El lenguaje Python

M2: FUNDAMENTOS DE PROGRAMACIÓN PYTHON PARA INGENIEROS DE DATOS

|AE1: APLICAR LAS CARACTERÍSTICAS PRINCIPALES DEL LENGUAJE PYTHON Y SU ENTORNO PARA RESOLVER DISTINTAS PROBLEMÁTICAS.

alkemy

Introducción👩‍💻

Python es uno de los lenguajes de programación más populares y ampliamente utilizados en el mundo hoy en día, y su crecimiento no muestra señales de desaceleración. Este lenguaje fue creado por Guido van Rossum a finales de los años 80 y ha evolucionado para convertirse en una **herramienta versátil que es esencial en el desarrollo de software moderno.** Originalmente, Python fue diseñado con la intención de ser un lenguaje fácil de aprender y usar, ofreciendo una sintaxis limpia y legible que hace que el código sea comprensible para personas de diferentes niveles de experiencia en programación. A diferencia de otros lenguajes que pueden requerir una curva de aprendizaje pronunciada, Python es una opción accesible, permitiendo a los desarrolladores **concentrarse en la resolución de problemas sin tener que preocuparse excesivamente por la sintaxis.**

Esta lección tiene como objetivo proporcionar una introducción completa a Python, desde sus **aplicaciones prácticas en diversos campos hasta sus características principales y las herramientas necesarias para trabajar con él de manera eficiente.** Python se utiliza en ciencia de datos, desarrollo web, inteligencia artificial, automatización, y muchas otras áreas, lo que hace que sea un lenguaje valioso para cualquier profesional en tecnología. A lo largo de este manual, exploraremos cada uno de estos aspectos en profundidad, y presentaremos también las versiones más importantes de Python, señalando las diferencias clave entre Python 2 y Python 3, un tema crucial para quienes trabajan en proyectos con código heredado.

Aprendizajes esperados

Al finalizar esta lección, serás capaz de:

- Identificar y describir aplicaciones en donde Python es utilizado, como en ciencia de datos, desarrollo web, inteligencia artificial, entre otros.
- Explicar la historia y el propósito del lenguaje Python, y entender cómo su diseño y características lo han convertido en una herramienta popular en la programación moderna.
- Reconocer y diferenciar las versiones principales de Python, comprendiendo las diferencias clave entre Python 2 y Python 3, y la razón por la cual Python 3 es recomendado para nuevos proyectos.
- Comprender las principales características de Python, incluyendo su tipado dinámico, orientación a objetos, y compatibilidad multiplataforma, y cómo estas características facilitan el desarrollo de proyectos de diversas complejidades.
- Configurar y utilizar distintos entornos de trabajo en Python, tales como Anaconda, Spyder, Jupyter Notebooks, Colab y Visual Studio Code, eligiendo el entorno adecuado según el tipo de proyecto.

Desarrollo

1.1 Describir las Aplicaciones en donde puede ser Utilizado el Lenguaje Python

Python tiene aplicaciones en casi todos los campos de la tecnología moderna, lo cual es una prueba de su versatilidad. La **ciencia de datos** es una de las áreas donde Python ha ganado un lugar preeminente. Con bibliotecas como **Pandas**, **NumPy** y **Matplotlib**, Python permite a los analistas de datos realizar tareas complejas de manipulación, análisis y visualización de datos. Estas bibliotecas simplifican el procesamiento de grandes volúmenes de información y permiten crear gráficos y reportes visuales de manera sencilla. A medida que crece la demanda de expertos en ciencia de datos, Python continúa siendo el lenguaje preferido para el análisis de datos debido a su facilidad de uso y su amplia comunidad de soporte.

En el campo de la **inteligencia artificial (IA) y el aprendizaje automático (machine learning)**, Python también es ampliamente utilizado gracias a herramientas como **TensorFlow**, **Keras** y **scikit-learn**, que facilitan la creación y entrenamiento de modelos de aprendizaje automático. Estas bibliotecas han sido diseñadas para permitir que los desarrolladores creen modelos de predicción, clasificación y procesamiento de lenguaje natural sin tener que escribir algoritmos complejos desde cero. Esto hace que Python sea una opción ideal tanto para investigadores como para ingenieros de IA que necesitan prototipar rápidamente modelos y probar ideas innovadoras en un entorno accesible.

