests.
- Soporte de **entornos virtuales**.
- Soporte de múltiples versiones de Python.
- Coloreado de sintaxis, comprobación de PEP-8, formateado automático, optimización de importaciones, etc.
- Diferentes sistema de **control de versiones**.
- Admite extenderlo con **plugins**.
- Se pueden explorar **bases de datos** (versión pro).
- Soporte de **análisis científico** (versión pro).
- Permite **desarrollo web profesional** (versión pro).

Es desarrollado y mantenido por JetBrains, empresa especializada en creación de entornos de desarrollo profesionales, por lo que el soporte es excepcional.

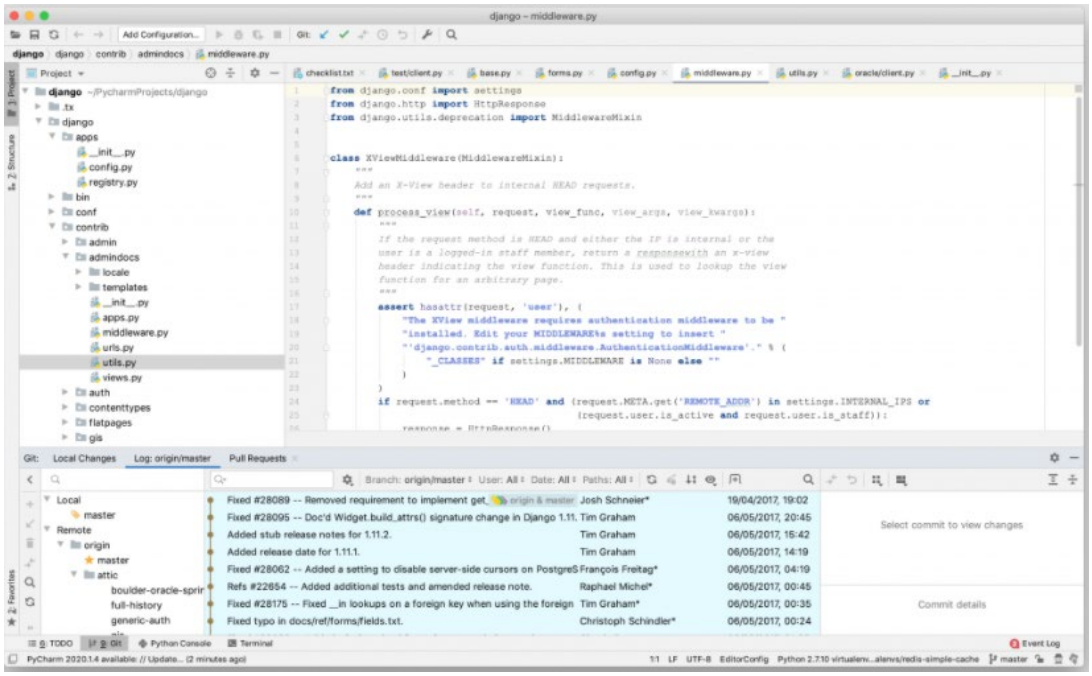


*imagen: fuente propia*

Pycharm está disponible en dos versiones, una community que es gratuita y otra de pago para profesionales con muchas más funcionalidades y se puede descargar en https://www.jetbrains.com/pycharm/.

**Spyder**

Spyder es un entorno de desarrollo orientado a las aplicaciones científicas de software libre, aunque también soporta desarrollo de aplicaciones Python en general. Las principales características son las siguientes:

- Permite trabajar con numerosas librerías científicas como NumPy, SciPy, Pandas o SymPy.
- Soporta librerías gráficas como matplotlib.
- Tiene soporte para IPython y notebooks.
- Está integrado por defecto en Anaconda.
- Tiene un navegador de ficheros.
- Permite depurar código.

