

# Parte 1. Fundamentos de la programación

Unai Pérez-Goya

Área de Lenguajes y Sistemas Informáticos

Curso 2024/2025

## Parte 1.

# Fundamentos de la programación

Sesión I: Introducción a los computadores y Lenguajes de programación

# Tabla de contenidos

[¿Por qué informática?](#)

[Historia de los Computadores](#)

[Ordenadores actuales](#)

[Lenguajes de programación](#)

[Lenguaje de programación python](#)

[Resumen](#)

# ¿Por qué informática?



# ¿Por qué informática?

- ▶ Es una competencia que actualmente se requieren en cualquier profesión.
- ▶ En agricultura de precisión.
  - ▶ En grandes parcelas: En la imagen por satélite para el seguimiento de las parcelas, seguimiento de la humedad y del estado de las plantas.
  - ▶ Parcelas medianas: Drones para seguimiento.
  - ▶ Pesticida y riego específico dependiendo del estado de la Tierra
  - ▶ Seguimiento digital de cada planta en explotaciones ecológicas
- ▶ Investigación: procesar datos.
  - ▶ Procesar los datos de un experimento, sobre todo cuando la cantidad de datos es grande.

# ¿Dónde se puede aplicar?



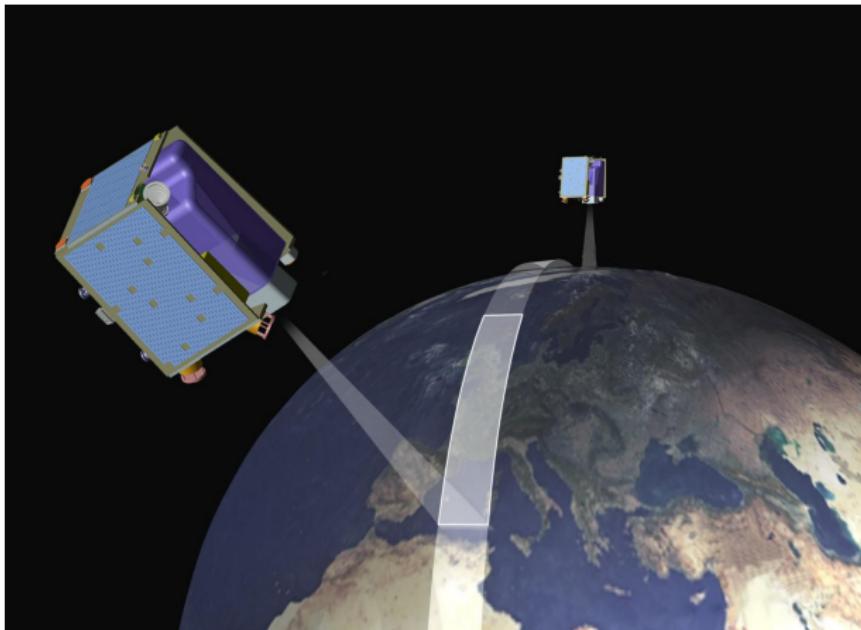
**Figura:** Cadena industrial de procesamiento de té

# ¿Dónde se puede aplicar?



**Figura:** Cadena industrial de procesamiento de botellas

# ¿Dónde se puede aplicar?



**Figura:** Satélite Rapid Eye obteniendo información

# ¿Dónde se puede aplicar?



**Figura:** Cuenca de Pamplona en color derivado del satélite Sentinel-2.

# ¿Dónde se puede aplicar?



**Figura:** Índice NDVI de la cuenca de Pamplona derivado del satélite Sentinel-2.

# ¿Dónde se puede aplicar?



Figura: (a) Estacion Dole (b) Estación meteorológica (c) Estación meteorológica

# ¿Dónde se puede aplicar?



**Figura:** Un sensor en una parcela rural

# Historia de los Computadores

# Desarrollo de la informática

## Informática

El computador o ordenador no es un invento de una persona concreta, sino el resultado de la evolución de ideas y logros relacionados con diferentes campos de estudio.

- ▶ Electrónica
- ▶ Mecánica
- ▶ Materiales
- ▶ Lógica
- ▶ Algebra
- ▶ y cómo no programación

# Desarrollo de la informática

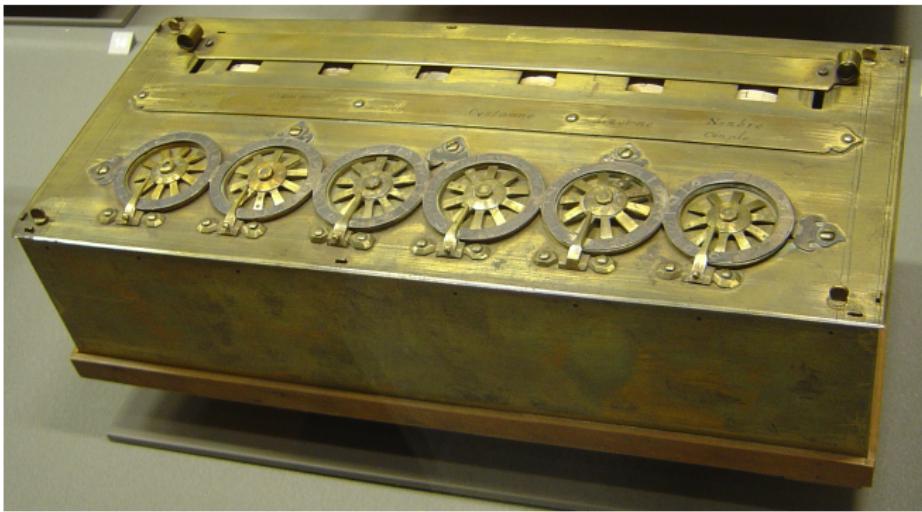
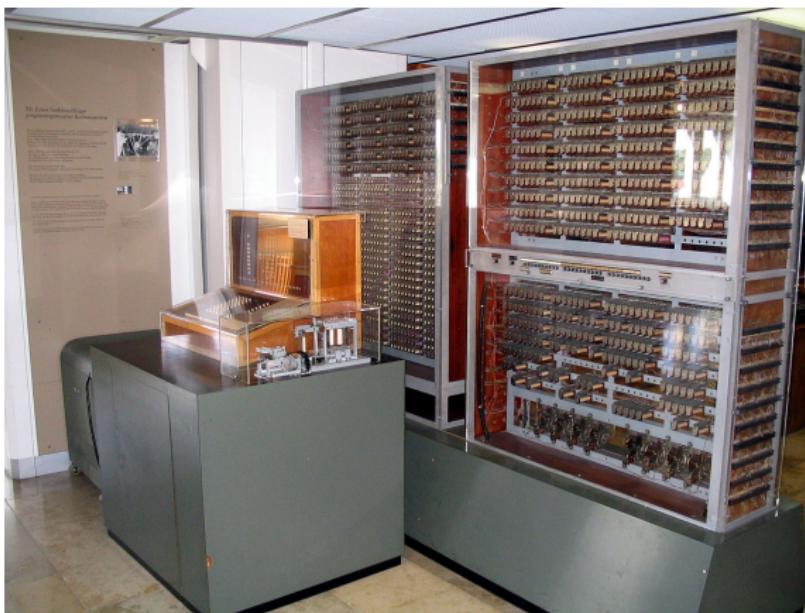


Figura: Pascaline, calculadora.

