

Lógica de programación I

Juan Pablo Restrepo Uribe

Ing. Biomedico

MSc. Automatización y Control Industrial

jprestrepo@correo.iue.edu.co

2023-2

Institución Universitaria de Envigado

Retomando

- ¿Cuáles son los pasos o recomendaciones para la realización de un algoritmo?
- ¿Cuáles son las características principales de un algoritmo?
- ¿Los algoritmos deben ser computacionales?
- ¿Podemos hacer algoritmos de las siguientes actividades?
 - Lavar las manos.
 - Barrer.
 - Tender la cama.
 - Bañarse.
 - Tirar la basura.

Haga los respectivos algoritmos

Retomando

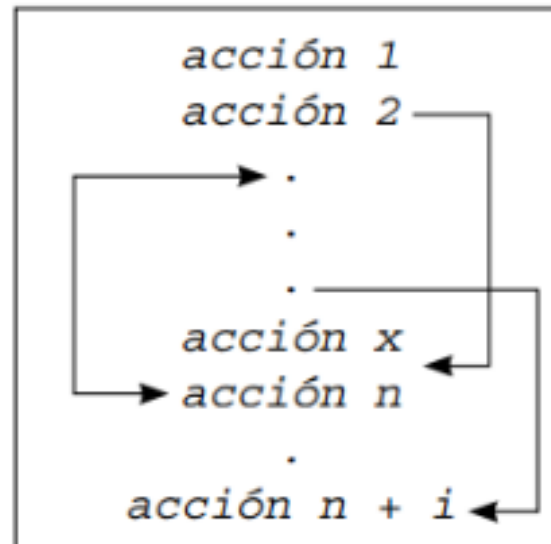
- ¿Qué tipo de algoritmos existen?
- ¿Cómo se resuelve el problema de la mochila?

Unidad 1: Definiciones Básicas

- Tipos de instrucción
 - Entrada
 - Salida
 - Proceso
 - Selectivas
 - Repetitivas
- Tipos de datos
- Variables y constantes
- Operadores aritméticos

Tipos de instrucción

Una instrucción de máquina es una operación elemental que un programa puede solicitar a un **procesador** para que la ejecute. Una instrucción de máquina es pues una orden básica que el ordenador directamente puede interpretar y ejecutar, sin requerir ningún paso previo o **traducción previa**.



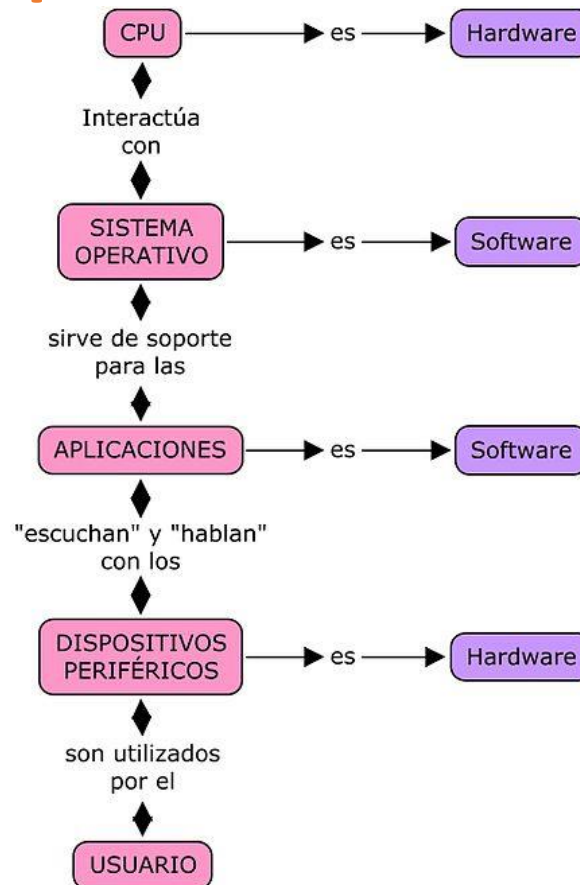
Tipos de instrucción

Una instrucción de máquina o instrucción-máquina es una operación elemental que un programa puede solicitar a un procesador para que la ejecute.¹ Una instrucción de máquina es pues una orden básica que el ordenador directamente puede interpretar y ejecutar, sin requerir ningún paso previo o traducción previa.

-u 100 1a

0CFD:0100	BA0B01
0CFD:0103	B409
0CFD:0105	CD21
0CFD:0107	B400
0CFD:0109	CD21

Tipos de instrucción



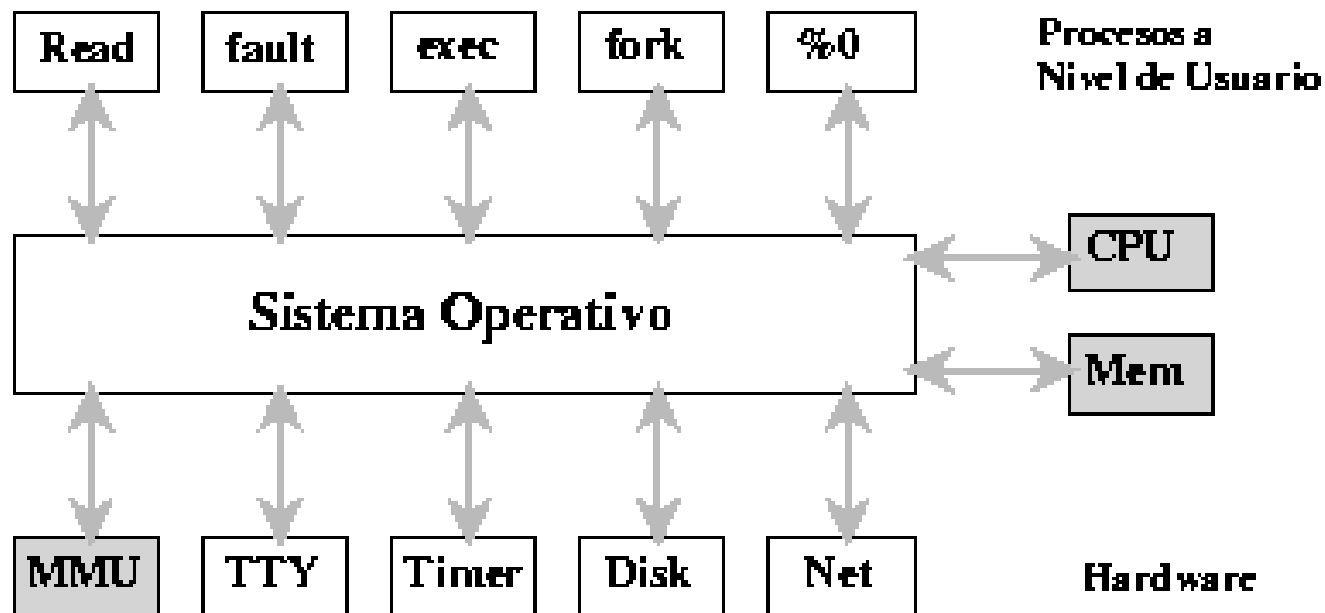
Tipos de instrucción

- **Instrucciones de transferencia de datos:** en este tipo de instrucciones, se transfieren datos desde una localización a otra. Los pasos que se siguen para realizarlo son:
 - Determinación de las direcciones de origen y destino de memoria.
 - Realización de la transformación de memoria virtual a memoria real.
 - Comprobación de la caché.
 - Inicio del proceso de lectura/escritura en la memoria.
- **Instrucciones aritméticas:** pueden implicar transferencia de datos antes y/o después. Realizan operaciones aritméticas de las que se encarga la ALU. Se pueden clasificar en de 1 operando (valor absoluto, negación) y 2 operandos (suma, resta).

