



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA

2021/2022

# Programación

## Tema 1.

Algoritmos,  
programación,  
ordenadores,  
Python,  
Thonny, y  
testing

# Índice

1. Algoritmos y problemas
2. Programas y la actividad de la Programación
3. Ordenadores
4. Lenguajes de programación
5. Compiladores e intérpretes
6. El lenguaje Python

# Lectura y ejercicios (en PoliformaT)

Tema1-teoria-Python-reader.pdf

## Programación en Python

### TEMA 1

Introducción: Algoritmos, programación, ordenadores, Python, Thonny y testing.

Universidad Politécnica de Valencia

2021-2022

#### Índice

1	Introducción	2
2	Arquitectura hardware de los ordenadores	3
3	Programación	4
4	Palabras y frases	5
5	Instalando Thonny, un entorno integrado de desarrollo (IDE)	6
6	Conversando con Python	7
7	Terminología: intérprete y compilador	9
8	Escribiendo un programa	10
9	¿Qué es un programa?	12
10	Los bloques de construcción de los programas	13
11	¿Qué cosas pueden ir mal?	14
12	Testing y depuración de los programas	15
13	El camino del aprendizaje	16
14	Glosario	17
	Ejercicios de tipo test	18
	Ejercicios respuesta abierta	19

---

El contenido de este boletín está basada en material de diferente libros open source:

- *Python for everybody*, Copyright 2009 - Charles Severance.
- *Think Python: How to Think Like a Computer Scientist*, Copyright 2015 - Allen Downey.

Ambos trabajos están registrados bajo una Licencia Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. Este licencia está disponible en <http://creativecommons.org/licenses/by-nc-sa/3.0/>.

---

# 1. Algoritmos

Un **algoritmo** consiste de

- una **secuencia de instrucciones** especifica de **acciones** que hay que ejecutar

y

- un **orden** en que hay que ejecutarlas, para completar una **tarea** determinada.

# 1. Algoritmos para humanos

Mientras suena la música:

Mano izquierda hacia adelante  
Mazo derecha hacia adelante  
Mano izquierda hacia atrás  
Mano derecha hacia atrás  
Mano izquierda a hombro derecho  
Mano derecha a hombro izquierdo  
Mano izquierda a la nuca  
Mazo derecha a la nuca  
Mano izquierda a caldera derecha  
Mano derecha a caldera izquierda  
Mano izquierda a nalgas izquierdas  
Mano derecha a nalgas derechas  
Meneo  
Meneo  
Salto



<https://www.youtube.com/watch?v=zWaymcVmJ-A>

# 1. Algoritmos para humanos



secuencia de instrucciones

+

orden

## Receta de paella de marisco paso a paso

Paella marinera con mariscos. Receta típica de la cocina española.

**Tipo de receta** Arroces  
**Tipo de cocina** Española  
**Keyword** Receta tradicional

**Tiempo de preparación** 20 minutos

**Tiempo de cocción** 25 minutos

**Tiempo total** 45 minutos

**Porciones** 4

**Calorías** 340 kcal

**Autor** Comedera.Com



★★★★★  
4.53 de 105 votos

Imprimir

### Ingredientes

- 2 tazas de arroz
- 4 tazas de caldo de gambas
- ½ kg de calamares
- ½ kg de gambas frescas
- 1 puñado de conchas de mar almejas y mejillones
- 1 cebolla
- 1 tomate mediano
- ½ pimiento rojo
- 2 dientes de ajo
- ½ taza de guisantes
- 1 ramo de hojas de perejil fresco
- Aceite de oliva
- Sal
- Pimienta
- Colorante amarillo ó algunas hebras de azafrán

### Preparación

1. Limpia y pela las gambas. Usa las conchas y cabeza para hacer caldo. Resérvalo.
2. También limpia los calamares y pícalos en ruedas.
3. Limpia también las conchas de mar con abundante agua para que suelten cualquier residuo de arena.
4. En una paellera, ó sartén muy grande y profunda, sofríe en aceite de oliva, la cebolla, el ajo, pimiento y tomate picados en cuadritos pequeños. Agrégale un poco de pimienta y sal.
5. Pon los calamares y luego las conchas de mar. Deja que se cocinen unos minutos. Verás que comienza a hacerse un caldo, esto está bien.
6. Agrega las 2 tazas de arroz y revuelve para que se mezcle todo. Seguidamente ponle 4 tazas del caldo de gambas que hiciste anteriormente. Si no te alcanza, completa con agua.
7. Revuelve bien. Agrega una cucharadita de colorante amarillo ó las hebras de azafrán y deja hervir por unos 3 minutos.
8. Aun con líquido en la paellera, agrega los guisantes frescos, las gambas y el perejil previamente picado muy pequeñito. Chequea la sal y la pimienta y agrega de ser necesario.
9. Deja cocinar hasta que esté casi seco el líquido. En este momento puedes ponerle unas tiras de pimiento para decorar y algunos langostinos con su concha.
10. Baja el fuego y tapa.
11. Deja cocinar por 15 minutos y prueba el grano. Si está listo retira del fuego y sirve tu paella de marisco con un chorro de aceite de oliva por encima para darle aun más sabor.

# 1. Algoritmos para ordenadores

- Los ordenadores se construyen con un solo propósito
  - hacer las cosas por nosotros
- Son perfectos para procesar una **secuencia de instrucciones** específica en un determinado **orden** para **completar una tarea**.
- Pero, necesitamos hablar su idioma para describirles **qué instrucciones** queremos que realicen y en **qué orden**



¿qué es la siguiente cosa que quieres que haga?

What next?

What next?

What next?

What next?

What next?

What next?



# 1. Algoritmos

Un **algoritmo** es una **secuencia de instrucciones** específica de las acciones que ha de ejecutar, y en qué **orden**, para completar una **tarea** determinada.

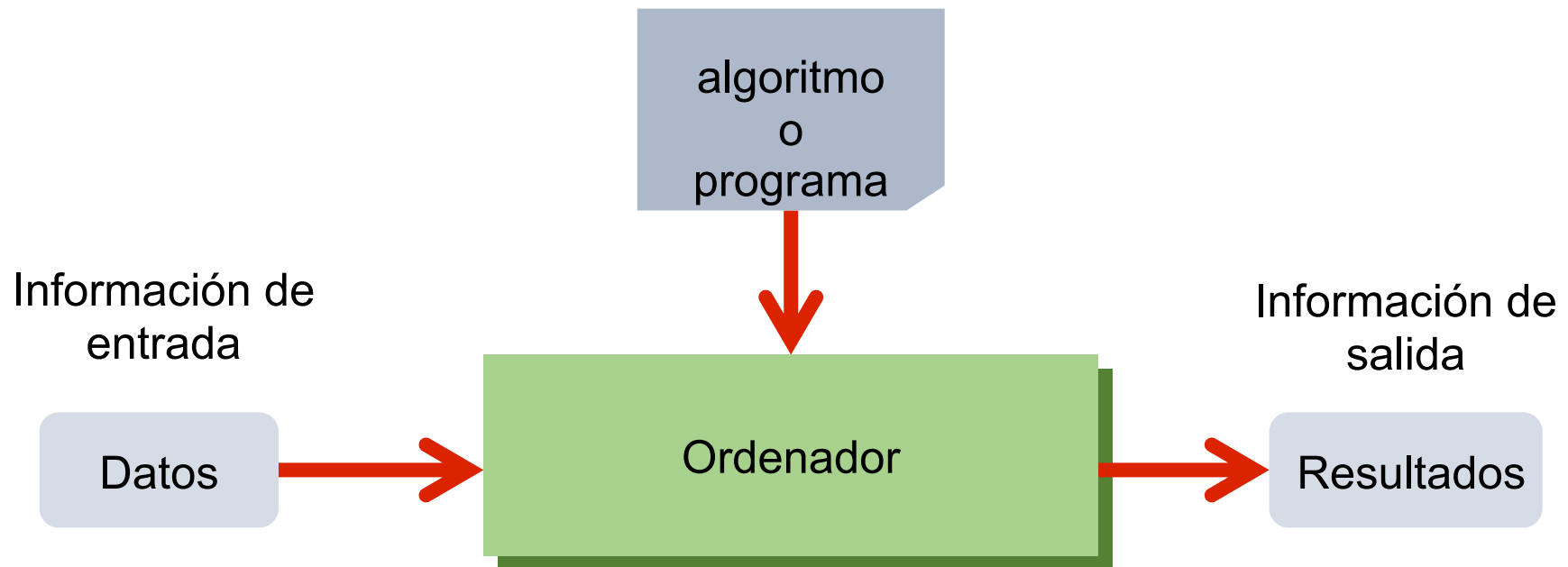
- Algoritmos para humanos en lenguaje natural
- Algoritmos para ordenadores en un lenguaje de programación

Un **programa** es un algoritmo escrito en un lenguaje de programación para que el ordenador puede completar la tarea.



