IntroduccionPython

January 7, 2026

Introducción a Python

Nombre: Alejandro Bedoya

Principales características

Python es un lenguaje fuertemente tipado que permite crear aplicaciones web, de escritorio, ciencia de datos, automatización, inteligencia artificial y videojuegos. Además, es interpretado, lo que significa que un traductor convierte el código en lenguaje máquina sin necesidad de compilar.

Python no necesita declarar el tipo de variable, ya que lo detecta automáticamente por su tipado dinámico. A la vez, es fuertemente tipado, por lo que no mezcla tipos de datos distintos. Por ejemplo: "5" + 3 generará un error.

Su código es muy legible, lo que lo convierte en un lenguaje de alto nivel. Python gestiona la memoria automáticamente, aunque para tareas de alto rendimiento se apoya en bibliotecas escritas en C++.

Python es multiparadigma, permitiendo programar con programación orientada a objetos, funciones o estilo procedural.

Los bloques de código se definen mediante indentación y no con llaves. Además, es multiplataforma y cuenta con una extensa biblioteca estándar para trabajar con archivos, fechas, web, matemáticas y bases de datos.

¿Por qué utilizar Python?

Python es uno de los lenguajes más usados en el mundo y se aplica en áreas como ciencia de datos, educación, gestión de sistemas, cine, inteligencia artificial y desarrollo de software.

Cuenta con una comunidad muy activa y un gran ecosistema de librerías, lo que permite resolver problemas sin empezar desde cero y aumenta considerablemente la productividad del desarrollador.

Python se integra fácilmente con otras herramientas y lenguajes, soporta varios estilos de programación, es multiplataforma, portable y puede compilarse para mejorar el rendimiento.

¿Qué se puede hacer con Python?

Se pueden crear herramientas tipo shell para administración de sistemas, automatizar tareas, manipular archivos y ejecutar comandos directamente desde los programas.

Python también se utiliza para el desarrollo de interfaces gráficas (GUI), permitiendo crear aplicaciones con ventanas, botones y formularios de forma sencilla.

Es muy fuerte en el área de IoT y redes, trabajando con XML y JSON, obteniendo información desde páginas web, comunicándose con sockets y transfiriendo archivos vía FTP.

En bases de datos facilita la conexión, consultas y administración de información en MySQL, PostgreSQL y MongoDB.

En el campo científico destaca en computación numérica con NumPy, análisis de lenguaje natural, machine learning e inteligencia artificial.

Es ampliamente utilizado en análisis de datos, álgebra lineal, modelado estadístico, visualización, business intelligence y almacenamiento de información.

The Zen of Python — Tim Peters

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one - and preferably only one - obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than right now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea - let's do more of those!

Básicamente lo que nos quiere decir es que Python promueve escribir código claro, simple y legible, priorizando la belleza y la claridad sobre la complejidad innecesaria. Es mejor ser explícito que implícito, mantener soluciones sencillas y evitar estructuras complicadas. Además, los errores no deben ignorarse, se debe evitar adivinar cuando algo es ambiguo y debe existir una forma clara y única de hacer las cosas, usando correctamente los namespaces para mantener el código ordenado.

Intérprete de Python y Ejecución de Scripts

Antes de Comenzar

Se pueden utilizar diferentes formas para ejecutar código:

Línea de comandos, Shell de Python o modo interactivo (python / ipython)

IDE: Eclipse, PyCharm, Sublime, Nano, VSCode, Atom, Spyder

Google Colab, Azure Notebooks, Jupyter

¿Qué es un intérprete?

Es el traductor entre tú y la computadora: el programa que lee tu código y lo ejecuta línea por línea, convirtiéndolo en acciones que la computadora entiende directamente.

Existen varias implementaciones:

CPython (implementación en C). Es el más común.

Jython (implementación en Java).

IronPython (implementación en .NET).

Perspectiva de Python

Compilación de código fuente a bytecode

Python traduce tu código a bytecode y lo guarda como archivos .pyc para no tener que volver a traducirlo cada vez.

Python Virtual Machine (PVM)

Ejecuta las instrucciones en bytecode.

¿Cómo puedes ejecutar tus scripts?

Línea de Comandos

Ejecuta python archivo.py o py archivo.py.

También puedes abrir Python desde el menú de inicio.

Cuando ves » estás en una sesión interactiva.

Es útil para debug y pruebas rápidas.

```
print('Hello world')
```

console:

Hello world

Inconveniente: los programas ejecutados en la línea de comandos desaparecen tras finalizar.

REPL (READ, EVALUATE, PRINT AND LOOP)

Es un sistema interactivo para comunicarse con un lenguaje (Python en este caso) directamente con el ordenador.

READ – Permite al ordenador leer la entrada.

EVALUATE – El código es procesado.

PRINT – Muestra los resultados en pantalla.

LOOP – Continúa con la conversación.

Ficheros

Permiten guardar nuestros programas para poder ejecutarlos después. Generalmente son archivos de texto con extensión .py. En Linux es importante incluir al inicio del archivo el #!/usr/bin/env python3.

Un archivo puede llamarse script o programa cuando es el principal, y módulo cuando se importa usando import.

Para ejecutar un fichero se pasa su nombre al comando python. Por ejemplo: python hola_soy_alejandro.py

En Linux, si el archivo tiene permisos de ejecución y el shebang correcto, se puede ejecutar directamente con ./script1.py. También es posible ejecutar el archivo haciendo doble clic.

Instalación de Librerías

Se realiza con el comando pip install libreria. Para gestión de paquetes y entornos virtuales se usa pipenv install entorno + libreria.

[5]:
```
# Para comentar una linea, se utiliza #
# """
# Para comentar en bloque, se usan las 3 comillas, solo que ahorita no vale
# """
print("Siempre vamos a poder 'poner' los comentarios en forma de salida para␣
 ↪poder ver los resultados ")
ITQ = "Instituto Superior Tecnológico Quito"
print(ITQ)
```

```
Siempre vamos a poder 'poner' los comentarios en forma de salida para poder ver
los resultados
Instituto Superior Tecnológico Quito
```

Performance Sample, NumPy vs Plain Python

[6]:
```
import numpy as np #Importamos la librería de numpy para poder medir por los␣
 ↪arrays
my_arr = np.arange(10000000) #Utilizamos Numpy Arrays

my_list = list(range(10000000)) #Utilizamos listas

#es un comando de Jupyter para medir cuánto tarda el bloque.
#Ejecuta 10 veces la operación
#NumPy multiplica todos los valores de una sola vez usando código optimizado en␣
 ↪C.
%time for _ in range(10): my_arr2 = my_arr * 2

#También se ejecuta 10 veces.
#Aquí Python tiene que recorrer elemento por elemento con un for.
#Esto es mucho más lento porque cada operación se hace en Python puro.
%time for _ in range(10): my_list2 = [x * 2 for x in my_list]
```

```
CPU times: total: 344 ms
Wall time: 344 ms
CPU times: total: 5.94 s
Wall time: 6.01 s
```

GitHub

Ver cuaderno en GitHub

En caso que no se vea en el PDF: https://github.com/Addrriel/MachineLearningCuadernazo/blob/main/clases/Int

[ ]: