



**Junta de Andalucía**  
Consejería de Educación y Deporte

# UT 1

## Introducción a la programación

—

## Módulo de Programación 1º DAW



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).

Autor: Fran Gómez

# Objetivos



- Conocer los **conceptos básicos** relacionados con la programación y el diseño de aplicaciones
- Describir los **paradigmas** de programación más usados
- Clasificar los **lenguajes** de programación.
- Configurar el **entorno de desarrollo**
- Crear **algoritmos y programas** básicos

# RA y CE implicados



**RA1:** Reconoce la estructura de un programa informático, identificando y relacionando los elementos propios del lenguaje de programación utilizado.

- a) Se han identificado los bloques que componen la estructura de un programa informático
- b) Se han creado proyectos de desarrollo de aplicaciones
- c) Se han utilizado entornos integrados de desarrollo.
- d) Se han identificado los distintos tipos de variables y la utilidad específica de cada uno.
- e) Se ha modificado el código de un programa para crear y utilizar variables.
- f) Se han creado y utilizado constantes y literales.
- g) Se han clasificado, reconocido y utilizado en expresiones los operadores del lenguaje.
- h) Se ha comprobado el funcionamiento de las conversiones de tipo explícitas e implícitas.
- i) Se han introducido comentarios en el código.

# Índice de contenidos



1. Introducción
  - 1.1. Evolución de la programación
  - 1.2. Conceptos básicos
  - 1.3. Clasificación de los lenguajes
2. Paradigmas de programación
  - 2.1. Declarativa
  - 2.2. Imperativa
  - 2.3. Estructurada
  - 2.4. Modular
3. Estructura de un programa
4. Elementos de un programa
  - 4.1. Palabras reservadas
  - 4.2. Operadores
  - 4.3. Tipos de datos
  - 4.4. Estructuras
    - 4.4.1. Secuenciales
    - 4.4.2. Selectivas
    - 4.4.3. Iterativas
    - 4.4.4. Modulares
5. Pseudocódigo
6. Diagramas de flujo
7. Ejercicios
8. Recursos y referencias

# Introducción



## Mi primer programa en bash/shell

```
C:\Windows\system32>
Hello World
Press any key
```



```
helloWorld.bat - Notepad
File Edit Format View Help
ECHO OFF
ECHO Hello World
PAUSE
```

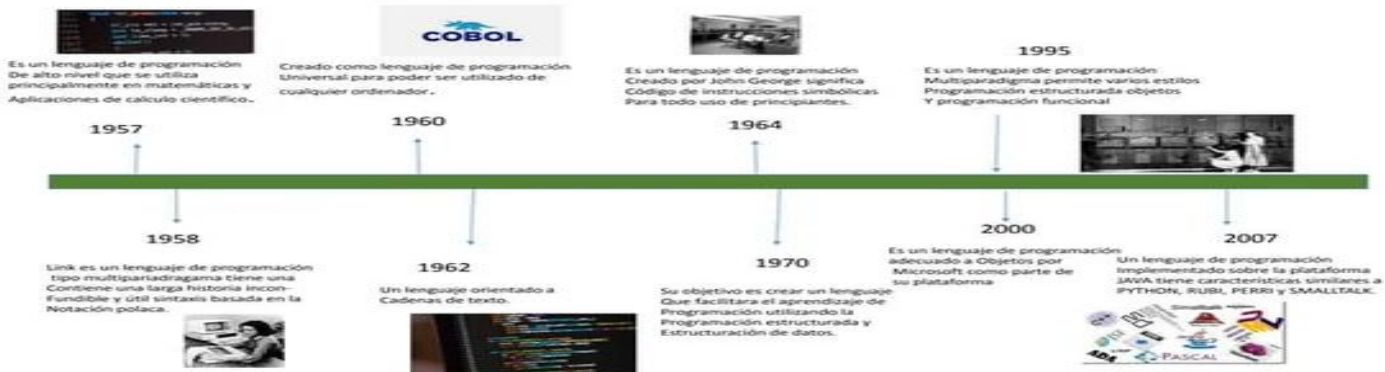
# Evolución de la programación



## Ejercicio 1

Crea una línea del tiempo con los items que más destacarías en la historia de la programación.

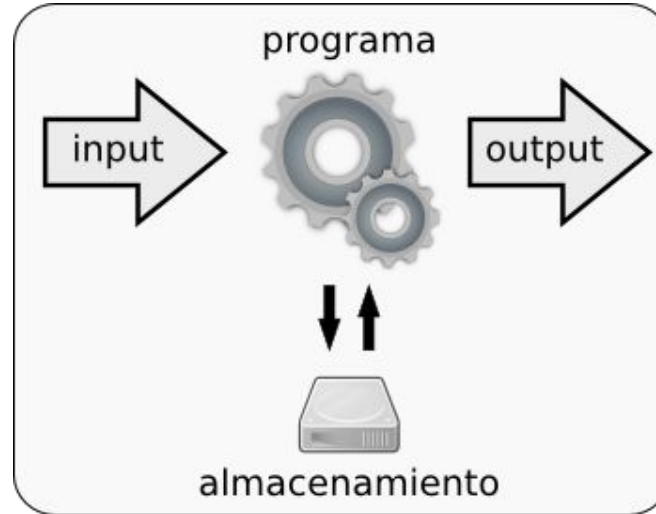
1. Año
2. Nombre (logotipo)
3. Creador
4. Alguna característica



# Conceptos básicos



- Programación
- Algoritmo
- Programar
- Código
- Lenguaje binario
- Lenguaje ensamblador
- Lenguajes de alto nivel



- Entrada
- Salida
- Cambio de estado
- Lógica
- Datos

## Ejercicio 2

Describe en qué consiste la programación y qué diferencia hay entre los algoritmos y Código.



# Clasificación de los lenguajes



- Propósito:
  - General vs específico
- Tipo:
  - Scripting (o interpretado)
  - Compilado
  - Marcado
- Uso:
  - Desarrollo Web
  - Juegos
  - Administración de Sistemas
  - Móvil / Multiplataforma
  - IA, BigData, Machine Learning...
- Plataforma:
  - Web
    - Frontend
    - Backend
  - Desktop
  - Móvil
  - Embedded
- Paradigma:
  - Imperativo
  - Estructurado
  - Orientado a objetos
  - Declarativo

# Clasificación de los lenguajes



## Ejercicio 3

Crea una tabla comparativa de los lenguajes más populares, clasificándolos por parámetros ante

¿Cuál elegirías para programar? ¿por qué?

Lenguaje	Propósito	Tipo	Uso Principal	Plataforma	Ventajas	Desventajas
Python	General	Interpretado	Ciencia de datos, ML, desarrollo web	Web, ciencia de datos, ML	Fácil de aprender, gran comunidad	Velocidad de ejecución
Java						
JavaScript						
PHP						
C#						
C						
C++						
GO						
Rust						
Swift						
Ruby						
Bash/Batch/PowerShell						

# Paradigmas de programación



- **Imperativa**  $\Rightarrow$  Instrucciones paso a paso. “Cómo”
  - **Estructurada** o procedural  $\Rightarrow$  Instrucciones siguiendo estructuras. Eliminar Goto.
  - **Modular**  $\Rightarrow$  Divide el programa en partes. Subrutinas.
- **Declarativa**  $\Rightarrow$  Resultado deseado. “Qué”
  - **Funcional**  $\Rightarrow$  funciones matemáticas
  - **Lógico**  $\Rightarrow$  reglas lógicas
- **Orientada a objetos**  $\Rightarrow$  Agrupa Datos y Lógica

Queremos programar la obtención de los clientes de Madrid mayores de 30 años.

```
SELECT nombre, apellido  
FROM clientes  
WHERE ciudad = 'Madrid' AND edad > 30;
```

**Declaración, no instrucción:** No le dices al ordenador cómo buscar a los clientes, simplemente **declaras** que quieres los nombres y apellidos de aquellos que cumplen las condiciones.

**Foco en el resultado:** Te centras en el qué quieres obtener, no en el **cómo** lo vas a obtener.

**Motor de base de datos:** El motor de base de datos se encarga de optimizar la consulta y encontrar los datos que cumplen con los criterios especificados.

Queremos programar la elaboración de un sándwich

```
#include <stdio.h>

int main() {
    printf("Tomando una rebanada de pan\n");
    printf("Untando mantequilla\n");
    printf("Colocando queso\n");
    printf("Tomando otra rebanada\n");
    printf("Cerrando el sándwich\n");
    return 0;
}
```

**Instrucciones secuenciales:** Cada línea de código representa una instrucción específica que se ejecuta una después de la otra.

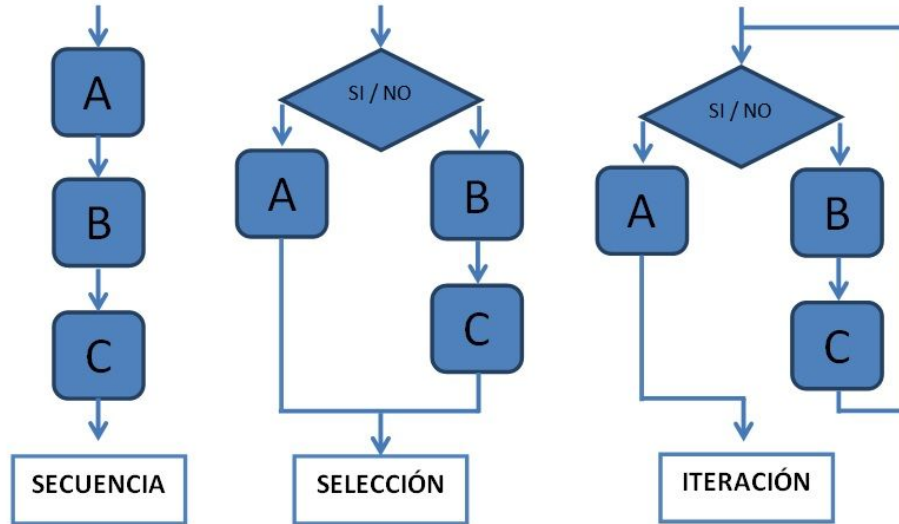
**Estado mutable:** El estado del sándwich va cambiando a medida que se ejecutan las instrucciones.

**Foco en el proceso:** Nos enfocamos en describir cómo hacer el sándwich, paso a otro.

# Programación Estructurada



- Secuencial
- Alternativa
- Iterativa

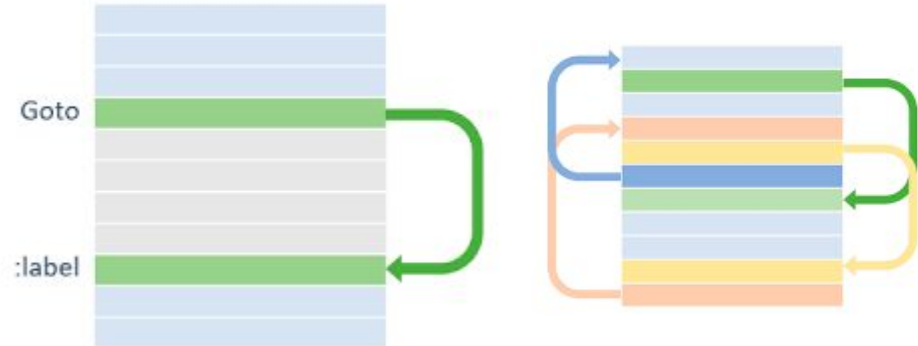


Sentencia GOTO

Código espagueti

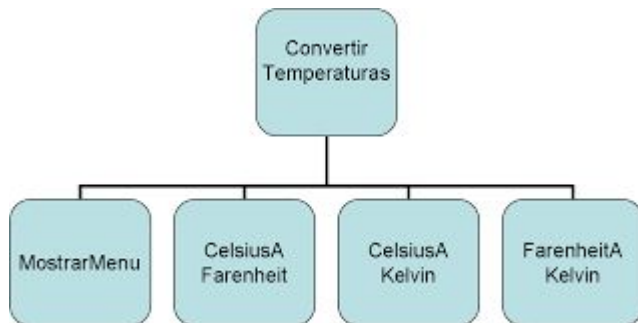


Java Hispano



```
goto etiqueta;  
  
...  
...  
...  
  
etiqueta: sentencia;
```

- Los programas crecen y se hacen más complejos  $\Rightarrow$  dividirlos en módulos
- Módulo = subprograma o rutina  $\Rightarrow$  funciones
- Librerías