# 1. Algoritmos para ordenadores

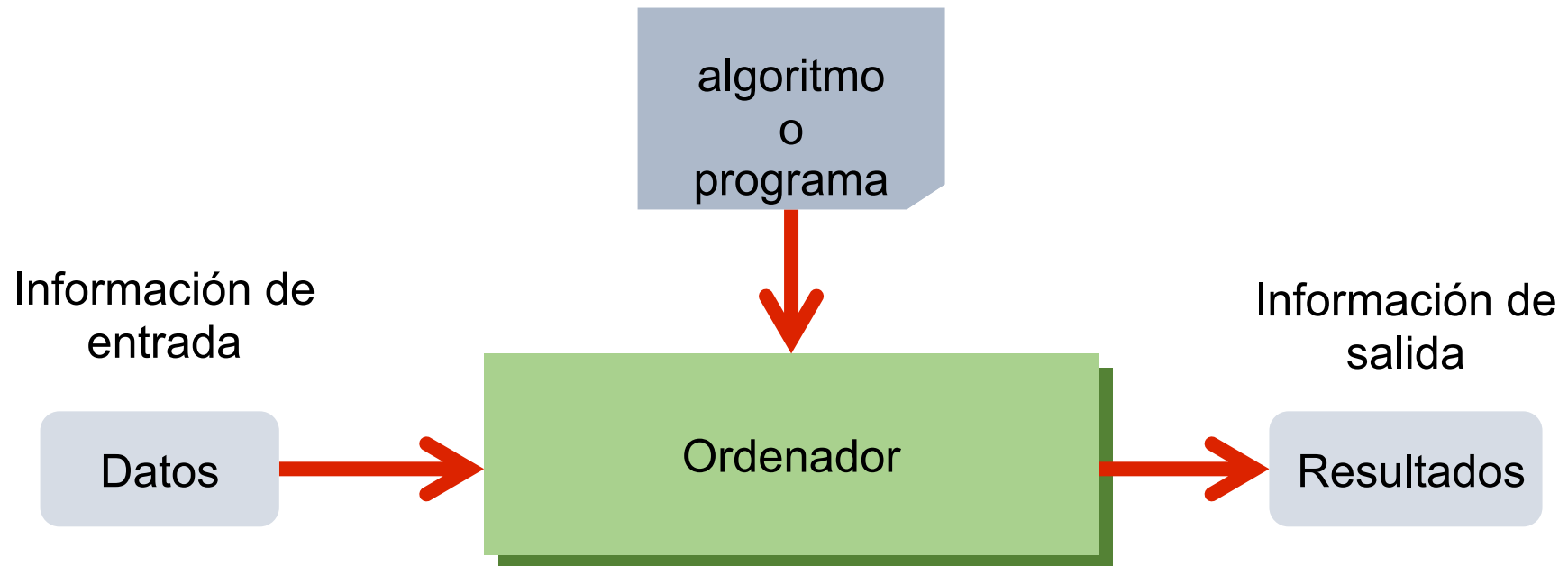
- Ordenadores hacen lo que nosotros queremos.
- Indicamos lo que queremos que hacen a través de un **programa**.
- Un programa es una **secuencia de instrucciones**
- Ordenadores procesan estos **programas**.
- Significa que procesan una secuencia de instrucciones específica en un determinado orden para completar **una tarea**.



# 1. ¿Qué tareas? problemas algorítmicos

- Un **programa** es un algoritmo escrito en un lenguaje de programación para que el ordenador puede **completar una tarea**.
- ¿qué tareas? ¿qué problemas?
- Por ejemplo problemas algorítmicos o computacionales: aquellos que se pueden resolver de forma “mecánica” o “automática”, es decir, aplicando unas reglas conocidas y preestablecidas,
- por ejemplo los relacionados con
  - el análisis de texto ,
  - el cálculo numérico,
  - la manipulación gráfica
  - Etc.

# 1. Tareas que son fáciles para ordenadores y no tanto para humanos



Dado un texto

La cantidad de veces que cada palabra se encuentra en el fichero y que palabra aparece más que otras.

1. ¿Que palabra es la que más ocurre en esta frase?

the clown ran after the car and the car ran into the tent and  
the tent fell down on the clown and the car



# 1. Contar las palabras

- La palabra: “the”
- Para humanos es difícil
- Imagina un texto más largo!
- Para ordenadores es fácil, son muy buenos procesando texto!
- Pero tenemos que hablar su lenguaje para poder instruirlos y decir ¿cómo? queremos procesar
- A través de programas en un lenguaje de programación



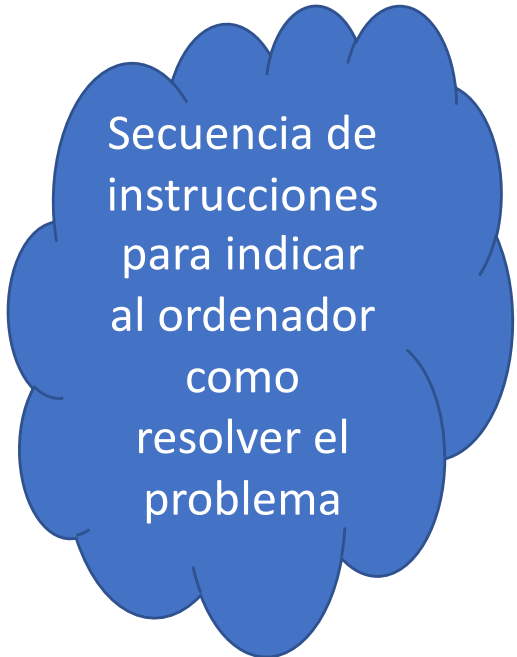
# 1. Programa en Python: words.py

```
text = input('Enter text:')

counts = dict()
words = text.split()
for word in words:
    counts[word] = counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```



Secuencia de  
instrucciones  
para indicar  
al ordenador  
como  
resolver el  
problema

# 1. Más ejemplos de problemas algorítmicos

- Ejemplos concretos de problemas algorítmicos:
  - Determinar el producto de dos números  $a$  y  $b$ .
  - Determinar la raíz cuadrada positiva de un número  $n$  cualquiera.
  - Determinar si el número  $n$ , entero mayor que 1, es primo.
  - Dada una lista de palabras, determinar las palabras repetidas.
  - Determinar si una palabra está en un diccionario.
  - Separar una palabra en sus sílabas.
  - Ordenar y listar alfabéticamente un conjunto de palabras.
  - Dibujar en pantalla un círculo de radio  $r$ .
  - Comparar documentos según cierto criterio establecido.
  - ...

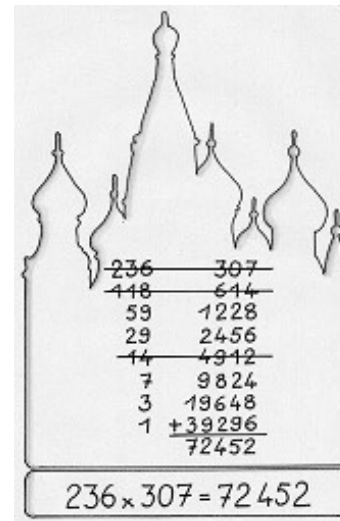
# 1. Ejemplos de algoritmos y problemas

## Multiplicación de dos números naturales: 2 algoritmos

- Multiplicación clásica

$$\begin{array}{r} 981 \\ \times 1234 \\ \hline 3924 \\ 2943 \\ 1962 \\ 981 \\ \hline 1210554 \end{array}$$

- Multiplicación *Rusa*



236	307
118	614
59	1228
29	2456
14	4912
7	9824
3	19648
1	39296
<hr/>	
72452	

$236 \times 307 = 72452$



# 1. Ejemplos de algoritmos y problemas

## Algoritmo de la multiplicación rusa

Poner aqui el  
numero más  
pequeño

Saber hacer:

- Division entera entre dos  $\div 2$
- Multiplicar por dos  $\times 2$
- Sumar  $+$

	Multiplicando	Multiplicador
	34	56
$\div 2$	17	112
$\times 2$	8	224
	4	448
	2	896
	1	1792
	0	3584

¿impar?

+

1904

## Actividad: algoritmo de multiplicación rusa

	<div>/</div> <div>2</div>	<div>*</div> <div>2</div>
	Multiplicando	Multiplicador
	77	1234

¿impar?

+

# Actividad: algoritmo de multiplicación rusa

- Corta con tijeras las siguientes piezas
- Compone el algoritmo utilizando las piezas

2x **incrementar**  **en**

**dividir**  **por**

3x **fijar**  **en**

**es impar(**  **)**

**≠**

**si**

4x **multiplicando**


4x **multiplicador**

2x **producto**

**981**

**1234**

**hacer**   
  
  
**mientras**



# 1. Problemas, algoritmos y programas

## Actividad: algoritmo de multiplicación rusa

fijar	multiplicando	en	981
-------	---------------	----	-----

fijar	multiplicador	en	1234
-------	---------------	----	------

fijar	producto	en	0
-------	----------	----	---

mientras	multiplicando	≠	0	hacer
----------	---------------	---	---	-------

si	es impar(	multiplicando	)
----	-----------	---------------	---

incrementar	producto	en	multiplicador
-------------	----------	----	---------------

dividir	multiplicando	por	2
---------	---------------	-----	---

incrementar	multiplicador	en	multiplicador
-------------	---------------	----	---------------

# 1. Problemas, algoritmos y programas

## **Actividad: algoritmo de multiplicación rusa**

- ❑ Resuelve el cuestionario en Lessons

# 1. Programa en Python: rusa.py

Secuencia de instrucciones para indicar al ordenador como resolver el problema

