



Curso de  
**Algoritmos y  
Pensamiento Lógico**

Ana Belisa Martínez

---

¿Por qué es  
importante entrenar  
tu pensamiento lógico?

---

¿Alguna vez has  
pensado cómo  
optimizar las tareas  
de tu vida diaria?

Tomar la ruta  
más corta



Usar la mitad del  
tiempo en tu rutina



---

# Imagina una fuente con una capacidad de agua limitada

Debes aprovecharla al máximo

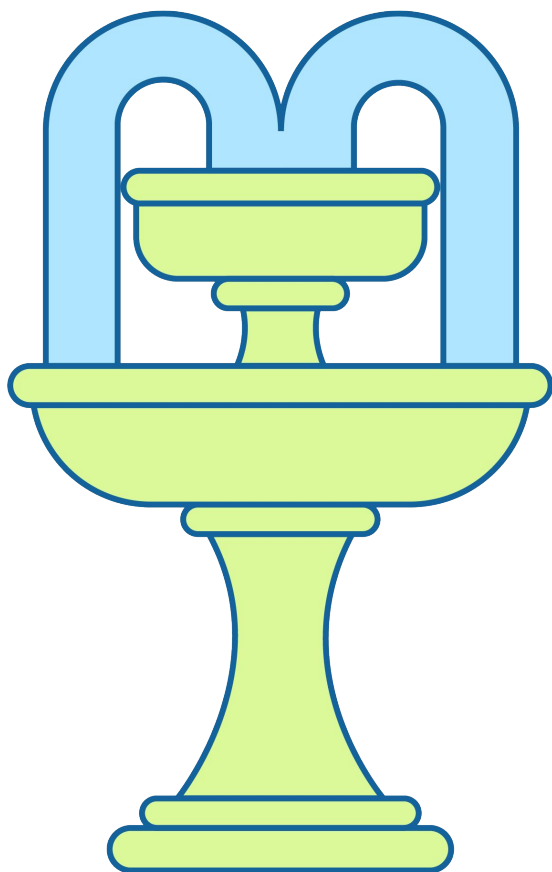
¿La dejas encendida  
todo el tiempo?



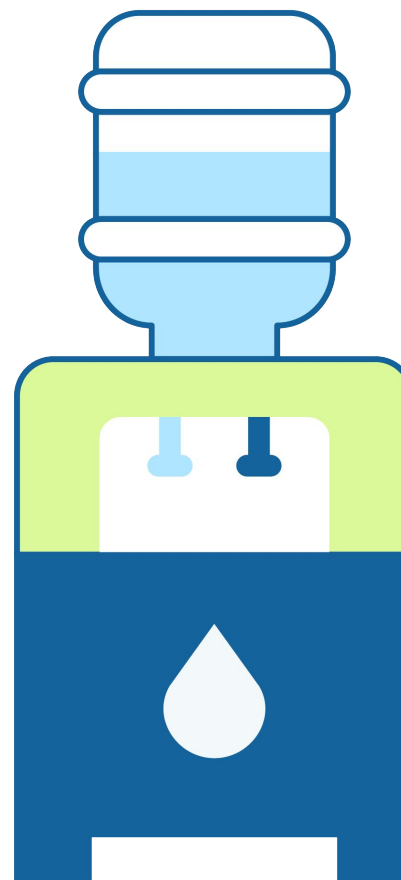
¿O generas una solución  
para habilitarla solo  
cuando la necesitas?







