Herramientas para Programar en Python

Entornos de Desarrollo Integrado (IDEs) y Editores de Código

Prof. D.Sc. BARSEKH-ONJI Aboud
Facultad de Ingeniería
Universidad Anáhuac México

8 de diciembre de 2025

Agenda de la Sesión

1. ¿Por qué Aprender a Programar?
2. ¿Cómo 'Entiende' la Máquina a Python?
3. Introducción a Python
4. Herramientas para Escribir Python (IDEs)
5. Recomendación para la Clase

Agenda

1. ¿Por qué Aprender a Programar?
2. ¿Cómo 'Entiende' la Máquina a Python?
3. Introducción a Python
4. Herramientas para Escribir Python (IDEs)
5. Recomendación para la Clase

El Diálogo entre Humanos y Computadoras

El Problema Fundamental

Las computadoras son increíblemente poderosas, pero solo entienden un lenguaje: el **código binario**.

Lenguaje de Máquina (Binario)

```
01101011 01101011
11000000 00110101
10111101 00100100
```

Esto es incomprensible y casi imposible de escribir para un humano.

La Solución: Lenguajes de Programación

Son lenguajes formales que actúan como un **puente**. Nos permiten escribir instrucciones de forma legible para nosotros, que luego son traducidas a un formato que la computadora puede ejecutar.

Niveles de Abstracción

No todos los lenguajes son iguales

Los lenguajes de programación se clasifican por su **nivel de abstracción**, es decir, qué tan lejos están del hardware de la computadora.

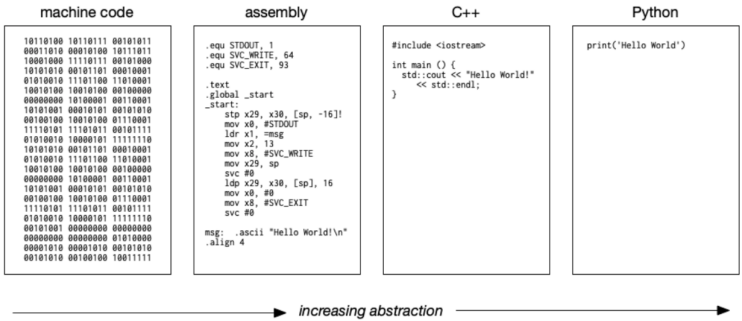

Figura 1: Aumento de la abstracción desde el código máquina hasta Python.

Niveles de Abstracción

Lenguajes de Alto Nivel (como Python)

Nos permiten expresar ideas complejas de forma simple, sin preocuparnos por los detalles de la máquina (registros, memoria, etc.).

Agenda

1. ¿Por qué Aprender a Programar?
2. ¿Cómo 'Entiende' la Máquina a Python?
3. Introducción a Python
4. Herramientas para Escribir Python (IDEs)
5. Recomendación para la Clase

Traduciendo nuestras ideas a código máquina

Compilación (Ej: C++)

- Un **compilador** traduce **todo** el código fuente a un archivo ejecutable en código máquina **antes** de que el programa se ejecute.

Traduciendo nuestras ideas a código máquina

Compilación (Ej: C++)

- Un **compilador** traduce **todo** el código fuente a un archivo ejecutable en código máquina **antes** de que el programa se ejecute.
- El resultado es un programa muy rápido, pero específico para una arquitectura (ej. Windows x86).

Interpretación (Ej: Python)

- Un **intérprete** lee el código fuente línea por línea y lo ejecuta **al momento**.

Traduciendo nuestras ideas a código máquina

Compilación (Ej: C++)

- Un **compilador** traduce **todo** el código fuente a un archivo ejecutable en código máquina **antes** de que el programa se ejecute.
- El resultado es un programa muy rápido, pero específico para una arquitectura (ej. Windows x86).

Interpretación (Ej: Python)

- Un **intérprete** lee el código fuente línea por línea y lo ejecuta **al momento**.
- No se crea un archivo ejecutable final, lo que lo hace más flexible y portable.

Nota

La distinción real es más compleja. Python, de hecho, usa un paso intermedio.

¿Qué pasa al hacer clic en 'Run'?