# Desarrollo de la informática

- ▶ Los ordenadores son calculadoras complejas
  - ▶ XVII – En los siglos XX las calculadoras se hicieron más complejas
- 1801:** francés Joseph Marie Jacquard, utilizó mecanismo de tarjetas perforadas para programar máquinas de tejer
- 1843:** Ada Augusta Lovelace sugirió la idea de que las tarjetas perforadas repetían ciertas operaciones
- 1854:** el lógico inglés George Boole publica su Álgebra de Boole. El sistema de Boole redujo a argumentos lógicos las permutaciones de tres operadores básicos algebraicos: y, o, y no.
- 1936:** Alan Turing describe la máquina de Turing, la cual formaliza el concepto de algoritmo.
- 1941:** La computadora Z3 fue creada por Konrad Zuse. Fue la primera máquina programable y completamente automática.

# Desarrollo de la informática



**Figura:** Réplica de Zuse Z3 exhibida en el Museo de Alemania, Munich.

## Ordenadores modernos: años 50–70

- 1953 : IBM fabrica su primera computadora a escala industrial, la IBM 650. Se amplía el uso del lenguaje ensamblador para la programación de las computadoras.
- 1957 : Jack S. Kilby construye el primer circuito integrado.
- 1958 : comienza la segunda generación de computadoras, caracterizadas por usar circuitos transistorizados en vez de válvulas al vacío.
- 1964 : la aparición del IBM 360 marca el comienzo de la tercera generación de computadoras. Las placas de circuito impreso con múltiples componentes elementales pasan a ser reemplazadas con placas de circuitos integrados.
- 1966 : aparecen los primeros ensayos que más tarde definirán lo que hoy es la programación estructurada.
- 1973 : El primer ordenador que utilizó el concepto de computadora

# Ordenadores modernos



**Figura:** El Xerox Alto, desarrollado en el Xerox PARC en 1973, fue uno de los primeros ordenadores personales

# Ordenadores actuales



# Ordenadores

- ▶ Entonces, ¿qué es un ordenador o una computadora?

## Desktop PC System



Figura: Ordenador convencional de escritorio

# Pero... ¿qué es un ordenador?

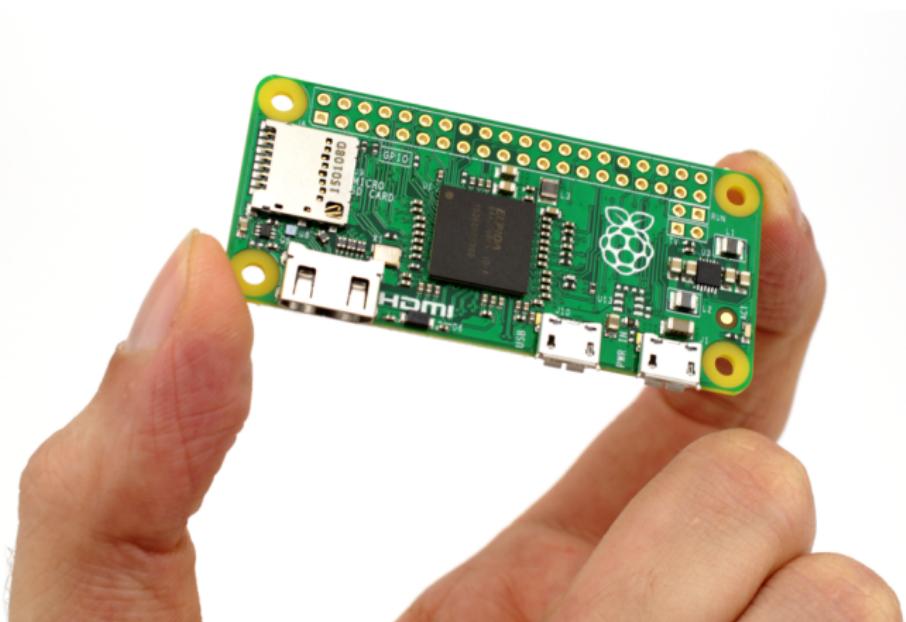
- ▶ Para un Ingeniero un ordenador es una máquina **cualquiera** que contiene los siguientes elementos:
  - ▶ Central Processing unit (CPU).
  - ▶ Random Access Memory (RAM).
  - ▶ Graphic Processing Unit (GPU).
  - ▶ Memoria a largo plazo: Hard Drive Disk (HDD) o Solid State Disk (SSD).
  - ▶ Comunicación: Ethernet, WiFi, Bluethooth ...

# Ordenadores integrados



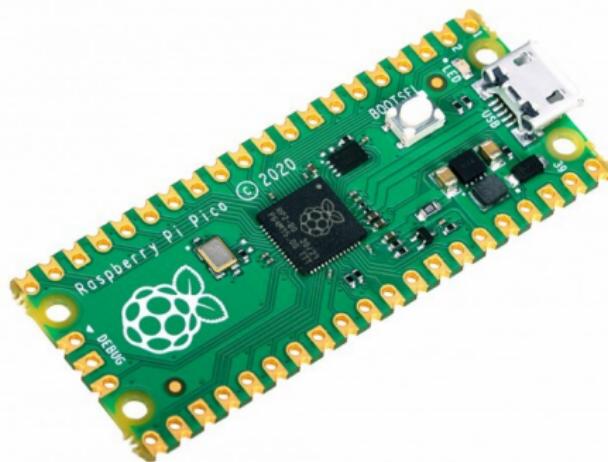
**Figura:** Pequeño ordenador Raspberry Pi

# Ordenadores integrados



**Figura:** Raspberry Pi cero ordenador aún más pequeño

# Ordenadores integrados



**Figura:** Raspberry Pi cero ordenador aún más pequeño

# Sistemas operativos

- ▶ **UNIX**: sistema operativo Unix que sigue una especificación única o estándar. (<https://www.opengroup.org/>)
- ▶ **GNU/Linux (Linux Is Not Unix)**: es un sistema operativo tipo Unix formado por software libre y libre (<https://www.gnu.org/>)
- ▶ **MacOS**: es un sistema operativo basado en Unix desarrollado y comercializado por Apple Inc.  
(<https://www.apple.com/residuo/ventura/>)
- ▶ **Windows**: es el grupo de sistema operativo desarrollado y comercializado por Microsoft.  
(<https://www.microsoft.com/en-us/windows/>)

