

Programación en Python: Estadística a través de problemas resueltos

Manuel José Martínez-Santaolalla Martínez
Isabel María del Águila Cano

Programación en Python: Estadística a través de problemas resueltos

texto:

Manuel José Martínez-Santaolalla Martínez
Isabel María del Águila Cano

Textos Docentes n.º 200

edición:

Editorial Universidad de Almería, 2025

editorial@ual.es

www.ual.es/editorial

Telf/Fax: 950 015459

α

ISBN: 978-84-1351-388-1



Esta obra se edita bajo una licencia Creative Commons
CC BY-NC-ND (Atribución-NoComercial-Compartirigual) 4.0 Internacional



En este libro puede volver al índice
pulsando el pie de la página

Índice

| | |
|---|-----------|
| Prefacio | 1 |
| Introducción a la Programación | 3 |
| Conceptos básicos de la informática | 3 |
| Ejecución de programas. Lenguajes de Programación. | 9 |
| Instrucciones básicas | 17 |
| Identificadores y variables | 18 |
| Primer ejemplo en Python | 20 |
| Tipos de datos simples | 23 |
| Tipos de datos compuestos | 32 |
| Instrucción de asignación | 44 |
| Instrucción de entrada y salida | 46 |
| Funciones integradas e importadas | 49 |
| Diseño y Control de Flujo en Programación | 52 |
| Elaboración de soluciones a problemas. Algoritmos. | 52 |
| Tipos de instrucciones de control del programa | 56 |
| Instrucción secuencial | 56 |
| Instrucción selectiva. Condicionales | 58 |
| Instrucción iterativa. Bucles | 70 |
| Problemas resueltos con iteraciones | 81 |
| Técnicas y herramientas de iteración | 86 |
| Criterios de calidad de los programas | 93 |
| Funciones. Diseño Modular | 97 |
| Módulos y estructura general de un programa | 98 |
| Comunicación entre funciones | 102 |
| Funciones importables | 109 |
| Generación de números aleatorios | 110 |
| Validación de entrada con funciones y excepciones | 116 |
| Problemas resueltos con funciones | 118 |
| Recursividad | 124 |

| | |
|---|------------|
| Herramientas para estadística, probabilidad y tratamiento de datos | 127 |
| Cálculo de estadísticos en Python sin uso de librerías | 127 |
| Librería itertools: combinaciones y permutaciones | 134 |
| Librería statistics | 139 |
| Librerías de terceros | 143 |
| NumPy (Numerical Python) | 145 |
| Pandas (Panel Data Analysis) | 151 |
| Matplotlib: visualización de datos | 160 |
| | |
| Casos aplicados de análisis de datos y la probabilidad: de estadística descriptiva a la inferencia estadística | 178 |
| Uso de librerías Python para la estadística | 179 |
| Asimetría y curtosis | 181 |
| Regresión lineal | 182 |
| Probabilidad | 183 |
| Distribuciones de probabilidad | 190 |
| Ley de los sucesos raros o ley de las probabilidades pequeñas. (Aproximación de la Binomial a la Poisson) | 193 |
| Teorema Central del Límite | 199 |
| Teorema de Moivre-Laplace | 201 |
| Intervalos de confianza y contraste de hipótesis | 203 |
| Resolución con Pandas: Series y Dataframe | 207 |
| Tratamiento de datos en un caso aplicado | 212 |

Prefacio

Después de varios años impartiendo la asignatura “Estadística e Informática” en la Universidad de Almería, en particular encargándonos del bloque correspondiente a Informática, hemos llegado a una conclusión que se ha ido reforzando curso tras curso: la necesidad de disponer de un material didáctico específico que, por una parte, sirva como introducción a la Un modo puede calcularse de foprogramación para estudiantes sin experiencia previa y, por otra, permita abordar la implementación de conceptos estadísticos mediante el lenguaje Python, ya sea de forma nativa o con el apoyo de librerías especializadas.

Este libro nace, por tanto, de una necesidad tanto práctica como docente. Queríamos ofrecer un recurso que integrara dos mundos que muchas veces se presentan por separado: la lógica y estructura de la programación, y la aplicación directa de estos conocimientos a problemas reales de estadística. En un contexto donde la capacidad de analizar datos y automatizar procesos se vuelve cada vez más relevante, aprender a programar ya no es solo una habilidad complementaria, sino una herramienta esencial.

Hemos elegido Python no solo por su creciente popularidad en la comunidad científica y académica, sino también por su sencillez, claridad sintáctica y su enorme ecosistema de librerías. Su curva de aprendizaje moderada lo convierte en un lenguaje ideal para introducirse en la programación, y su potencia lo hace igualmente válido para abordar problemas complejos de análisis de datos y es una herramienta clave en diversas disciplinas.