```
rusa.py x
1
2 # Calcular el producto del multiplicando x multiplicador
3 # Aplicando el algoritmo para multiplicacion a la Rusa
4
5 multiplicando = 981
6 multiplicador = 1234
7 producto = 0
8
9 while multiplicando != 0:
10     if multiplicando % 2 != 0:
11         producto = producto + multiplicador
12         multiplicando = multiplicando//2
13         multiplicador = multiplicador * 2
14
15 print ("El resultado es ", str(producto))

Shell x
>>> %Run rusa.py
El resultado es 1210554
>>>
```

Para copiarlo en <https://code.sololearn.com/#py>

```
# Calcular el producto del multiplicando x multiplicador  
# Aplicando el algoritmo para multiplicacion a la Rusa
```

```
multiplicando = 981  
multiplicador = 1234  
producto = 0
```

```
while multiplicando != 0:  
    if multiplicando % 2 != 0:  
        producto = producto + multiplicador  
    multiplicando = multiplicando//2  
    multiplicador = multiplicador * 2
```

```
print ("El resultado es ", str(producto))
```

## 2. Programas y la actividad de la Programación

- Un **algoritmo** es una **secuencia de instrucciones** específica de las acciones que ha de ejecutar y en qué **orden** para completar una **tarea** determinada.
- Un **procesador** de un ordenador es la entidad capaz de interpretar y ejecutar un cierto repertorio de instrucciones.
- Un **lenguaje de programación** es una notación, conjunto de reglas y definiciones que determinen:
  - tanto lo que se puede escribir en un programa (y el procesador puede interpretar)
  - como el resultado de la ejecución de este programa por el procesador.
- Un **programa** es un algoritmo escrito en un lenguaje de programación para que el ordenador puede completar la tarea
- La **Programación** es la disciplina relacionada con:
  - La resolución de problemas mediante algoritmos
  - Escribir el algoritmo como programa
  - Testear el programa ejecutándolo con el ordenador



## 2. Programas y la actividad de la Programación

- La **programación** es la disciplina relacionada con:
  - **Resolver problemas** mediante algoritmos
  - **Codificar el algoritmo como programa** en un determinado lenguaje
  - **Testear el programa** ejecutándolo con el ordenador
- Ya vemos que es mucho más que solo aprender un lenguaje de programación.
- Creatividad para resolver problemas
- Motivación para aprender un lenguaje
- Ser critico testeando tu programa

### 3. Ordenadores (definiciones)

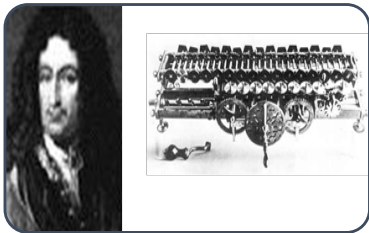
- Ordenador: conjunto de sistemas mecánicos y electrónicos diseñado para la gestión automática de los datos y que puede ser programado
- Hardware (lo que se puede romper con un martillo)
  - Parte física de un ordenador