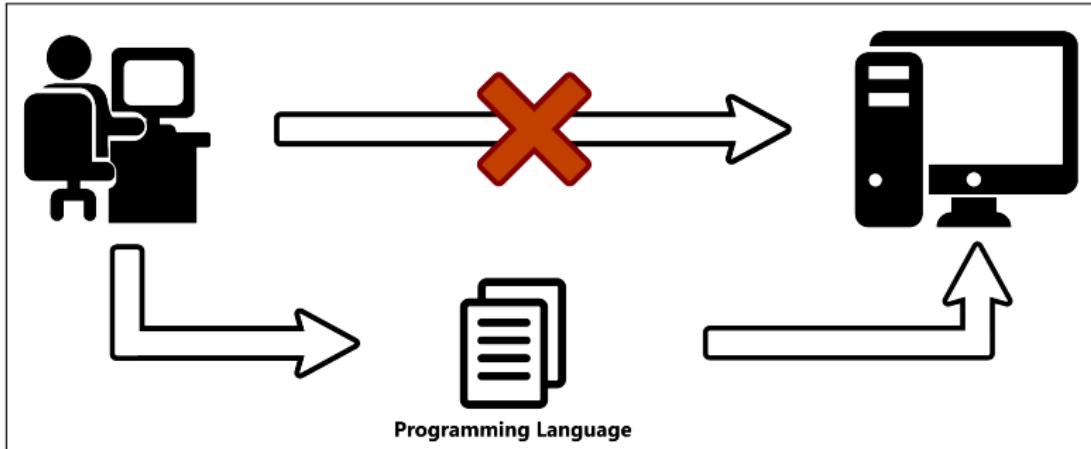
# Lenguajes de programación



# Lenguajes de programación

- ▶ Una máquina sólo hace lo que se le ordena. Esto es literal.
- ▶ Para poder dar órdenes necesitamos capacidad de comunicación con las máquinas.
- ▶ Pero... el ordenador moderno (eléctrico) solo entiende órdenes básicas o lenguaje binario.
- ▶ La comunicación entre máquina humana es imposible.
- ▶ Por eso... es necesario algo intermedio , existe la necesidad de un lenguaje específico.

# Lenguajes de programación



# Programación

- ▶ La programación consiste en crear programas con un lenguaje intermedio que el ser humano pueda entender.
- ▶ Normalmente se requieren dos capacidades:
  - a) Ofrecer una solución estructurada para resolver un problema específico;
  - b) Crear una estructura específica pensada en lenguaje de programación e indicar paso a paso.
- ▶ El segundo es la capacidad técnica y requiere un enfoque especial.

# Lenguajes de programación vs. lenguajes naturales

**P:** Entonces, ¿qué es un lenguaje de programación?

**R:** Es un grupo de normas para explicar al ordenador **las tareas a realizar**.

**R:** **Gramática** es un conjunto de normas. Se parece al lenguaje natural, pero...

- ▶ Se añaden operaciones matemáticas, palabras en inglés y mezclas de simbologías específicas.

**!!** Es posible que la misma **palabra** o **frase** indique algo diferente en dos lenguajes diferentes.

# Lenguajes de programación

## Historial de lenguajes de programación

- ▶ Los lenguajes de programación existen desde la época en que se crearon los ordenadores.
  - ▶ Cobol y Lisp se presentaron en los años 50;
  - ▶ En los años 60 aparecen Algol y Fortran;
  - ▶ ...
- ▶ Se han adaptado a los nuevos tiempos.
  - ▶ Las necesidades de los nuevos sistemas informáticos
  - ▶ Modas y las prioridades de los desarrolladores
  - ▶ ...

# Lenguajes de programación

Ejemplo: Pascal

```
1 program fifteen;
2 var a:char;
3 begin
4     write('Dame un caracter: ');
5     readln(a);
6     case (a) of
7         'A'..'Z': writeln('Es mayuscula.');
8         'a'..'z': writeln('Es minuscula.');
9         '0'..'9': writeln('Es un digito.');
10        else
11            writeln('Es un signo de puntuacion')
12    end
13 end.
```

Código: Ejemplo de fragmento de código que identifica el tipo de variable en Pascal leguaia.

# Lenguajes de programación

Ejemplo: Java

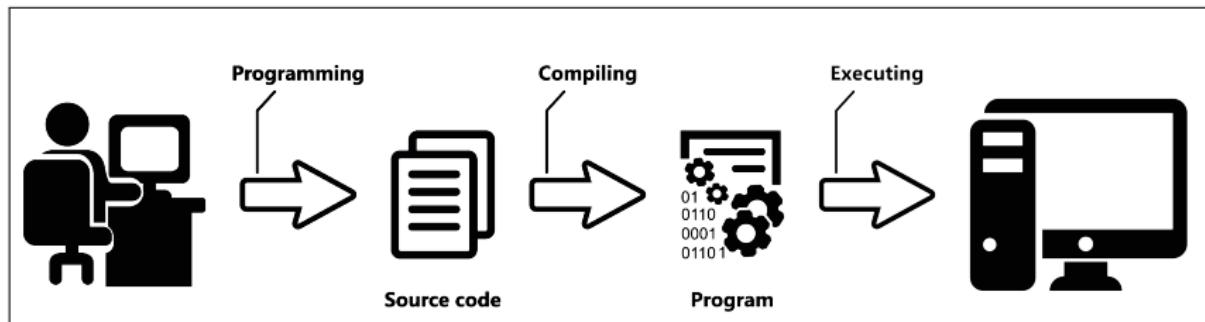
```
1 import java.util.ArrayList;
2 public class Main {
3     public static void main(String[] args) {
4         ArrayList<String> cars = new ArrayList<String>();
5         cars.add("Volvo");
6         cars.add("BMW");
7         cars.add("Ford");
8         cars.add("Mazda");
9         cars.remove(0);
10        System.out.println(cars);
11    }
12 }
```

Código: Ejemplo para crear la lista en Java leguaia y eliminar el primer elemento.

# Compilación

- ▶ Una persona escribe un archivo de texto en cualquier lenguaje de programación...
- ▶ ...pero la máquina solo entiende el lenguaje máquina.
- ▶ El lenguaje de programación es un programa **compilador** que traduce la máquina al lenguaje.
- ▶ Por ejemplo, si desplegamos un archivo **.exe** aparecerán caracteres incomprendibles para nosotros. Esto es información compilada y en lenguaje máquina.

# Proceso de compilación



# Lenguajes compilados

- ▶ Todos los lenguajes mencionados anteriormente son **lenguajes compilados**.
- ▶ Hacer cambios → nueva compilación → nuevo programa.
- ▶ Los programas compilados se adaptan a una máquina y a un sistema operativo concreto.
- ▶ Ejemplos:
  - ▶ Cobol;
  - ▶ Pascal;
  - ▶ C/C++;
  - ▶ Java.