# Clasificación de los lenguajes



## Ejercicio 3 (Continuación)

Completa el ejercicio 3 añadiendo una columna para indicar el paradigma

¿Cambiar por q


Lenguaje	Propósito	Tipo	Uso Principal	Plataforma	Ventajas	Desventajas	Paradigma
Python	General	Interpretado	Ciencia de datos, ML, desarrollo web	Web, ciencia de datos, ML	Fácil de aprender, gran comunidad	Velocidad de ejecución	Multiparadigma
Java							
JavaScript							
PHP							
C#							
C							
C++							
GO							
Rust							
Swift							
Ruby							
Bash/Batch/PowerShell							

## Your Wooclap question will appear here

1

Install the **Chrome** or **Firefox** extension 

2

Click on “Add a Wooclap vote” and make sure you are logged into your Wooclap account 

3

Ensure you are in **presentation mode** on Google Slides 

**wooclap**

# Estructura

ENCABEZADOS /

PROGRAMA nomb

CONSTANTES

Declaración y a

VARIABLES

Declaración de

FUNCIONES / MÓD

INICIO

Instrucciones c

FIN

```
// Importaciones
import java.util.Scanner;

public class ProgramaCompleto {

    // Declaración de variables
    static int edad = 25;
    static String nombre = "Ana";

    // Funciones / Métodos
    public static String saludar(String persona, int edad) {
        // Devuelve un mensaje de saludo con nombre y edad
        return "Hola " + persona + ", tienes " + edad + " años.";
    }

    public static double calcularAreaCirculo(double radio) {
        // Calcula el área de un círculo
        return Math.PI * radio * radio;
    }

    // Bloque principal / Ejecución
    public static void main(String[] args) {
        String mensaje = saludar(nombre, edad);
        System.out.println(mensaje);

        double radio = 5;
        double area = calcularAreaCirculo(radio);
        System.out.printf("El área del círculo de radio %.2f es %.2f\n", radio, area);
    }
}
```

```
# Importaciones
import math

# Declaración de variables
edad = 25
nombre = "Ana"

# Funciones / Módulos
def saludar(persona, edad):
    """Devuelve un mensaje de saludo con nombre y edad"""
    return f"Hola {persona}, tienes {edad} años."

def calcular_area_circulo(radio):
    """Calcula el área de un círculo"""
    return math.pi * radio ** 2

# Bloque principal / Ejecución
if __name__ == "__main__":
    mensaje = saludar(nombre, edad)
    print(mensaje)

    radio = 5
    area = calcular_area_circulo(radio)
    print(f"El área del círculo de radio {radio} es {area:.2f}")
```

# Elementos de un programa



1. **Entrada**
  2. Procesamiento
  3. Salida
  4. Palabras reservadas
  5. Comentarios
- **Datos:** La información que el programa recibe del usuario o de otros dispositivos para realizar sus cálculos o tomar decisiones.
  - **Dispositivos de entrada:** Teclado, mouse, escáner, micrófonos, etc.

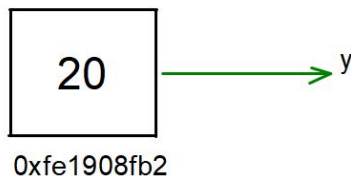
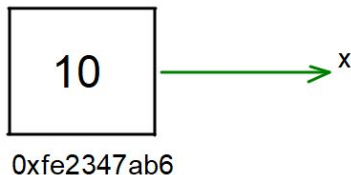


# Elementos de un programa



1. Entrada
2. **Procesamiento**
3. Salida
4. Palabras reservadas
5. Comentarios

```
instrucción 1
instrucción 2
.
.
.
instrucción n
```



- **Instrucciones:** Órdenes o sentencias. Unidad mínima.
- **Variables:** Espacios en la memoria del ordenador donde se almacenan los datos que el programa está utilizando.
- **Constantes:** como una variable cuyo valor inicial no cambia a lo largo del programa
- **Literales:** datos “*a pelo*” en el código.

# Elementos de un programa

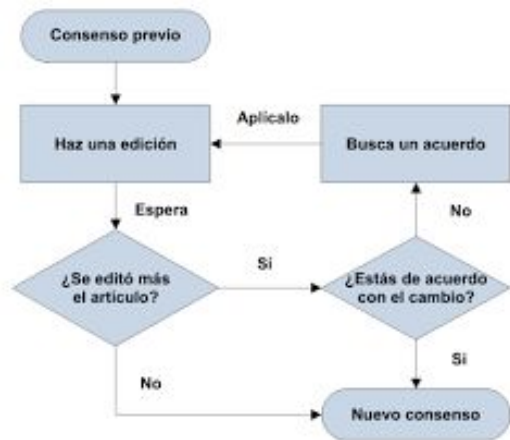


1. Entrada
2. **Procesamiento**
3. Salida
4. Palabras reservadas
5. Comentarios

- **Operadores:** Símbolos que permiten realizar operaciones matemáticas, lógicas o de comparación sobre los datos.