La estructura del libro refleja esta doble vocación. Los primeros capítulos se centran en una introducción progresiva a la programación estructurada: desde los tipos de datos básicos, pasando por estructuras de control, hasta llegar al diseño modular con funciones. Esta base permite que el lector pueda enfrentarse con confianza a problemas más complejos.

A medida que se avanza, el texto se adentra en la implementación de técnicas estadísticas, comenzando con cálculos simples sin librerías, para luego introducir herramientas especializadas como `itertools`, `statistics`, `numpy`, `pandas` o `matplotlib`. Al final, se presentan casos aplicados en los que se pone en práctica todo lo aprendido, abordando temas como la regresión lineal, la probabilidad, el Teorema Central del Límite, y los intervalos de confianza, entre otros.

Nuestro objetivo ha sido que este libro sea práctico y accesible, tanto para estudiantes universitarios

como para cualquier persona interesada en iniciarse en la programación y la estadística aplicada. Cada concepto va acompañado de ejemplos resueltos y simulaciones que permiten consolidar lo aprendido y, sobre todo, ver cómo la teoría se transforma en soluciones reales mediante código.

Esperamos que estas páginas sirvan no solo como material de estudio, sino también como una experiencia y punto de partida para que el lector descubra el valor de la programación como herramienta esencial para entender mejor el mundo a través de los datos.

Introducción a la Programación

Aunque no es estrictamente necesario para aprender a programar, es útil comprender qué significa realmente la computación o la programación, así como conocer los componentes y el funcionamiento básico de una computadora. Por ello, antes de adentrarse en la construcción de programas informáticos, este libro resume brevemente estos conceptos para luego pasar a los fundamentos propios de la programación de computadoras usando Python.

Conceptos básicos de la informática

La **informática** según la RAE es

Conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de los ordenadores.

Se puede considerar la unión de dos elementos: INFORmación + autoMÁTICA.

En literatura anglosajona nos encontramos con los términos: Computer Science (ciencia de los computadores), Computer Engineering o IT Engineering. En literatura francófona: Informatique.

La **programación** según la RAE es

Acción o efecto de programar

Programar según la RAE es:

1. Formar programas, previa declaración de lo que se piensa hacer y anuncio de las partes de que se ha de componer un acto o espectáculo o una serie de ellos.
2. Idear y ordenar las acciones necesarias para realizar un proyecto.
3. *Preparar ciertas máquinas o aparatos para que empiecen a funcionar en el momento y en la forma deseados.*
4. *Elaborar programas para su empleo en computadoras.*

Máquinas programables

Una **máquina** es un cierto dispositivo físico capaz de realizar un cierto trabajo u operación, bien de forma manual (una garrucha) o bien de forma automática (un ascensor).

Este concepto puede extenderse a cuando se consideran máquinas que no existen físicamente, pero en las que puede describirse, concebirse y simular su comportamiento, se denominan **máquinas virtuales**.

En el caso del ascensor su funcionamiento está fijado en el momento de su construcción dependiendo de los dispositivos y conexiones entre ellos.

Existen otras máquinas automáticas (que actúan por si solas) denominadas **máquinas programables**, cuyo comportamiento fijo se completa con un *programa* que permite modificar el comportamiento de la máquina base. Tal como se muestra en la siguiente figura, basta con cambiar el programa para tener una nueva máquina con un comportamiento diferente.

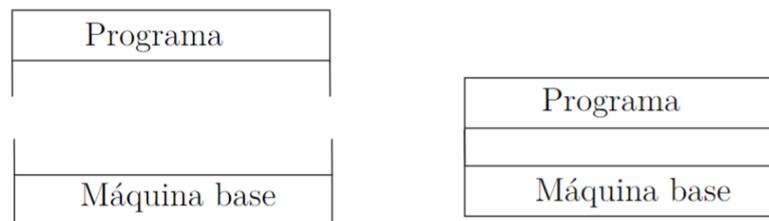


Figura 1: Máquinas programables

Concepto de cómputo

La definición de diccionario de cómputo indica que es el *cálculo destinado a obtener el resultado o valor de algo*.

Por lo tanto, un cómputo implica el procesamiento de información, que tradicionalmente es numérica, aunque puede extenderse a otro tipo de información como texto o sonido.

Cualquier cómputo se puede describir de diferentes maneras. Por ejemplo, el cómputo para **Calcular la nota final** de un alumno que ha realizado tres exámenes a lo largo del curso, pero sabiendo que si en uno de los exámenes tiene un cero, entonces su nota es cero podría hacerse de dos formas:

- Una opción es utilizar la siguiente fórmula

$$Nota \approx \sqrt[3]{nota_1 * nota_2 * nota_3}$$

- Pero también se puede describir como un proceso

```
1. Nota = 0;
2. Si nota1 no es cero Nota = Nota + nota1
   - Sino Nota = 0 e ir a paso 5
3. Si nota2 no es cero Nota = Nota + nota2
   - Sino Nota = 0 e ir a paso 5
4. Si nota3 no es cero Nota = Nota + nota3
   - Sino Nota = 0 e ir a paso 5
5. Nota = Nota / 3
```

Concepto de computador

Un **computador** es una máquina programable para el tratamiento de información, es decir realiza cálculos. Posee unos elementos **fijos** o máquina de base llamada **hardware** y otros modificables que son los programas o **software**.