En el **desarrollo web**, Python cuenta con frameworks populares como **Django** y **Flask**. Django, por ejemplo, es un framework de alto nivel que permite desarrollar aplicaciones web completas y seguras en poco tiempo, con un enfoque en la reutilización de código y la optimización de procesos. Flask, en cambio, es un framework ligero que permite construir aplicaciones web de forma más sencilla y es ideal para proyectos pequeños o para aquellos que requieren flexibilidad en el diseño.

Además, Python es ampliamente utilizado en la **automatización de tareas**. Su sintaxis simple permite a los desarrolladores crear scripts que automatizan tareas repetitivas, optimizando flujos de trabajo y aumentando la productividad. En muchas empresas, Python se utiliza para la creación de herramientas que automatizan la gestión de archivos, la administración de sistemas y el análisis de logs, entre otros procesos.

En **desarrollo de videojuegos**, Python también tiene su lugar con herramientas como **Pygame**, que permite a los desarrolladores crear juegos simples, y **Unity** que, aunque está más orientado a C#, puede integrar scripts de Python. Si bien no es tan común en el desarrollo de juegos grandes, Python es una excelente opción para prototipos de juegos y proyectos de aprendizaje.

Python es ampliamente reconocido por su versatilidad y se aplica en múltiples campos tecnológicos, entre los cuales destacan:

- **Ciencia de Datos y Análisis de Datos**: Con herramientas como Pandas, NumPy y Matplotlib, Python facilita la manipulación, análisis y visualización de grandes volúmenes de datos, permitiendo a los analistas generar insights clave para la toma de decisiones. Además, bibliotecas avanzadas como Scikit-learn y Statsmodels permiten implementar modelos de machine learning y análisis estadístico.

```python
import pandas as pd
import matplotlib.pyplot as plt

# Crear un DataFrame de ejemplo
data = {'Ventas': [200, 300, 400, 350], 'Año': [2019, 2020, 2021, 2022]}
df = pd.DataFrame(data)

# Visualización simple
plt.plot(df['Año'], df['Ventas'], marker='o')
plt.title("Crecimiento de Ventas Anuales")
plt.xlabel("Año")
plt.ylabel("Ventas")
plt.show()
```

- **Inteligencia Artificial y Aprendizaje Automático**: En IA, Python sobresale por sus bibliotecas como TensorFlow, Keras y PyTorch, que son ampliamente utilizadas para crear redes neuronales, modelos predictivos y sistemas de procesamiento de lenguaje natural. Su estructura flexible y amigable acelera el prototipado de modelos complejos en investigación y desarrollo.

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

# Cargar datos y dividirlos en entrenamiento y prueba
data = load_iris()
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=

# Entrenar el modelo
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Realizar predicciones
print("Predicción:", model.predict(X_test[:5]))
```

- **Desarrollo Web**: Python es común en backend de aplicaciones web, especialmente con frameworks como Django y Flask, que proporcionan soluciones seguras y escalables. Django, por ejemplo, es un framework de alto nivel que permite la creación de aplicaciones robustas, mientras que Flask es más ligero y flexible, adecuado para proyectos de menor escala.

```python
from flask import Flask

# Crear una instancia de la aplicación Flask
app = Flask(__name__)

# Definir una ruta para la página principal
@app.route('/')
def home():
    return "¡Bienvenido a mi sitio web con Flask!"

# Ejecutar la aplicación
if __name__ == '__main__':
    app.run(debug=True)
```

- **Automatización y Scripting**: Uno de los usos más populares de Python es la automatización de tareas repetitivas. Scripts en Python son usados en administración de sistemas, procesamiento de archivos y automatización de pruebas, facilitando flujos de trabajo eficientes.

```python
import os

folder = '/ruta/de/tu/carpeta'
for count, filename in enumerate(os.listdir(folder)):
    new_name = f"archivo_{count}.txt"
    os.rename(os.path.join(folder, filename), os.path.join(folder, new_name))
```

- **Desarrollo de Juegos**: Aunque Python no es el principal lenguaje en esta área, herramientas como Pygame permiten a los desarrolladores prototipar videojuegos de forma sencilla, siendo una opción educativa y accesible para aprender sobre mecánicas de juego y programación en general.