Problema



Solución

Podrías dejarla caer  
hasta que acabe



O podrías agregar  
una condición para  
que salga solo  
cuando la vas a usar



**Cada botón del dispensador es  
una condición**





# Parece simple

Porque es simple



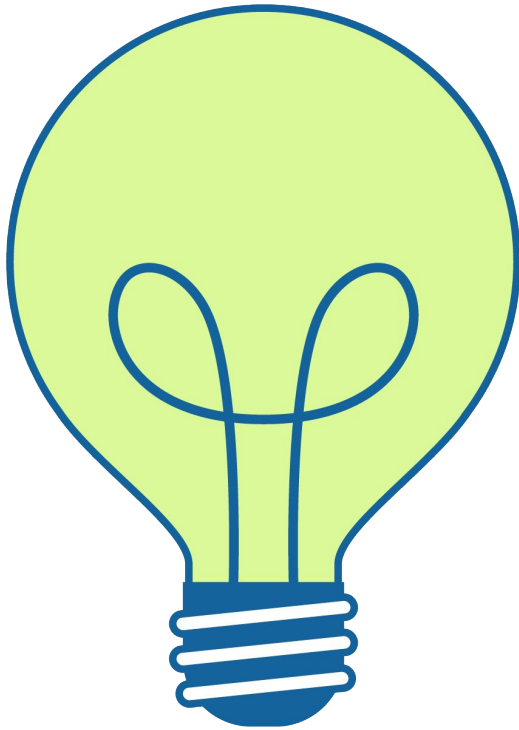
# Para eso estás aquí

Para entender que puedes encontrar soluciones óptimas a problemas simples y complejos.

