

Identificadores

Nombres dados a objetos

`mi_var iable, área_1, _temp`

Palabras Clave

No pueden usarse como identificadores

`and` Operador lógico

`as` Crea alias

`assert` Depura

`break` Sale de un loop

`class` Define una clase

`continue` Continúa la iteración de un loop

`def` Define una función

`del` Elimina un objeto

`elif` Crea una condición

`else` Crea una condición

`except` Da lugar a una excepción

`false` Valor booleano

`finally` Siempre se ejecuta, independientemente de si hay una excepción o no

`for` Crea un loop

`from` Importa cosas específicas de un módulo

`global` Declarar una variable global

`if` Crea una condición

`import` Para importar un módulo

`in` Verifica si un valor específico está dentro de una secuencia

`is` Prueba si dos variables son iguales

`lambda` Crea una función anónima

`none` Representa un valor null

`nonlocal` Declara una variable no local

`not` Operador Lógico

`or` Operador Lógico

`raise` Plantea una excepción

Palabras Clave (cont)

`return` Sale de una función y retorna un valor

`true` Valor booleano

`try` Comienza una sentencia

`try...e xcept`

`while` Comienza un loop

`with` Simplifica el manejo de excepciones

`yield` Devuelve una lista de valores de un generador

Booleanos

`True` 1

`False` 0

`es_verdad = True`

`no_es_verdad = False`

Operadores Numéricos

`x + y` Suma

`x - y` Resta

`x * y` Multiplicación

`x / y` División, devuelve un flotante

`x // y` División entera

`x % y` Módulo o resto

`x y**` Potencia

`-x` Negativo

`+x` Positivo

`abs(x)` Valor absoluto de x

`divmod (x , y)` Cociente y resto de x entre y

`pow (x , y)` Eleva x a la potencia de y

`pow (x , y , z)` Calcula (x ** y) % z

`round (x, n)` Redondea x a n decimales

Conversión de Enteros

`bin (i)` Convierte i a binario

`hex (i)` Convierte i a hexadecimal

`int (x)` Convierte x a entero

`int (s, base)` Convierte la cadena s a entero en la base especificada

`oct (i)` Convierte i a octal

Números Complejos

`a + bj` Define un número complejo (a real, b imaginario)

`z.real` Parte real de z

`z.imag` Parte imaginaria de z

`z.conj-ugate()` Cambia el signo de la parte imaginaria

`complex(x, y)` Crea un número complejo a partir de x (real) y y (imaginario)

Escapes de Strings

`\newline` Escape

`\\` Barra invertida (\)

`\'` Comilla simple (')

`\"` Comilla doble (")

`\a` Campana ASCII

`\b` Retroceso ASCII

`\f` Salto de página ASCII

`\n` Nueva línea ASCII

`\N{name}` Carácter Unicode con el nombre dado

`\ooo` Carácter con valor octal ooo

`\r` Retorno de carro ASCII

`\t` Tabulación ASCII

`\uhhhh` Carácter Unicode de 16 bits hhhh

`\Uhhhhhhhh` Carácter Unicode de 32 bits hhhhhhhh

`\v` Tabulación vertical ASCII

`\xhh` Carácter con valor hexadecimal hh

By [schmidtlady16](#)

Published 30th October, 2024.

Last updated 30th October, 2024.

Page 1 of 2.

Sponsored by [Readable.com](#)

Measure your website readability!

<https://readable.com>

Slicing y Striding de Strings

<code>s[i]</code>	Devuelve el carácter en la posición i
<code>s[start:end]</code>	Extrae una subcadena desde start hasta end (excluye end)
<code>s[start:end:step]</code>	Extrae caracteres desde start hasta end con un intervalo step
<code>s[:end]</code>	Extrae desde el inicio hasta end
<code>s[start:]</code>	Extrae desde start hasta el final de la cadena
<code>s[::-1]</code>	Invierte la cadena

Operadores Bit a Bit

<code>i j</code>	OR bit a bit entre i y j
<code>i ^ j</code>	XOR bit a bit entre i y j
<code>i & j</code>	AND bit a bit entre i y j
<code>i << j</code>	Desplaza i a la izquierda j bits
<code>i >> j</code>	Desplaza i a la derecha j bits
<code>~i</code>	Invierte los bits de i



By **[schmidtlady16](#)**

cheatography.com/schmidtlady16/

Published 30th October, 2024.

Last updated 30th October, 2024.

Page 2 of 2.

Sponsored by **[Readable.com](#)**

Measure your website readability!

<https://readable.com>