Los computadores actuales son máquinas con programa almacenado de forma que la modificación de los programas no implica la modificación de los elementos físicos, y por tanto, se necesitan sistemas de almacenamiento y de comunicación del ordenador con el entorno.

Un **computador o computadora** es una máquina formada por elementos de tipo **electrónico**, capaces de aceptar unos datos de **entrada**, realizar con ellos gran variedad de tareas (**operaciones**) y proporcionar la información resultante a través de un medio de **salida**, bajo el control de un **programa** previamente **almacenado** en el propio computador. Esto puede incluir tanto dispositivos físicos como servidores, laptops, tablets, entre otros, así como también máquinas virtuales desplegadas en la nube.

Las **máquinas virtuales** en la nube son entornos computacionales virtuales que se ejecutan en infraestructuras de computación remota y están disponibles a través de Internet. Estas máquinas virtuales proporcionan capacidades informáticas similares a las de una computadora física, pero con la flexibilidad y escalabilidad adicionales ofrecidas por la computación en la nube. Por tanto, más que hablar de elementos electrónicos se tiene que hablar de **unidades funcionales**, ya que no tienen porque existir asignadas físicamente a una máquina y que son redimensionables. Se definen o alquilan componentes/unidades lógicas sobre servidores remotos para crear computadoras personalizadas virtuales. Es decir, la máquina base se define de forma virtual.

Un **programa** es la descripción de un cómputo en un lenguaje de programación, que como se ha visto en el ejemplo del cálculo de la nota puede tener muchas formas diferentes, aunque este libro se basa en la definición de programas como un conjunto de **instrucciones**, la llamada **programación estructurada**.

Estructura de una computadora

Los componentes electrónicos de una computadora son cada día más complejos, y muchos de ellos incluso se duplican. Existen numerosas combinaciones de dispositivos/componentes físicos que pueden ensamblarse para crear computadoras. Independientemente de las diferencias físicas, los ordenadores pueden dividirse en varias **unidades lógicas** o secciones, que pueden o no definirse en la nube como los componentes de una máquina virtual. Estas unidades son: Unidades de entrada, de salida, de memoria, de control, aritmético lógica y de memoria secundaria.

Unidad de entrada

Esta unidad obtiene la información (datos y programas informáticos) de los dispositivos de entrada y la pone a disposición de las demás unidades para su procesamiento. Los computadores reciben la mayoría de las entradas de los usuarios a través de teclados, pantallas táctiles, ratones y touchpads, aunque también pueden utilizar otros dispositivos como escáneres, micrófonos, cámaras web, sensores biométricos y lectores de códigos de barras.

Unidad de salida

Esta unidad envía la información que el computador ha procesado a uno o varios dispositivos de salida para que esté disponible fuera del ámbito del computador, y para ser usada directamente por el usuario o bien por otro computador. La mayor parte de la información sale de los ordenadores a través de las pantallas o se reproduce en audio o vídeo sobre teléfonos inteligentes, tabletas, ordenadores y pantallas gigantes en estadios deportivos, o muy habitualmente se transmite por Internet. Otras formas de salida son la vibración de teléfonos inteligentes, mandos de juegos y dispositivos de realidad virtual.

También es habitual que la información se transmita a dispositivos de almacenamiento secundario, como las unidades de estado sólido (SSD), los discos duros, memorias o lápices USB y unidades de DVD. Son unidades donde la información se hace **persistente**, es decir, no se pierde aunque se apague la máquina.

Unidad de memoria

Esta unidad es un almacén de acceso rápido y capacidad relativamente baja que mantiene la información introducida a través de la unidad de entrada, haciéndola disponible cuando sea necesario. La unidad de memoria también guarda la información procesada hasta que puede ser colocada en la unidad de salida.

La información de la unidad de memoria es volátil. Se pierde cuando se apaga el ordenador. La unidad de memoria suele denominarse memoria principal, memoria primaria o RAM (Random Access Memory).

Se divide en posiciones (**palabras de memoria**) de un determinado tamaño. Se accede a cada posición a través de un número (**dirección de memoria**).

Unidad Central aritmético lógica (ALU)

Esta unidad fabrica datos, es decir, realiza cálculos (suma, resta, multiplicación y división) y toma decisiones (por ejemplo, comparar dos elementos de la unidad de memoria para determinar si son iguales). En los sistemas actuales, la ALU forma parte de la siguiente unidad lógica.