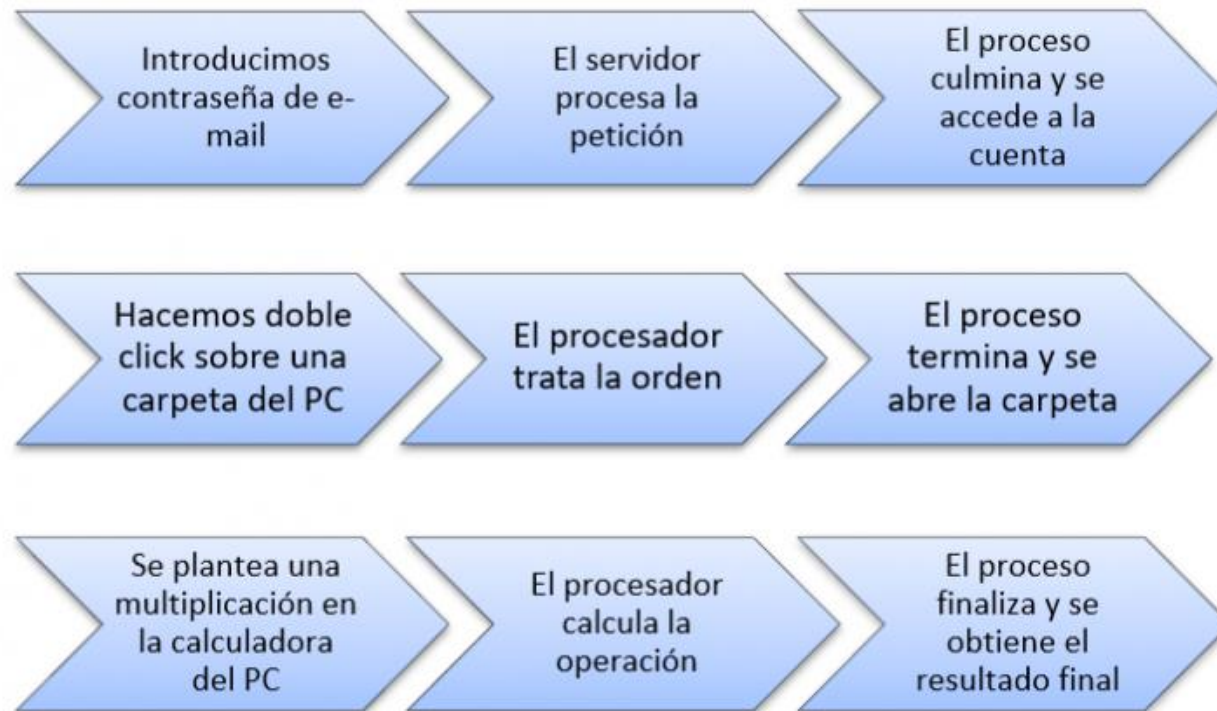
Tipos de instrucción

- **Instrucciones lógicas:** al igual que las aritméticas, la ALU se encarga de realizar estas operaciones, que en este caso son de tipo lógico.
- **Instrucciones de conversión:** similares a las aritméticas y lógicas. Pueden implicar lógica especial para realizar la conversión.
- **Instrucciones de transferencia de control:** actualizan el contador de programa (PC). Administran las llamadas/retornos a las subrutinas, el paso de parámetros y el enlazado.
- **Instrucciones de E/S (entrada/salida):** administran los comandos de entrada/salida. Si hay un mapa de memoria de entrada/salida, determina la dirección de este mapa.

Tipos de instrucción



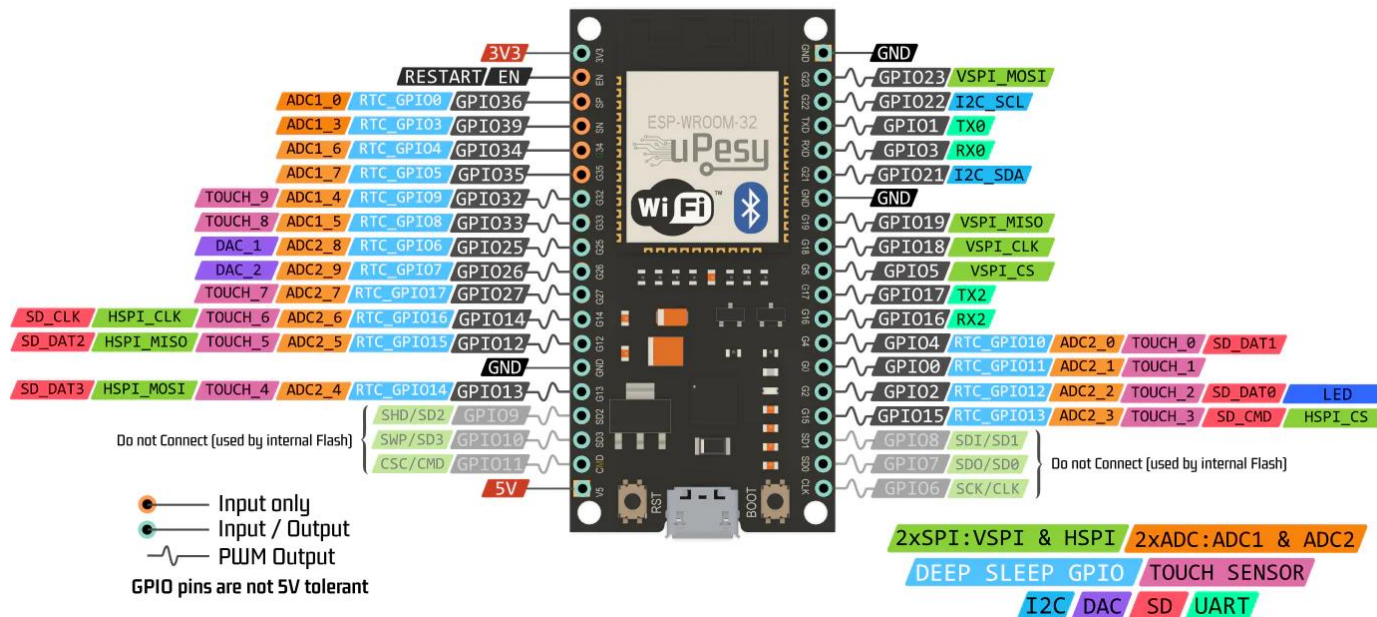
Tipos de instrucción



Instrucciones de entrada y salida (E/S)

Este grupo de instrucciones forma un caso especial dentro de las instrucciones privilegiadas, principalmente porque acceden a dispositivos que están compartidos.

