SESIÓN INTRODUCCIÓN A PYTHON Y ENTORNO DE TRABAJO

CONTENIDOS:

- Reseña del Lenguaje Python.
- Propósito del Lenguaje Python.
- Principales características del Lenguaje.
- Versiones de Python.
- Entorno de trabajo y herramientas:
    - Entorno Anaconda.
    - Editor Spyder.
    - Jupyter Notebook.
    - Google Colab.
    - Visual Studio Code.

RESEÑA DEL LENGUAJE PYTHON

Python es un lenguaje de programación de alto nivel y de propósito general, creado en 1991 por el programador holandés Guido van Rossum. Su diseño se centra en la simplicidad, la legibilidad y la accesibilidad, lo que ha convertido a Python en una de las opciones preferidas para personas que se inician en el mundo de la programación, así como para desarrolladores experimentados.

A diferencia de otros lenguajes que requieren una sintaxis compleja, Python se enfoca en una estructura clara y concisa que se asemeja al lenguaje natural. Esto significa que el código en Python es generalmente más fácil de escribir y comprender, facilitando la colaboración y la revisión de proyectos en equipo. Gracias a esta claridad y facilidad de uso, Python es especialmente útil para aprender conceptos básicos de programación y lógica computacional sin sentirse abrumado por una sintaxis difícil.

Ilustración 1 Logo Python

PROPÓSITO DEL LENGUAJE PYTHON

Python fue creado con el propósito de hacer la programación más accesible y práctica para una amplia gama de usuarios, desde principiantes hasta expertos. El diseño del lenguaje prioriza la simplicidad y la claridad, facilitando que cualquiera, sin importar su experiencia, pueda aprenderlo rápidamente y comenzar a desarrollar aplicaciones útiles en poco tiempo. Su propósito no se limita a un sector o industria específica, sino que abarca una gran variedad de campos, haciendo de Python un **lenguaje de propósito general**.

Accesibilidad y facilidad de uso

Python está diseñado para ser fácil de entender y escribir, gracias a su sintaxis intuitiva y cercana al lenguaje natural. A diferencia de otros lenguajes que requieren estructuras de código complejas, Python permite escribir programas de manera más directa y sencilla. Esto lo convierte en un excelente punto de partida para quienes están aprendiendo a programar, permitiéndoles concentrarse en los conceptos fundamentales de la programación sin quedar atrapados en los detalles técnicos del lenguaje.

Adaptabilidad y uso en múltiples disciplinas

Python es uno de los lenguajes de programación más versátiles y se emplea en una variedad de sectores y aplicaciones, tales como:

- **Desarrollo web**: Python facilita la creación de aplicaciones web con frameworks como Django y Flask, que proporcionan herramientas integradas para la administración de bases de datos, la creación de interfaces, y la gestión de usuarios, entre otros.
- **Ciencia de datos y análisis**: Python es uno de los lenguajes preferidos en la ciencia de datos debido a su facilidad para manejar y analizar grandes volúmenes de datos. Herramientas como pandas, NumPy, y Matplotlib permiten trabajar con datos de forma rápida y eficaz, y hacen posible aplicar técnicas estadísticas y de visualización sin complicaciones.
- **Inteligencia artificial (IA) y aprendizaje automático (ML)**: Python es el lenguaje predilecto en el desarrollo de algoritmos de IA y ML. Con bibliotecas como TensorFlow, Keras y scikit-learn, los desarrolladores pueden construir, entrenar y evaluar modelos complejos de IA y ML, desde la detección de patrones en datos hasta la creación de redes neuronales avanzadas.

- **Automatización de tareas y scripting**: Python permite automatizar tareas repetitivas de forma sencilla. Esto es especialmente útil para trabajos administrativos o de gestión de sistemas, donde se pueden escribir scripts que realicen tareas como mover archivos, limpiar bases de datos o recolectar información de Internet.