Unidad central de proceso

Esta unidad es la que coordina y supervisa el funcionamiento de las demás unidades. La unidad central de proceso (CPU) indica:

- a la unidad de entrada, cuándo leer información y llevarla a la unidad de memoria,
- a la ALU cuándo utilizar la información de la unidad de memoria en los cálculos, y
- a la unidad de salida cuándo enviar datos desde la memoria a los dispositivos de salida,
- cuándo y cómo se accede a la memoria.

La mayoría de los ordenadores actuales tienen procesadores multinúcleo que implementan de forma económica múltiples procesadores en un único chip de circuito integrado. Estos procesadores pueden realizar muchas operaciones simultáneamente. Un procesador de doble núcleo tiene dos CPU, un procesador octa-core tiene ocho. Intel tiene algunos procesadores con hasta 72 núcleos.

Unidad de almacenamiento secundario

Es la unidad encargada de guardar la información de forma permanente, a largo plazo y con gran capacidad. Se considera tanto un dispositivo de entrada como de salida, ya que permite tanto almacenar como recuperar información. Los programas y datos que son utilizados por las demás unidades del sistema se transfieren a dispositivos de almacenamiento secundario (también conocidos como memoria secundaria) cuando no están en uso activo, y pueden permanecer allí durante horas, días, meses o incluso años antes de volver a ser requeridos. Por esta razón, también se le denomina memoria masiva.

La característica principal del almacenamiento secundario es su persistencia: la información no se pierde al apagar el computador, a diferencia de lo que sucede con la memoria primaria. Aunque acceder a esta información toma más tiempo debido a su menor velocidad de lectura y escritura, su gran ventaja es el bajo coste por unidad de almacenamiento.

Entre los dispositivos de almacenamiento secundario más comunes se encuentran las unidades de estado sólido (SSD), las memorias flash USB o lápices de memoria, los discos duros mecánicos (HDD), así como también las unidades ópticas como los discos Blu-Ray que permiten lectura y escritura de datos.

Además, en la actualidad, existen soluciones de almacenamiento secundario virtuales o en la nube, que permiten almacenar y acceder a datos sin necesidad de que estén físicamente en el computador. Estos servicios en línea hacen posible compartir, respaldar y recuperar archivos desde cualquier lugar con conexión a Internet.

Representación de la información

De las definiciones de **dato**, según la RAE, son del contexto de la computación la 1 y la 3, en concreto la tercera solamente es relativa a la informática.

1. m. Información sobre algo concreto que permite su conocimiento exacto o sirve para deducir las consecuencias derivadas de un hecho. A este problema le faltan datos numéricos.
2. m. Documento, testimonio, fundamento.
3. m. *Inform.* Información dispuesta de manera adecuada para su tratamiento por una computadora.

Más específicamente un **dato** en informática es: una representación formalizada de hechos o conceptos susceptible de ser comunicada o procesada.

Existen diversos **tipos de datos**, y por tanto, su representación es diferente:

- Numéricos (12, 28.5): reales o enteros
- Alfabéticos (Ana)
- Alfanuméricos: 23456X, M-6995
- Imágenes, sonido, video.

Se llama **representación de la información** a la forma en la que se almacenan y recogen los datos para poder ser procesados en la computadora. Se debe tener en cuenta que en los medios electrónicos de procesamiento solamente disponen de **dos estados**, interruptor abierto-cerrado.

Esto se traslada a que la información procesada por un computador son siempre unos y ceros. Lo que lleva a la necesidad de **traducir** cualquier dato a una combinación de esos dos símbolos es decir **notación binaria** o sistema de numeración binario.

| | | | | | | | | |
|------------|---|---|----|----|-----|-----|-----|-----|
| Nº decimal | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Nº binario | 0 | 1 | 10 | 11 | 100 | 101 | 110 | 111 |

Observando la tabla se comprueba que, por ejemplo, para representar el “6” en decimal son necesarias tres posiciones en binario. Cada una de estas posiciones es un **bit**, siendo la unidad más pequeña de información un uno o un cero. Cabe decir que en la computación cuántica se amplía este concepto incorporando la idea de que cada bit puede ser cero, uno, o no conocerse su valor.

Veáse un ejemplo: representar en binario todos los posibles estados emocionales de una persona.

Estados = { Tranquilo, Feliz, Emocionado, Preocupado, Triste, Enojado, Sorprendido, Nervioso, Cansado, Motivado }

Son necesarias cuatro cifras binarias. 2^4 es el máximo número de elementos a representar, en este caso, ejemplo hay 10 estados.

Un **byte** es un conjunto de 8 bits. También se llama Octeto o Carácter (porque con un byte se codifica un carácter). Ésta es la unidad de almacenamiento de información en informática (por ejemplo, tamaño de memoria o tamaño de almacenamiento secundario) que siempre se mide en potencias de dos.

Los **qubits** (cúbit) pueden representar simultáneamente 0 y 1. El estado de un qubit puede depender del estado de otro, lo que proporciona una mayor capacidad de procesamiento y almacenamiento de información.