ESP32 Wroom DevKit Full Pinout



Instrucciones de entrada y salida (E/S)

Caracteres ASCII de control			Caracteres ASCII imprimibles				ASCII extendido (Página de código 437)									
00	NULL	(carácter nulo)	32	espacio	64	@	96	`	128	Ç	160	à	192	Ł	224	Ó
01	SOH	(inicio encabezado)	33	!	65	A	97	a	129	ü	161	í	193	ł	225	ô
02	STX	(inicio texto)	34	"	66	B	98	b	130	é	162	ó	194	Ł	226	ö
03	ETX	(fin de texto)	35	#	67	C	99	c	131	â	163	ú	195	ł	227	õ
04	EOT	(fin transmisión)	36	\$	68	D	100	d	132	ä	164	ñ	196	—	228	ö
05	ENQ	(consulta)	37	%	69	E	101	e	133	å	165	Ñ	197	†	229	õ
06	ACK	(reconocimiento)	38	&	70	F	102	f	134	ä	166	ª	198	ä	230	µ
07	BEL	(timbre)	39	'	71	G	103	g	135	ç	167	º	199	Ä	231	þ
08	BS	(retroceso)	40	(72	H	104	h	136	ê	168	¿	200	ℒ	232	þ
09	HT	(tab horizontal)	41)	73	I	105	i	137	ë	169	®	201	ℒ	233	Û
10	LF	(nueva línea)	42	*	74	J	106	j	138	è	170	™	202	ℒ	234	Ü
11	VT	(tab vertical)	43	+	75	K	107	k	139	ï	171	½	203	ℒ	235	Ù
12	FF	(nueva página)	44	,	76	L	108	l	140	î	172	¼	204	ℒ	236	Ý
13	CR	(retorno de carro)	45	-	77	M	109	m	141	í	173	⅓	205	ℒ	237	Ÿ
14	SO	(desplaza afuera)	46	.	78	N	110	n	142	Ä	174	«	206	ℒ	238	—
15	SI	(desplaza adentro)	47	/	79	O	111	o	143	Å	175	»	207	ℒ	239	.
16	DLE	(esc.vínculo datos)	48	0	80	P	112	p	144	É	176	•	208	ø	240	≡
17	DC1	(control disp. 1)	49	1	81	Q	113	q	145	æ	177	•	209	Ð	241	±
18	DC2	(control disp. 2)	50	2	82	R	114	r	146	Æ	178	•	210	É	242	•
19	DC3	(control disp. 3)	51	3	83	S	115	s	147	ö	179	•	211	Ê	243	¾
20	DC4	(control disp. 4)	52	4	84	T	116	t	148	ö	180	•	212	Ë	244	¶
21	NAK	(conf. negativa)	53	5	85	U	117	u	149	ó	181	À	213	ì	245	§
22	SYN	(inactividad sinc)	54	6	86	V	118	v	150	û	182	Á	214	í	246	÷
23	ETB	(fin bloque trans)	55	7	87	W	119	w	151	ù	183	Â	215	î	247	•
24	CAN	(cancelar)	56	8	88	X	120	x	152	ÿ	184	Ã	216	ï	248	•
25	EM	(fin del medio)	57	9	89	Y	121	y	153	Ö	185	•	217	Ĵ	249	•
26	SUB	(sustitución)	58	:	90	Z	122	z	154	Ü	186	•	218	Ĵ	250	•
27	ESC	(escape)	59	;	91	[123	{	155	ø	187	•	219	Ĵ	251	•
28	FS	(sep. archivos)	60	<	92	\	124		156	£	188	•	220	Ĵ	252	•
29	GS	(sep. grupos)	61	=	93]	125	}	157	Ø	189	•	221	Ĵ	253	•
30	RS	(sep. registros)	62	>	94	^	126	~	158	x	190	¥	222	Ĵ	254	■
31	US	(sep. unidades)	63	?	95	_			159	f	191	•	223	Ĵ	255	nbsp
127	DEL	(suprimir)														

<https://elcodigoascii.com.ar/>

Creación de un proceso

- Arranque del sistema
- Una petición deliberada del usuario para crear un proceso.
- El inicio de un trabajo por lotes.

La forma de creación de procesos en Unix es a través de una llamada al sistema fork la cual creará un proceso hijo en total semejanza al padre, hasta que el recién proceso decida cambiar su imagen en memoria, incluso obtener sus propios descriptores de archivos abiertos.

Creación de un proceso

System Monitor

File View Settings Help

Process Table System Load

End Process... Quick search

Own Processes Tools

Name	Username	PID	TTY	Niceness	CPU %	CPU Time	IO Read	IO Write	Virtual Size	Memory	Shared Mem
firefox	vulphere	2130		0	1%	18:24		54 K/s	3,116,732 K	274,996 K	215,291 K
WebExtensions	vulphere	2335		0		3:56			1,899,080 K	225,444 K	131,991 K
Discord	vulphere	4149		0		25:30			2,467,916 K	224,880 K	287,931 K
Web	vulphere	2928		0		7:06			1,859,968 K	154,480 K	125,661 K
plasmashell	vulphere	1725		0		2:13			2,666,276 K	141,892 K	129,481 K
Web	vulphere	2231		0		20:50			2,131,628 K	128,760 K	136,891 K
Web	vulphere	15422		0		0:31			1,871,484 K	123,828 K	131,841 K
mysqld	vulphere	1873		0		0:06			899,160 K	57,424 K	19,181 K
kwin_x11	vulphere	15379		0	1%	0:18			3,141,696 K	55,884 K	85,471 K
Discord	vulphere	4079		0		2:34			1,674,940 K	43,484 K	82,541 K
spectacle	vulphere	15661		0		0:00			475,960 K	37,232 K	55,251 K
Web	vulphere	15486		0		0:00			1,413,992 K	25,236 K	59,901 K
ksysguard	vulphere	15652		0	3%	0:22			553,088 K	20,524 K	70,021 K
akonadi_mailfilter_agent	vulphere	1983		0		0:02			911,880 K	17,156 K	74,201 K
akonadi_sendlater_agent	vulphere	1991		0		0:03			886,352 K	17,032 K	70,281 K
akonadi_archivemail_agent	vulphere	1968		0		0:02			903,796 K	16,944 K	71,601 K
krunner	vulphere	1723		0		0:14			558,092 K	16,600 K	57,871 K
Discord	vulphere	4106		0		0:41			323,072 K	15,324 K	44,351 K
konsole	vulphere	2544		0		0:07			477,308 K	14,900 K	56,731 K
kalarm	vulphere	1794		0		0:02			585,104 K	13,196 K	66,321 K
kmix	vulphere	1761		0		0:05			1,072,172 K	12,788 K	52,681 K
polkit-kde-authentication-agent-1	vulphere	1729		0		0:02			606,160 K	12,544 K	52,051 K
kded5	vulphere	1658		0		0:20			1,477,728 K	11,880 K	56,701 K
xdg-desktop-portal-kde	vulphere	14242		0		0:00			521,016 K	9,948 K	40,381 K
xdg-desktop-portal-gtk	vulphere	14248		0		0:00			472,472 K	7,788 K	13,941 K
korgac	vulphere	1775		0		0:01			544,344 K	6,712 K	50,971 K
akonadi_notes_agent	vulphere	1989		0		0:02			596,388 K	6,268 K	43,281 K
akonadi_indexing_agent	vulphere	1973		(Batch) 19		0:01			421,076 K	6,176 K	40,551 K
baloorunner	vulphere	15351		0		0:02			268,774,044 K	5,864 K	59,231 K
ksmserver	vulphere	1716		0		0:02			427,992 K	5,812 K	39,501 K
akonadi_newmailnotifier_agent	vulphere	1988		0		0:01			454,276 K	5,756 K	43,101 K
akonadi_maildispatcher_agent	vulphere	1982		0		0:02			503,724 K	5,644 K	40,361 K
kglobalaccel5	vulphere	1683		0		0:01			327,656 K	5,472 K	32,971 K

85 processes CPU: 6% Memory: 2.1 GiB / 7.6 GiB Swap: 780.0 KiB / 11.7 GiB

Terminación de un proceso

El ciclo de vida de un proceso es fácil, depende de la creación, la ejecución de instrucciones y la terminación. Cabe señalar que un proceso en el transcurso de su ciclo puede estar en diferentes estados.

Salida normal.

Salida por error.

Error fatal.

Eliminado por otro proceso.

Estados de un proceso

Los estados de un proceso obedecen a su participación y disponibilidad dentro del sistema operativo. Los **procesadores** sólo pueden ejecutar un solo proceso a la vez, turnándolos para el uso de este. Existen procesos no apropiativos o cooperativos que básicamente ocupan todo el tiempo del procesador hasta que ellos deciden dejarlo.