  - Dispositivos de entrada: ratón, teclado, joystick, red, etc.
  - Dispositivos para el proceso de la información: procesador, memoria principal, placa base, tarjeta gráfica,....
  - Dispositivos de almacenamiento de la información: disco duro, cds, DVDs, memorias USB, ..., etc.
  - Dispositivos de salida: monitor, impresora, red, plotter,...
- Software
  - Parte lógica de un ordenador: sistema operativo, programas, datos, etc
  - Sistema operativo
  - Programas
  - Datos

### 3. Ordenadores

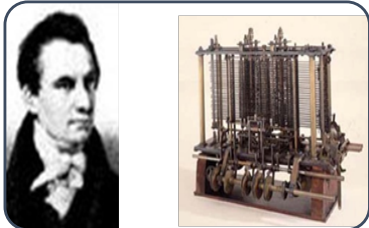
Los orígenes de los ordenadores se encuentran en las máquinas de cálculo, máquinas para resolver problemas específicos.



Máquina sumadora de Pascal, alrededor de 1640.



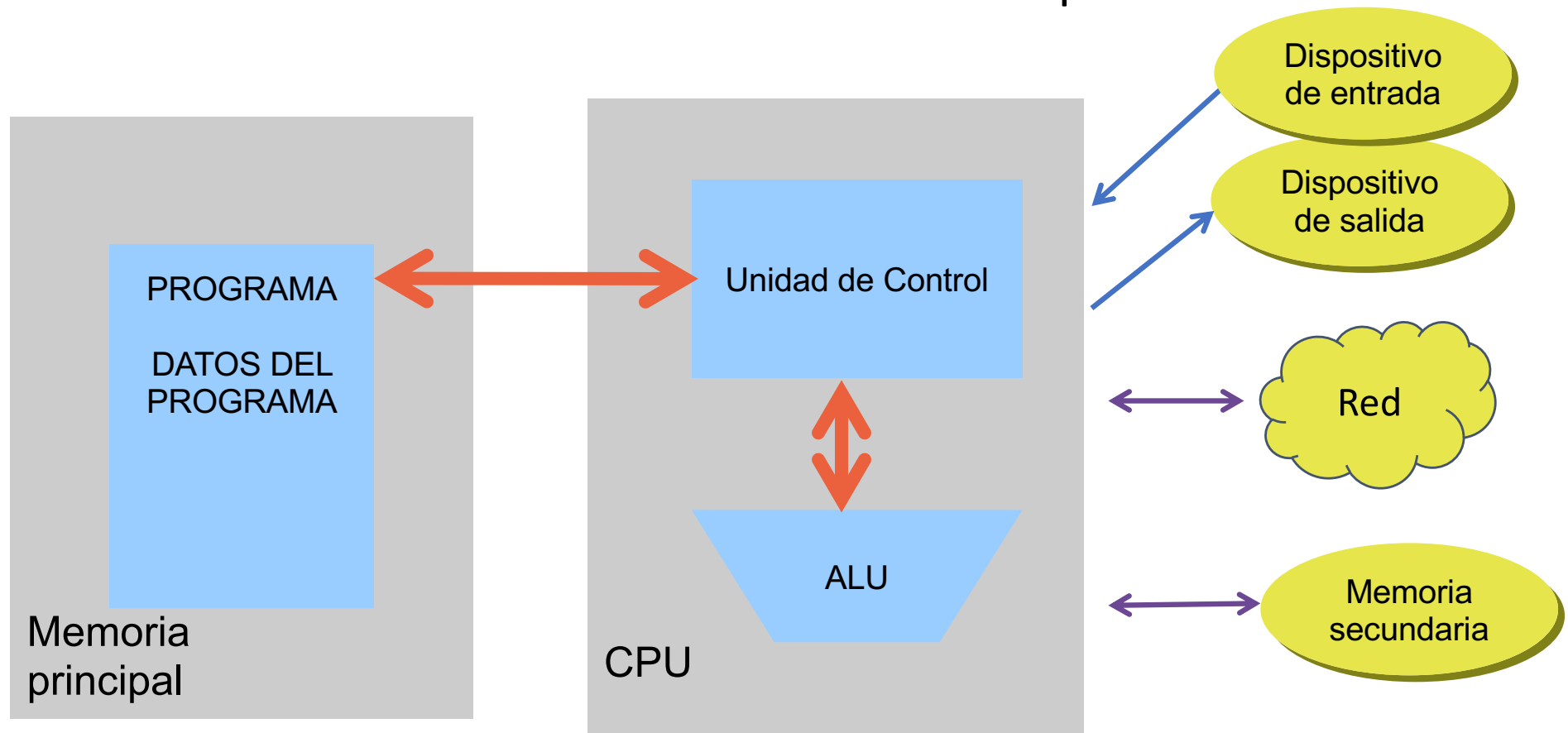
Máquina de Leibniz, alrededor de 1670 (sumas, restas, productos, divisiones, raíces cuadradas).



Máquina diferencial de Babbage, alrededor de 1820, tabulación de polinomios y Máquina analítica de Babbage, diseño alrededor de 1840, *máquina programable* que se debería usar con un lenguaje de programación precursor de los lenguajes modernos.

### 3. Ordenadores (arquitectura)

- En el siglo 20 se obtuvieron los primeros ordenadores o máquinas de propósito general.
- La **máquina von Neumann**, arquitectura desarrollada en los 40, y usada en los ordenadores electrónicos de la época.



### 3. Ordenadores (más definiciones)

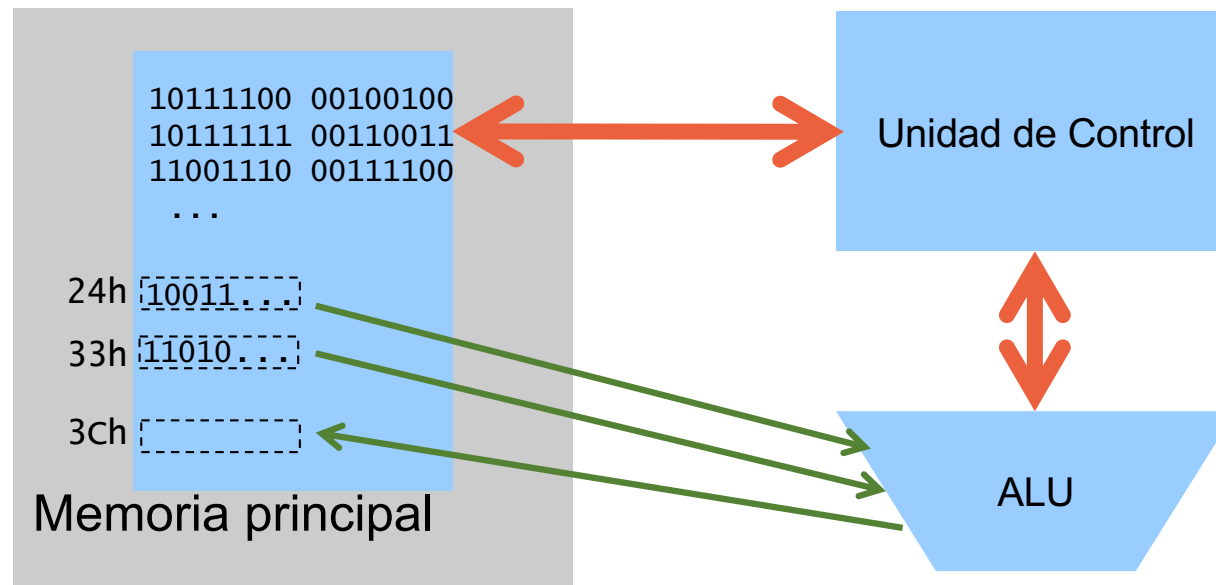


- Unidad de procesamiento central (**CPU**): El corazón de cualquier computadora. Es lo que ejecuta el software que escribimos. También recibe el nombre de “CPU” por sus siglas en inglés (Central Processing Unit), o simplemente, “el procesador”.
- Unidad aritmética lógica (**ALU**): es un circuito digital que calcula operaciones aritméticas (como suma, resta, multiplicación, etc.) y operaciones lógicas. También conocida como “ALU” (siglas en inglés de arithmetic logic unit).
- **Memoria Principal**: Almacenamiento pequeño y temporario pero rápido –que se pierde al reiniciar– se la conoce como RAM
- **Memoria Secundaria**: Almacenamiento permanente y grande pero más lento – la información permanece hasta que se la elimina– unidad de disco, tarjeta de memoria

## 4. Lenguajes de programación

- A nivel máquina, un programa y los datos, se codifican como números en binario: ***lenguaje máquina***.
- Ejemplo: Multiplicar unos ciertos valores en memoria.

Código de la operación	Posición de memoria sobre la que trabaja	
10111100	00100100	cargar el valor que está en 24h
10111111	00110011	multiplicar por el valor que está en 33h
11001110	00111100	almacenar el resultado en 3Ch