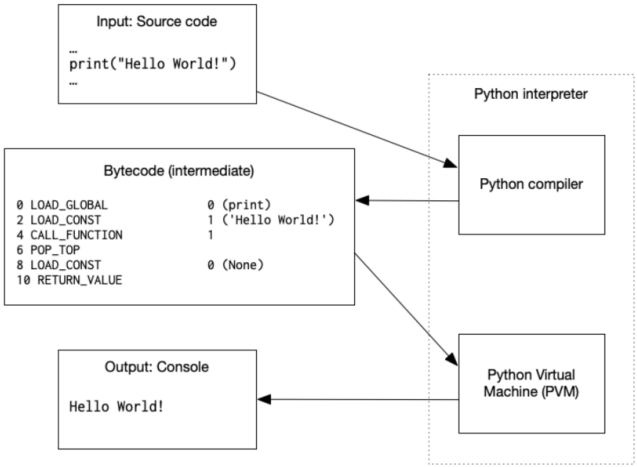

Figura 2: Ejecución de un programa en Python.

¿Qué pasa al hacer clic en 'Run'?

De Código Fuente a Ejecución

Python combina lo mejor de ambos mundos con un proceso de dos pasos.

1. El intérprete lee tu archivo `.py` y lo traduce a un código intermedio llamado **bytecode**.

¿Qué pasa al hacer clic en 'Run'?

De Código Fuente a Ejecución

Python combina lo mejor de ambos mundos con un proceso de dos pasos.

1. El intérprete lee tu archivo `.py` y lo traduce a un código intermedio llamado **bytecode**.
2. Este bytecode (que es portable) es ejecutado por la **Máquina Virtual de Python (PVM)**.

Agenda

1. ¿Por qué Aprender a Programar?
2. ¿Cómo 'Entiende' la Máquina a Python?
3. Introducción a Python
4. Herramientas para Escribir Python (IDEs)
5. Recomendación para la Clase

¿Por qué elegimos Python?

Figura 3: Guido van Rossum, creador de Python.

Filosofía de Diseño

La filosofía de Python enfatiza la **legibilidad** y la **simplicidad**. El objetivo es permitir a los programadores escribir código claro y lógico.

Características Clave

- **Sintaxis simple:** Se parece al inglés, facilitando el aprendizaje.
- **Versatilidad:** Útil para desarrollo web, ciencia de datos, IA, automatización y más.
- **Gran comunidad:** Enorme cantidad de librerías y frameworks (NumPy, Pandas, Django).

Popularidad y Aplicaciones

Python en el Mundo Real

Python se ha convertido en el lenguaje de facto en muchas de las áreas tecnológicas más importantes.

Principales Campos de Aplicación

- Ciencia de Datos y Machine Learning
- Desarrollo Web (Backend con Django/Flask)
- Automatización de Tareas (Scripting)
- Cómputo Científico y de Ingeniería

Popularidad y Aplicaciones

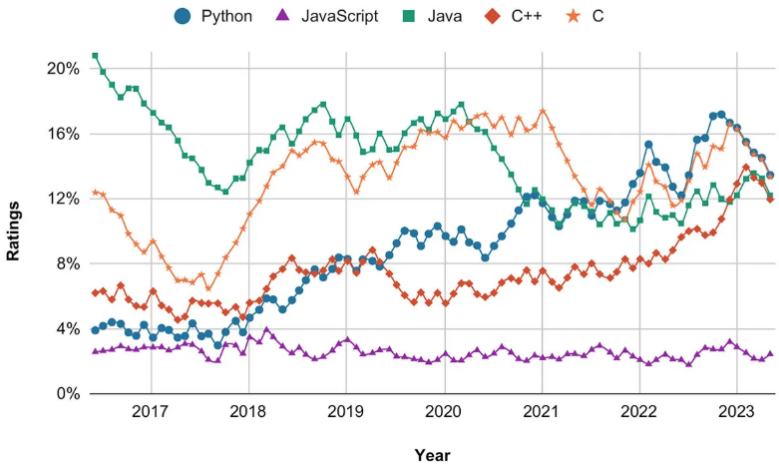

Figura 4: Popularidad de lenguajes (Índice TIOBE).