# Problemas de programas compilados

- ▶ Los lenguajes compilados presentan algunos problemas.
- ▶ Principal:
  - ▶ Después de realizar un cambio hay que compilar de nuevo.
- ▶ Secundarios:
  - ▶ Crean problemas a la hora de cambiar de plataforma;
  - ▶ No se pueden utilizar de forma interactiva (más adelante entenderéis esto).

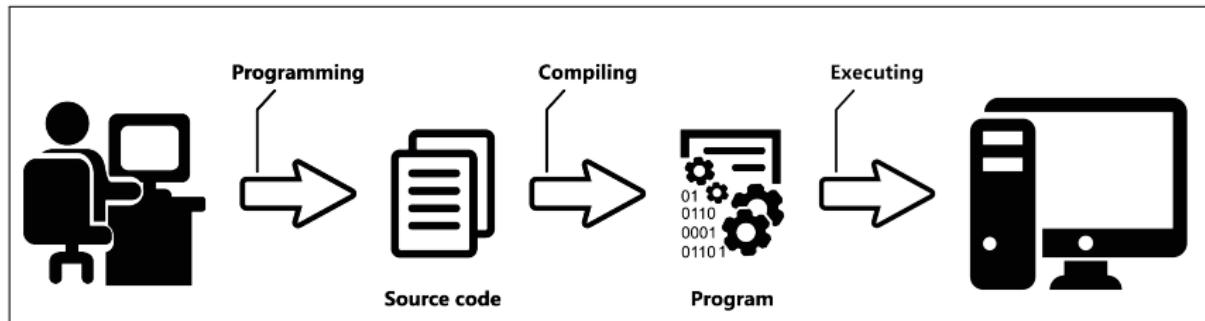
# Lenguajes interpretados

- ▶ Para hacer frente a los problemas de los programas compilados se creó un nuevo paradigma de programación: **lenguajes interpretados**.
- ▶ No es obligatorio compilar (no se debe compilar y ejecutar todo el programa).
- ▶ Las aplicaciones e instrucciones escritas se enviarán a una máquina **simulada** que se esté ejecutando en un ordenador.
- ▶ Esta máquina se conoce como **máquina intérprete**.

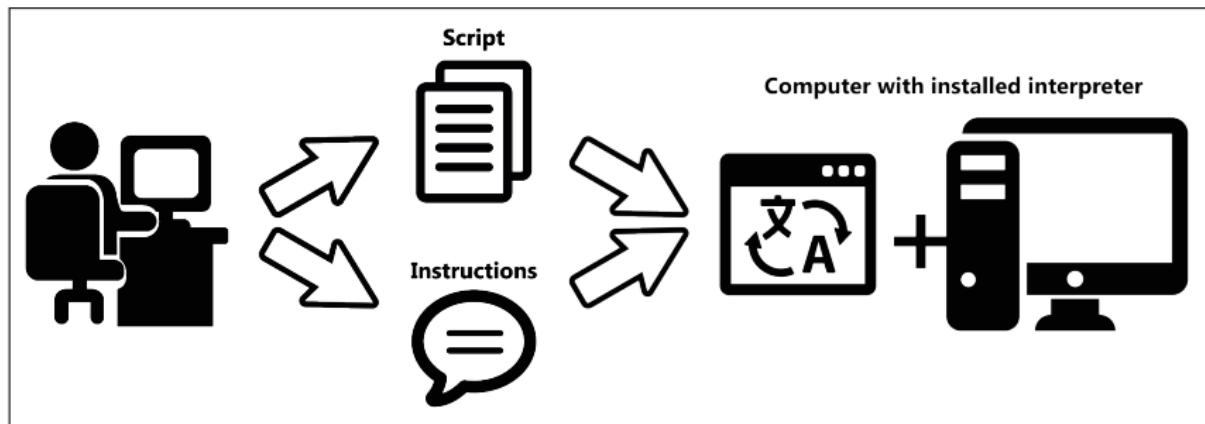
# Concepto de la máquina de intérprete

- ▶ La **máquina intérprete** es una **aplicación** que ejecuta (**interpreta**) el lenguaje escrito y las órdenes en un lenguaje de programación.
  - ▶ En principio, no compila el programa.
  - ▶ Ejecuta las instrucciones una a una.
- ▶ En un modo, el programa escrito se ejecuta sobre otra aplicación predefinida (**máquina de intérprete**).
- ▶ Aunque solo entenderá las instrucciones de un lenguaje de programación específico.

# Compilado vs interpretado



# Compilado vs interpretado



# Algunos ejemplos de lenguajes interpretados

- ▶ Los lenguajes interpretados son muy cómodos para trabajar y están ganando cuota de mercado.
- ▶ Algunos ejemplos de lenguajes interpretados:
  - ▶ Lisp;
  - ▶ MatLab;
  - ▶ Python;
  - ▶ R;
  - ▶ PhP.

## Algunas preguntas

**P:** ¿Todos los lenguajes son interpretados o compilados?

**R:** No, existen otros paradigmas intermedios.

**R:** Los lenguajes Java o .NET son precompilados.

**P:** ¿Un lenguaje puede ser compilado e interpretado?

**R:** Sí.

**R:** Mejor dicho, todos los lenguajes interpretados se pueden compilar para ejecutarlos sin utilizar intérpretes.

# Lenguaje de programación python



# En esta asignatura utilizaremos Python

- ▶ Python es un lenguaje interpretado.
- ▶ Ahora está de moda, se puede ver en muchas aplicaciones informáticas (páginas web, computación científica, aplicaciones GIS,...)
- ▶ Es sencillo, intuitivo y poco restrictivo.
  - ▶ Puede que demasiado poco...



# Historial de Python

- ▶ Diseñado por Guido van Rossum.
- ▶ En 1991 se publicó la versión 0.9.0.
  - ▶ Es decir, se publicaron las reglas del lenguaje y la primera versión de la máquina interprete.
- ▶ En 1994 apareció la versión 1.0 y se hizo popular. Muy popular...
- ▶ Ahora está administrada por una fundación.

# Python 2.x vs Python 3.x

- ▶ Python es muy flexible (¿demasiado?), y con el paso de los años se le han ido añadiendo posibilidades. La versión 2 llegó a ser demasiado grande, demasiado compleja; tenía demasiados compromisos con el pasado (pre-compatibilidad).
- ▶ Para la publicación de la versión 3 la **comunidad**, decidió proceder a la limpieza de la segunda versión.
  - ▶ La versión 3 no es pre-compatible, es decir, una aplicación escrita en python 2.7 no funcionará en la máquina de intérprete de 3.x
  - ▶ Python 3.x es más limpio y tiene futuro (excepto viejos sistemas).
- ▶ En esta asignatura utilizaremos Python 3.x.

# Código en python

**P:** ¿Cómo es el código de Python? ¿Qué aspecto tiene un programa Python?

**R:** Es especial:

- ▶ La menor longitud posible, sin separadores ni marcadores.
- ▶ Lo más legible posible.
- ▶ Muy flexible (hay muchas maneras de hacer lo mismo).