Eficiencia y productividad en el desarrollo

Uno de los principales propósitos de Python es hacer que la programación sea productiva. Esto significa que los desarrolladores pueden trabajar con eficiencia y rapidez, incluso en proyectos complejos. La simplicidad de Python permite a los desarrolladores crear **prototipos rápidos** y realizar ajustes sin perder mucho tiempo. Esta capacidad es especialmente valiosa en proyectos que requieren experimentación, como en el desarrollo de productos de software o en investigación científica, donde los cambios rápidos y las pruebas constantes son esenciales.

Comunidad y soporte

Otro propósito fundamental de Python es fomentar una comunidad de apoyo y colaboración. Debido a que es de código abierto, Python permite a cualquier usuario contribuir a su desarrollo. Esto ha creado una comunidad global que trabaja para mejorar continuamente el lenguaje y compartir recursos de aprendizaje, herramientas y bibliotecas. Además, la comunidad mantiene una vasta cantidad de documentación y foros donde programadores de todos los niveles pueden resolver dudas, recibir apoyo y colaborar en proyectos. Esta comunidad contribuye a que Python sea un lenguaje accesible y siempre actualizado.

PRINCIPALES CARACTERÍSTICAS DEL LENGUAJE

Python presenta varias características que lo distinguen:

1. **Sintaxis sencilla y legible**: El código de Python está diseñado para ser fácil de leer, eliminando la necesidad de muchos símbolos y reduciendo la complejidad visual.
2. **Multiparadigma**: Python admite diferentes estilos de programación, incluyendo la programación orientada a objetos, funcional e imperativa. Esto permite a los desarrolladores trabajar con el estilo que mejor se ajuste a sus necesidades.

3. **Interpretado y dinámico**: Python no necesita ser compilado antes de ejecutarse, y permite declarar variables sin especificar un tipo fijo, lo que facilita una programación más rápida y flexible.
4. **Gran biblioteca estándar y extensiones**: Python incluye una amplia colección de bibliotecas estándar que cubren tareas comunes, desde manejo de archivos hasta comunicación en red. Además, existen miles de bibliotecas externas que se pueden integrar para aplicaciones específicas como el análisis de datos (por ejemplo, con pandas o NumPy).
5. **Multiplataforma**: Python se puede ejecutar en distintos sistemas operativos, como Windows, macOS y Linux, facilitando el desarrollo de aplicaciones que sean portables.

VERSIONES DE PYTHON

Desde su creación en 1991 por Guido van Rossum, Python ha pasado por varias versiones importantes, cada una de las cuales ha aportado cambios significativos al lenguaje. A lo largo de su historia, Python ha sido mejorado continuamente para adaptarse a las necesidades de sus usuarios y de la industria. A continuación, conoceremos las distintas versiones de Python:

Primeras versiones de Python (1.x)

Python comenzó su desarrollo a fines de los años 80 y fue lanzado oficialmente en su versión 1.0 en enero de 1994. Estas primeras versiones de Python (1.x) establecieron la base de muchas de las características que siguen siendo esenciales en el lenguaje. En esta etapa, Python ya era un lenguaje de propósito general, pero contaba con una sintaxis simple y orientada a objetos que lo diferenciaba de otros lenguajes de programación de la época, como Perl o Tcl. Las primeras versiones incluyeron características clave como las funciones lambda, el manejo de excepciones y los primeros pasos en la recolección automática de basura (garbage collection), que permitieron una gestión de memoria más eficiente.

Las versiones posteriores dentro de la serie 1.x, como Python 1.5 (lanzada en 1997) y Python 1.6 (lanzada en el año 2000), continuaron añadiendo mejoras y refinamientos, preparando el camino para Python 2. Sin embargo, estas versiones tenían limitaciones en cuanto a compatibilidad y rendimiento, lo que impulsó la necesidad de una versión más avanzada.

Python 2.x

Python 2 fue lanzado en octubre de 2000 y fue una versión fundamental en la historia de Python, pues introdujo numerosas mejoras y características avanzadas que hicieron que el lenguaje fuera aún más popular. Algunas de las características destacadas de Python 2 incluyen:

- **List comprehensions**: que permitían escribir listas de manera más compacta y expresiva.
- **Mejora en la gestión de excepciones**: simplificando la forma de manejar errores y hacer el código más robusto.
- **Soporte para Unicode**: que hizo posible manejar texto en diferentes idiomas, abriendo el lenguaje a un uso global.
- **Introducción de módulos importantes**: como el módulo xml para manejo de datos en formato XML, muy útil en aplicaciones web.