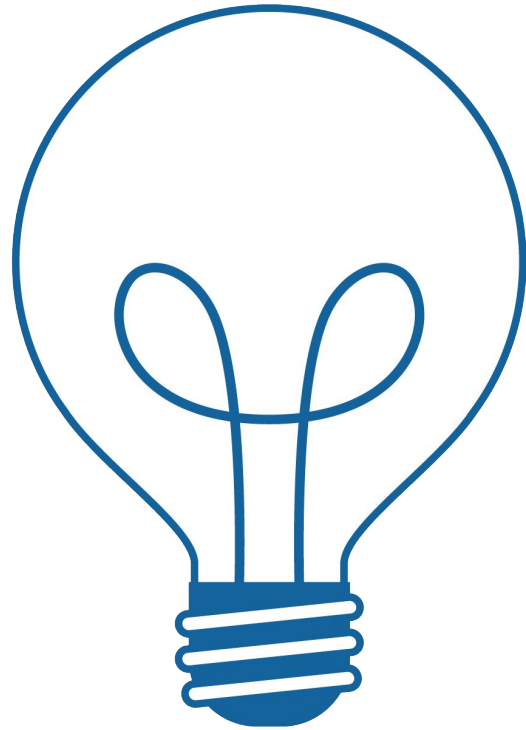


# El Sistema Binario

100101011001

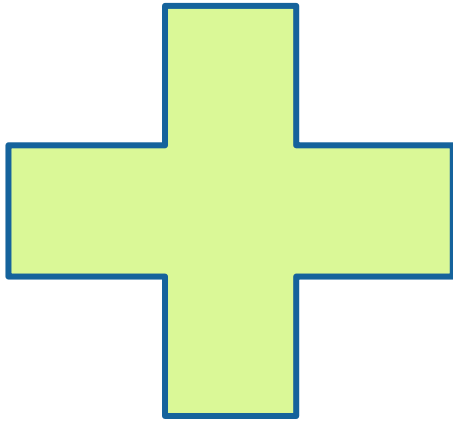


1



0





**True**

1



**False**

0



# Convertir 34 a binario

$$34 / 2 = 17 \text{ (queda 0)}$$



# Convertir 34 a binario

$$34 / 2 = 17 \text{ (queda 0)}$$

$$17 / 2 = 8 \text{ (queda 1)}$$

# Convertir 34 a binario

$$34 / 2 = 17 \text{ (queda 0)}$$

$$17 / 2 = 8 \text{ (queda 1)}$$

$$8 / 2 = 4 \text{ (queda 0)}$$

# Convertir 34 a binario

$$34 / 2 = 17 \text{ (queda 0)}$$

$$17 / 2 = 8 \text{ (queda 1)}$$

$$8 / 2 = 4 \text{ (queda 0)}$$

$$4 / 2 = 2 \text{ (queda 0)}$$

# Convertir 34 a binario

$$34 / 2 = 17 \text{ (queda 0)}$$

$$17 / 2 = 8 \text{ (queda 1)}$$

$$8 / 2 = 4 \text{ (queda 0)}$$

$$4 / 2 = 2 \text{ (queda 0)}$$

$$2 / 2 = 1 \text{ (queda 0)}$$

# Convertir 34 a binario

$$34 / 2 = 17 \text{ (queda 0)}$$

$$17 / 2 = 8 \text{ (queda 1)}$$

$$8 / 2 = 4 \text{ (queda 0)}$$

$$4 / 2 = 2 \text{ (queda 0)}$$

$$2 / 2 = 1 \text{ (queda 0)}$$

$$1 / 2 = 0 \text{ (queda 1)}$$

# Convertir 34 a binario

$$34 / 2 = 17 \text{ (queda 0)}$$

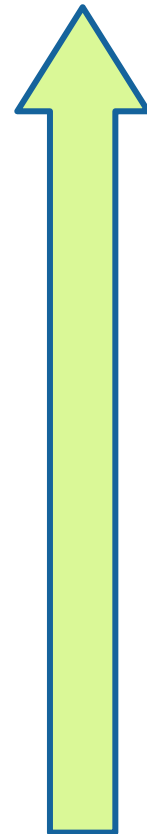
$$17 / 2 = 8 \text{ (queda 1)}$$

$$8 / 2 = 4 \text{ (queda 0)}$$

$$4 / 2 = 2 \text{ (queda 0)}$$

$$2 / 2 = 1 \text{ (queda 0)}$$

$$1 / 2 = 0 \text{ (queda 1)}$$





$$34 = 100010$$



---

# Operaciones Aritméticas



Suma



# Suma

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 10$$



# Suma

1

1 0 0 1 1 0 0 0

+ 0 0 0 1 0 1 0 1

---

1 0 1 0 1 1 0 1



Resta

# Resta

$$0 - 0 = 0$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$$10 - 1 = 1$$



# Resta

$$\begin{array}{r} 10011000 \\ - 00010101 \\ \hline 10000011 \end{array}$$



---

# Multiplicación y División

---

Reto:

Traduce el año actual -  
tu edad para  
encontrar el año en  
que naciste



# Algoritmos

1

2

3

4

.

.





Preciso  
Definido  
Finito

---

# Gráfico y No Gráfico



# Metodología para crear un algoritmo

# Definir el problema





# Analizar el problema



# Diseñar el algoritmo



# Prueba de escritorio



---

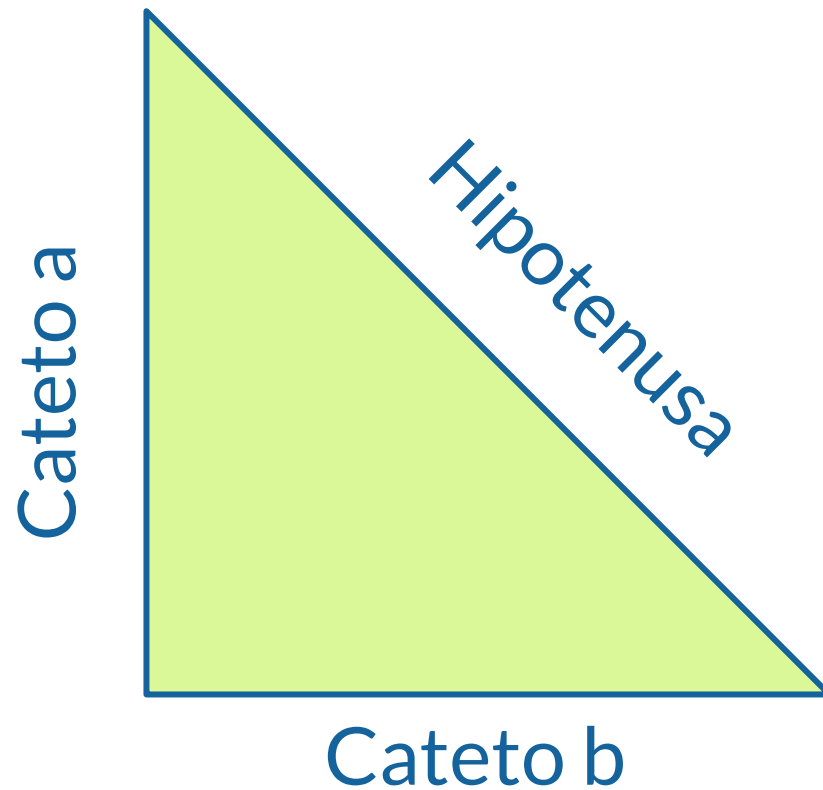
Calcular la  
hipotenusa de un  
triángulo rectángulo