Múltiplos del byte

| acrónimo | nombre | equivalencia |
|----------|----------|--|
| KB | Kilobyte | 2^{10} bytes = 1024 bytes $\approx 10^3$ bytes |
| MB | Megabyte | 2^{20} bytes = 1048576 bytes $\approx 10^6$ bytes |
| GB | Gigabyte | 2^{30} bytes = 1073741824 bytes $\approx 10^9$ bytes |
| TB | Terabyte | 2^{40} bytes $\approx 10^{12}$ bytes |
| PB | Petabyte | 2^{50} bytes $\approx 10^{15}$ bytes |
| EB | Exabyte | 2^{60} bytes $\approx 10^{18}$ bytes |

Ejecución de programas. Lenguajes de Programación.

Desde el punto de vista del software almacenado en la memoria de una computadora se pueden observar distintas capas que definen su comportamiento.

- Software de base o programas del sistema, son programas que gestionan el funcionamiento de la computadora:
 - *Sistema Operativo* - Pertenece a los componentes fijos de un máquina dada. Porque si bien es un elemento lógico, es necesario para que la máquina funcione. Un móvil o un portátil sin sistema operativo no es más que un bonito pisapapeles.
 - *Utilidades para construcción/ejecución de aplicaciones*. Cabe destacar aquí el concepto de **lenguaje máquina**, puesto que cada modelo de computadora física tiene una forma de entender los programas que depende de cuál sea su arquitectura y sus componentes. El lenguaje máquina es el sistema de codificación que tiene cada computadora/procesador para almacenar los programas, que lo define el fabricante de la máquina y se almacena en la propia máquina.
- Software de aplicación son los programas que se cargan sobre la memoria para ejecutar una tarea.

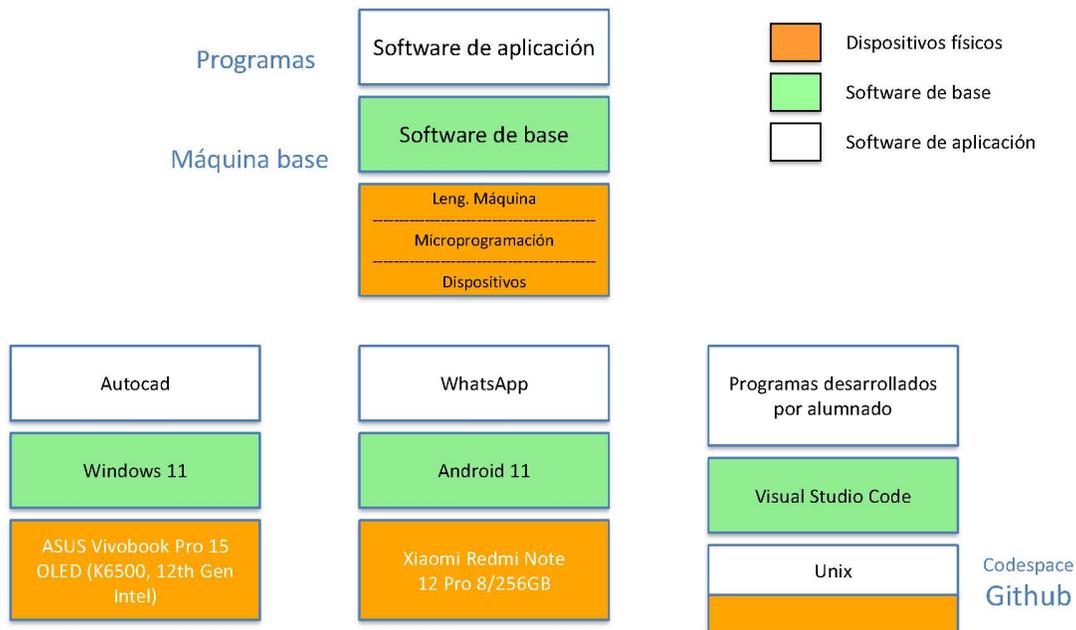


Figura 2: Distintas configuraciones de máquinas base y programas

En general cualquier software es una lista de instrucciones máquina que al ejecutarse producen un resultado concreto. Deben estar expresados en un lenguaje que entienda la máquina, o bien los programadores deben escribirlos en algún lenguaje de más alto nivel (lenguaje de programación) que ha de ser traducido a lenguaje máquina utilizando algún tipo de software de base como son los compiladores o los intérpretes.

Todo programa comienza con idea, algo que se quiere hacer. Generalmente ese algo se plantea como solución a un problema específico. La elaboración de esta solución requiere el diseño de un **algoritmo**.

Los algoritmos son soluciones abstractas a problemas, generalmente son codificados posteriormente en un lenguaje de programación, y luego se traducen al lenguaje máquina que es el único que una computadora puede ejecutar. Con sus resultados, este algoritmo en forma de programa solucionará el problema real para el que se concibió. Los algoritmos son independientes del lenguaje de programación y de la máquina que lo ejecute.