### 3. Lenguajes de programación

- Es evidente que los programas escritos en lenguaje máquina resultan ilegibles.
- Los ***lenguajes ensambladores*** usan mnemónicos e identificadores para instrucciones y datos.

Load 24,	10111100	00100100	cargar el valor en 24h
Multiply 33,	10111111	00110011	multiplicar por el valor valor en 33h
Store 3C,	11001110	00111100	almacenar en 3Ch

- Los dos son lenguajes tan próximos a la máquina que se conocen como ***lenguajes de bajo nivel***.
- Sería deseable poder expresarlo con una notación del siguiente estilo:

c <-- a\*b

En donde a, b, c serían nombres para los datos a multiplicar y el resultado respectivamente.

### 3. Lenguajes de programación

- Frente a los lenguajes de bajo nivel se tienen los ***lenguajes de alto nivel*** que:
  - Disponen de operadores y estructuras más próximas a las humanas, lo que permite al programador dar órdenes de forma más sencilla al computador.
  - Son más seguros que el código máquina y ayudan a no cometer errores evidentes.
  - El código es transportable (independiente de la máquina).
  - El código es más legible.
- Los lenguajes de alto nivel que contienen una instrucción como la del ejemplo anterior:

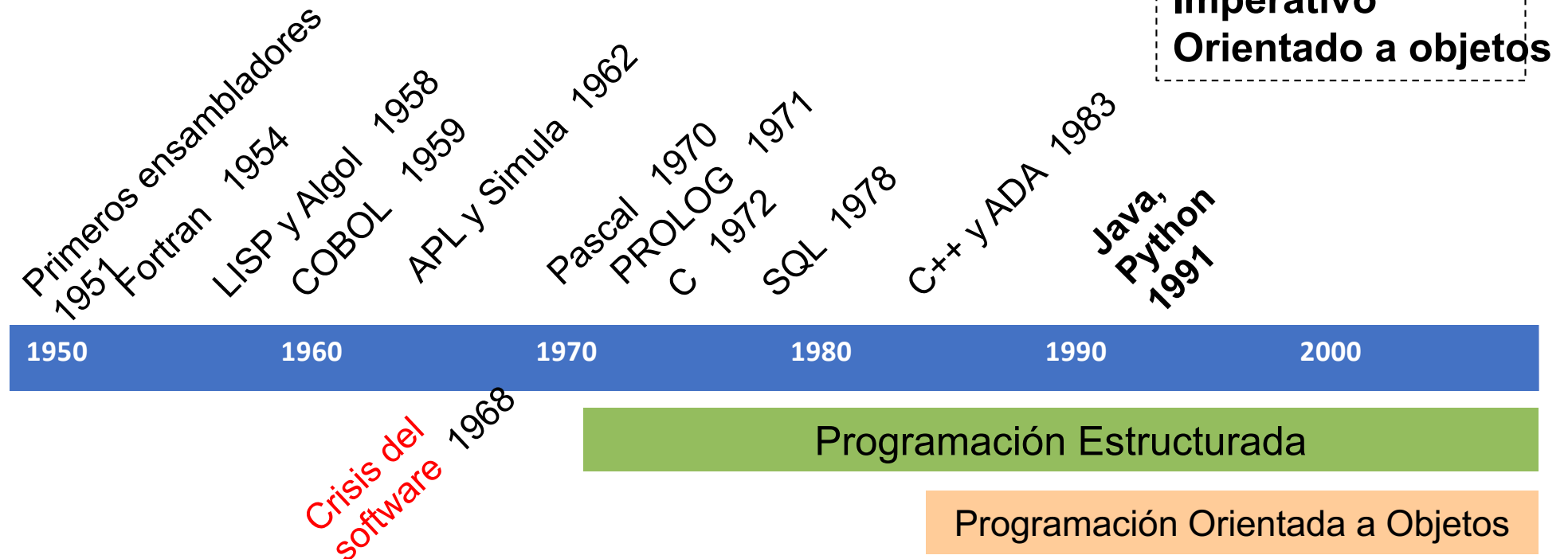
`c <-- a*b`

se denominan ***imperativos***, aunque existen otros modelos como los ***lógicos*** y ***funcionales***.

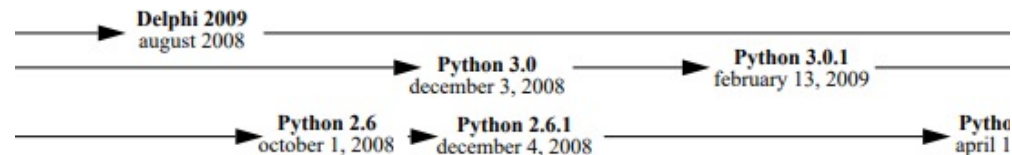


### 3. Lenguajes de programación

- Lista cronológica reducida:



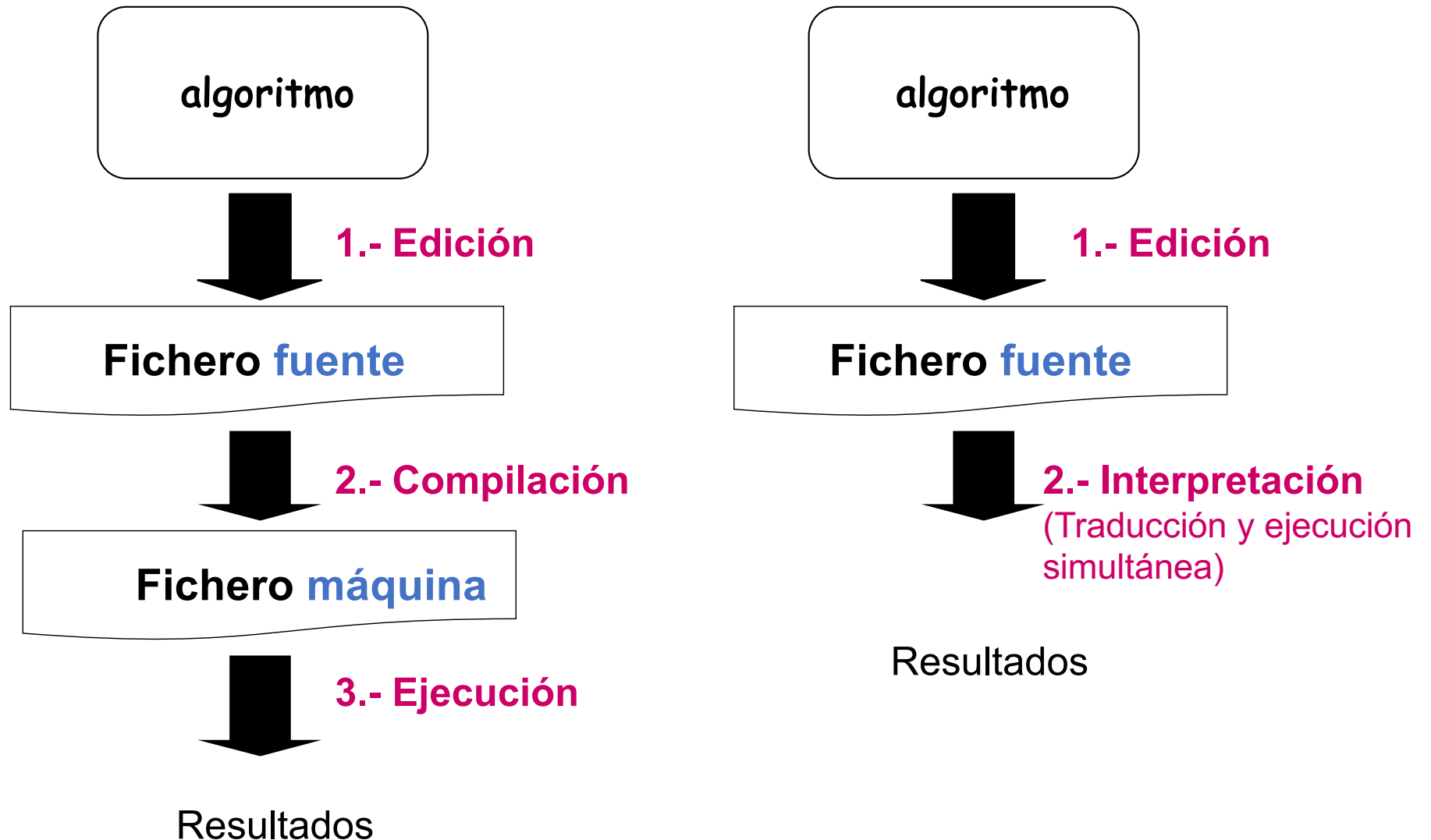
- Árbol genealógico de los lenguajes de programación



## 5. Compiladores e intérpretes

- Existen dos maneras de traducir un programa escrito en un lenguaje de alto nivel a lenguaje máquina: la ***interpretación*** y la ***compilación***.
- En la ***interpretación*** se traduce a lenguaje máquina cada instrucción del lenguaje de alto nivel, una a una, en tiempo de ejecución.
- En la ***compilación*** se traducen (*compilan*) por medio de un programa (*compilador*) todas las instrucciones del lenguaje a lenguaje máquina, previamente a su ejecución.

## 5. Compiladores e intérpretes

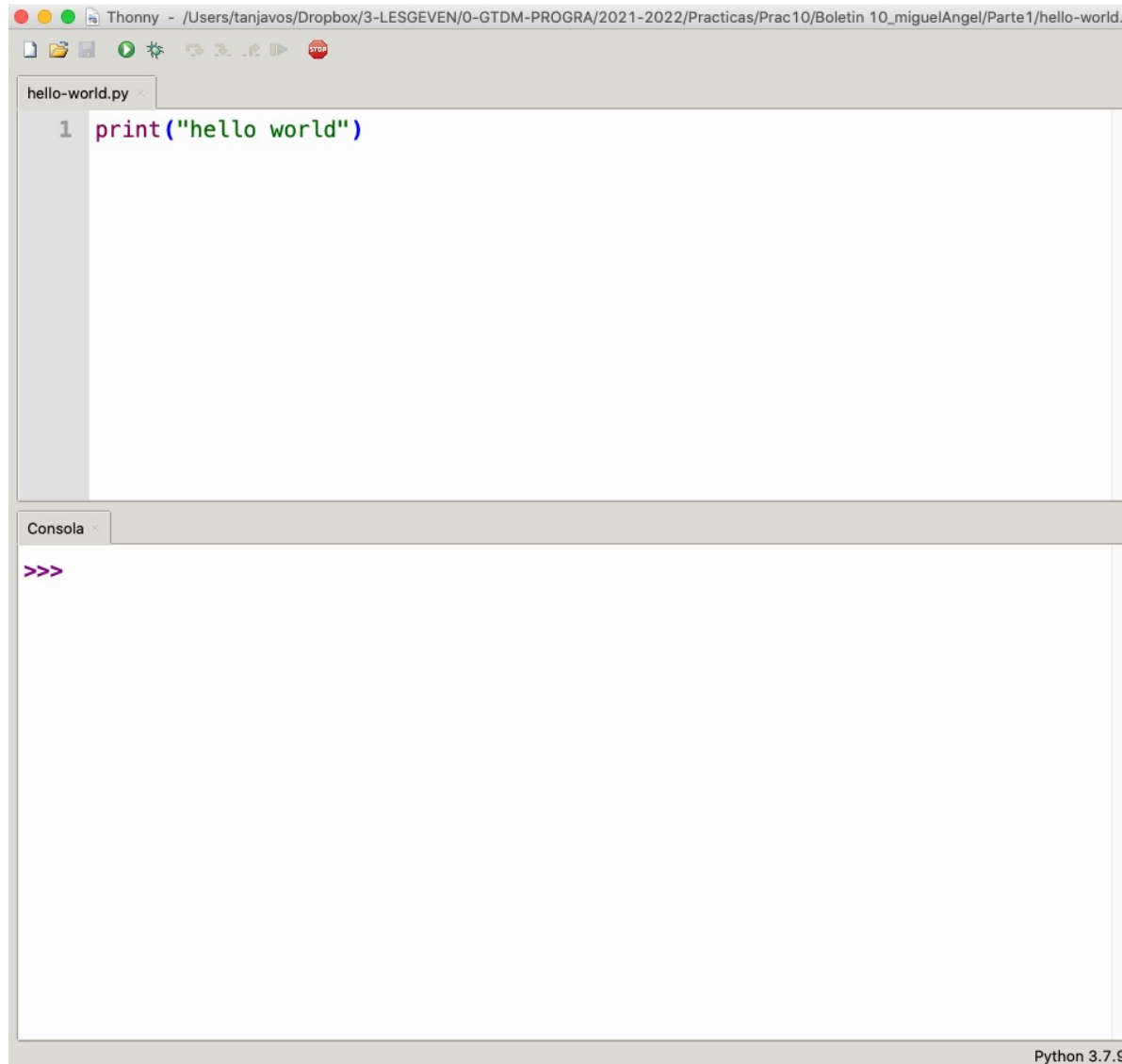


## 6. El lenguaje Python



- El lenguaje que vamos a aprender es Python.
- Python es un tipo de serpiente
- Es un lenguaje muy expresivo
- Fácil de leer
- Tiene entornos de desarrollo (IDEs) que facilitan la programación
- Es un lenguaje interpretado, permite trabajar:
  - Interactivo (hablar directamente y dar instrucciones)
  - Con un scripts (escribir las instrucciones en un fichero .py)
- Lo arrancamos y empezamos!

# Entorno de desarrollo: thonny.org



Editor de programas

Consola o Shell

## 6. Hablar con Python

¿qué es la  
siguiente  
cosa que  
quieres que  
hago?



