Tema 2:

Introducción al lenguaje Python (2h)

1. Introducción al lenguaje Python (4h)

1.1. Breve historia de Python

1.2. Filosofía y principios (*The Zen of Python*)

1.3. ¿Por qué Python?

1.4. Ventajas y desventajas frente a otros lenguajes (C, Java, JavaScript, etc.)

1.5. Principales áreas de aplicación (web, IA, ciencia de datos, automatización, etc.)

1.6. Parte práctica (ejemplos y ejercicios)

1.7. Distribución sugerida del tiempo

Profesor: Salvador Martínez Bolinches

Centro: IES Font de Sant Lluís

Año: 2025

Objetivos de aprendizaje

- Reconocer el origen, filosofía y principios de diseño de Python.
- Comparar ventajas y limitaciones frente a otros lenguajes.
- Identificar ámbitos de aplicación habituales y no habituales.
- Utilizar el intérprete de forma elemental y evidenciar el *Zen of Python* en micro-tareas.
- Argumentar, con criterio técnico, decisiones básicas de adopción del lenguaje.

1.1. Breve historia de Python

Python es un lenguaje de programación de **alto nivel**, **interpretado**, **multiplataforma** y de propósito general, creado por **Guido van Rossum** en 1991.

Su nombre no proviene de la serpiente, sino del grupo humorístico británico *Monty Python*. Guido buscaba un nombre corto, original y llamativo, que rompiera con los formalismos de otros lenguajes.

Línea temporal destacada

- **1991**: Publicación de la primera versión (Python 0.9.0).
- **2000**: Nace **Python 2.0**, con nuevas características, pero con problemas de compatibilidad futura.
- **2008**: Se lanza **Python 3.0**, no compatible con Python 2.
- **2020**: Fin del soporte oficial para Python 2.
- **Actualidad**: Python 3.x es el estándar, con versiones actualizadas cada 12-18 meses.

👉 Idea clave: **Python 3 es el presente y futuro del lenguaje.**

1.2. Filosofía y principios de Python (*The Zen of Python*)

El Zen de Python es un conjunto de principios que guían el diseño del lenguaje. Se puede consultar ejecutando en el intérprete:

```
import this
```

Algunos de los aforismos más relevantes:

- "Beautiful is better than ugly" (La belleza es mejor que la fealdad).
- "Simple is better than complex" (Lo simple es mejor que lo complejo).
- "Readability counts" (La legibilidad importa).
- "There should be one– and preferably only one –obvious way to do it." (Debería haber una, y preferiblemente solo una, forma evidente de hacerlo).

👉 Esto convierte a Python en un lenguaje **claro, legible y accesible** para principiantes, sin dejar de ser potente para expertos.

1.3. ¿Por qué Python?

Python se ha convertido en uno de los lenguajes más populares del mundo gracias a su:

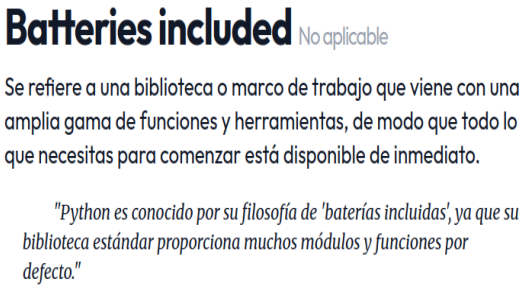
- **Simplicidad sintáctica** → muy parecido al lenguaje natural.
- **Gran comunidad** → abundancia de tutoriales, foros y documentación.
- **Multiplataforma** → funciona en Windows, Linux, macOS y otros sistemas.
- **Versatilidad** → se usa en ámbitos tan diversos como:
    - Desarrollo web (*Django*, *Flask*).
    - Ciencia de datos y estadística (*pandas*, *NumPy*).
    - Inteligencia artificial y machine learning (*TensorFlow*, *PyTorch*).
    - Automatización de tareas (*scripting*).
    - Programación de videojuegos (*pygame*).

📊 Según el índice **TIOBE (2023)**, Python suele aparecer en el **top 3 de lenguajes más usados del mundo**.

1.4. Ventajas y desventajas frente a otros lenguajes

Ventajas

- Sintaxis clara y concisa.
- Orientado a la productividad → menos líneas de código.
- Amplia librería estándar ("baterías incluidas").
- Comunidad activa y en crecimiento.
- Alta demanda en el mercado laboral.

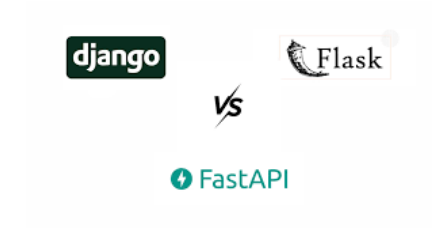


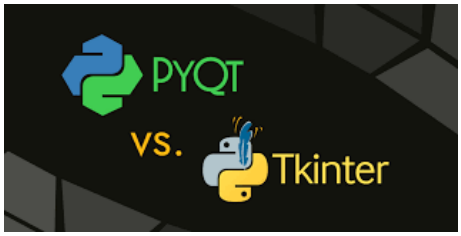
Desventajas

- Rendimiento inferior a lenguajes compilados (C, C++).
- No es el mejor para aplicaciones móviles nativas.
- Consumo de memoria más elevado.
- Fragmentación en cuanto a versiones (aunque cada vez menos).

