

Curso de nivelación de algoritmos

Algoritmo

Un **algoritmo** es una secuencia de **instrucciones**.

Algoritmo

Un **algoritmo** es una secuencia de **instrucciones**.

Ejemplo:

- 1.- Moje el cabello,
- 2.- Coloque champú,
- 3.- Masajee suavemente y deje actuar por 2 min.,
- 4.- Enjuague, y
- 5.- Repita el procedimiento (desde 1.-).

Algoritmo

Otro ejemplo:

Ingredientes: 15 huevos, 600 gramos de harina, 600 gramos de azúcar

- 1.- Mientras no estén espumosos, batir los huevos junto con el azúcar,
- 2.- agregar la harina en forma envolvente sin batir,
- 3.- batir suavemente,
- 4.- colocar en el horno a 180 grados,
- 5.- si le clavo un cuchillo y sale húmedo, entonces ir a 4.-
- 6.- retirar del horno,
- 7.- mientras no esté frío, esperar
- 8.- desmoldar y servir,
- 9.- fin

Instrucción

Una **instrucción** es una operación que:

- transforma los datos, o bien
- modifica el flujo de ejecución.

Instrucción

Una **instrucción** es una operación que:

- transforma los datos, o bien
- modifica el flujo de ejecución.

- 1.- Moje el cabello,
- 2.- Coloque champú,
- 3.- Masajee suavemente y deje actuar por 2 min.,
- 4.- Enjuague, y
- 5.- Repita el procedimiento (desde 1.-).

Instrucción

Una **instrucción** es una operación que:

- transforma los datos, o bien
- modifica el flujo de ejecución.

- 1.- Mientras no estén espumosos, batir los huevos junto con el azúcar,
- 2.- agregar la harina en forma envolvente sin batir,
- 3.- batir suavemente,
- 4.- colocar en el horno a 180 grados,
- 5.- si le clavo un cuchillo y sale húmedo, entonces ir a 4.-
- 6.- retirar del horno,
- 7.- mientras no esté frío, esperar
- 8.- desmoldar y servir,
- 9.- fin

Programa

Un programa es una implementación de un algoritmo en un lenguaje de programación.



```
te
: value;
fixPrecision : function(value){
  nan = isNaN(value);
  !this.allowDecimals || this.decimalPrecision = 0;
  return nan ? '' : value;
}

return parseFloat(parseFloat(value).toFixed(this.decimalPrecision));

function(){
  Blur : function(){
    var v = this.parseValue(this.getRawValue());
    if(!Ext.isEmpty(v)){
      this.setRawValue(this.fixPrecision(v));
    }
  }
}
```


Datos

Los programas manipulan **valores** de diferentes **tipos**.

Ejemplos:

- 1 es un valor de tipo **entero**.
- “Hola” es un valor de tipo **string**.
- False es un valor de tipo **bool (lógico)**.

Tipos de datos

Valores de verdad (bool):

Denotan el resultado de una evaluación lógica: los valores “verdadero” (*True*) y “falso” (*False*).

Tipos de datos

Operaciones de booleanos:

Operador	Operación
not	Negación
and	Conjunción
or	Disyunción

Tipos de datos

Negación de un booleano:

p	not p
True	False
False	True

Tipos de datos

Conjunción de booleanos:

p	q	p and q
True	True	True
True	False	False
False	True	False
False	False	False

Tipos de datos

Disyunción de booleanos:

p	q	p or q
True	True	True
True	False	True
False	True	True
False	False	False

Tipos de datos

Enteros (int):


Los enteros para una computadora son *levemente* diferentes que los enteros matemáticos.


Tipos de datos

Enteros (int):

Los enteros para una computadora son *levemente* diferentes que los enteros matemáticos.

Están acotados por encima y por debajo.


$-\infty, \dots, -2, -1, 0, 1, 2, \dots, \infty$ 


$-9.223.372.036.854.775.808, \dots, -2, -1, 0, 1, 2, \dots, 9.223.372.036.854.775.807$ 

Tipos de datos

Enteros (int):

Los enteros para una computadora son *levemente* diferentes que los enteros matemáticos.
Están acotados por encima y por debajo.