---

# Calcular la hipotenusa de un triángulo rectángulo

Este es el problema

# Analizar el problema





# Teorema de Pitágoras

$$h^2 = a^2 + b^2$$

$$h = \sqrt{h^2}$$

# Pasos para resolverlo

- Necesito el valor de los catetos.
- Elevar al cuadrado el valor de los catetos.
- Sumo los dos valores.
- Saco raíz cuadrada del resultado.
- Imprimo el valor de la hipotenusa.



# Prueba de escritorio

$$h^2 = 3^2 + 4^2$$

$$h^2 = 9 + 16$$

$$h^2 = 25$$

$$h = \sqrt{25}$$

$$h = 5$$

---

**Reto:**  
**Diseña un algoritmo  
para cocinar  
quesadillas**



# Diagramas de flujo



**Inicio / Fin**



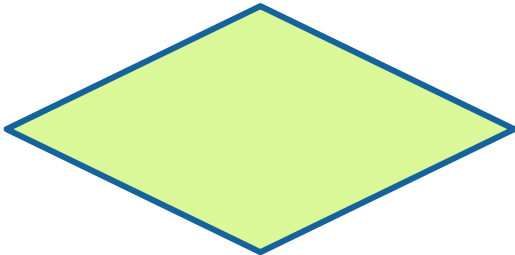
**Línea de flujo**



**Entrada / Salida**



**Proceso**



**Decisión**



# Ejemplo práctico

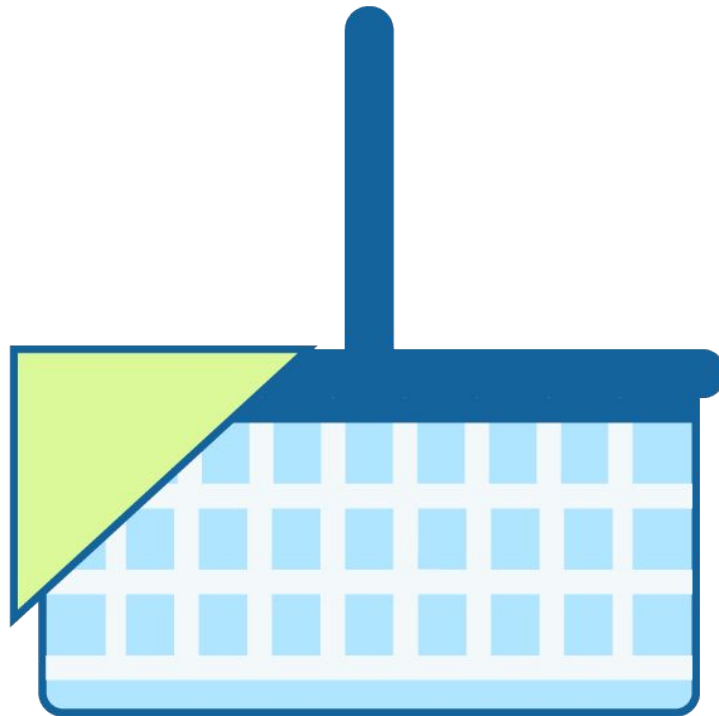
---

**Reto: Representa  
el funcionamiento  
de un reloj**



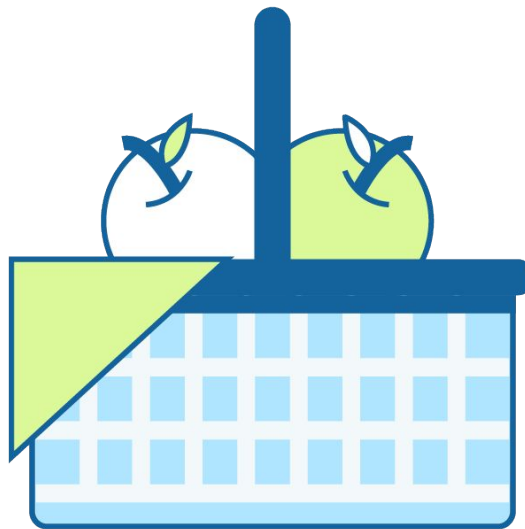
# Variables y Constantes

# Variables y constantes

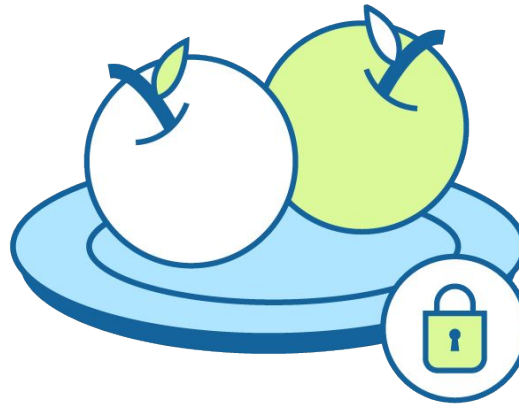




# Variables



# Constantes





# Tipos de datos



# Tipos de datos

Número	Texto	Lógica
Enteros Decimales	Cadenas Caracteres	True False

---

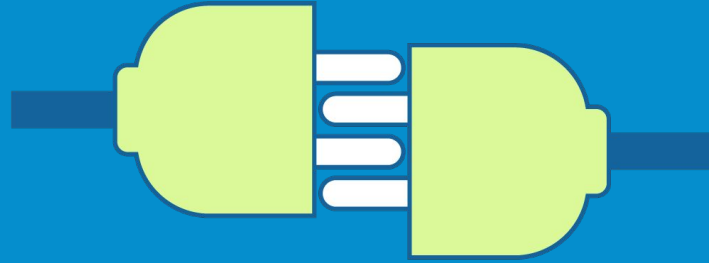
# Strings y concatenación

“Yo soy una cadena  
de texto”



“Hola”  
“mi nombre es”  
“Ana”





“Hola mi nombre  
es Ana”