# Código en python

- ▶ Estos son los consejos o *consignas* en inglés que se siguieron durante la creación de Python:
  - ▶ Beautiful is better than ugly;
  - ▶ Explicit is better than implicit;
  - ▶ Simple is better than complex;
  - ▶ Complex is better than complicated;
  - ▶ Readability counts.
- ▶ Los diseñadores del lenguaje han creado las normas de escritura basándose en estas *consignas*.

# Código en python

Comparada con C

```
1 def print_up_to(x):  
2     for i in range(1,x):  
3         print(i)  
4  
5  
6  
7
```

**Código:** Los números que hay de 1 a 'x' en Python.

```
1 void printUpTo(int x){  
2     int i;  
3     for(i=1;i<=x;i++){  
4         printf('%d', i)  
5     }  
6     return  
7 }
```

**Código:** Los números que hay de 1 a 'x' en C.

# Código en python

Comparada con java

```
1 def print_up_to(x):  
2     for i in range(1,x):  
3         print(i)  
4  
5  
6
```

Código: Los números que hay de 1 a 'x' en Python.

```
1 public static void  
2     printUpTo(int x){  
3         for(int i=1;i<=x;i++){  
4             System.out.println(  
5                 )  
6         }  
7     }
```

Código: Los números que hay de 1 a 'x' en java.

# Código en python

Comparada con pascal

```
1 def print_up_to(x):
2     for i in range(1,x):
3         print(i)
4
5
6
7
8
```

Código: Los números que hay de 1 a 'x' en Python.

```
1 procedure printUpTo(x:
2     integer)
3 var
4     i:integer;
5 begin
6     for i:=1 to x do
7         writeln(i)
end
```

Código: Los números que hay de 1 a 'x' en pascal.

# Características técnicas de Python

- ▶ Puede soportar casi cualquier paradigma de programación: programación estructurada, orientada a objeto y funcional ([lambda calculus](#)).
- ▶ El sistema tiene un tipado fuerte, implícito y dinámico. Sea cual sea su significado.
- ▶ La memoria se gestiona automáticamente por el recolector de basura.
  - ▶ Si tienes experiencia en C, y para hacerte una idea, python evita la memoria inservible y las violaciones de segmentos.

## Algunas preguntas

**P:** ¿Se puede compilar Python?

**R:** Sí, a veces

**P:** ¿Puedo hacer con Python lo mismo que con otros lenguajes?

**R:** es similar a Java, C#, etc.

**R:** es inferior a C en algunos aspectos, como la libertad de gestión de la memoria, pero más cómoda.

**P:** ¿Python se utiliza en el mundo profesional?

**R:** Sí. En diferentes ámbitos.

# Instalar python

# ¿Qué necesito para ejecutar python?

- ▶ Necesitas [Instalar Python](#).
  - ▶ ¿Instalar un lenguaje de programación?
- ▶ Bueno, consiste en instalar y ejecutar la [máquina intérprete](#) de python.
  - ▶ Es decir, la [máquina intérprete](#) es un programa que admite órdenes de python.
  - ▶ Existe un [máquina oficial](#), y esa es la que vamos a utilizar.
    - ▶ Python Standard Distribution (PSD).
  - ▶ En la asignatura se aceptarán otras implementaciones pero deberéis mirarlas e instalarlas por vuestra cuenta.

## Instalar python

# Instalar Python

- ▶ Descargar e instalar:

The screenshot shows the Python.org homepage with a dark blue header. The main navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the header is a search bar and social sharing links. A large central area features a code snippet demonstrating a Fibonacci series implementation in Python 3, followed by a link to learn more about defining functions. Below this, a promotional message encourages quick work and system integration, with a 'Learn More' button. At the bottom of the page, there are four main sections: 'Get Started', 'Download', 'Docs', and 'Jobs', each with a brief description and a corresponding icon.

# Python 3: Fibonacci series up to n

```
>>> def fib(n):  
>>>     a, b = 0, 1  
>>>     while a < n:  
>>>         print(a, end=' ')  
>>>         a, b = b, a+b  
>>>     print()  
>>> fib(1000)  
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

**Functions Defined**

The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. [More about defining functions in Python 3](#)

Python is a programming language that lets you work quickly and integrate systems more effectively. [» Learn More](#)

---

**Get Started**  
Whether you're new to programming or an experienced developer, it's easy to learn and use Python.  
[Start with our Beginner's Guide](#)

**Download**  
Python source code and installers are available for download for all versions! Not sure which version to use? Check here.  
[Latest: Python 3.5.0 - Python 2.7.10](#)

**Docs**  
Documentation for Python's standard library, along with tutorials and guides, are available online.  
[docs.python.org](#)

**Jobs**  
Looking for work or have a Python related position that you're trying to hire for? Our [relaunched community-run job board](#) is the place to go.  
[jobs.python.org](#)

# Instalar Python

**P:** ¿Qué estoy instalando exactamente con ese PSD?

**R:** Algunas cosas. En resumen, y de cara a lo visto hasta ahora:

- ▶ Interprete makina, y...
  - ▶ un monton de librerías
- 
- ▶ Librería ≡ Son acciones y funciones ya definidas (y probadas) para que el programador no tenga que escribir desde 0.

# Maquina interprete

- ▶ La máquina interprete se puede utilizar de dos formas:
  - a) Modo ejecución;
  - b) Consola interactiva
- ▶ Sobre todo es importante el concepto de **sesión**.
  - ▶ La principal implicación de la sesión es que se pueden guardar los resultados de las órdenes
  - ▶ Las órdenes ejecutadas anteriormente no se 'eliminan'
- ▶ La diferencia la veremos en el entorno de desarrollo.

# Entorno de desarrollo



# Programas en python

- ▶ Los programas creados en esta asignatura serán compactos *script*, es decir, **programas complejos**.
- ▶ De una ejecución a otra no se guarda nada.
  - ▶ Las variables (valores guardados) desaparecerán de una ejecución a otra
  - ▶ La máquina intérprete se abre y cierra en cada ejecución.
  - ▶ Como ocurre en los lenguajes tradicionales (Pascal, C, etc.).

# Programas en python

- ▶ Los programas que crearemos serán secuenciales
  - ▶ Las instrucciones se ejecutarán línea a línea
  - ▶ Para ejecutar la segunda línea será necesario ejecutar previamente la primera