Una analogía de la vida real sería cuando un cocinero con vista cansada quiere hacer una receta. La receta de un plato de cocina (algoritmo) puede expresarse en inglés, francés o español (lenguaje), e indicará la misma preparación independientemente del cocinero (máquina). Después, el cocinero concreto que lea la receta, debe poder entender lo escrito y como el cocinero es corto de vista, necesita sus gafas (compilador) para que le traduzcan los garabatos a letras. Sin gafas el cocinero solamente ve garabatos y necesita que algo se los traduzca (gafas).

Un programa contiene la información codificada del comportamiento deseado o descrito en el algoritmo. Cada modelo de computadora podrá utilizar una forma particular de codificación dependiendo de sus dispositivos físicos, no es lo mismo un móvil que una computadora de sobremesa. La forma de codificar programas de una máquina particular se llama código máquina o lenguaje máquina. Los programas en lenguaje máquina no son legibles, y se suelen llamar **programas objeto** (obj), siendo el resultado de la traducción de un programa escrito en un lenguaje de programación.

Los lenguajes de programación sirven para representar programas de forma simbólica en forma de texto, abstrayéndose del lenguaje máquina que es lo realmente ejecutable. Un **lenguaje de programación** es un idioma artificial diseñado para que sea fácilmente entendible por un humano y traducible por una máquina (empleando una pieza de software de base que lo traduzca). También son llamados lenguajes de alto nivel.

Un programa escrito en un lenguaje de alto nivel está formado por símbolos tomados de un determinado repertorio (componentes **léxicos**). Se construyen siguiendo unas reglas precisas (**sintaxis**). Incorpora unas reglas de **semántica** que determinan el significado de cada construcción.

Tipos de lenguaje de programación

- Lenguajes de marcado: No son considerados lenguajes de programación como tal. Permiten añadir códigos en formato texto (marcas) para indicar cómo se debe mostrar la información. Ejemplos: markdown, latex, HTML.
- Lenguajes de programación. Es un conjunto de reglas y símbolos que permiten a un programador **escribir instrucciones** que una computadora puede entender y ejecutar. Hay muchos tipos y de muchas categorías. Se clasifican según su:
 - *Nivel de abstracción* teniendo tanto lenguajes de **bajo nivel**, en los que se programa sobre el hardware (i.e. ensambladores); como lenguajes de **alto nivel**, donde hay una serie de símbolos y reglas que definen instrucciones más complejas.
 - *Estilo de programación*: funcional, orientada a objetos, declarativa, guiada por eventos, concurrente e **imperativa**. En esta última es donde hay que definir el detalle de lo que hay que hacer siendo el estilo empleado en este libro sobre Python.
 - *Dominio de aplicación*. Existiendo lenguajes especializados en aplicaciones: científico tecnológicas, gestión de la información, inteligencia artificial, programación de sistemas, o aplicaciones para la Web.
 - *Ámbito de uso*: Web, móviles, tradicional (equipos fijos y procesamiento local) y/o hardware (Programación de sistemas, diseño de circuitos, . . .)
 - *Forma de traducción*: Compiladores o Intérpretes.

Traductores: Compiladores e intérpretes

Hay diferentes estrategias para poder conseguir/ejecutar el lenguaje máquina a partir de un programa escrito en un lenguaje de programación (también llamado programa fuente). Se utiliza un software que ha de procesar el programa. Estos **procesadores de lenguajes o traductores** manipulan la descripción simbólica de un algoritmo escrito en un lenguaje de programación para obtener el código objeto que ya si es ejecutable en la máquina, como las gafas del cocinero. Existen dos categorías en los traductores:

- **Compilador:** una vez traducido el programa fuente, su ejecución (programa objeto) es independiente del compilador (se traduce una vez y se ejecuta múltiples veces).
- **Intérprete:** el programa fuente se traduce instrucción a instrucción cada vez que se ejecuta (no se crea el programa objeto).

Algunos lenguajes de programación

Python

- Se diseñó para ser un lenguaje muy legible, utilizando palabras clave en inglés donde otros lenguajes usan signos de puntuación, facilitando la comprensión y mantenimiento del código.
- Multiparadigma soportando por ejemplo la programación orientada a objetos, la programación imperativa y, en menor medida, la programación funcional.
- Multiplataforma porque es compatible con múltiples sistemas operativos, como Windows, macOS y diversas distribuciones de Linux, permitiendo su ejecución en diferentes entornos sin necesidad de modificaciones en el programa.
- Cuenta con una gran cantidad de librerías y paquetes que extienden sus funcionalidades. Una librería es un conjunto de módulos que ofrecen herramientas reutilizables para diversas tareas, como matemáticas, manipulación de datos, desarrollo web, inteligencia artificial, entre otras. Ejemplos destacados son *Numpy* para cálculo numérico y *Pandas* para análisis de datos.
- Interpretado: se requiere tener Python disponible para ejecutar las instrucciones.