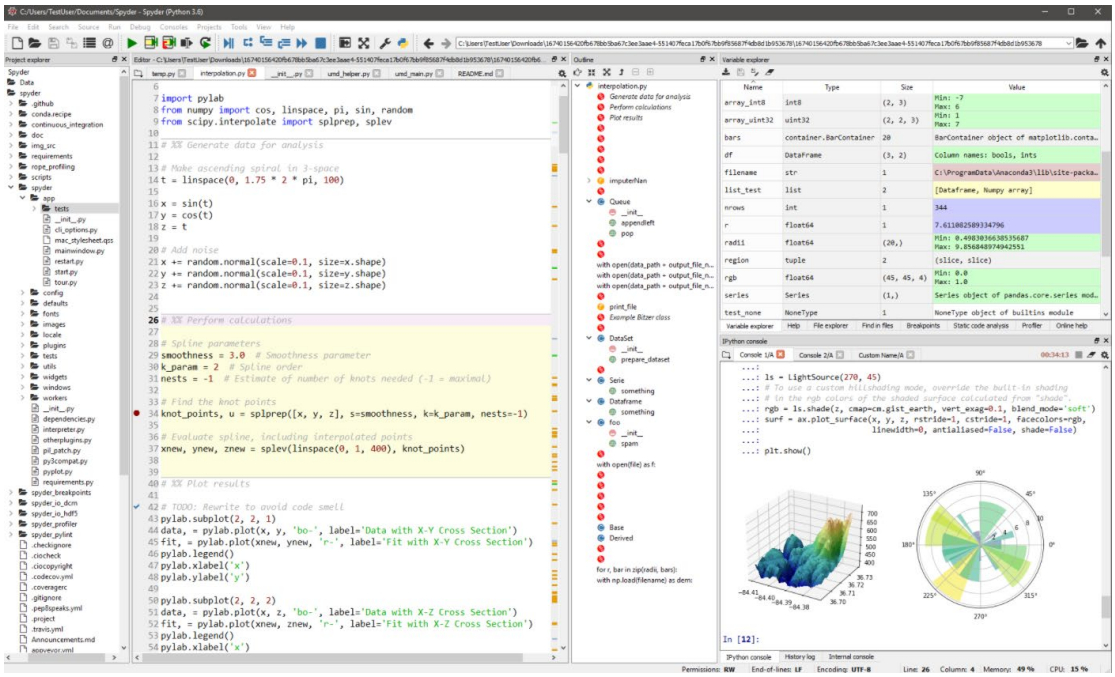


*imagen: fuente propia*

**Google Colaboratory**

**¿Qué es Colab?**

Colab, también conocido como "Colaboratory", te permite programar y ejecutar Python en tu navegador con las siguientes ventajas:

- No requiere configuración
- Acceso a GPUs sin coste adicional
- Permite compartir contenido fácilmente
- Colab puede facilitar tu trabajo, ya seas estudiante, científico de datos o investigador de IA. No te pierdas el vídeo de Introducción a Colab aquí para obtener más información. O simplemente empieza con los pasos descritos en el link.

Con Colab, puedes importar un conjunto de datos de imágenes, entrenar un clasificador de imágenes con dicho conjunto de datos y evaluar el modelo con tan solo usar unas pocas líneas de código. Los cuadernos de Colab ejecutan código en los servidores en la nube de Google, lo que te permite aprovechar la potencia del hardware de Google, incluidas las GPU y TPU, independientemente de la potencia de tu equipo. Lo único que necesitas es un navegador.