Python 2 fue ampliamente adoptado y se mantuvo como la versión estándar durante muchos años. Sin embargo, algunas de sus limitaciones comenzaron a hacerse evidentes, especialmente en cuanto a la consistencia y el rendimiento. Por ejemplo, Python 2 utilizaba la división entera en lugar de la división de punto flotante por defecto, lo que generaba confusión y errores en los cálculos. Además, la comunidad de Python deseaba mejorar aspectos de la sintaxis y la estructura que no podían solucionarse sin realizar cambios importantes. Esto llevó al desarrollo de una nueva versión que sería incompatible con Python 2: Python 3.

Python 3.x

Python 3 fue lanzado en diciembre de 2008 con el objetivo de corregir muchas de las limitaciones y problemas de diseño que existían en Python 2. Esta versión trajo una serie de mejoras en términos de rendimiento, legibilidad y consistencia en la sintaxis. Algunas de las mejoras clave incluyeron:

- **Nueva división de enteros y flotantes**: en Python 3, la división (/) devuelve un número decimal de forma predeterminada, mientras que la división de enteros (//) se utiliza para resultados enteros. Esto resolvió muchos problemas de precisión en los cálculos.
- **Mejor manejo de cadenas de texto**: en Python 3, todas las cadenas (str) son Unicode por defecto, facilitando el trabajo con texto en diferentes idiomas y simplificando el código.
- **Simplificación de la sintaxis**: Python 3 introdujo una sintaxis más limpia, como el uso de print() como función en lugar de comando, mejorando la legibilidad del código.

- **Optimizaciones de rendimiento**: Python 3 es más rápido y eficiente en comparación con Python 2, con mejoras en la administración de memoria y el procesamiento de datos.

Python 3 no es compatible con Python 2, lo que provocó una migración gradual. Muchas aplicaciones antiguas aún requerían Python 2, por lo que ambas versiones coexistieron durante varios años. La Fundación Python brindó soporte a Python 2 hasta el 1 de enero de 2020, fecha en la que se discontinuó oficialmente, incentivando a las empresas y desarrolladores a completar la transición a Python 3.

Actualizaciones continuas de Python 3.x

Desde su lanzamiento, Python 3 ha seguido evolucionando con actualizaciones periódicas que mejoran la eficiencia, la seguridad y las funcionalidades del lenguaje. Estas actualizaciones incluyen nuevas bibliotecas, mejoras en el rendimiento y funciones como:

- **Operadores de asignación ampliada**: como el operador :=, introducido en Python 3.8, que permite asignar valores dentro de expresiones.
- **Bibliotecas para el desarrollo de IA y ciencia de datos**: Python 3 continúa siendo el lenguaje principal en estos campos, y las actualizaciones lo mantienen a la vanguardia en cuanto a soporte de herramientas y bibliotecas.
- **Mejoras en la concurrencia y el manejo de tareas asíncronas**: para facilitar el desarrollo de aplicaciones de alto rendimiento.

La evolución de Python y su futuro

La evolución de Python refleja su capacidad de adaptarse a las necesidades de la industria y de la comunidad. A través de estas versiones y actualizaciones continuas, Python ha logrado mantenerse como uno de los lenguajes de programación más populares en el mundo. Hoy en día, Python 3 es la versión recomendada y la única que cuenta con soporte activo, mientras que la comunidad continúa trabajando en futuras mejoras que harán de Python un lenguaje aún más robusto, versátil y accesible para el desarrollo de software.