$+$ ,  $-$ ,  $*$ ,  $/$ ,  $=$

- **Estructuras de control:** Mecanismos que permiten controlar el flujo de ejecución del programa, como las condicionales (si, entonces, sino) y los bucles (para, mientras).



# Elementos de un programa



1. Entrada
2. **Procesamiento**
  - **Expresiones:** Son combinaciones de literales, constantes, variables y operadores para ejecutar una operación.  
  
7 + 3                      edad < 12                      PI \* r^2
3. Salida
4. Palabras reservadas
5. Comentarios
  - **Asignación:** Operación que toma el valor de una expresión y lo almacena en una variable  
  
nombre = "Fran"                      suma = 2 + 3

# Elementos de un programa



1. Entrada
2. Procesamiento
3. **Salida**
4. Palabras reservadas
5. Comentarios

- **Resultados:** La información que el programa genera a partir del procesamiento de los datos de entrada.
- **Dispositivos** de salida: Pantalla, impresora, altavoces, etc.

```
C:\Windows\system32\cmd.exe - helloWorld.bat
Hello World
Press any key to continue . . .
```





# Elementos de un programa



1. Entrada
2. Procesamiento
3. Salida
4. **Palabras reservadas**
5. Comentarios

Comandos que entiende el programa



## Java keywords

short	if	implements	finally	throw
boolean	void	int	long	while
case	do	switch	private	interface
abstract	default	byte	else	try
for	double	class	catch	extends
final	transient	float	instanceof	package
continue	native	public	break	char
protected	return	static	super	synchronized
this	new	throws	import	volatile

No podemos usarlas para nombres de variables, algoritmos, etc.

# Elementos de un programa



1. Entrada
  2. Procesamiento
  3. Salida
  4. Palabras reservadas
  5. **Comentarios**
- Texto que añade claridad al código, explicando qué hace cada parte del programa. Son muy útiles para documentar el código y facilitar su comprensión por parte de otros programadores o por uno mismo en el futuro.

# Imprime el resultado  
*print (resultado);*

# Ejemplo de Programa



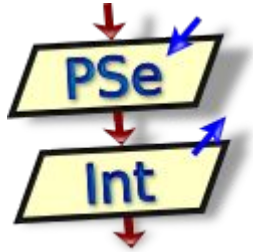
Programa que calcula el área de un círculo.

**Entrada:** El radio del círculo (un número ingresado por el usuario).

**Procesamiento:** Una variable llamada "radio" para almacenar el valor ingresado. Una constante llamada "pi" con el valor 3.14159. Una sentencia que multiplica el radio por sí mismo y luego por pi para calcular el área.

**Salida:** El resultado del cálculo (el área del círculo), mostrado en pantalla.

**Comentarios:** Explicaciones sobre cómo se realiza el cálculo y el significado de cada variable.



Crear un algoritmo sin usar un lenguaje de programación concreto.