Los posibles estados que puede tener un proceso son ejecución, bloqueado y listo:

- **Ejecución:** es un proceso que está haciendo uso del procesador.
- **Bloqueado:** No puede ejecutarse hasta que un evento externo sea llevado a cabo.
- **Listo:** ha dejado disponible al procesador para que otro proceso pueda ocuparlo.

Tipos de procesos

- Procesos en primer plano:**
 Interactúan con el usuario, es decir, el usuario proporciona los datos que el proceso utilizará.
- Procesos en segundo plano:** Son creados para tareas bien definidas y no necesitan la intervención del usuario, por ejemplo, se puede tener un proceso en segundo plano para revisar la temperatura del disco duro constantemente.

archivo Opciones Vista

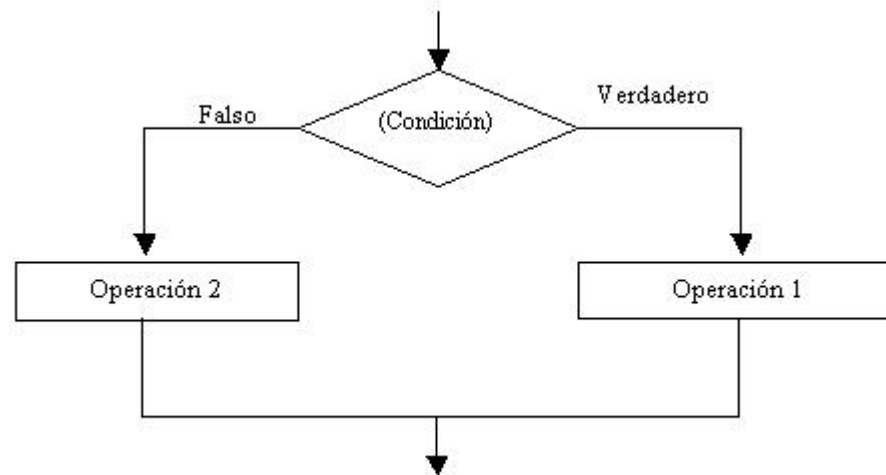
Procesos Rendimiento Historial de aplicaciones Inicio Usuarios Detalles Servicios

Nombre	Estado	100% CPU	26% Memoria	3% Disco	0% Red	1% GPU	Motc
Aplicaciones (1)							
> Administrador de tareas		2.8%	21.2 MB	0 MB/s	0 Mbps	0%	
Procesos en segundo plano (82)							
.NET Runtime Optimization Service		14.6%	6.6 MB	0 MB/s	0 Mbps	0%	
ACMON (32 bits)		0%	0.9 MB	0 MB/s	0 Mbps	0%	
Aislamiento de gráficos de dispositivo de audio de Windows		0%	1.3 MB	0 MB/s	0 Mbps	0%	
Antimalware Service Executable		4.8%	103.0 MB	0.1 MB/s	0 Mbps	0%	
Aplicación de subsistema de cola		0%	3.5 MB	0 MB/s	0 Mbps	0%	
Application Web Server Daemon (32 bits)		0%	4.2 MB	0 MB/s	0 Mbps	0%	
ASLDR Service (32 bits)		0%	0.7 MB	0 MB/s	0 Mbps	0%	
ASUS GiftBox Desktop (32 bits)		0%	0.7 MB	0 MB/s	0 Mbps	0%	
ASUS Patch For Touch Panel		0%	0.9 MB	0 MB/s	0 Mbps	0%	
ASUS Product Register Program (32 bits)		0%	1.2 MB	0 MB/s	0 Mbps	0%	
ASUS Smart Gesture Center		0%	1.9 MB	0 MB/s	0 Mbps	0%	
ASUS Smart Gesture Helper		0%	0.5 MB	0 MB/s	0 Mbps	0%	
ASUS Smart Gesture Loader		0%	1.2 MB	0 MB/s	0 Mbps	0%	
ASUS USB Charger Plus (32 bits)		0%	0.8 MB	0 MB/s	0 Mbps	0%	
ATK Media (32 bits)		0%	0.8 MB	0 MB/s	0 Mbps	0%	
ATKOSD2 (32 bits)		0%	1.1 MB	0 MB/s	0 Mbps	0%	
Cargador de CTF		0%	2.5 MB	0 MB/s	0 Mbps	0%	
Conexant Audio Message Service		0%	1.0 MB	0 MB/s	0 Mbps	0%	

Menos detalles Finalizar tarea

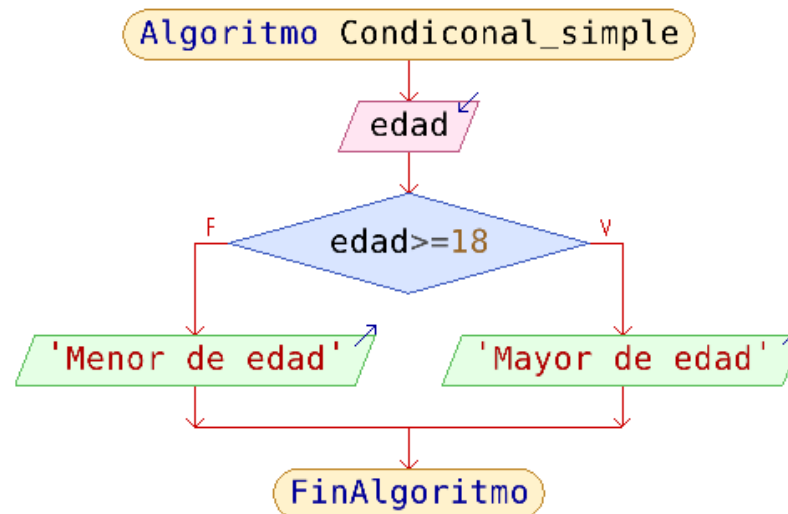
Estructuras condicionales

Se definen como una estructura que permite controlar de cierto modo el flujo de una acción o de acciones del programa, ayudando a que el desarrollo de la lógica de programación sea de manera óptima.



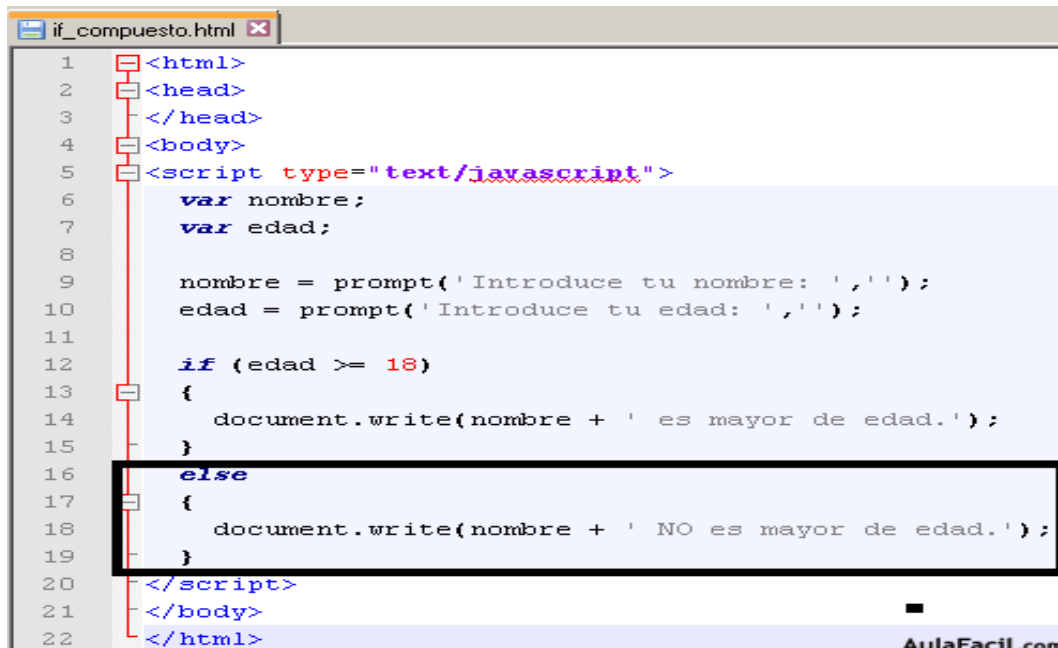
Estructuras condicionales

Todas estas acciones permiten confirmar la validez del flujo del programa en relación a una expresión lógica escogiendo entre dos o más acciones. De esta manera, evalúan las condiciones e inmediatamente ejecutan una sentencia. Sin embargo, se debe tener cuidado al realizarlo, pues no es posible hacerlo todas al mismo tiempo, sino una a la vez.