$-\infty, \dots, -2, -1, 0, 1, 2, \dots, \infty$ 

$-9.223.372.036.854.775.808, \dots, -2, -1, 0, 1, 2, \dots, 9.223.372.036.854.775.807$ 

¿Por qué esas cotas?

Porque Python usa 64 bits para representar enteros.

(Más adelante vamos a ver qué problemas puede traer esta representación.)

Tipos de datos

Operaciones de enteros:

Operador	Operación	Ejemplo
+	Suma	$3 + 4 \rightarrow 7$
-	Resta	$5 - 6 \rightarrow -1$
*	Producto	$2 * 8 \rightarrow 16$
/	División	$5 / 2 \rightarrow 2$
%	Resto	$5 \% 2 \rightarrow 1$
-	Negación (unaria)	-6

Tipos de datos

Comparaciones entre enteros:

Operador	Operación
$i==k$	Igualdad
$i!=k$	Distinto
$i<k$	Comparación por menor
$i>k$	Comparación por mayor
$i\leq k$	Comparación por menor o igual
$i\geq k$	Comparación por mayor o igual

Tipos de datos

Listas:

Una lista es una colección de valores que se acceden mediante un índice.

45	657	6756	4	23	5	324	2344
0	1	2	3	4	5	6	7

Tipos de datos

Operaciones básicas de listas:

Operador	Operación
<code>[]</code>	Lista vacía.
<code>c.append(x)</code>	Agrega el valor x al final de la lista c.
<code>c[i]</code>	i-ésimo elemento de la lista c.
<code>len(c)</code>	Longitud de la lista c.

¡Atención! Los índices válidos de una lista de N elementos no van de 1 a N, sino de 0 a N-1.

Ejemplo: Si c tiene 8 elementos, entonces `c[8]` está indefinido, y genera un error.

(En Python hay **muchas** más operaciones de listas, que vamos a ver más adelante.)

Tipos de datos

Cadena de caracteres (string, o 'str'):

Un **character** es un símbolo válido en la computadora:

- abcdefghijklmnopqrstuvwxyz
- ABCDEFGHIJKLMNOPQRSTUVWXYZ
- 1234567890
- !@#\$%^&*()-_+=~`':;,."<>?/
- etc.

Un **string** es una cadena o secuencia de caracteres.

En Python se puede tratar como una lista:

```
a = 'Hola, mundo!'
print len(a)      → Output: ?
print a[1]        → Output: ?
```

Tipos de datos

Cadena de caracteres (string, o 'str'):

Un **character** es un símbolo válido en la computadora:

- abcdefghijklmnopqrstuvwxyz
- ABCDEFGHIJKLMNOPQRSTUVWXYZ
- 1234567890
- !@#\$%^&*()-_+=~`':;,."<>?/
- etc.

Un **string** es una cadena o secuencia de caracteres.

En Python se puede tratar como una lista:

```
a = 'Hola, mundo!'
print len(a)      → Output: 12
print a[1]        → Output: o
```

Tipos de datos

Operaciones básicas de cadenas de caracteres:

Operador	Operación
<code>c[i]</code>	i-ésimo caracter de la cadena c.
<code>len(c)</code>	Longitud de la cadena c.
<code>+</code>	Concatena dos cadenas. Ej: 'hola'+'chau' → 'holachau'
<code>==</code> <code>!=</code> <code><</code> <code><=</code> <code>></code> <code>>=</code>	Comparaciones lexicográficas entre cadenas.
...	...

Tipos de datos - Resumen

Tipo de datos	Ejemplos
bool	True, False
int	3, 0, -5
float	3.0, 0.0, -5.0, 3.141592
list	['pepe', 'coco'], [10, 20, 30, 40, 50], []
string	'pepe', 'coco', "

Memoria

Durante la ejecución de un programa, sus datos se almacenan en la memoria.

La memoria de una computadora es una secuencia numerada de “celdas” o “posiciones de memoria”, en las cuales podemos almacenar datos.

Unidad elemental: el **bit**, que toma valores **0** ó **1**.