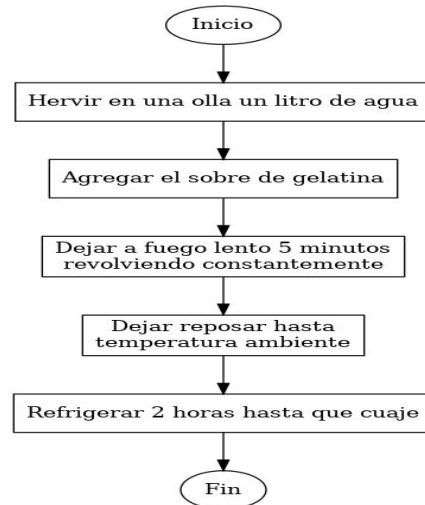
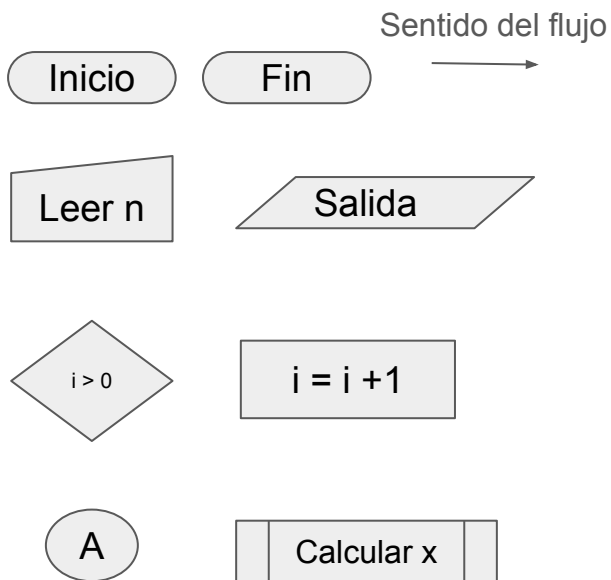
- Lenguaje informal → Cada uno con sus palabras
- Ayuda a ver la lógica antes de pasar a codificar en un lenguaje real
- Facilita comunicación entre programadores
- Sirve para documentar el código
- Útil para aprender a programar

```
1  Algoritmo calcular_area_circulo
2      // Declaración de variables
3      Definir radio, area Como Real
4
5      // Entrada de datos
6      Escribir "Ingrese el radio del círculo: "
7      Leer radio
8
9      // Cálculo del área
10     area ← pi * radio * radio
11
12     // Salida de resultados
13     Escribir "El área del círculo es: ", area
14
15 FinAlgoritmo
```

# Diagramas de flujo



## Representación gráfica de un programa



## Ejercicio 4

Escribe un algoritmo y dibuja el diagrama para calcular el área de un círculo.

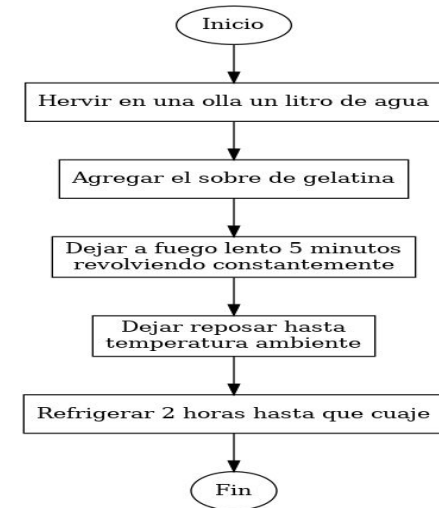
Debe solicitar al usuario los datos de entrada y mostrar el resultado cuando finalice.

No es necesario que uses una sintaxis de un lenguaje concreto sino tus propias palabras.

## Ejemplos de Algoritmos

### RECETA DE COCINA: Cómo preparar una gelatina

1. Inicio
2. Hervir en una olla un litro de agua
3. Agregar el sobre de gelatina
4. Dejarlo a fuego lento cinco minutos revolviendo constantemente
5. Dejar reposar para que tome temperatura ambiente
6. Refrigerar dos hora hasta que cuaje
7. Fin