Agenda

1. ¿Por qué Aprender a Programar?
2. ¿Cómo 'Entiende' la Máquina a Python?
3. Introducción a Python
4. Herramientas para Escribir Python (IDEs)
5. Recomendación para la Clase

¿Qué es un Entorno de Desarrollo Integrado (IDE)?

El desafío

Escribir código involucra muchas tareas: editar, ejecutar, encontrar errores, instalar librerías... Hacerlo con herramientas separadas es ineficiente.

¿Qué es un Entorno de Desarrollo Integrado (IDE)?

El desafío

Escribir código involucra muchas tareas: editar, ejecutar, encontrar errores, instalar librerías... Hacerlo con herramientas separadas es ineficiente.

La solución: Un Entorno de Desarrollo Integrado (IDE)

Un IDE es una aplicación que **consolida todas las herramientas esenciales** para el desarrollo de software en una sola interfaz gráfica.

Componentes Clave de un IDE

1. Editor de Código Fuente

Un editor de texto 'inteligente'.

- Resaltado de sintaxis
- Autocompletado
- Formateo automático

2. Depurador (Debugger)

La herramienta más importante para encontrar y corregir errores.

- Puntos de ruptura (*breakpoints*)
- Ejecución paso a paso
- Inspección de variables

3. Automatización y Ejecución

- Botón de 'Play' para correr el script.
- Consola integrada para ver la salida.

Elegir la herramienta adecuada para el trabajo

Plataforma/Editor	Nivel	Ideal para...	Ventajas Principales	Desventajas Principales
`Google Colab`	Fácil	Machine Learning, colaboración	Cero instalación, GPU gratis	Dependencia de internet, temporal
`Thonny`	Muy Fácil	Principiantes absolutos	Simple, depurador visual	Limitado para proyectos grandes
`Jupyter Notebook`	Fácil	Ciencia de datos, experimentación	Interactivo, visualización	No ideal para software robusto
`VS Code`	Intermedio	Desarrollo general, personalización	Versátil, ligero, extensiones	Requiere configuración inicial
`Spyder`	Fácil-Inter.	Ingeniería, computación científica	Similar a MATLAB, expl. de variables	Interfaz densa para principiantes
`PyCharm (Comm.)`	Inter.-Avan.	Desarrollo profesional	Muy potente, autocompletado	Curva de aprendizaje, pesado

Agenda

1. ¿Por qué Aprender a Programar?
2. ¿Cómo 'Entiende' la Máquina a Python?
3. Introducción a Python
4. Herramientas para Escribir Python (IDEs)
5. Recomendación para la Clase

¿Qué usaremos en este curso?

Nuestra Elección: La Distribución Anaconda

Para asegurar que todos tengamos el mismo entorno y minimizar problemas de instalación, usaremos la **distribución de Anaconda**.

¿Por qué Anaconda?

- Es una instalación única que incluye:
  - El lenguaje **Python**.
  - **Jupyter Notebook** y **Spyder**.
  - Cientos de librerías científicas preinstaladas (NumPy, Pandas, Matplotlib).
- El **Anaconda Navigator** nos da una interfaz gráfica para lanzar las aplicaciones.

Pasos para la Instalación

1. Ir a la página oficial de descargas de Anaconda:
   https://www.anaconda.com/download

Pasos para la Instalación

1. Ir a la página oficial de descargas de Anaconda:
   `https://www.anaconda.com/download`
2. Elegir la opción **'Distribution Installers'** (la versión completa), no Miniconda.

Pasos para la Instalación

1. Ir a la página oficial de descargas de Anaconda:
   https://www.anaconda.com/download
2. Elegir la opción **'Distribution Installers'** (la versión completa), no Miniconda.
3. Descargar el instalador gráfico para tu sistema operativo (Windows, macOS, Linux).

Pasos para la Instalación

1. Ir a la página oficial de descargas de Anaconda: https://www.anaconda.com/download
2. Elegir la opción **'Distribution Installers'** (la versión completa), no Miniconda.
3. Descargar el instalador gráfico para tu sistema operativo (Windows, macOS, Linux).
4. Ejecutar el instalador y seguir las instrucciones. Se recomienda dejar las opciones por defecto.

Examples

¡Y eso es todo! Estarás listo para nuestra primera práctica de programación.

¿Preguntas?