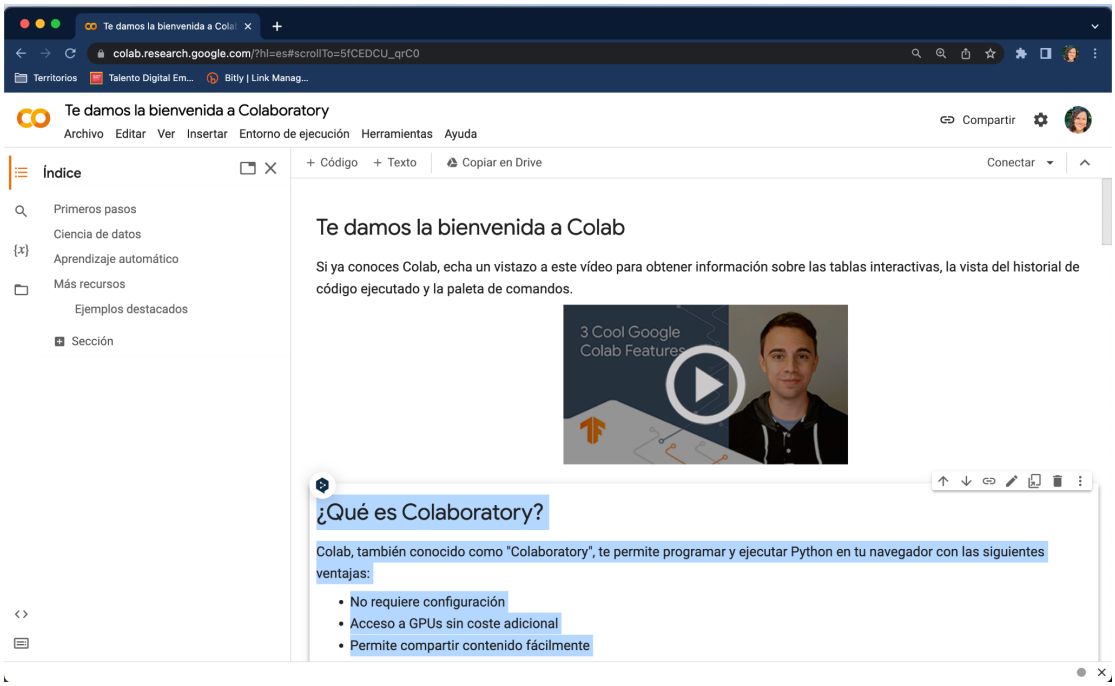


*imagen: fuente propia*

**IDLE**

IDLE es un entorno de desarrollo presente en la mayoría de instalaciones de Python y que permite hacer una edición simple de código y depuración del mismo, entre sus características se pueden destacar las siguientes:

- **Coloreado** de sintaxis Python.
- Se incluye en instalaciones de Python en Windows y MacOS X.

- **Depurador** de ejecuciones.
- **Shell de Python** en el IDE.

El código fuente de este entorno de desarrollo se encuentra presente en la librería estándar y utiliza tkinter para su interfaz gráfica.

**IDEs generalistas usando Python**

**Visual Studio code**

Visual Studio Code es un entorno de desarrollo de software libre, liberado y mantenido por Microsoft.

Es uno de los IDEs más utilizados en todo el mundo gracias a que soporta un gran número de lenguajes de programación incluyendo Java, JavaScript, Go, Node.js o C++ entre otros.

La integración con Python se hace a través de un plugin que instala parte específica para el lenguaje como:

- **Coloreado propio** de sintaxis.
- Capacidad de **ejecución de código Python**.
- Integración con **Jupyter Notebooks**.
- **Depurador** avanzado de variables.
- **Ejecución partes** de código.
- Autocompletado avanzado en Python (**IntelliSense**).
- **Integración con consola interactiva de Python**.

Adicionalmente se pueden añadir más plugins desde el marketplace de VS como gather o pylance para ayudar a exportar Jupyter notebooks o el autocompletado.

Para cualquier lenguaje versus da soporte a:

- Posibilidad de extender el IDE con plugins.
- **Sistema de control de versiones.**
- **Explorador de sistema de ficheros.**
- Soporte **multi-ventana.**
- Integración con Microsoft Azure para desplegar aplicaciones.

Se puede descargar en: https://code.visualstudio.com/

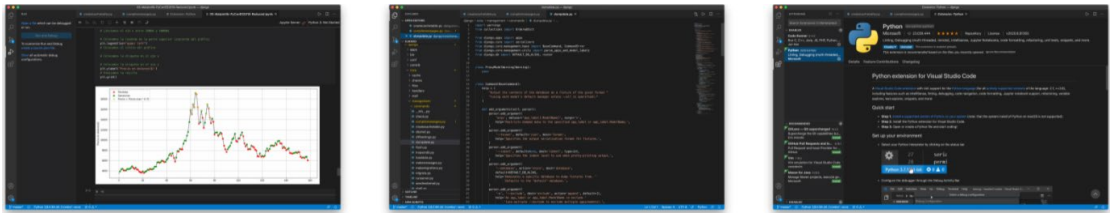


*imagen: fuente propia*

**PyDev o LiClipse**

PyDev es un IDE de Python para Eclipse. Eclipse es un entorno de desarrollo software libre orientado al desarrollo de aplicaciones Java, pero con PyDev se integra:

- Soporta intérpretes de CPython, Jython e IronPython.
- Integración con Django.
- Permite autocompletar.
- Coloreado de sintaxis de Python.
- Dispone de depuración.
- Integración de tests.

La forma más simple de obtener PyDev en eclipse es instalando LiClipse, dado que ya trae por defecto el soporte para Python y para PyDev.