```
Shell x
Python 3.7.2 (bundled)
>>> |
```

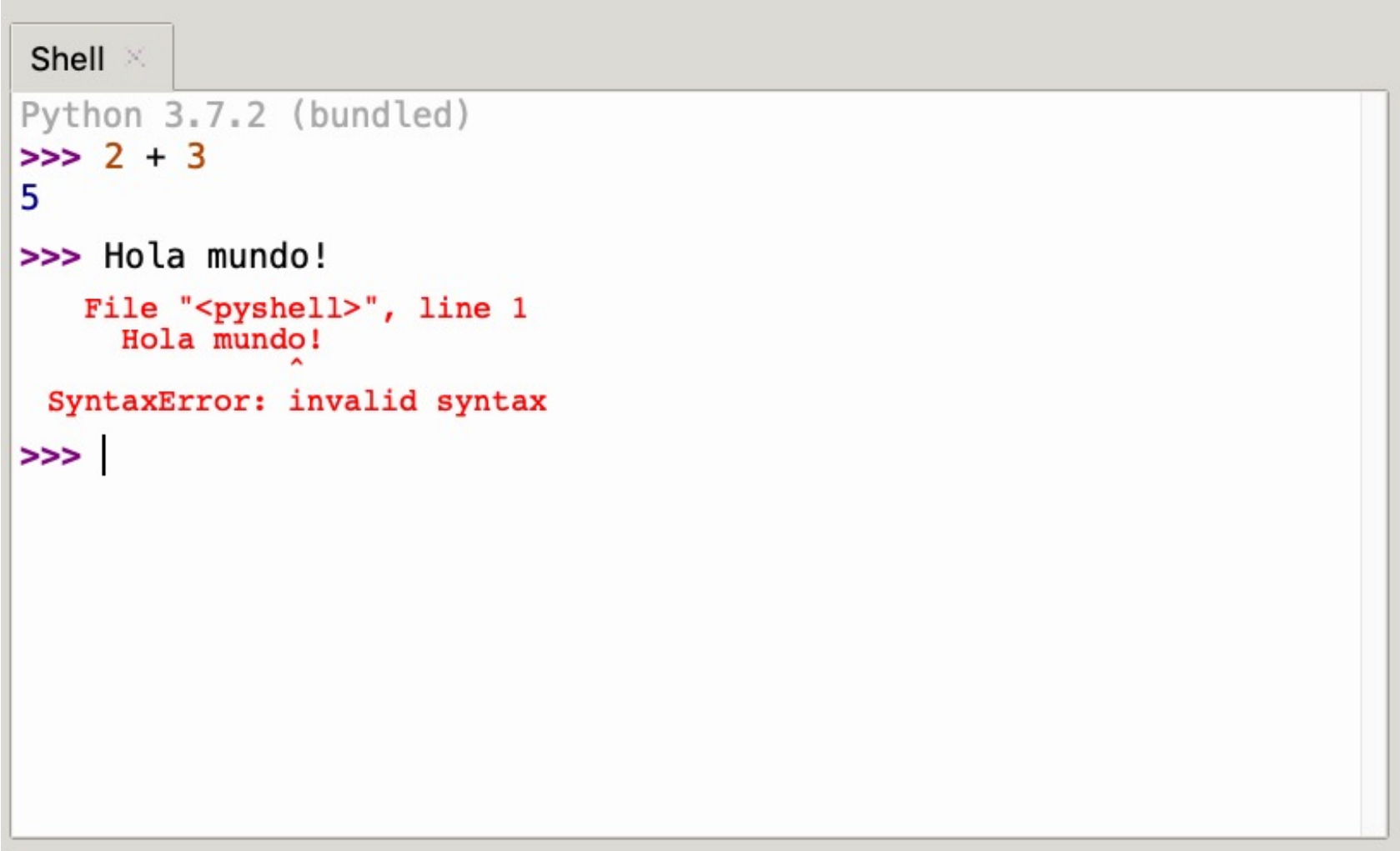
## 6. Hablar con Python



A screenshot of a Python Shell window. The window has a title bar with the text "Shell" and a close button. The main area of the window displays the text "Python 3.7.2 (bundled)" in a light gray font. Below this, the prompt ">>>" is followed by the expression "2 + 3" in orange. The result "5" is displayed in blue. The prompt ">>>" is shown again on the next line.