```
1 numero_entero = 5
2 otro_numero_entero = 3
3 resultado = numero_entero + otro_numero_entero
4 print(resultado)
```

Código: Un programa en python.

# Programas en python

- ▶ Para ejecutar *scripts* en el lenguaje python, se pueden utilizar múltiples aplicaciones, pero lo más recomendable es utilizar un **IDE**;
- ▶ Los archivos de texto se pueden crear/modificar directamente desde el **IDE**;
- ▶ Es decir:
  - ▶ Primero, se crear un archivo de texto utilizando el editor de texto integrado
  - ▶ Luego, se guarda el archivo de texto
  - ▶ Para terminar, dar la orden de ejecutar el fichero completo.

# Ejecución de script de python

- ▶ El desarrollo de la asignatura se realizará con el entorno de desarrollo integrado (IDE) **Spyder**.
  - ▶ IDE ≡ Integrated Development Environment.
- ▶ Permite distintos modos de ejecución, en la creación del código de ayuda, etc.



# Ejecución de código: Spyder IDE

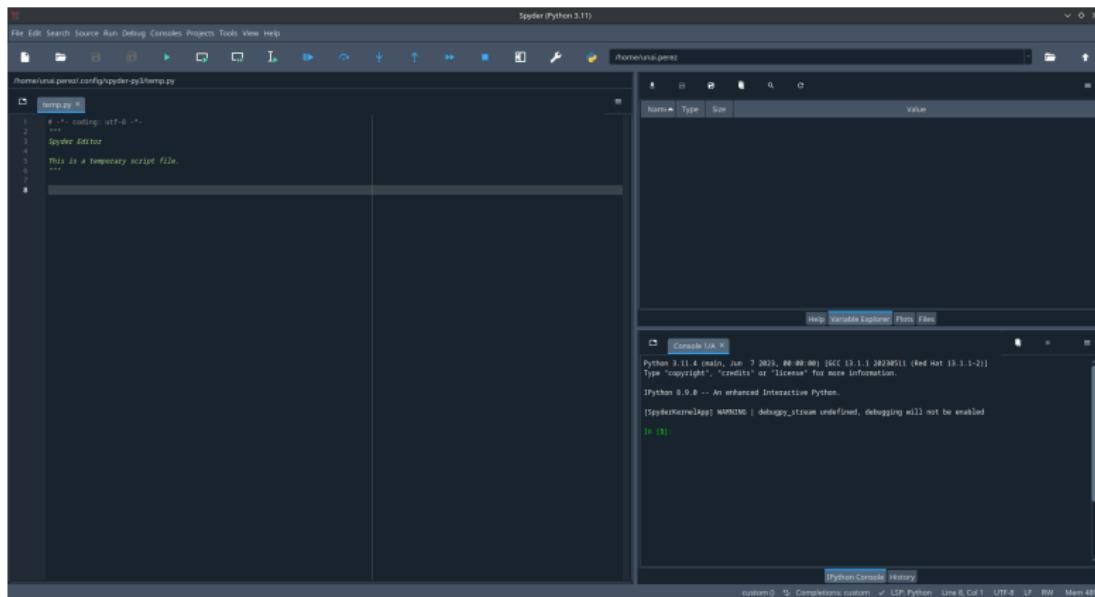


Figura: Ejemplo de Spyder IDE

## Entorno de desarrollo

## Ejecución de código: Spyder IDE

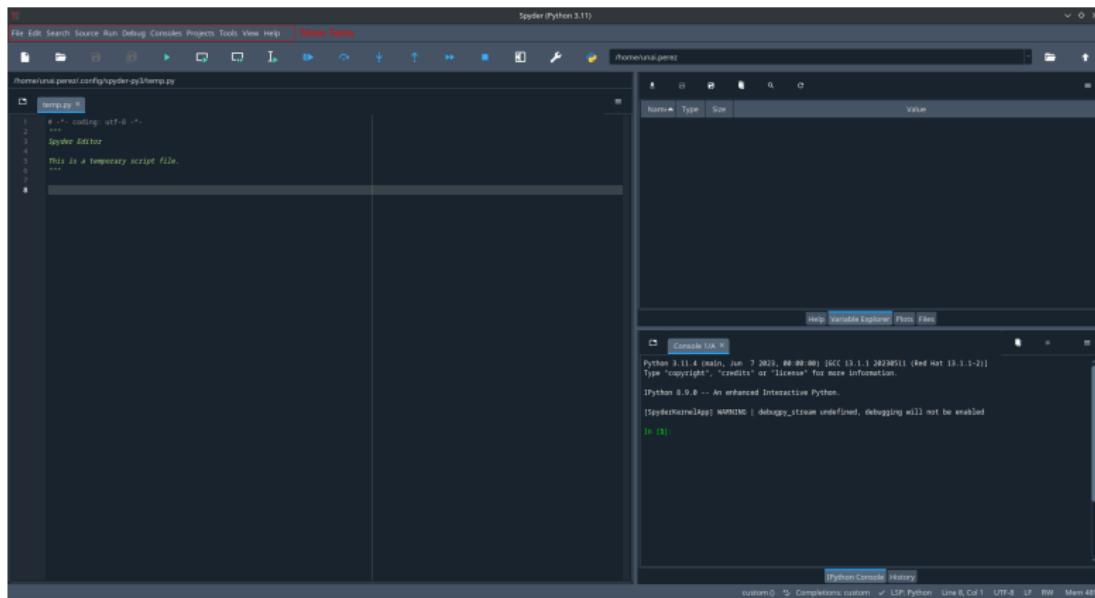


Figura: Ejemplo de Spyder IDE: barra de menú

## Entorno de desarrollo

## Ejecución de código: Spyder IDE

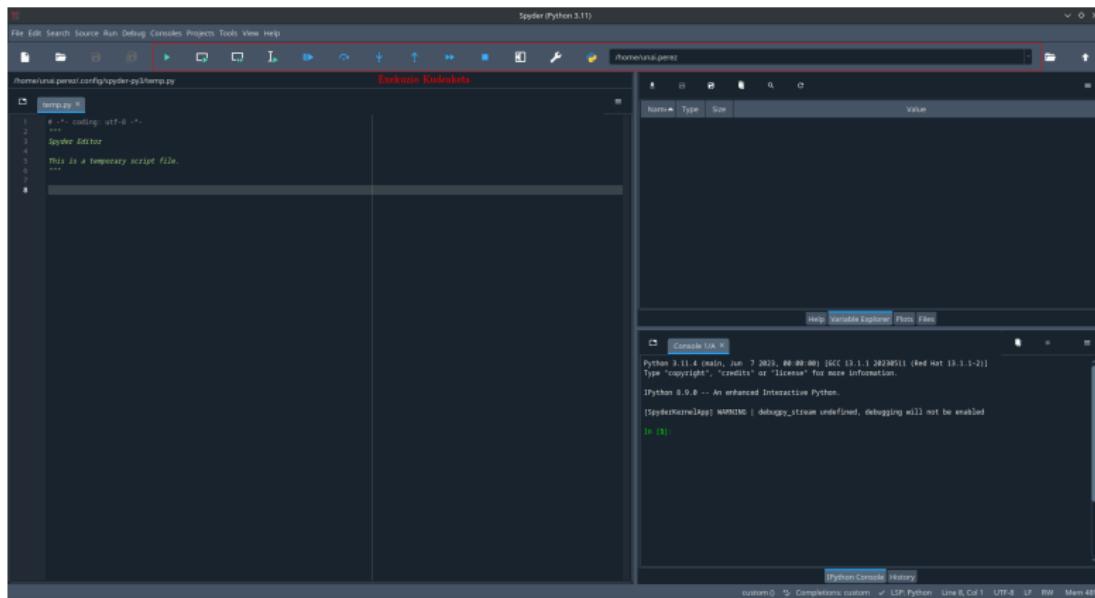


Figura: Gestión de ejecución de Spyder IDE

## Entorno de desarrollo

## Ejecución de código: Spyder IDE

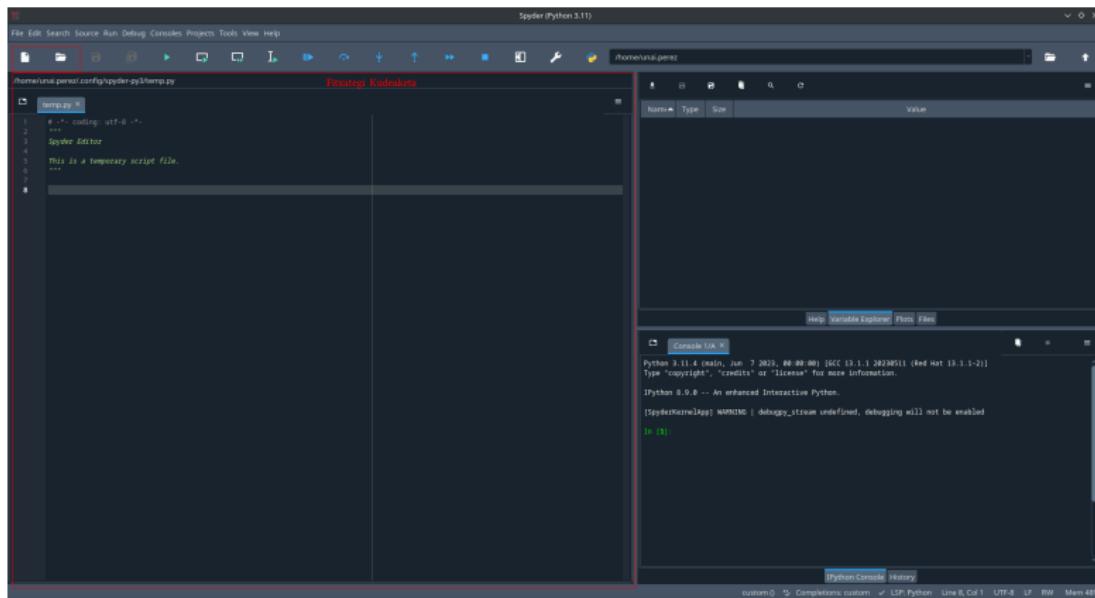