# Char y String

‘H’ - Carácter  
“Hola” - Cadena





# Números

Y operaciones matemáticas básicas

int, float, short, long



---

# Suma

valor + valor  
5 + 7

---

# Resta

valor - valor

9 - 6

---

# Multiplicación

valor \* valor

3 \* 4

---

# División

valor / valor  
10 / 2



**El tipo de dato puede  
afectar el resultado**



---

# Operaciones matemáticas compuestas

$$5 - 3 * 2 + 4 - 4 / 2$$





$$5 - \underline{3 * 2} + 4 - \underline{4 / 2}$$

$$5 - 6 + 4 - 2 = 1$$



$$(5 - 3 * 2) + (4 - 4 / 2) + 8$$




$$(5 - \underline{3 * 2}) + (4 - \underline{4 / 2}) + 8$$

$$(\underline{5 - 6}) + (\underline{4 - 2}) + 8$$

$$-1 + 2 + 8 = 9$$





# True y false

Booleanos y tablas de verdad

---

# Tablas de verdad

¿Qué son?



# Composición

Valor de proposición 1	Valor de proposición 2	Conectiva Lógica
V	F	Resultado
F	V	Resultado

# Negación

Una proposición que es verdadera es falsa.

P	$\sim P$
V	F
F	V

# Conjunción

Es verdadera solo si ambas son verdaderas, y es falsa cuando al menos uno de los valores es falso.

<b>p</b>		<b>q</b>	<b><math>p \wedge q</math></b>
<b>v</b>	and	<b>v</b>	<b>v</b>
<b>v</b>	and	<b>f</b>	<b>f</b>
<b>f</b>	and	<b>v</b>	<b>f</b>
<b>f</b>	and	<b>f</b>	<b>f</b>

# Disyunción inclusiva

Es verdadera cuando por lo menos una es verdadera; de lo contrario, será falsa.