```
Shell ×  
Python 3.7.2 (bundled)  
>>> 2 + 3  
5  
>>>
```

## 6. Hablar con Python



```
Shell x
Python 3.7.2 (bundled)
>>> 2 + 3
5
>>> Hola mundo!
      File "<pyshell>", line 1
        Hola mundo!
            ^
      SyntaxError: invalid syntax
>>> |
```



## 6. Errores de sintaxis en Python

- Ordenadores son mucho más estrictos entendiendo frases escritos en un lenguajes de programación que nosotros humanos con el lenguaje natural.
- Python te esta diciendo que no te entiende
- Es más estricto que nosotros humanos

## 6. Recuerda estas instrucciones para humanos

Mientras suena la música:

Mano izquierda hacia adelante  
Mazo derecha hacia adelante  
Mano izquierda hacia atrás  
Mano derecha hacia atrás  
Mano izquierda a hombro derecho  
Mano derecha a hombro izquierdo  
Mano izquierda a la nuca  
Mazo derecha a la nuca  
Mano izquierda a caldera derecha  
Mano derecha a caldera izquierda  
Mano izquierda a nalgas izquierdas  
Mano derecha a nalgas derechas  
Meneo  
Meneo  
Salto



<https://www.youtube.com/watch?v=anzzNp8HIVQ>

## 6. Error de sintaxis

Mientras suena la música:

Mano izquierda hacia adelante

**Mazo** derecha hacia adelante

Mano izquierda hacia atrás

Mano derecha hacia atrás

Mano izquierda a hombro derecho

Mano derecha a hombro izquierdo

Mano izquierda a la nuca

**Maho** derecha a la nuca

Mano izquierda a **caldera** derecha

Mano derecha a **caldera** izquierda

Mano izquierda a nalgas izquierdas

Mano derecha a nalgas derechas

Meneo

Meneo

Salto



<https://www.youtube.com/watch?v=gwWRjvwILKg>

## 6. El lenguaje Python

- Para conseguir que Python nos entiende tenemos que hablar su lenguaje
- Cualquier lenguaje tiene:
  - Vocabulario (palabras)
    - Artículos (el, la, los, etc.)
    - Sustantivos (serpiente, libro, estudiante, etc.)
    - Adjetivos (gracioso, bonito, feo, etc.)
    - Etc.
  - Estructuras para hacer frases (oraciones) con estas palabras
    - El serpiente gracioso.
    - El libro bonito.
  - Estructuras para juntar frases y crear un relato
    - Un serpiente gracioso y el libro bonito.

## 6. El lenguaje Python

- En Python:
  - Vocabulario (palabras)
    - Variables
    - Constantes
    - Funciones y operadores
    - Palabras reservadas
  - Estructura para las frases
    - Patrones de sintaxis validos para crear instrucciones
  - Estructura para contar un relato
    - Patrones para estructurar instrucciones y crear un programa que resuelva un problema
      - Estructura secuencial
      - Estructura condicional (p.ej. if-then)
      - Estructura repetida (p.ej. while)

## 6. Ejemplos de sintaxis para frases (o instrucciones)

```
x = 2
```

```
x = x + 2
```

```
print(x)
```



Asignación

Asignación con calculo

Función para imprimir

Variable

Operador

Constante

Función

## 6. Ejemplo

Shell ×

```
>>> x = 2
>>> y = 4
>>> z = x + y
>>> print (z)
```

6

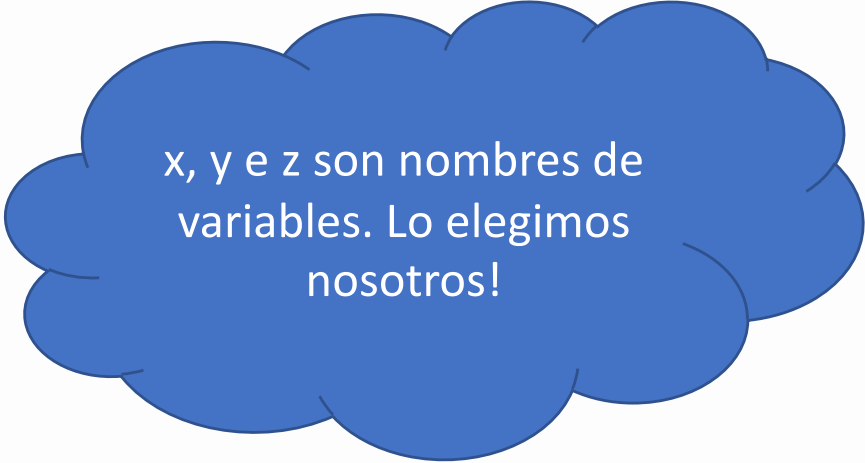
```
>>>
```

Guardar el valor 2 en la variable x

Guardar el valor 4 en la variable y

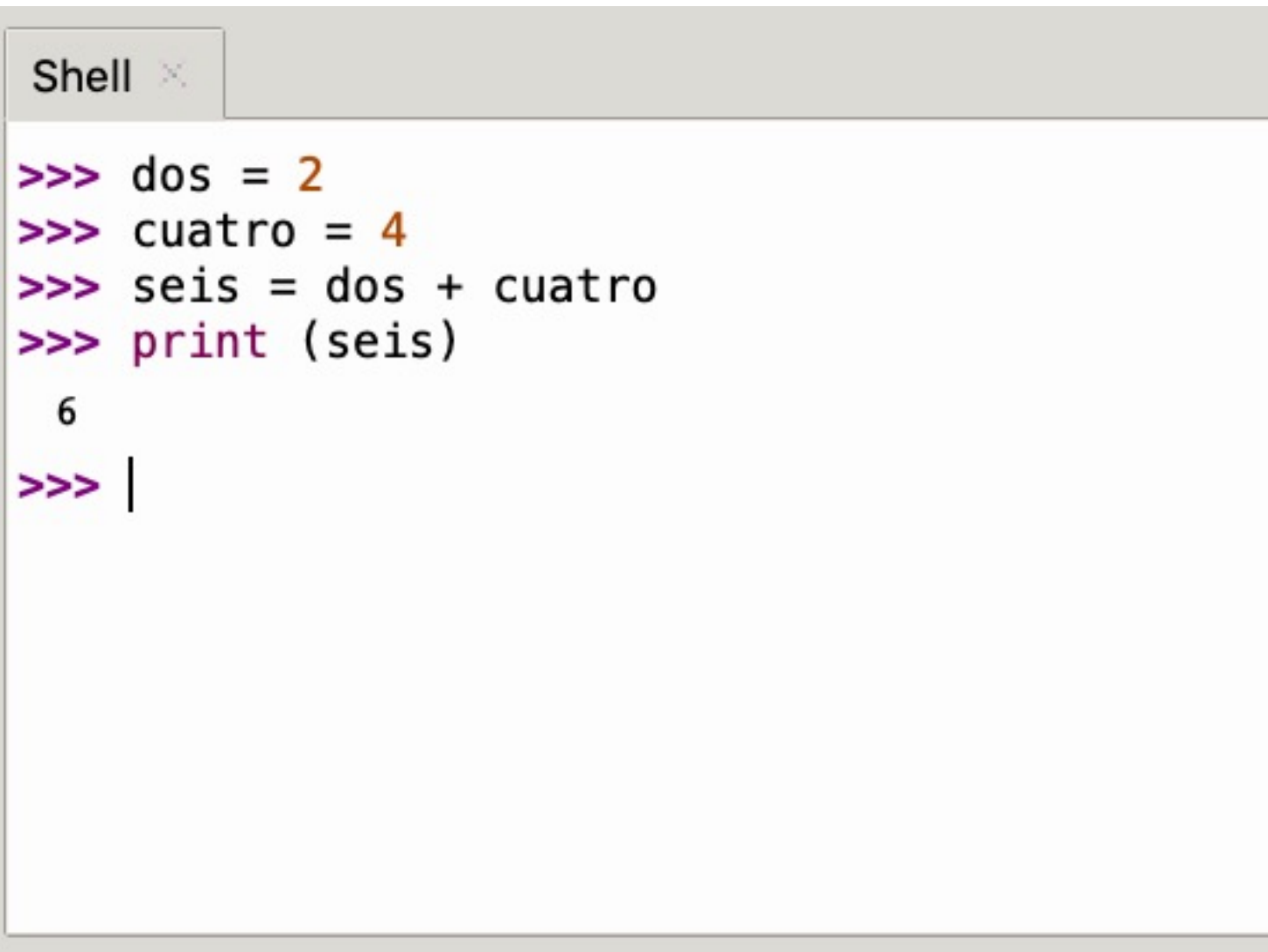
Suma el valor de variable x e y, e guarda el resultado en la variable z

Imprimir el valor de la variable z



x, y e z son nombres de variables. Lo elegimos nosotros!

## 6. Ejemplo: lo mismo con otros nombres