C

- Utilizado para programación de sistemas.
- Gestión de memoria - Soporta la característica de asignación dinámica de memoria.
- Muy eficiente, de hecho la librería Numpy de Python está escrita en C.
- Estándar ISO/IEC 9899:2018 (C18)
- Familia de lenguajes C: C++ y C#
- Compilado. desde el archivo fuente `.c` se genera un ejecutable `.exe`. Aunque suele ser transparente desde los entornos de construcción de programas, la generación del ejecutable tiene dos fases:

- Compilar: Crea los objetos (`.o` o `.obj`).
- Enlazar o Linkar: Une los objetos para crear ejecutables. Busca librerías y las añade si es necesario.

Java

- Lenguaje portable, basado en C.
- Utilizado para aplicaciones en internet.
- A la vez compilado e interpretado. Con el compilador se traducen archivos fuente `.java`, a un conjunto de instrucciones llamados bytecodes, en un archivo `.class` que son independientes del tipo de ordenador. El intérprete ejecuta estas instrucciones en un ordenador específico (máquina virtual java).

Hola Mundo

Un programa es por tanto un conjunto de **“texto”** o **bloque de código** escrito con unas normas de sintaxis marcadas por el lenguaje de programación que busca hacer algo y que se guarda en un archivo plano (`.c`, `.py`, `.java`). Sea el siguiente ejemplo:

Problema: Escribir un mensaje de saludo al usuario en la pantalla

Solución: Cada lenguaje tiene una forma de expresarlo. Nótese que cada lenguaje tiene su forma de organizar los bloques de código e incorporar comentarios. Por ejemplo, en C se utiliza `/*` y `*/`, en Java se utiliza `//` para los comentarios y en el caso de Python se usa `#`.

Solución en ensamblador

```
section .data
    msg db "Hola Mundo!", 0ah
section .text
    global _start
_start:
    mov rax, 1
    mov rdi, 1
    mov rsi, msg
    mov rdx, 13
    syscall
    mov rax, 60
    mov rdi, 0
    syscall
```

Solución en C

```
/* Programa que muestra un mensaje en pantalla */
#include <stdio.h>

int main() {
    printf("Hola Mundo!\n");
    return 0;
}
```

Solución en Java

```
// Programa que muestra un mensaje en pantalla
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {
        System.out.println("Hola Mundo!");
    }
}
```

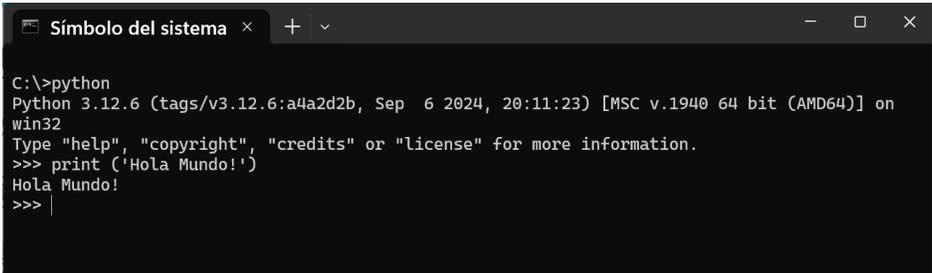
Solución en Python

```
# Programa que muestra un mensaje en pantalla
print('Hola Mundo!')
```

Pero ¿cómo se **ejecuta** este código? Tal como ya se ha dicho, algunos programas tienen que ser compilados y otros interpretados por un traductor del lenguaje concreto. En el caso de C, se debe compilar el archivo que contiene el código para obtener un archivo ejecutable que se podrá usar sin que el compilador de C esté instalado en la misma máquina. Sin embargo, en Python, bien sea teniendo el código en un archivo `.py` o bien escribiendo directamente las instrucciones, para poder “saludar” es imprescindible tener un intérprete Python disponible en la máquina donde se ejecute.

El código Python (siempre con el intérprete presente) se ejecuta de diversas formas. Tres de los modos básicos de uso de Python son el modo interactivo, el modo script y los cuadernos.