**Jupyter Notebook - Entornos interactivos basados en la web**

En esta categoría tenemos a Jupyter Notebook. Este entorno de desarrollo es una aplicación web que podemos correr en un navegador como Chrome, Firefox, etc. En concreto, esta aplicación facilita la creación de cuadernos (*notebooks*) compuestos por celdas. En estas celdas podemos desarrollar nuestro código Python e irlas ejecutando una a una. Este entorno es muy popular en *data science* y *machine learning*, ya que permite visualizar gráficos y tener el código que los ha generado en un mismo documento.

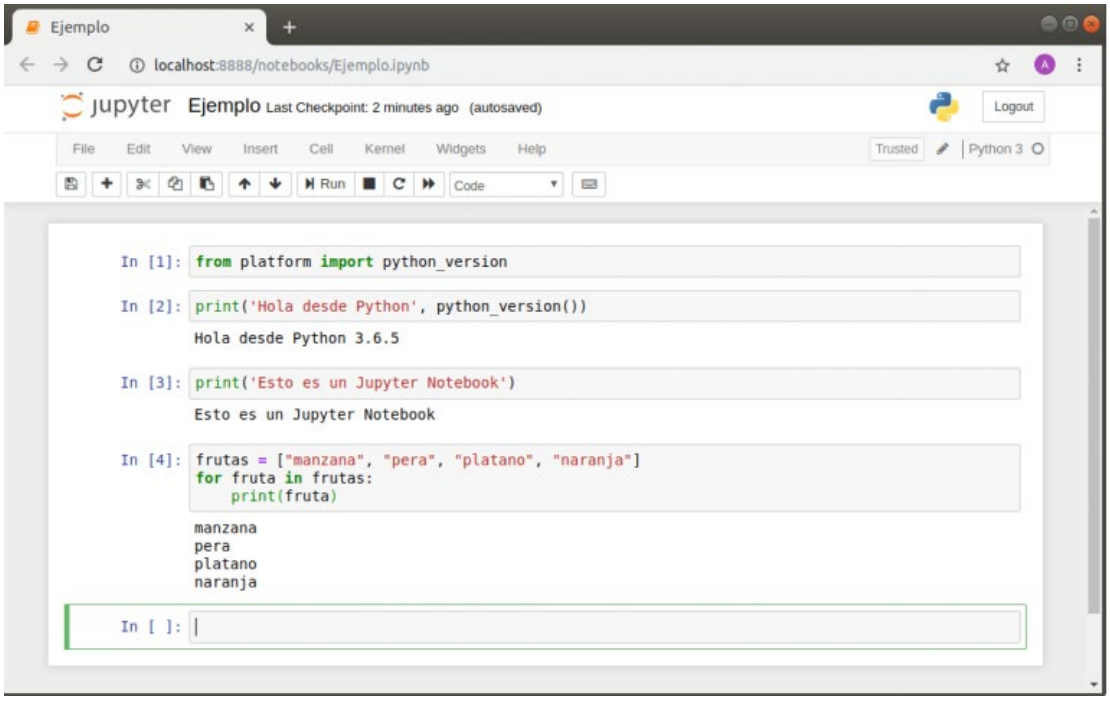


*imagen: fuente propia*

Jupyter Notebook es una aplicación cliente-servidor lanzada en 2015 por la organización sin ánimo de lucro Proyecto Jupyter. Permite crear y compartir **documentos web en formato JSON** que siguen un esquema versionado y una lista ordenada de celdas de entrada y de salida. Estas celdas albergan, entre otras cosas, código, texto (en formato Markdown), fórmulas matemáticas y ecuaciones, o también contenido multimedia (Rich Media). El programa se ejecuta desde la **aplicación web cliente que funciona en cualquier navegador estándar.** El requisito previo es instalar y ejecutar en el sistema el servidor Jupyter Notebook.

Los documentos creados en Jupyter pueden exportarse, entre otros formatos, **a HTML, PDF, Markdown o Python** y también pueden compartirse con otros usuarios por correo electrónico, utilizando Dropbox o GitHub o mediante el visor integrado de Jupyter Notebook.