```python
import pygame
pygame.init()

# Configuración de la pantalla
screen = pygame.display.set_mode((800, 600))
pygame.display.set_caption("Juego de Movimiento")

# Definir el color y la posición del cuadrado
color = (0, 128, 255)
x, y = 400, 300
speed = 5

# Bucle principal del juego
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Obtener las teclas presionadas
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        x -= speed
    if keys[pygame.K_RIGHT]:
        x += speed
    if keys[pygame.K_UP]:
        y -= speed
    if keys[pygame.K_DOWN]:
        y += speed

    # Dibujar el cuadrado en la pantalla
    screen.fill((0, 0, 0))  # Limpiar la pantalla con color negro
    pygame.draw.rect(screen, color, (x, y, 50, 50))  # Dibujar el cuadrado
    pygame.display.flip()  # Actualizar la pantalla

pygame.quit()
```

1.2 El Lenguaje Python

Python es un lenguaje de programación interpretado, lo que significa que su código es ejecutado directamente por un intérprete sin necesidad de compilarse, permitiendo una depuración más sencilla. Además, Python se caracteriza por tener un tipado dinámico y fuerte, lo que significa que no es necesario declarar el tipo de las variables antes de usarlas, pero estas no se convierten automáticamente entre tipos sin intervención explícita. Esto simplifica mucho la escritura de código, pero a la vez evita errores comunes que podrían pasar desapercibidos en lenguajes de tipado débil.

Uno de los principios fundamentales de Python es la legibilidad del código. La estructura del lenguaje es intuitiva, y utiliza indentación para delimitar bloques de código en lugar de llaves u otros símbolos. Este diseño fomenta un estilo de programación que facilita la

colaboración y el mantenimiento del código a largo plazo. Para quienes están familiarizados con otros lenguajes de programación, la sintaxis de Python puede parecer sorprendentemente sencilla al principio, pero su diseño minimalista es uno de los factores que han contribuido a su éxito y adopción masiva.

Cada entorno tiene características específicas que permiten trabajar con Python de manera eficiente, según el tipo de proyecto:

- **Anaconda**: Es una distribución de Python que incluye numerosas bibliotecas preinstaladas, ideal para ciencia de datos y machine learning. Anaconda facilita la gestión de entornos y paquetes, permitiendo a los desarrolladores crear entornos virtuales para distintos proyectos sin conflictos entre dependencias.
- **Spyder**: Un IDE optimizado para ciencia de datos, incluido en Anaconda. Su interfaz permite ejecutar scripts de manera interactiva y visualizar datos en tiempo real, lo que lo convierte en una herramienta valiosa para el análisis de datos similar a Matlab.
- **Jupyter Notebooks**: Popular entre científicos de datos, este entorno permite crear cuadernos interactivos que combinan código, texto y gráficos en un solo documento, ideal para análisis exploratorio de datos y presentación de resultados de forma narrativa y visual.
- **Google Colab**: Similar a Jupyter, pero basado en la nube, lo que permite a los usuarios ejecutar código sin requerir un equipo de alta capacidad. Colab es ideal para colaboración y trabajos en equipo, ya que permite compartir cuadernos fácilmente y aprovechar GPUs para cálculos intensivos.
- **Visual Studio Code (VS Code)**: Este editor de texto versátil, con extensiones para Python, ofrece un entorno de desarrollo completo. Permite realizar debugging, integración con control de versiones y configuración de entornos virtuales, siendo una opción flexible para desarrolladores de aplicaciones más complejas.

Para cada entorno de trabajo, incluimos detalles específicos y sugerimos en qué casos puede ser útil.

Entorno de Trabajo	Características	Ejemplo de Uso	Ventajas
Anaconda	Incluye una gran cantidad de paquetes y un gestor de entornos virtuales.	Ciencia de Datos y Machine Learning.	Evita conflictos de dependencias; ideal para análisis de datos y modelos de machine learning.

Spyder	IDE orientado a ciencia de datos, similar a Matlab.	Desarrollo en análisis numérico y data science.	Visualización directa de datos y depuración de código en tiempo real.
Jupyter Notebooks	Permite combinar texto, gráficos y código en un solo documento.	Proyectos de análisis exploratorio.	Facilita el análisis paso a paso y la presentación de resultados.