ENTORNO DE TRABAJO Y HERRAMIENTAS PARA PYTHON

Para desarrollar en Python, existen diversas herramientas y entornos que facilitan la escritura, ejecución y depuración del código. A continuación, encontrarás algunos de los entornos y editores más comunes para trabajar en Python:

Entorno Anaconda

Anaconda es una distribución de Python que incluye gran cantidad de bibliotecas y herramientas necesarias para el análisis de datos, la ciencia de datos, el aprendizaje automático y otras áreas relacionadas. Es muy útil cuando se trabaja con bibliotecas como **NumPy**, **Pandas**, **Matplotlib** y **Scikit-Learn**. Anaconda facilita la gestión de entornos virtuales, lo que permite trabajar en diferentes proyectos con distintas versiones de Python y bibliotecas sin conflictos. Además, incluye **Conda**, un administrador de paquetes y entornos.

Para instalar Anaconda:

1. Visita anaconda.com y descarga la versión correspondiente a tu sistema operativo.
2. Sigue las instrucciones del instalador y selecciona la opción de instalar Python y Conda.
    - Asegúrate de marcar la opción para agregar Anaconda al PATH (en Windows), lo cual facilita su uso desde la terminal.
3. Una vez instalado, puedes abrir Anaconda Navigator, que es una interfaz visual que permite acceder fácilmente a varias herramientas, incluidas algunas que usaremos en este curso: **Jupyter Notebook** y **Spyder**.

Editor Spyder

Spyder (Scientific Python Development Environment) es un entorno de desarrollo integrado (IDE) diseñado para científicos y analistas de datos. Es parte de la distribución Anaconda y tiene una interfaz similar a la de MATLAB, lo que facilita su uso en proyectos de ciencia de datos. Spyder permite visualizar y depurar el código de manera sencilla y tiene herramientas integradas para trabajar con datos, lo que lo hace ideal para quienes se están iniciando en el análisis de datos y desean una interfaz intuitiva.

Para abrir Spyder:

1. Instala Anaconda y abre el **Anaconda Navigator**.
2. Selecciona Spyder desde el menú principal y espera a que el entorno se inicie.

Jupyter Notebook

Jupyter Notebook es una herramienta que permite escribir y ejecutar código en Python en un formato de bloques o celdas. Es muy utilizada en el ámbito académico, científico y de análisis de datos, ya que permite combinar código, texto, imágenes y gráficos en un solo documento, lo que facilita la creación de informes interactivos y la visualización de resultados. Al ser también parte de la distribución Anaconda, Jupyter Notebook está listo para usarse una vez que se instala Anaconda.

Para abrir Jupyter Notebook:

1. Inicia **Anaconda Navigator** y selecciona Jupyter Notebook.
2. Se abrirá en tu navegador web y te permitirá crear nuevos notebooks en formato .ipynb.

Google Colab

Google Colab es un servicio gratuito de Google que permite ejecutar código Python en la nube. Es una excelente herramienta para quienes no desean instalar Python en su equipo, o para aquellos que necesitan acceso a recursos de procesamiento como GPU para tareas de aprendizaje profundo. Google Colab es especialmente útil para proyectos de ciencia de datos y aprendizaje automático. Además, los archivos de Colab se guardan en Google Drive, lo que facilita el acceso y la colaboración en proyectos.

Para comenzar a usar Google Colab:

1. Accede a Google Colab y conéctate con tu cuenta de Google.
2. Puedes cargar notebooks existentes desde Google Drive o crear uno nuevo directamente en la plataforma.

Visual Studio Code (VS Code)

Visual Studio Code (VS Code) es un editor de código gratuito y altamente personalizable de Microsoft. VS Code ofrece un soporte completo para Python a través de una extensión específica, permitiendo la depuración, el autocompletado y la gestión de entornos virtuales. Además, su interfaz es intuitiva y permite instalar muchas otras extensiones que pueden facilitar el trabajo en Python, incluyendo herramientas de análisis de código y bibliotecas populares.

Pasos de descarga e instalación de Visual Studio Code

En Windows

1. Ve a la página de descarga de Visual Studio Code.
2. Descarga el instalador para tu sistema operativo.
3. Sigue las instrucciones de instalación.
4. Una vez instalado, abre Visual Studio Code.