```
Shell x
>>> dos = 2
>>> cuatro = 4
>>> seis = dos + cuatro
>>> print (seis)
6
>>> |
```



## 6. Palabras reservadas para Python

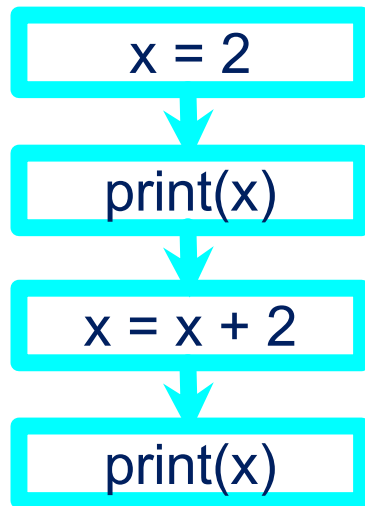
- No se pueden usar como nombre de variables

False	class	return	is	finally
None	if	for	lambda	continue
True	def	from	while	nonlocal
and	del	global	not	with
as	elif	try	or	yield
assert	else	import	pass	
break	except	in	raise	

## 6. Componer instrucciones para crear programas

- Estructura secuencial
- Estructura condicional (p.ej. if-then)
- Estructura repetida (p.ej. while)

## 6. Secuencial



Program:

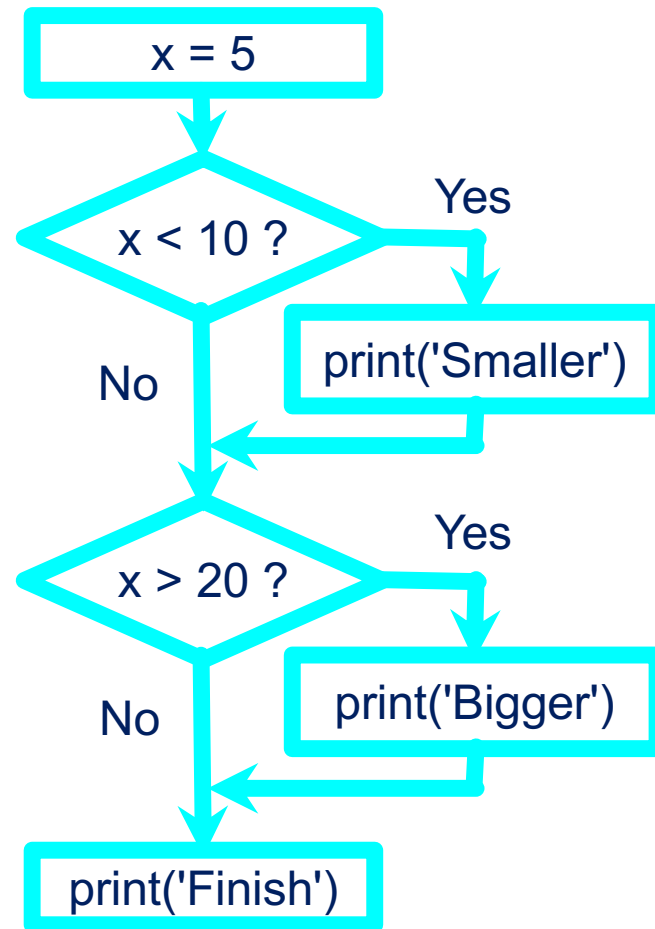
```
x = 2  
print(x)  
x = x + 2  
print(x)
```

Output:

2  
4

Una instrucción detrás de otra

## 6. Condicional



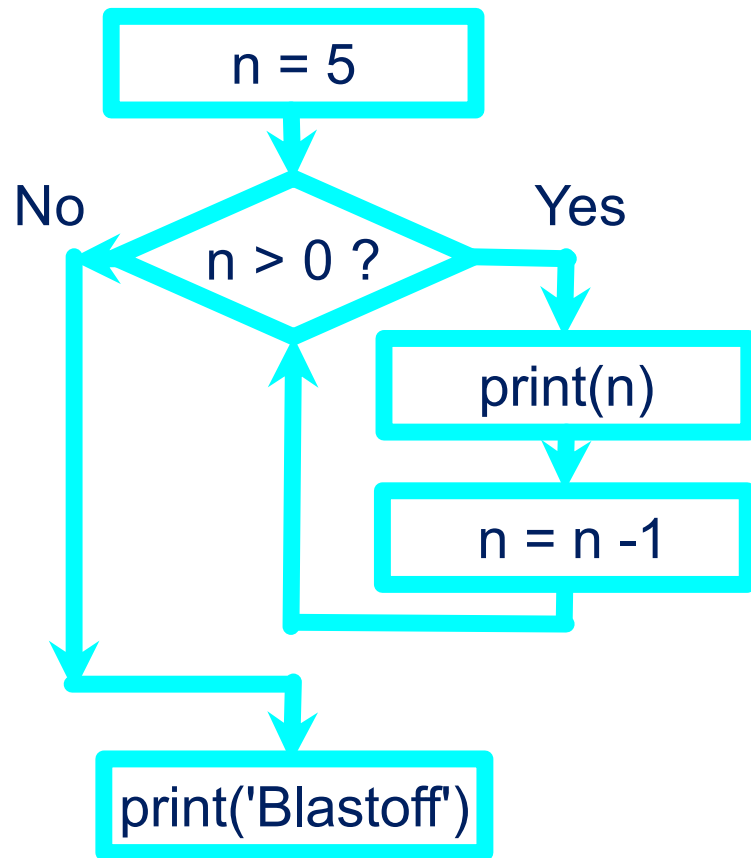
Program:

```
x = 5
if x < 10:
    print('Smaller')
if x > 20:
    print('Bigger')
print('Finish')
```

Output:

Smaller  
Finish

## 6. Repetida



Program:

```
n = 5
while n > 0 :
    print(n)
    n = n - 1
print('Blastoff!')
```

Output:

5  
4  
3  
2  
1  
Blastoff!

## 6. Sintaxis y Semántica

- Errores de sintaxis
  - Significa que hemos violado las reglas de gramática
  - **Girar a la izquierda**
- Errores de semántica
  - No significa lo que pretendes decir
  - **Todos los profesores de programación están muy buenos**

## 6. Sintaxis y Semantica

- Errores de sintaxis
  - Significa que hemos violado las reglas de gramatica de Python
  - Detectados por el interpreter
- Errores de semantica
  - No hace lo que pretende hacer

```
Shell x
Python 3.7.2 (bundled)
>>> 2 + 3
5
>>> Hola mundo!
      File "<pyshell>", line 1
        Hola mundo!
            ^
      SyntaxError: invalid syntax
>>> |
```

```
Shell x
>>> print (dos + tres)
9
>>>
```

## 6. El camino hacia el aprendizaje





¡Solo hay una forma de aprender a programar!



practicando, practicando, practicando, practicando