Soporte físico: electrónico, magnético, óptico, ...

Memoria

8 bits	= 1 byte	→ Unidad mínima más usada.
1024 bytes	= 1 KB (kilobyte)	
1024 KB	= 1 MB (megabyte)	
1024 MB	= 1 GB (gigabyte)	
1024 GB	= 1 TB (terabyte)	
1024 TB	= 1 PB (petabyte)	
1024 PB	= 1 EB (exabyte)	
...		

(Nota: Para discos rígidos, se suele usar 1000 en lugar de 1024.)

Variable

Una **variable** es como una cajita en la que podemos colocar un dato particular...

Variable

Una **variable** es como una cajita en la que podemos colocar un dato particular...

Papá Noel vive en el polo norte junto con un montón de duendes que hacen regalos para los chicos, los camellos de los reyes magos se comen el pasto y el agua que les dejás por la noche, si tomás vino y comés sandía te morís, etc.

Variable

Una **variable** es como una cajita en la que podemos colocar un dato particular...

Papá Noel vive en el polo norte junto con un montón de duendes que hacen regalos para los chicos, los camellos de los reyes magos se comen el pasto y agua que les dejás por la noche, si tomás vino y comés sandía te morís, etc.

Una **variable** es un nombre que denota una **dirección de memoria** en la que se almacena un valor. De esa dirección es posible la lectura y/o modificación del valor almacenado.

Expresión

Una **expresión** es una combinación de valores, variables y operadores.

La **evaluación** de una expresión arroja como resultado un valor.

Ejemplos:

¿Qué valores resultan de evaluar estas expresiones?

`1 + 1`

`'ho' + 'la'`

`(1 > 0) or (not ('a' < 'b'))`

`len('hola') + 6`

`(5.6 > 2.0) and (len('hola') < 2)`

Nuestra primera instrucción: la **Asignación**

VARIABLE = EXPRESIÓN

Almacena el valor de la *EXPRESIÓN* en la dirección en memoria denotada por *VARIABLE*.

Ejemplos:

$x = 1000$

$x = x$

$x = y$

$x = x + y * 22 / 33$

$1000 = x$

$x + 1 = y$

Nuestra primera instrucción: la **Asignación**

VARIABLE = EXPRESIÓN

Almacena el valor de la *EXPRESIÓN* en la dirección en memoria denotada por *VARIABLE*.

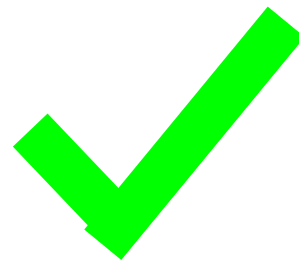
Ejemplos:

$x = 1000$

$x = x$

$x = y$

$x = x + y * 22 / 33$



~~$1000 = x$~~
 ~~$x + 1 = y$~~

Literal

Un **literal** es un valor particular utilizado directamente en el código del programa.

Ejemplos:

```
x = 10
```

```
x = x + 7.45
```

```
c = 'a'
```

```
b = (pepe > 'coco') or not True
```

Programa

Un **programa** es una secuencia de **instrucciones**.

En particular,
una asignación es un programa!

Secuencialización

PROG1; PROG2

PROG1 y *PROG2* son programas. Se ejecuta primero *PROG1*; una vez finalizado, se ejecuta *PROG2*.

Ejemplo:

```
a = 10
print 'a vale', a
b = a + 1
print 'b vale', b
```

Estado

Se denomina **estado** al valor de todas las variables de un programa en un punto de su ejecución, más la siguiente instrucción a ejecutar.

Es una “foto” de la memoria en un momento determinado.