<b>p</b>		<b>q</b>	<b><math>p \vee q</math></b>
<b>v</b>	or	<b>v</b>	<b>v</b>
<b>v</b>	or	<b>f</b>	<b>v</b>
<b>f</b>	or	<b>v</b>	<b>v</b>
<b>f</b>	or	<b>f</b>	<b>f</b>

# Disyunción exclusiva

Es falsa si ambos componentes son verdaderos o falsos.

p		q	$p \underline{\vee} q$
v	xor	v	f
v	xor	f	v
f	xor	v	v
f	xor	f	f

# Condicional

Devuelve falso cuando el primer valor es verdadero y el segundo falso.

<b>p</b>		<b>q</b>	<b>p-&gt;q</b>
<b>v</b>	then	<b>v</b>	<b>v</b>
<b>v</b>	then	<b>f</b>	<b>f</b>
<b>f</b>	then	<b>v</b>	<b>v</b>
<b>f</b>	then	<b>f</b>	<b>v</b>

# Bicondicional

Es verdadera cuando ambas proposiciones son verdaderas o ambas falsas.

<b>p</b>		<b>q</b>	<b><math>p \leftrightarrow q</math></b>
<b>v</b>	if	<b>v</b>	<b>v</b>
<b>v</b>	if	<b>f</b>	<b>f</b>
<b>f</b>	if	<b>v</b>	<b>f</b>
<b>f</b>	if	<b>f</b>	<b>v</b>



# Arrays

Vectores y arreglos



Conjunto de elementos  
del mismo tipo  
ordenados en fila



0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---



0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---



0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---



9

A	M	B	N	C	O	D	P	E
---	---	---	---	---	---	---	---	---



A	M	B	N	C	O	D	P	E
---	---	---	---	---	---	---	---	---

0

Pueden tener diferentes tamaños

a	b	c	d	e	f
---	---	---	---	---	---

79	8	56
----	---	----

A	M	B	N	C	O	D	P	E	X
---	---	---	---	---	---	---	---	---	---

Se pueden ordenar

a	b	c	d	e	f
---	---	---	---	---	---

8	56	79
---	----	----

A	B	C	D	E	M	N	O	P	X
---	---	---	---	---	---	---	---	---	---



Se pueden recorrer



Se puede acceder a una posición específica



3



# Estructuras de control

---

Nos ayudan a  
construir el flujo de  
nuestras tareas

# If / Else



# If / Else Switch



If / Else  
Switch  
While



If / Else  
Switch  
While  
Do While

---



If / Else  
Switch  
While  
Do While  
For

# If / Else



Si  $x < 20$  entonces  
retorna “ok”



Si  $x < 20$  entonces  
retorna “ok”

Si no



Si  $x < 20$  entonces  
retorna “ok”

Si no

retorna “no”



---

Puedes usar  
operadores de  
comparación



# Switch

```
switch(numero) {  
  case 1:  
    "El número es 1"  
  break;  
  case 2:  
    "El número es 2"  
  break;  
  case 3:  
    "El número es 3"  
  break;  
  default:  
    "El número es mayor a 3"  
}
```



```
switch(numero) {  
case 1:  
    "El número es 1"  
break;  
case 2:  
    "El número es 2"  
break;  
case 3:  
    "El número es 3"  
break;  
default:  
    "El número es mayor a 3"  
}
```

```
switch(numero) {  
case 1:  
    "El número es 1"  
break;  
case 2:  
    "El número es 2"  
break;  
case 3:  
    "El número es 3"  
break;  
default:  
    "El número es mayor a 3"  
}
```

```
switch(numero) {  
  case 1:  
    "El número es 1"  
    break;  
  case 2:  
    "El número es 2"  
    break;  
  case 3:  
    "El número es 3"  
    break;  
  default:  
    "El número es mayor a 3"  
}
```

```
switch(numero) {  
  case 1:  
    "El número es 1"  
    break;  
  case 2:  
    "El número es 2"  
    break;  
  case 3:  
    "El número es 3"  
    break;  
  default:  
    "El número es mayor a 3"  
}
```

```
switch(numero) {  
  case 1:  
    "El número es 1"  
  break;  
  case 2:  
    "El número es 2"  
  break;  
  case 3:  
    "El número es 3"  
  break;  
  default:  
    "El número es mayor a 3"  
}
```