Los dos componentes principales de Jupyter Notebook son un conjunto de **núcleos** (Interpreter) y el **Dashboard**. Cada núcleo o kernel es un motor de ejecución para un lenguaje que se encarga de procesar las solicitudes y devolver las respuestas apropiadas. El kernel por defecto es IPython, un intérprete de líneas de comandos que permite trabajar con Python. Gracias a los 50 kernels restantes, es posible trabajar también con otros lenguajes como C++, R, Julia, Ruby, JavaScript, CoffeeScript, PHP o Java. Por un lado, el Dashboard (panel de control) funciona como una interfaz de administración de cada uno de los kernels y, por otro, como un centro de control desde donde es posible crear nuevos documentos o abrir proyectos existentes. Jupyter Notebook está disponible bajo una **licencia BSD modificada,** por lo que cualquier usuario puede usarlo gratuitamente.

**¿Para qué se utiliza?**

Jupyter Notebook proporciona un entorno pensado para satisfacer necesidades concretas y ajustarse al **flujo de trabajo de la ciencia de datos y la simulación numérica**. En una sola interfaz, los usuarios pueden escribir, documentar y ejecutar código, visualizar datos, realizar cálculos y ver los resultados. Concretamente, la fase de prototipado incluye la ventaja de que **el código se organiza en celdas independientes**, es decir, es posible probar bloques concretos de código de forma individual. Gracias a que existen muchos kernels o núcleos adicionales, Jupyter no se limita al lenguaje de programación Python, lo que aporta muchísima flexibilidad a la hora de crear código y de hacer análisis.

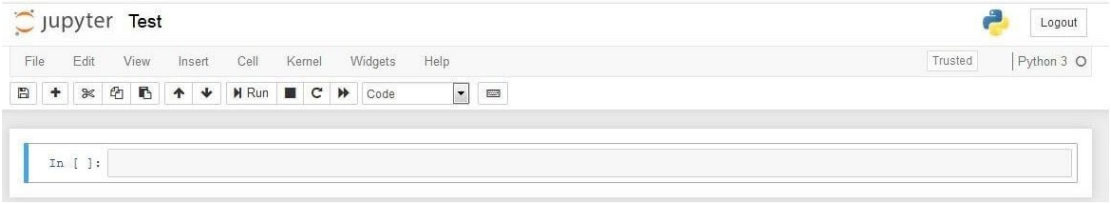
Algunos de los principales usos que se da a Jupyter Notebook:

- Depuración de datos: distinguir entre los datos que son importantes y los que no lo son al ejecutar un análisis de big data.
- Modelización estadística: método matemático para estimar la probabilidad de distribución de una característica concreta.
- Creación y entrenamiento de modelos de aprendizaje automático: diseño, programación y entrenamiento de modelos basados en aprendizaje automático
- Visualización de datos: representación gráfica de datos para visualizar con claridad patrones, tendencias, interdependencias, etc.

**¿Cómo funciona Jupyter Notebook?**

Si deseas sacar el máximo partido de todo lo que Jupyter Notebook, primero tienes que instalar la aplicación cliente-servidor del entorno de código en tu sistema (o, si lo prefieres, en la nube). El único requisito es tener instalado también la **versión actualizada de Python**. Por esta razón, el equipo de Jupyter recomienda descargar la distribución **Anaconda** que incluye Jupyter Notebook y Python, además de otros paquetes de software para ciencia de datos y cálculos científicos. Tras la instalación, se puede iniciar el servidor del Notebook utilizando la línea de comando y luego el Dashboard en el navegador que prefieras mediante el URL *'http://localhost:8888'*.

Ahí, los usuarios podrán crear nuevas carpetas en el directorio de **Jupyter Notebook,** abrir el editor de texto y el terminal integrados o crear un nuevo Proyecto Jupyter. Los proyectos de nueva creación contienen al principio nada más que un único campo de entrada vacío. Mediante la **barra del menú** se pueden añadir nuevos campos, importar bibliotecas o incrustar widgets (elementos interactivos). Además, en la barra hay **botones** que permiten ejecutar y detener los códigos ya finalizados, guardar o exportar el documento completo y seleccionar el kernel subyacente.


jupyter Test
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [ ]:

AWAKELAB

## Referencias


[1] Anaconda
https://www.anaconda.com/

[2] Instalación de Jupyter Notebook
https://www.youtube.com/watch?v=dycl0bntMwo

[3] Página Spyder
https://www.spyder-ide.org/

[4] Instalación Anaconda (Python) en Windows y empezar a programar (Spyder y Jupyter)
https://www.youtube.com/watch?v=52h3r_lROGY

[5] Curso Python para principiantes
https://www.youtube.com/watch?v=chPhlsHoEPo