Tipos de estructuras condicionales

La representación de cada una de las partes de esta estructura se hace con palabras en pseudocódigo, como lo son por ejemplo **if**, **else**, **then** o, traducidas al español **si**, **entonces**, **si... no**.



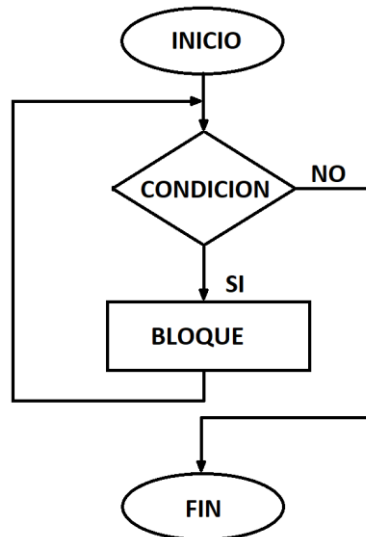
```
1 <html>
2 <head>
3 </head>
4 <body>
5 <script type="text/javascript">
6     var nombre;
7     var edad;
8
9     nombre = prompt('Introduce tu nombre: ', '');
10    edad = prompt('Introduce tu edad: ', '');
11
12    if (edad >= 18)
13    {
14        document.write(nombre + ' es mayor de edad. ');
15    }
16    else
17    {
18        document.write(nombre + ' NO es mayor de edad. ');
19    }
20 </script>
21 </body>
22 </html>
```

Tipos de estructuras condicionales

- **Simples:** Se definen como aquellas compuestas bajo una sola selección y condición. El ejemplo más claro de estructura selectiva simple en programación es falso/verdadero, en donde si el resultado es falso no hará nada, sin embargo, ejecutará la acción o acciones que se requieran en caso tal sea verdadero.
- **Dobles:** Al igual que las simples, están condicionadas únicamente por una condición, pero usando el mismo ejemplo de falso y verdadero. En caso de falso, ejecutará las acciones necesarias, y en caso de que sea verdadero, también las va a ejecutar.
- **Múltiples:** Estas se dan cuando tenemos la unión de varias estructuras selectivas simples, con la diferencia de que su código es mucho más pequeño que las simples.

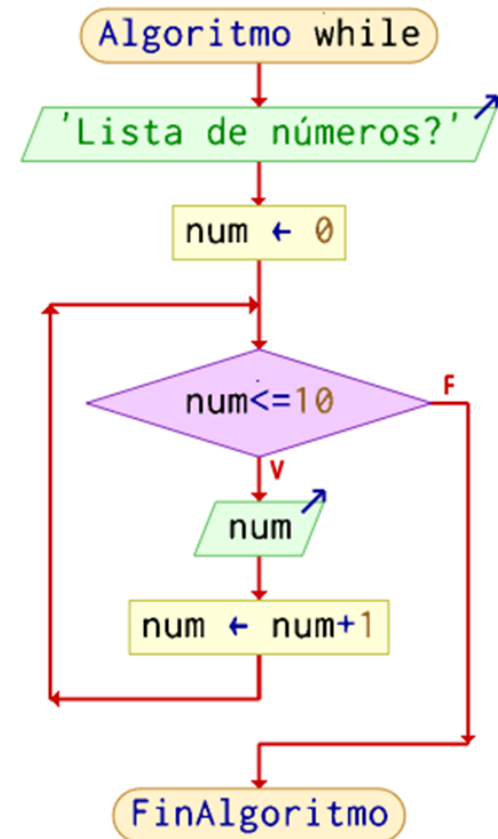
Estructuras repetitivas

En caso de que se tenga que repetir alguna acción un determinado número de veces, si se realiza de forma lineal, sin embargo, en la mayoría de los casos, el número de instrucciones varia, además que el programa no sería óptimo.



Estructuras repetitivas

Las estructuras repetitivas se utilizan para realizar un determinado tipo de instrucciones en un número finito de veces. Se caracterizan por tener un punto inicial de partida, una condición la cual se encarga de ejecutar un número determinado de acciones hasta que esta condición no sea válida (no se cumpla). Las sentencias repetitivas son **for**, **while** y **do while**.



Almacenamiento de datos

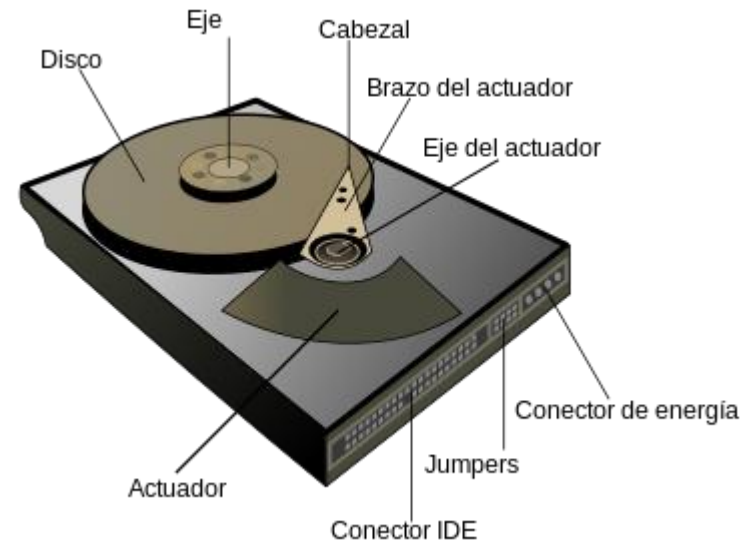
En informática, la memoria es el dispositivo que retiene, memoriza o almacena datos informáticos durante algún periodo de tiempo.