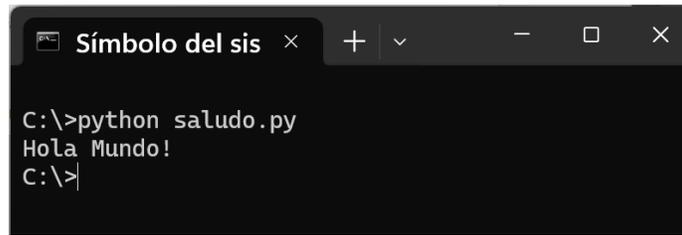
En el **modo interactivo**, introduciendo directamente fragmentos de código Python viendo sus resultados inmediatamente. El símbolo `>>>` indica que el intérprete está cargado.



```
Símbolo del sistema x + v - □ x
C:\>python
Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep 6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hola Mundo!')
Hola Mundo!
>>> |
```

Figura 3: Ejecución en modo interactivo

En el modo **script**, se ejecuta el código cargando desde un archivo con la extensión `.py` que lo contiene, son los llamados scripts, que podrían traducirse como secuencia de instrucciones, aunque no suele hacerse y el nombre en inglés es el más utilizado. El código, en modo texto, se coloca en un archivo llamándolo por ejemplo `saludo.py`. Este archivo se puede lanzar bien desde entornos de programación como **Visual Studio Code**, **Anaconda** o **WinPython** o directamente desde la consola llamando al intérprete, `python saludo.py`.



```
Símbolo del sis x + - □ x
C:\>python saludo.py
Hola Mundo!
C:\>
```

Figura 4: Ejecución en modo script sobre consola

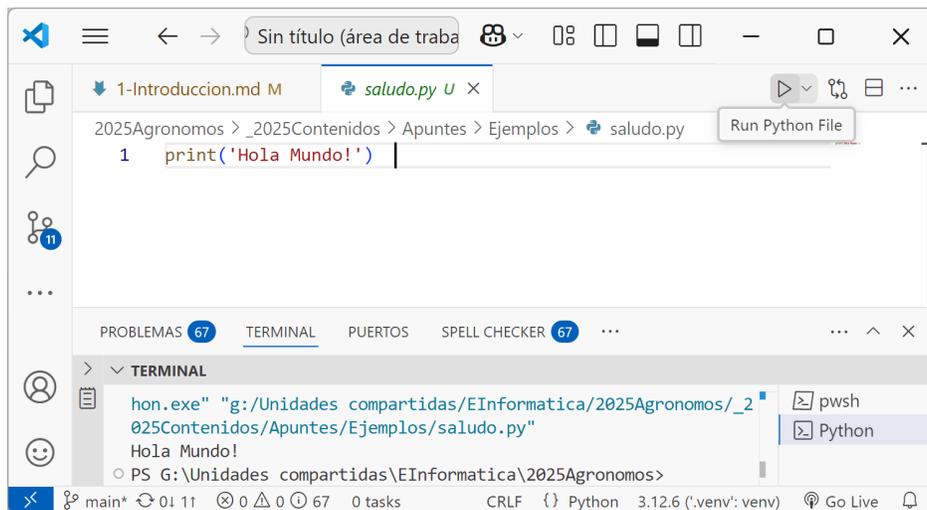


Figura 5: Ejecución en modo script desde Visual Studio Code

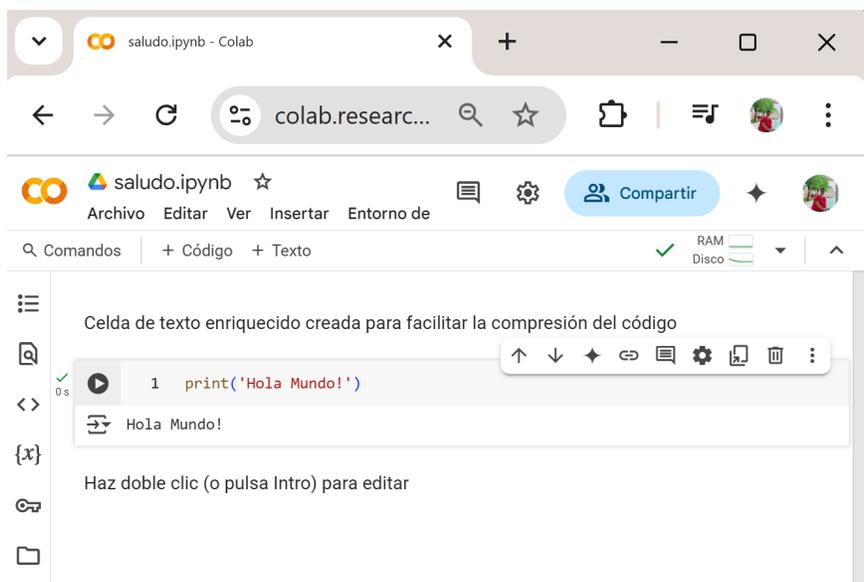


Figura 6: Cuaderno jupyter sobre Colab

Finalmente, sobre todo para soluciones finalistas, se pueden ejecutar fragmentos (chunks o snippets) de código Python sobre **cuadernos** Jupyter (Jupyter Notebook), en especial sobre **Colab de Google**, que es un servicio de alojamiento de cuadernos que no requiere configuración y que ofrece acceso sin coste a recursos de computación. Los cuadernos permiten combinar elementos de texto enriquecido (formato, tablas, figuras, ecuaciones, enlaces, etc.), código y el resultado de ejecutar el código en un único documento.