Google Colab	Basado en la nube, permite colaboración en tiempo real.	Proyectos colaborativos y ejecución de scripts intensivos con GPU.	Ejecución en la nube sin necesidad de instalación local.
Visual Studio Code	Soporta múltiples lenguajes y cuenta con depurador y extensiones.	Desarrollo de aplicaciones grandes.	Flexibilidad para integrar herramientas adicionales y personalizar el entorno.

1.3 Reseña del Lenguaje Python

Python fue lanzado en 1991 por Guido van Rossum, y su diseño fue inspirado por lenguajes como ABC, un lenguaje de programación creado para facilitar el aprendizaje de conceptos de programación. La primera versión de Python incluía características como excepciones, funciones, módulos y una amplia biblioteca estándar. En el año 2000, con el lanzamiento de Python 2, se introdujeron mejoras significativas que incluyeron una recolección de basura más eficiente y soporte para Unicode. Python 3 fue lanzado en 2008 con el objetivo de corregir inconsistencias y mejorar el rendimiento general del lenguaje. Esta versión marcó un cambio importante, ya que la sintaxis y algunas funcionalidades no eran compatibles con Python 2. A lo largo de los años, Python ha sido desarrollado y actualizado por una comunidad activa que sigue mejorando el lenguaje y ampliando su funcionalidad.

Desde su lanzamiento, Python ha experimentado un crecimiento exponencial en popularidad, impulsado en gran medida por su versatilidad y facilidad de aprendizaje. La comunidad de Python ha jugado un papel crucial en su evolución, desarrollando una gran cantidad de bibliotecas y frameworks que han expandido las capacidades del lenguaje en áreas emergentes como la ciencia de datos, la inteligencia artificial, y el

desarrollo web. Cada nueva versión de Python introduce mejoras en rendimiento, seguridad y funcionalidad, asegurando que el lenguaje se mantenga competitivo y relevante. Además, la decisión de descontinuar Python 2 en 2020 impulsó a muchos desarrolladores a adoptar Python 3, consolidando así una versión estándar para la mayoría de los proyectos modernos. Esta evolución constante y el apoyo comunitario han convertido a Python en un lenguaje confiable y flexible, utilizado tanto en entornos académicos como en aplicaciones industriales y comerciales.

1.4 Propósito del Lenguaje Python

El propósito de Python es ser un lenguaje versátil que pueda usarse en una gran variedad de aplicaciones. Esto se refleja en su diseño intuitivo, que permite a los programadores centrarse en la resolución de problemas en lugar de en la sintaxis. Python se adapta bien tanto a proyectos pequeños como a aplicaciones a gran escala, y su extensa biblioteca estándar incluye módulos para realizar tareas comunes como la manipulación de archivos, el procesamiento de texto y la gestión de redes. Python es ideal para equipos que buscan desarrollar código legible y fácil de mantener, lo que contribuye a una colaboración más fluida.

Python también se destaca por su filosofía de "baterías incluidas", lo que significa que su biblioteca estándar está equipada con módulos que cubren una amplia gama de necesidades, desde operaciones matemáticas complejas hasta gestión de datos y redes. Esta característica permite a los desarrolladores reducir la dependencia de bibliotecas externas, logrando soluciones rápidas y eficientes. Además, Python promueve prácticas de programación como la orientación a objetos, la programación funcional y el uso de excepciones para manejar errores de manera efectiva. Gracias a su comunidad activa y a la gran cantidad de recursos disponibles, Python continúa evolucionando y manteniéndose relevante en diversos sectores, desde la ciencia de datos y la inteligencia artificial hasta el desarrollo de aplicaciones web y el Internet de las cosas (IoT). Esta versatilidad, junto con su enfoque en la legibilidad y mantenibilidad del código, hace que Python sea una herramienta valiosa tanto para principiantes como para desarrolladores experimentados.

1.5 Principales Características del Lenguaje