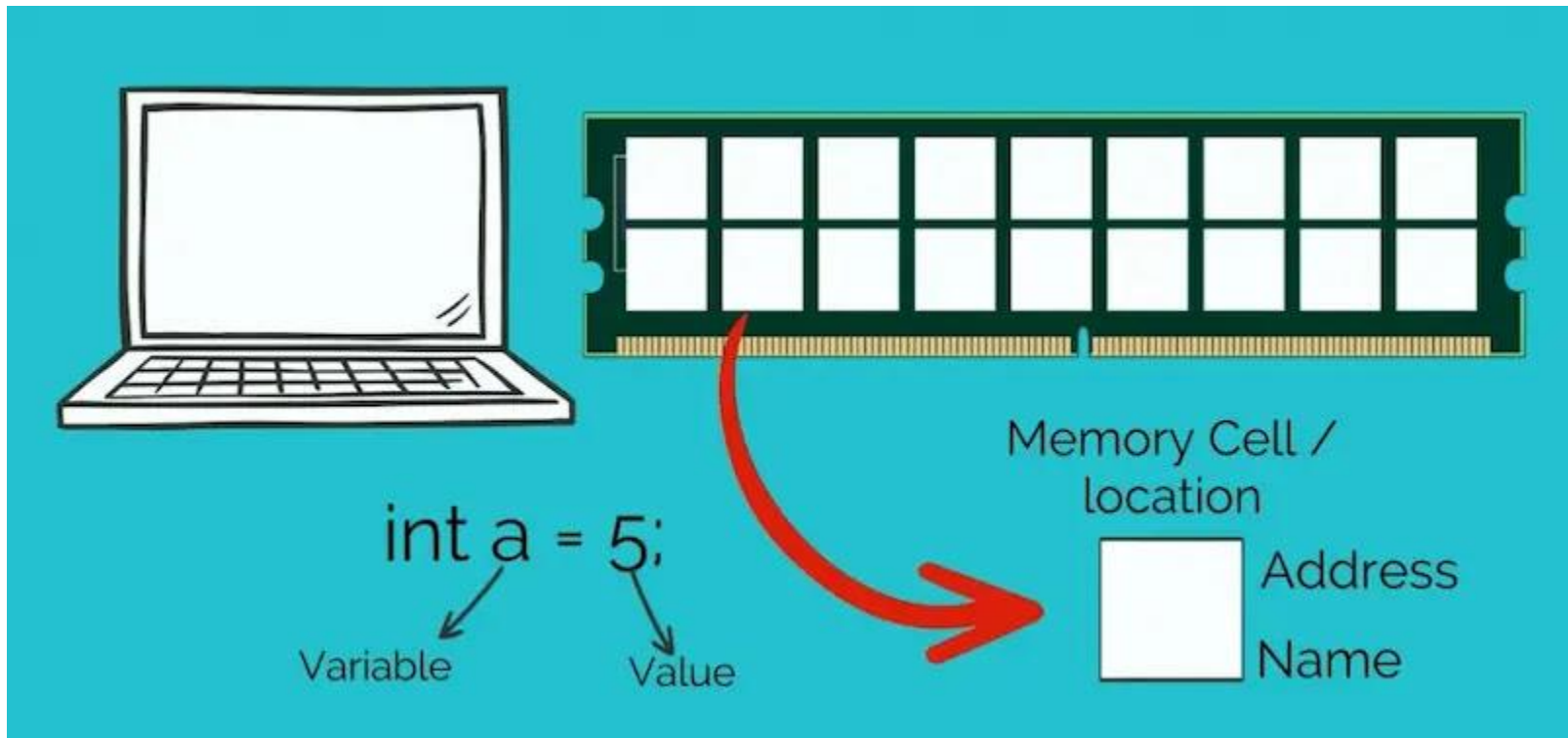
Almacenamiento de datos

Memoria suele referirse a una forma de almacenamiento de estados sólido, conocida como memoria **RAM** (memoria de acceso aleatorio), y otras veces se refiere a otras formas de almacenamiento rápido, pero temporal.

De forma similar, se refiere a formas de almacenamiento masivo, como **discos ópticos**, y tipos de almacenamiento magnético, como **discos duros** y otros tipos de almacenamiento, más lentos que las memorias RAM, pero de naturaleza más permanente.



Almacenamiento de datos



Tipos de datos

- **Numérico**
- **Número Entero**
- **Número Real** (con decimales)
- **Carácter**
- **Cadena de caracteres**
- **Booleano** (verdadero o falso)
- **Enumerado** (un conjunto de valores limitado)
- **Texto**: letras, caracteres, símbolos que representan otros idiomas.
- **Valores booleanos**: que son fundamentales para establecer condiciones de verdad o falsedad.
- **Listas**: para almacenar múltiples elementos de un mismo tipo.



Tipos de datos

- **Caracteres:** El tipo de dato carácter es un dígito individual el cual se puede representar como numéricos (0 al 9), letras (a-z) y símbolos (!"#\$%&/'\).
- **Caracteres Unicode:** El tipo de dato carácter unicode es una "extensión" del tipo de dato carácter, permite ampliar los símbolos de escritura, provee exactamente hasta 65535 caracteres diferentes.
- **Reales:** estos caracteres almacenan números muy grandes que poseen parte entera y parte decimal.
- **Booleanos:** Este tipo de dato se emplea para valores lógicos, los podemos definir como datos comparativos dicha comparación devuelve resultados lógicos (Verdadero o Falso).

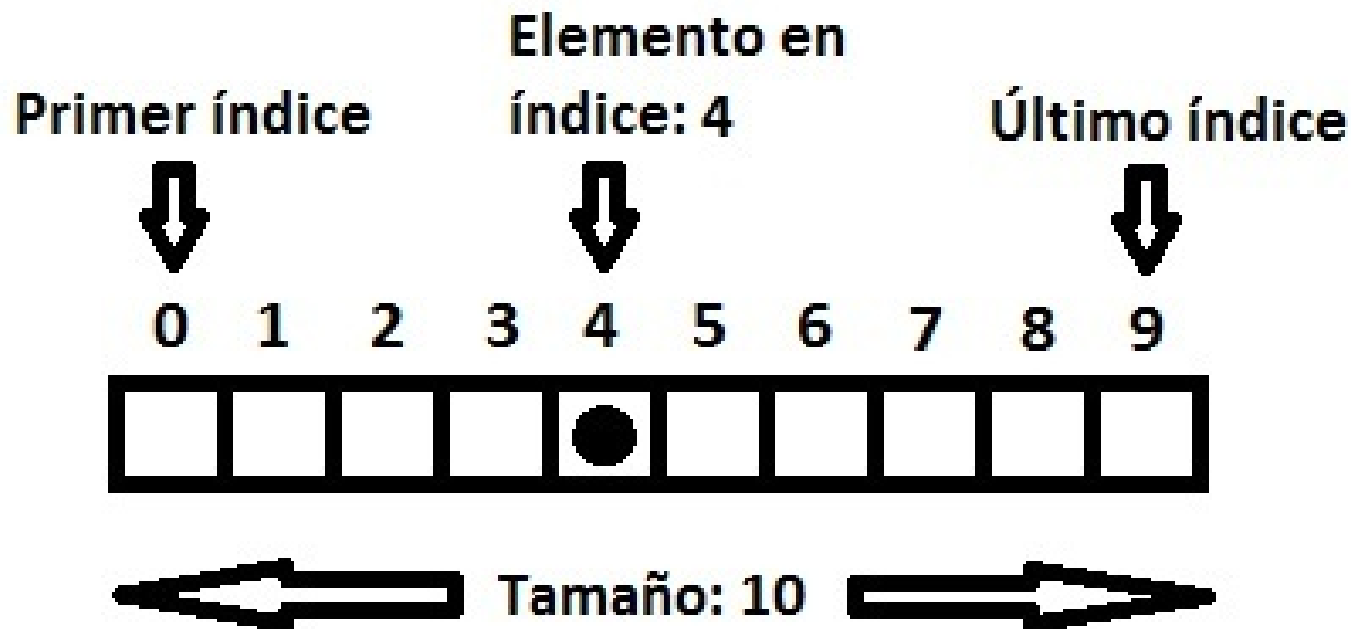
Tipos de datos

- **Tipos compuestos:** Los tipos compuestos se derivan de uno o más datos primitivos. A las distintas maneras de formar o combinar estos datos se les conocen con el nombre de “Estructura de datos”. Al combinarlo podemos crear un nuevo tipo, por ejemplo:

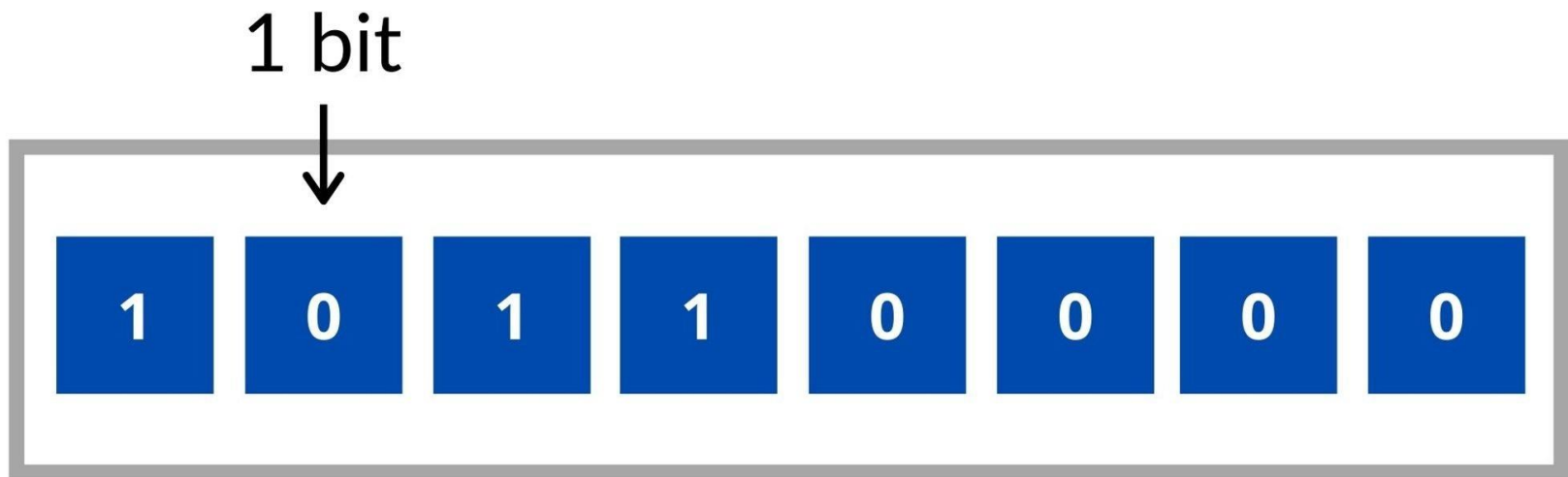
"array-de-enteros" es distinto al tipo "entero"

Un vector (array) almacena un número de elementos del mismo tipo en un orden específico. Los arrays pueden ser estáticos (con una medida fija) o dinámicos (crecer durante su ciclo de vida).

Tipos de datos



Tipos de datos: Pascal



8 bits = 1 byte

Tipos de datos: Pascal

Nombre	Memoria requerida	Rango	Descripción
Booleano	1bit	1 ~ 0	Verdad - Falso
Byte	1 byte (8 Bits)	0 ~ 255	Byte sin signo.
ByteSig	1 byte (8 Bits)	(-128) ~ 127	Byte con signo.
Word	2 byte (16 Bits)	0 ~ 65.535	Word sin signo.
WordSig	2 byte (16 Bits)	(-32768) ~ 32767	Word con signo.
Entero	4 byte (32 Bits)	0 ~ 4.294.967.295	Entero sin signo.
EnteroSig	4 byte (32 Bits)	(-2.147.483.648) ~ 2.147.483.647	Entero con signo.
Real	8 byte (64 Bits)	$(-1,79769313486232^{308}) \sim (-4,94065645841247^{-324})$	Número con coma flotante de doble precisión
Decimal	8 byte (64 Bits)	(-922.337.203.685.477,5800) ~ 922.337.203.685.477,5800	Número con coma fija de 4 decimales.
Cadena	1 byte por carácter	0 ~ 2000 millones de caracteres	Cadena de caracteres alfanumérica.

Tipos de datos: Java

Nombre	Declaración	Memoria requerida	Intervalo	Descripción
Booleano	boolean	-	true - false	Define una bandera que puede tomar dos posibles valores: true o false.
Byte	byte	1 byte (8 bits)	[-128 .. 127]	Representación del número de menor rango con signo.
Entero pequeño	short	2 byte (16 bits)	[-32,768 .. 32,767]	Representación de un entero cuyo rango es pequeño.
Entero	int	4 byte (32 bits)	$[-2^{31} .. 2^{31}-1]$	Representación de un entero estándar. Este tipo puede representarse sin signo usando su clase <i>Integer</i> a partir de la Java SE 8.
Entero largo	long	8 byte (64 bits)	$[-2^{63} .. 2^{63}-1]$	Representación de un entero de rango ampliado. Este tipo puede representarse sin signo usando su clase <i>Long</i> a partir de la Java SE 8.
Real	float	4 byte (32 bits)	$[\pm 3,4 \cdot 10^{-38} .. \pm 3,4 \cdot 10^{38}]$	Representación de un real estándar. Recordar que al ser real, la precisión del dato contenido varía en función del tamaño del número: la precisión se amplía con números más próximos a 0 y disminuye cuanto más se aleja del mismo.
Real largo	double	8 byte (64 bits)	$[\pm 1,7 \cdot 10^{-308} .. \pm 1,7 \cdot 10^{308}]$	Representación de un real de mayor precisión. Double tiene el mismo efecto con la precisión que float.
Carácter	char	2 byte (16 bits)	['\u0000' .. '\uffff'] o [0 .. 65.535]	Carácter o símbolo. Para componer una cadena es preciso usar la clase <i>String</i> , no se puede hacer como tipo primitivo.

Tipos de datos: Python

Tipo	Clase	Notas	Ejemplo
<code>str</code>	Cadena en determinado formato de codificación (UTF-8 por defecto)	Inmutable	<code>'Cadena'</code>
<code>bytes</code>	Vector o <i>array</i> de bytes	Inmutable	<code>b'Cadena'</code>
<code>list</code>	Secuencia	Mutable, puede contener objetos de diversos tipos	<code>[4.0, 'Cadena', True]</code>
<code>tuple</code>	Secuencia	Inmutable, puede contener objetos de diversos tipos	<code>(4.0, 'Cadena', True)</code>
<code>set</code>	Conjunto	Mutable, sin orden, no contiene duplicados	<code>{4.0, 'Cadena', True}</code>
<code>frozenset</code>	Conjunto	Inmutable, sin orden, no contiene duplicados	<code>frozenset([4.0, 'Cadena', True])</code>
<code>dict</code>	Diccionario	Grupo de pares clave:valor	<code>{'key1': 1.0, 'key2': False}</code>
<code>int</code>	Número entero	Precisión arbitraria	<code>42</code>
<code>float</code>	Número decimal	Coma flotante de doble precisión	<code>3.1415927</code>
<code>complex</code>	Número complejo	Parte real y parte imaginaria <i>j</i> .	<code>(4.5 + 3j)</code>
<code>bool</code>	Booleano	Valor booleano (verdadero o falso)	<code>True</code> o <code>False</code>

Variable

Una **variable** es un objeto del lenguaje cuyo valor se puede cambiar. Antes de utilizar una variable ésta debe de ser declarada. Al declarar una variable, se le asocia un identificador, es decir, un **nombre**, con un tipo de almacenamiento cuya forma determina la visibilidad y existencia de la variable.

El tipo de la variable nos indica el **conjunto de valores** que puede tomar y las operaciones que pueden realizarse con ella.