Figura: Gestión de ficheros en Spyder IDE.

# Ejecución de código: Spyder IDE

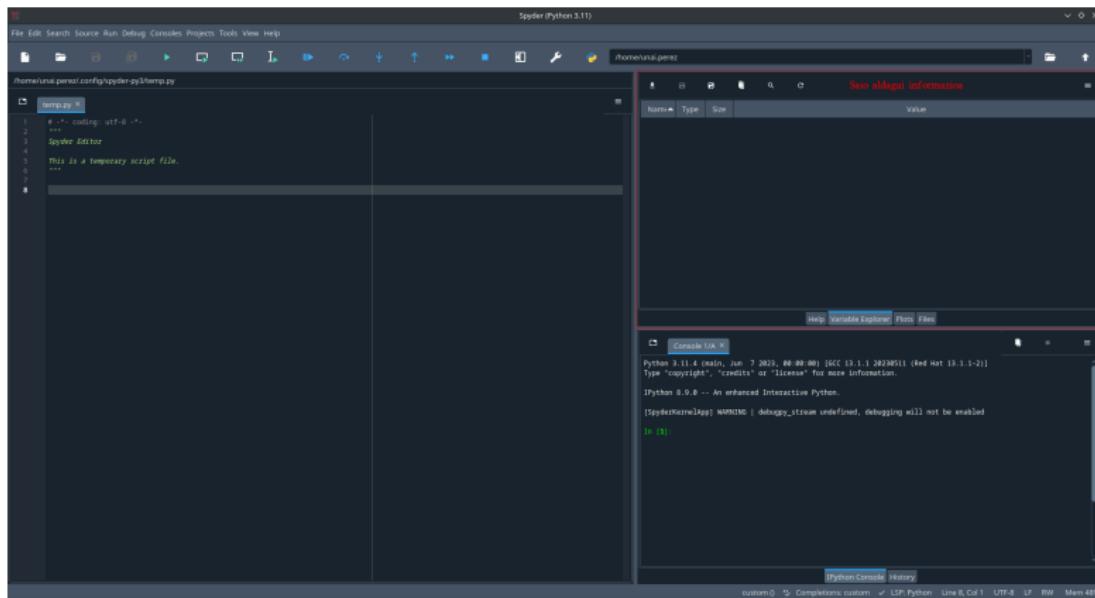


Figura: Visualización de las variables en Spyder IDE.

# Ejecución de código: Spyder IDE

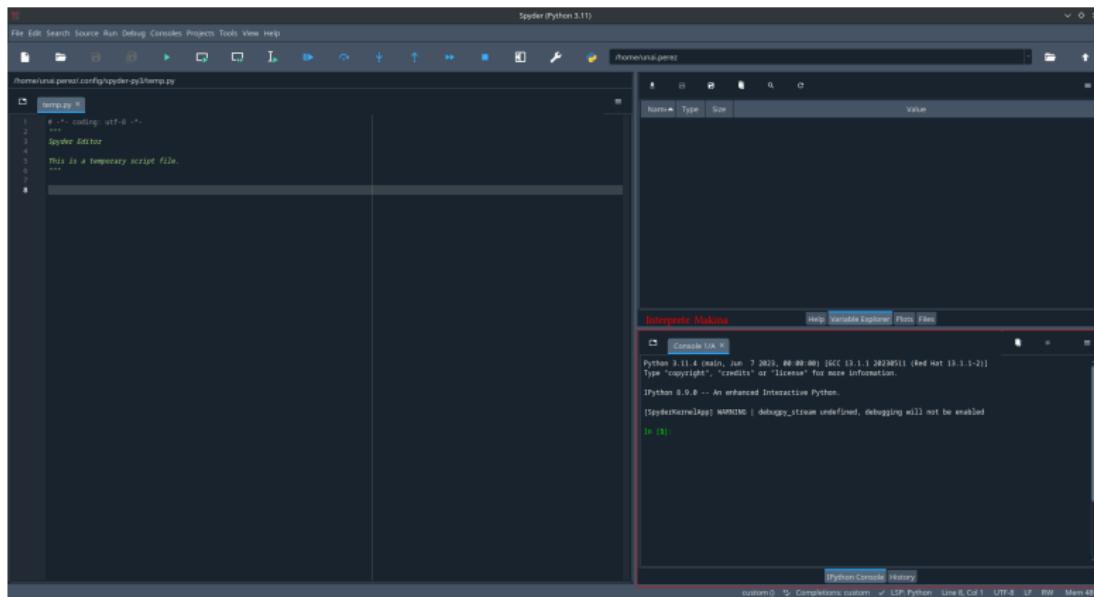


Figura: Maquínas interprete de Spyder IDE

# Instrucciones de python

- ▶ Python es un lenguaje interpretado por lo que la máquina intérprete puede recibir instrucciones al vuelo;
- ▶ Este modo de ejecución se asemeja a una calculadora avanzada;
- ▶ Entre los valores (numéricos) están admitidas las **operaciones matemáticas**;
  - ▶ `+, -, *, /, //, **, %, ...`
- ▶ Además de los operadores, existen **funciones**.

# La función print

- ▶ La función `print(values)` muestra en pantalla el contenido en la consola `values`.
- ▶ `print(values, sep, end, file, flush)`
- ▶ Valores admitidos en `values`:
  - ▶ Encadenar multiples valores;
  - ▶ Mostrar los valores calculados;
  - ▶ ...
- ▶ `sep`: cadena de caracteres por defecto con el valor “ ” (un espacio).
- ▶ `end`: adena de caracteres, por defecto, con el valor “\n”.
- ▶ ...

**!!** Este comando ha cambiado a partir de Python 3.

# Ejercicio

- ★ **Ejercicio** Qué instrucciones debes escribir en la máquina de intérprete para que calcule  $5^2 + \frac{2+7\times 4}{5}$ .

# Instrucciones y la función print

- ▶ Si le damos las siguientes instrucciones a la máquina de intérprete. ¿Qué aparece en pantalla?