Python es un lenguaje de tipado dinámico, lo que significa que no es necesario especificar el tipo de las variables al declararlas. Esta característica facilita la escritura de código y hace que Python sea accesible para principiantes. Otra característica importante es su orientación a objetos, lo que permite organizar el código en clases y objetos, facilitando la reutilización y la estructuración de grandes proyectos. Además, Python es multiplataforma y es compatible con Windows, macOS y Linux, lo que permite que los programas escritos en Python funcionen en diferentes sistemas operativos con pocas o ninguna modificación.

1.6 Versiones de Python

Las dos versiones principales de Python son Python 2 y Python 3. Aunque Python 2 ya no es compatible oficialmente, todavía se usa en algunos proyectos heredados. Python 3 introdujo mejoras importantes en la sintaxis y en la eficiencia, siendo la versión recomendada para proyectos nuevos. Las diferencias más notables incluyen el manejo de cadenas y las divisiones entre enteros. En Python 2, el uso de `print` requiere paréntesis, mientras que en Python 3 se introdujo como una función, lo cual estandarizó su uso. Las mejoras en la gestión de memoria y en el rendimiento hacen que Python 3 sea la mejor opción para la mayoría de los desarrolladores.

1.7 Entorno de Trabajo y Herramientas (Anaconda, Spyder, Jupyter Notebooks, Colab, Visual Studio Code)

Anaconda es una distribución que incluye Python y una gran cantidad de bibliotecas para ciencia de datos y aprendizaje automático. **Spyder** es un IDE incluido en Anaconda y se enfoca en la ciencia de datos, permitiendo ejecutar scripts y ver datos de manera similar a Matlab. **Jupyter Notebooks** es ideal para proyectos interactivos y es muy popular en ciencia de datos. **Colab**, por su parte, permite ejecutar Jupyter Notebooks en la nube y es una opción excelente para proyectos colaborativos. **Visual Studio Code** es un editor de texto versátil con extensiones para Python que ofrece funcionalidades avanzadas para el desarrollo en diversos lenguajes.

1.8 Comparación de Python con Otros Lenguajes

Python se destaca en el ámbito de la programación moderna por su simplicidad y versatilidad, lo que le permite adaptarse a una amplia gama de aplicaciones, desde ciencia de datos hasta desarrollo web e inteligencia artificial. Sin embargo, existen otros lenguajes populares como Java y C++ que también son ampliamente utilizados y que presentan características únicas.

Esta sección tiene como objetivo comparar a Python con algunos de estos lenguajes, resaltando sus diferencias en áreas clave como la sintaxis, el manejo de tipos de datos, y los casos de uso típicos. A través de esta comparación, veremos cómo Python se posiciona como una opción accesible y poderosa en contraste con la robustez de Java o la eficiencia de C++.

La siguiente tabla resume las principales diferencias y ventajas de cada lenguaje, proporcionando una visión clara de en qué contextos podría ser beneficioso utilizar uno u otro.

Funcionalidad	Python	Java	C++
Tipado	Dinámico	Estático	Estático

Sintaxis	Simple, legible	Más detallada, basada en clases	Compleja, basada en punteros
Automatización	Alto soporte (p.ej., scripts)	Moderado	Bajo
Uso en IA	Extensivo (TensorFlow, Keras)	Limitado	Muy limitado
Desarrollo Web	Django, Flask	Spring	Poco usado
Compatibilidad Multiplataforma	Amplia	Amplia	Amplia

Cierre ✍️

Este manual ha ofrecido una visión completa sobre el lenguaje Python, cubriendo desde sus aplicaciones y características hasta las herramientas y entornos más populares. Al finalizar esta lección, los lectores estarán preparados para comenzar a utilizar Python en proyectos diversos, aprovechando su simplicidad y poder. Python es un lenguaje que continúa evolucionando, y su comunidad activa sigue contribuyendo al crecimiento del ecosistema, lo que garantiza que Python siga siendo una opción relevante y poderosa para desarrolladores de todo el mundo.

Referencias📚

1. Documentación oficial de Python (https://docs.python.org/)
2. "Learning Python" de Mark Lutz (O'Reilly Media)
3. Real Python, tutoriales avanzados y básicos (https://realpython.com/)
4. Comparaciones de versiones en Python.org (https://www.python.org/doc/versions/)
5. GeeksforGeeks, artículos sobre Python y aplicaciones (https://www.geeksforgeeks.org/python-programming-language/)

¡Muchas gracias!

Nos vemos en la próxima lección