1.5. Principales áreas de aplicación

- **Desarrollo web**: *Django*, *Flask*, *FastAPI*.
- **Ciencia de datos y análisis**: *pandas*, *NumPy*, *Matplotlib*.
- **Inteligencia artificial y machine learning**: *scikit-learn*, *TensorFlow*, *PyTorch*.
- **Automatización / scripting**: escribir pequeños programas que ahorran trabajo repetitivo.
- **Administración de sistemas**: scripts para gestión de servidores.
- **Aplicaciones gráficas y de escritorio**: *Tkinter*, *PyQt*.
- **Programación educativa**: ideal para iniciarse en la programación.





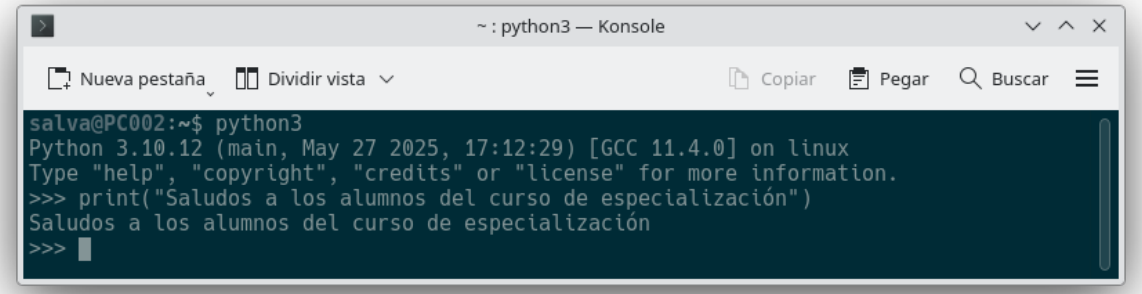
Un Python REPL (Read-Eval-Print Loop o Ciclo de Lectura-Evaluación-Impresión) es una consola interactiva que te permite ejecutar código Python línea por línea y ver los resultados de forma inmediata. Es un entorno donde puedes escribir fragmentos de código, que el REPL lee, los evalúa (ejecuta), imprime el resultado en la consola, y luego repite el proceso en un ciclo, permitiéndote interactuar continuamente con el intérprete de Python.

1.6. Parte práctica (ejemplos y ejercicios)

Ejemplo 1: Primer programa en Python

```
print("Hola, mundo")
```

Este ejemplo clásico ilustra cómo en Python no se necesitan librerías adicionales para imprimir en pantalla.



Ejemplo 2: Comparación con Java (simplicidad sintáctica)

En Java:

```
public class HolaMundo {
    public static void main(String[] args) {
        System.out.println("Hola, mundo");
    }
}
```

En Python:

```
print("Hola, mundo")
```

👉 Conclusión: Python es más conciso y legible.

Ejercicios propuestos

1. Abre el intérprete interactivo, ejecuta `import this`, y comenta qué principios del **Zen de Python** te parecen más relevantes para aprender a programar.
2. Elaborar un cuadro comparativo con las ventajas y desventajas de Python frente a otros lenguaje de programación (Java, JavaScript, C++, C#, …).
3. Buscar un caso real de uso de Python en la industria (ej. Netflix, Google, Spotify, NASA…).
4. Define “alto nivel”, “interpretado” y “multiparadigma”. Coloca un ejemplo de cada característica aplicada a Python.
5. Enumera **cinco** áreas en las que Python es común (p. ej., web, automatización…). Añade **un** ámbito donde **no** suele ser la primera opción y justifica por qué.
6. Define “multiplataforma” y explica qué consecuencias prácticas tiene al distribuir un script entre Windows, Linux y macOS.
7. Describe brevemente qué significa que Python tenga tipado **dinámico** y **fuerte**. Incluye un ejemplo breve (máx. 3 líneas) que lo ilustre.
8. Explica el lema “batteries included” (Baterías incluidas) y nombra **tres** módulos de la librería estándar que te parezcan útiles (sin usarlos aún).
9. Enumera **tres** beneficios concretos de una comunidad amplia (soporte, paquetes, documentación) y **una** posible desventaja (p. ej., fragmentación de soluciones).
10. Indica **dos** motivos por los que un script “portátil” podría fallar en otro sistema (p. ej., rutas, locales, dependencias externas) y cómo preverlos.

1.7. Distribución sugerida del tiempo (2h)

- **Teoría (1h)** → historia, filosofía, ventajas/desventajas, aplicaciones.
- **Ejemplos guiados (1h)** → primeros programas y comparaciones.