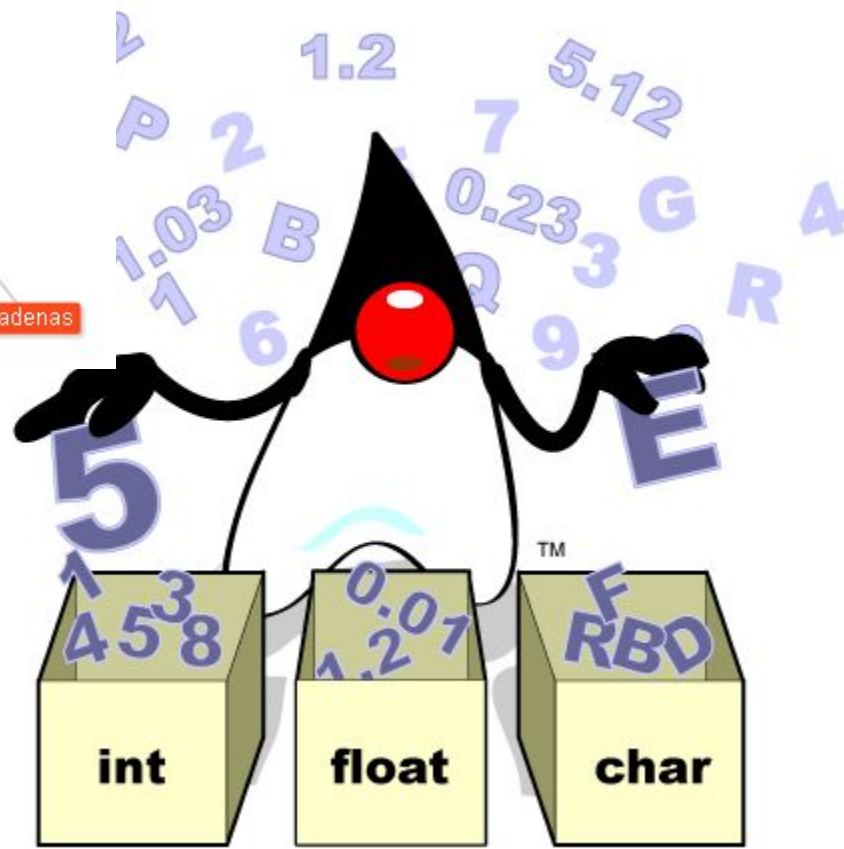


## Tipos básicos de datos



## Operadores Lógicos

P	Q	$\sim P$	$\sim Q$	$P \vee Q$	$P \wedge Q$
Verdadero	Verdadero	Falso	Falso	Verdadero	Verdadero
Verdadero	Falso	Falso	Verdadero	Verdadero	Falso
Falso	Verdadero	Verdadero	Falso	Verdadero	Falso
Falso	Falso	Verdadero	Verdadero	Falso	Falso





# Tipos de operadores

Tipo de operador	Símbolos python	Descripción
Aritméticos	<code>+, -, *, /, %, **, //</code>	Operaciones matemáticas: suma, resta, multiplicación, división, módulo, potencia, división entera
Relacionales / Comparación	<code>==, !=, &gt;, &lt;, &gt;=, &lt;=</code>	Comparan valores y devuelven <code>True</code> o <code>False</code>
Lógicos	<code>and, or, not</code>	Combina expresiones booleanas
De asignación	<code>=, +=, -=, *=, /=, %=</code>	Asignan valores a variables, con operaciones combinadas
Concatenación	<code>+</code>	Pega dos cadenas de caracteres
De identidad	<code>is, is not</code>	Comprueba si dos objetos son <b>el mismo objeto</b> en memoria
De pertenencia	<code>in, not in</code>	Comprueba si un elemento <b>pertenece a una colección</b>
Bit a bit / a nivel de bits	<code>&amp;, ^, ~, &lt;&lt;, &gt;&gt;</code>	



# Precedencia de los Operadores

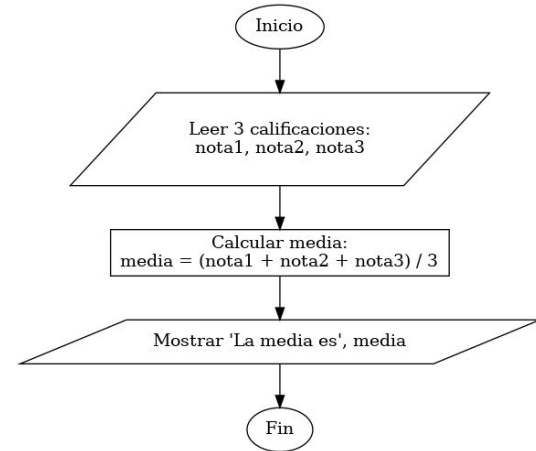


Descripción	Operadores
Postfijos	<code>i++, i--</code>
Unarios	<code>++i, --i</code>
Multiplicación y división	<code>*, /, %</code>
Suma y resta	<code>+, -</code>
Relacionales	<code>&gt;, &lt;, &gt;=, &lt;=</code>
Equivalencia	<code>==, !=</code>
AND lógico	<code>&amp;&amp;</code>
OR lógico	<code>  </code>
Asignación	<code>=</code>

## Ejercicio 5

Diagrama y algoritmo que calcule la nota media de tres actividades.

Usa los paréntesis para modificar la precedencia y así realizar correctamente el cálculo



1. Inicio

2. Leer valores de: Calificación1, Calificación2, Calificación3

3. Promedio = (Calificación1 + Calificación2 + Calificación3)/3