# Excepciones

Es un evento anormal  
que ocurre durante la  
ejecución

No es un “else”. No funciona como un “else”



---

# throw

La puedes invocar en el punto que desees





# Try ... catch

Intenta hacer **X** pero si falla haz **Y**

---

# finally

Lo que sucede después del try y catch

```
try {  
    Decir hola  
} catch(e) {  
    Ups, no pude decir hola :(  
} finally {  
    ¡Todo va a estar bien!  
}
```

```
try {  
    Decir hola  
} catch(e) {  
    Ups, no pude decir hola :(  
} finally {  
    ¡Todo va a estar bien!  
}
```

```
try {  
    Decir hola  
} catch(e) {  
    Ups, no pude decir hola :(  
} finally {  
    ¡Todo va a estar bien!  
}
```

```
try {  
    Decir hola  
} catch(e) {  
    Ups, no pude decir hola :(  
} finally {  
    ¡Todo va a estar bien!  
}
```

```
try {  
    Decir hola  
} catch(e) {  
    Ups, no pude decir hola :(  
} finally {  
    ¡Todo va a estar bien!  
}
```

```
try {  
    Decir hola  
} catch(e) {  
    Ups, no pude decir hola :(  
} finally {  
    ¡Todo va a estar bien!  
}
```



---

# ¿Qué es un ciclo?

for, while, do while



# Ciclo

Es una estructura de control que ejecuta un bloque de instrucciones de manera repetida.

---

# For

```
for (x=0; x<=30; x++)
```

for (x=0; x<=30; x++)

\_\_\_\_\_

```
for (x=0; x<=30; x++)
```

---

```
for (x=0; x<=30; x++)
```



```
for (x=0; x<=30; x++)
```

Estoy en la posición x



---

# While

```
while (x <= 10)
```



`while (x <= 10)`

—

**while** (x <= 10)

Soy menor que 10 x++



---

# Do ... while

```
do {  
    i = i + 1;  
    saludo  
} while (i < 5);
```

```
do {  
  i++
```

```
Aquí hago algo  
} while (i < 5)
```

---

```
do {  
    i ++
```

```
    Aquí hago algo  
} while (i < 5)
```

---



# Funciones

**Bloques de código  
que realizan una  
actividad específica**



---

¿Para qué sirven  
las funciones?





# Sirven para:

- Modularizar
- Optimizar
- Organizar
- Encapsular

Función simple y específica

```
function decirHola()  
  return "hola"
```



Función con parámetro


```
function muestraNombre(nombre)  
    return nombre
```



---

# ¿Parámetro?

Sí, parámetro



```
function suma(valor1, valor2)  
return valor1 + valor 2
```

valor1 y valor2 son los parámetros  
de la función.


---

Los parámetros son  
diferentes a los  
argumentos



# ¿Argumentos?

Sí, argumentos



# suma(3, 1)

3 y 1 son los argumentos, o sea, el valor que se le asignó a los parámetros al usar esa función.



---

# Volviendo a las funciones

Pueden ser tan simples o tan  
complejas como queramos

---

# Pero siempre deben ser puntuales

Y ocuparse de una sola tarea



# Modularización del código

Más que una buena práctica

Necesitas tener tu  
código modularizado



Deja que cada  
bloque haga una tarea  
particular



Esto permitirá  
que sea escalable



Seguramente  
estará optimizado



# Reutiliza y dinamiza





# A nivel de funciones y archivos



---

# ¿Qué es recursividad?

Funciones que se llaman a sí mismas

---

# Hay que tener precaución

Porque pueden ser infinitas

---

**Lo mejor es  
condicionarlas y  
usarlas sabiamente**

```
cuentaRegresiva(numero):  
    numero -= 1  
  
    if numero > 0:  
        muestra numero  
        cuentaRegresiva(numero)  
    else:  
        muestra "Feliz año nuevo"
```

---

# Diferencias entre lenguajes de programación

---

**No te quedes  
con un lenguaje  
exclusivamente**

Abre tu mente a la lógica de lo  
que debes hacer

---

# Diferencias sintácticas





# Finalización de líneas



# Tipado



# Indentación



# Corchetes

---

La lógica es lo  
más importante



# Curso de Prework