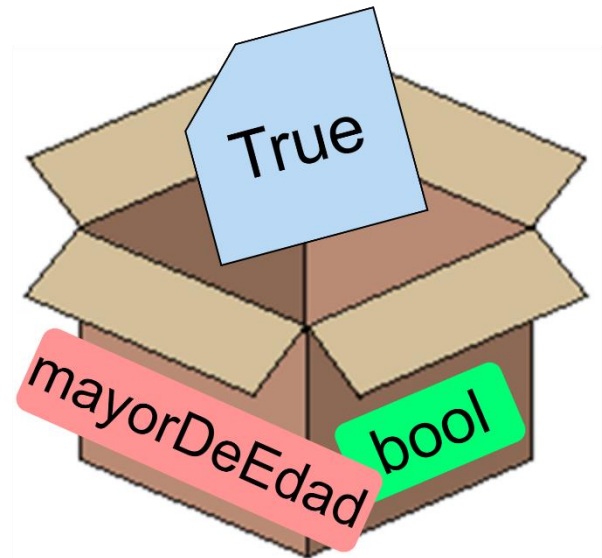
Variable

id	valor
edad	22
apellido	"Chang"
buscando	true
peso	9.35

Variable



```
1 mayorDeEdad = True  
2 print(type(mayorDeEdad))  
3 print(mayorDeEdad)
```



Variable

- **Variable global:** se considera una variable como global a no ser que esté declarada dentro de una definición de función. Las variables globales resultan visibles y disponibles para todas las sentencias de un script.
- **Variable local:** Las variables locales sólo resultan visibles y disponibles dentro de la función en la que están definidas.

```
1 #Local and Global variable
2
3 x = 23          #This is the Global Variable
4
5 def fnc_name():
6     y = 24      #This is Local Variable
7     print(y)
8
9 fnc_name()
10
11
```

Constantes

Una constante es un elemento de datos con nombre con un valor predefinido. No se puede cambiar el valor asignado a una constante predefinida.

palabra reservada
para indicar que
es una constante

tipo de dato

nombre de
la constante

```
const int buttonPin = 2;
```

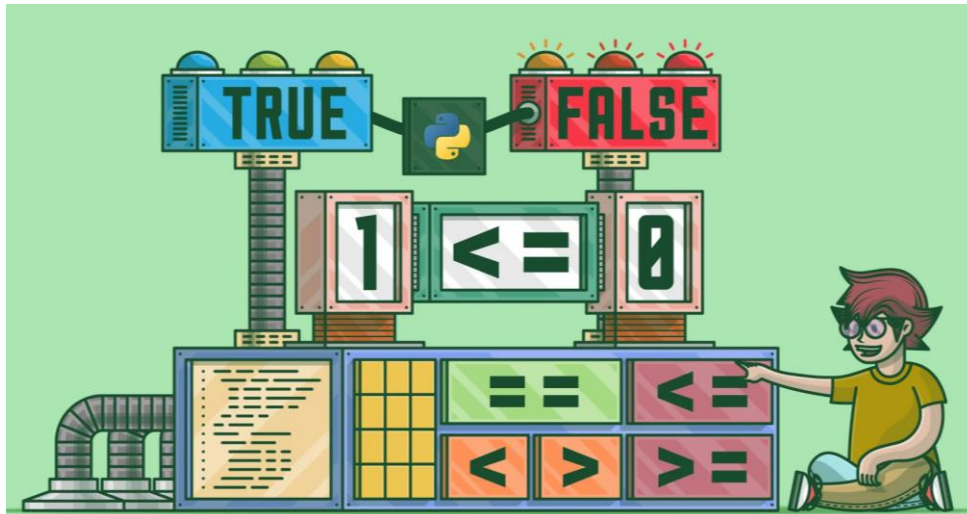
signo de
asignación

dato que
contiene

Constantes

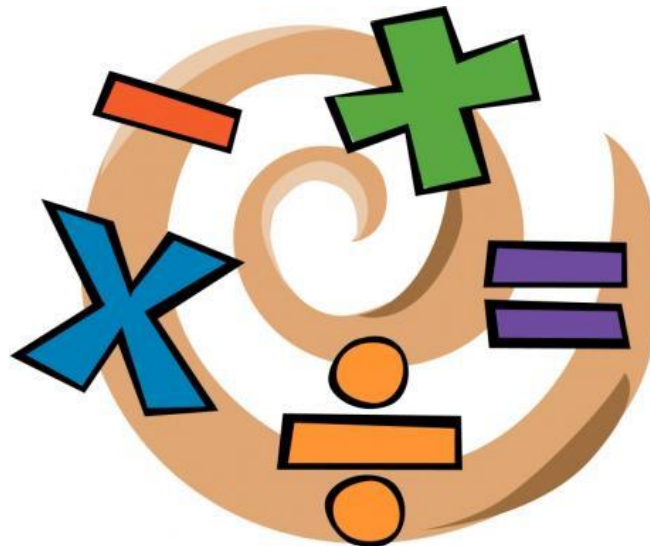
Las constantes predefinidas son:

- **NULL:** Una referencia vacía.
- **TRUE:** Equivalente al número 1.
- **FALSE:** Equivalente al número 0.



Operadores aritméticos

Las expresiones realizan acciones específicas, según un operador, con uno o dos operandos. Un operando puede ser una **constante**, una **variable** o el **resultado de una función**. Los operadores son **aritméticos**, **lógicos** y **relacionales**.



Operadores aritméticos

Realizan operaciones matemáticas, como sumas o restas con operandos. Hay dos tipos de operadores matemáticos: unarios y binarios.

- Los operadores **unarios** realizan una acción con un solo operando.
- Los operadores **binarios** realizan acciones con dos operandos.

Operadores aritméticos

Símbolo	Operación	Ejemplo	Descripción
+	Suma	$a + b$	Sumar los dos operandos
-	Resta	$a - b$	Restar el segundo operando del primero
*	Multiplicación	$a * b$	Multiplicar los dos operandos
/	División	a / b	Dividir el primer operando por el segundo
**	Potencia	$a ** b$	Elevar el primer operando a la potencia del segundo operando
%	Resto	$a \% b$	Dividir el primer operando por el segundo y dar como resultado la parte restante

Operador de asignación (=)

Utilice el operador de asignación (=) para copiar una constante, literal, resultado de expresión de variables o resultado de función en una variable.

Operadores lógicos

Símbolo	Operación	Ejemplo	Descripción
AND &&	AND	Expr1 \$\$ expr2	Verdadero si tanto expr1 como expr2 son verdaderas.
OR	O	Expr1 OR expr2	Verdadero si expr1 o expr2 son verdaderas.

Operadores relacionales

Símbolo	Operación	Ejemplo	Descripción
<	Menor que	$a < b$	Verdadero si a es menor que b.
>	Greater than (Mayor que)	$a > b$	Verdadero si a es mayor que b.
==	Igual	$a == b$	Verdadero si a es igual a b.
!=	No igual	$a \neq b$	Verdadero si a no es igual a b.
<=	Menor o igual que	$a \leq b$	Verdadero si a es menor o igual que b.
>=	Mayor o igual que	$a \geq b$	Verdadero si a es mayor o igual que b.

Ejercicios

Ejercicio 1: Si X, Y y Z son variables de tipo booleano con valores X = true, Y = false, Z = true, determina el valor de las siguientes expresiones lógicas:

- a) X and Y
- b) X and Z
- c) X or Y
- d) X or Z
- e) ! X
- f) (X and Y) or (X and Z)
- g) X or Y and Z

Ejercicios

Ejercicio 2: Si W, X, Y y Z son variables de tipo booleano con valores W = false, X = true, Y = true, Z = false, determina el valor de las siguientes expresiones lógicas:

- a) (X and Y) or (X and Z)
- b) (X and !Y) or (!X or Z)
- c) W or Y and X and W or Z
- d) X and !Y and !X or !W and Y
- e) (W or !Y) and X or Z
- f) X and Y and W or Z or X