```
1 5*2+(2+7*4)/5
```

Código: Instrucciones para ejecutar en la máquina interprete.

- ▶ Entonces, ¿para qué la función `print`?

```
1 # python script
2 5*2+(2+7*4)/5
```

Código: Instrucciones para ejecutar en la máquina interprete.

# Ejercicio

★ **Ejercicio** Crea un programa *python* en el que aparezcan secuencialmente cuatro `print` diferentes. En la primera visualizará dos números, en la segunda utilizará dos números y el argumento `sep`, en la tercera utilizará dos números y el argumento `end` y en la última dará valor a todos los números y argumentos.

Ejecuta varias veces el programa y dale diferentes valores tanto a `sep` como a `end`, ¿qué ocurre?

# Resultado del ejercicio

- ▶ ¿Qué es incorrecto en el siguiente código?

```
1 #python script
2 print(1,2)
3 print(1,2, sep=', ')
4 print(1,2, end='; ')
5 print(1,2, sep='/' ,end=':' )
```

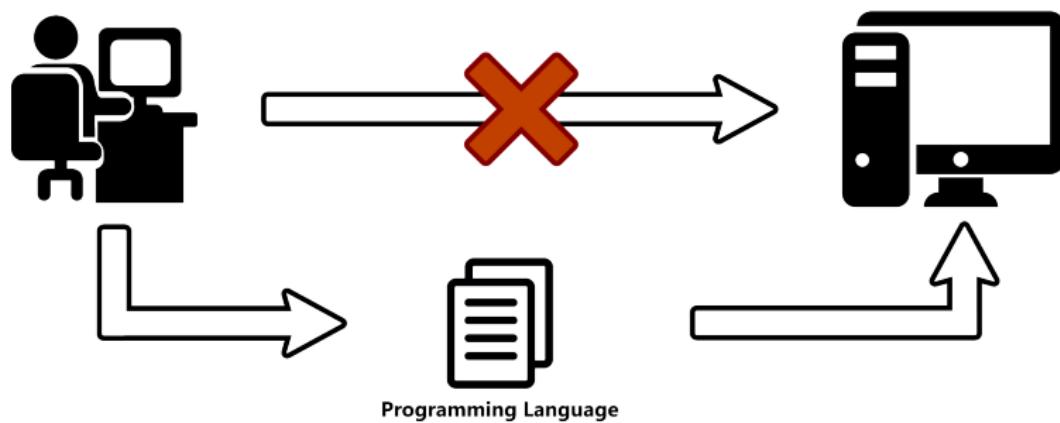
Código: Instrucciones para ejecutar en un script de python.

# Resumen



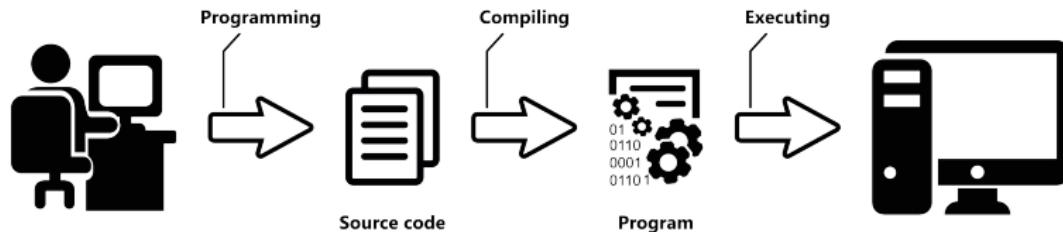
# Resumen

## Lenguajes de programación



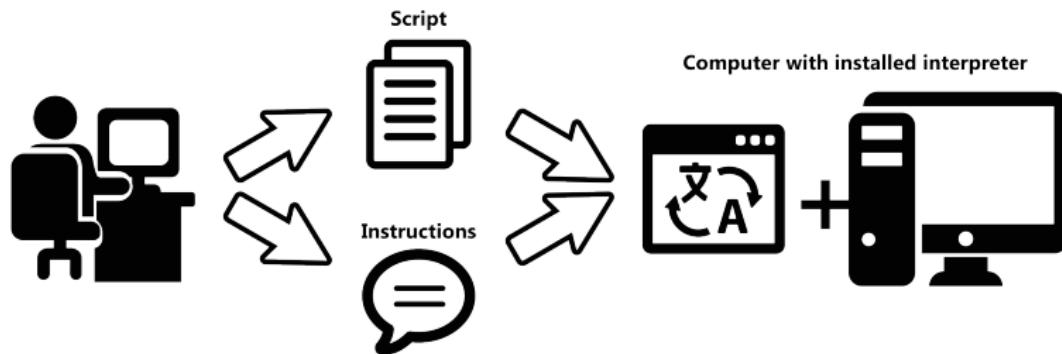
# Resumen

## El proceso de compilación



# Resumen

## Compilado vs. interpretado



# Resumen

Edukien zerrenda

1. Lenguajes de programación;
2. Languajes compilados [vs.](#) interpretados;
3. Introducción a Python;
4. Spyder IDE
5. Algunos ejemplos de python.