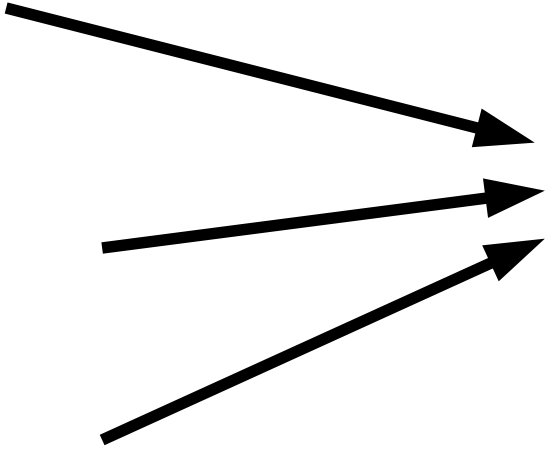
Estado

Ejemplo:

y = 10

x = y * 2

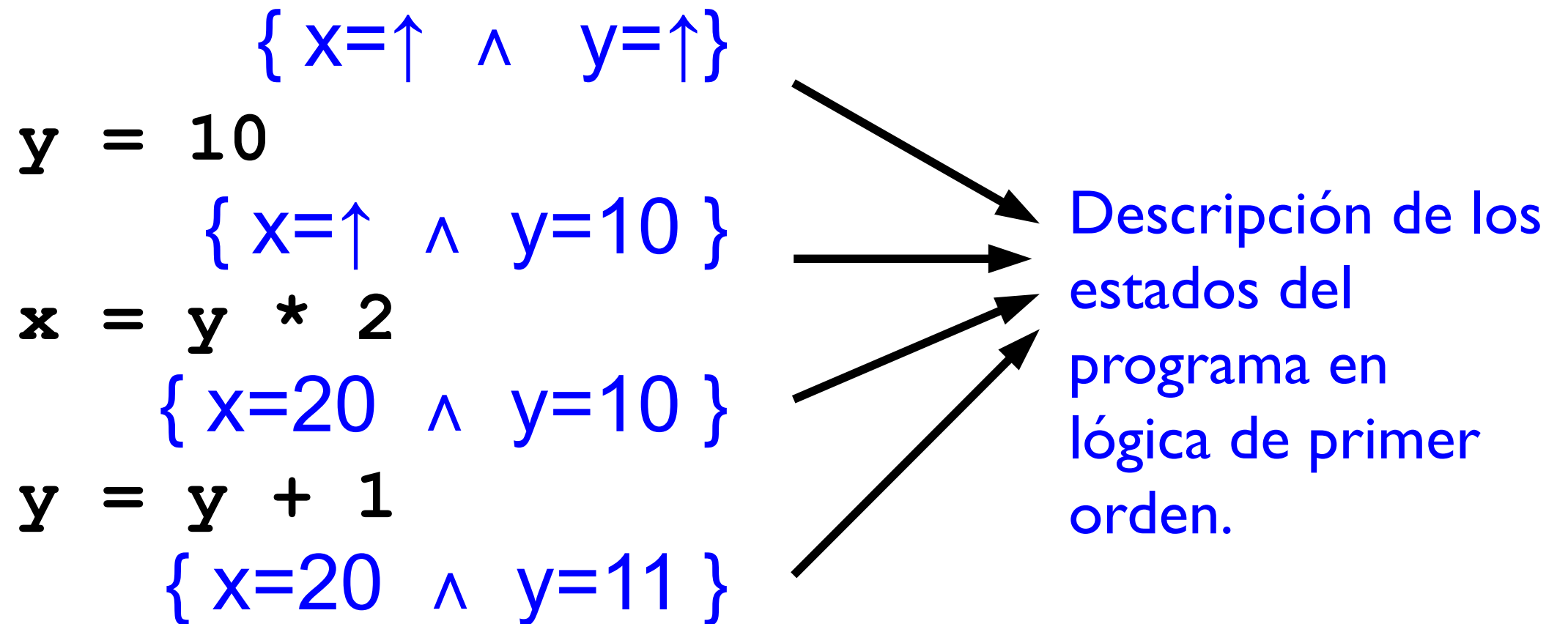
y = y + 1



Instrucciones en el
lenguaje de
programación Python.

Estado

Ejemplo:



\uparrow significa “valor indefinido”

Repaso de la clase de hoy

- Valor. Tipos de datos: bool, int, float, string, list.
- Expresiones, variables, literales.
- Memoria, estado.
- Programa, instrucción, asignación, secuencialización.

Temas de la clase que viene

- Condicionales, ciclos, funciones.