Variables y Tipos de datos

Tipos de datos built-in de Python
Asignación de variables
Operaciones y Operadores

Objetivos

- Seleccionar los tipos de datos y los operadores lógicos y relacionales apropiados para escribir expresiones
- Aplicar la precedencia de los operadores, el operador de asignación, la lógica usada en las operaciones booleanas y los tipos de datos para escribir expresiones
- Usar funciones del lenguaje de programación para manipular cadenas de caracteres

Tipos de datos built-in
Numéricos, Lógicos y Cadenas

Tipos de datos primitivos

- El lenguaje Python permite operar con los siguientes tipos de datos básicos:
  - Numéricos: enteros, reales o de punto flotante, y complejos
  - Lógicos: booleanos
  - Cadenas: secuencia o cadena de caracteres (tipo de dato estructurado)

Datos numéricos

Tipo	Nombre	Descripción	Ejemplo
Enteros	`int`	Números sin parte fraccionaria	`520` `-318`

Reales o de punto flotante	`float`	Números con parte fraccionaria
		o expresados en notación de potencias de 10
			`6.37`
			`-0.089`
			`4.1e-3`

Complejos	`complex`	Números con un componente real y uno imaginario	`(9-3j)` `(2.5+6.4j)`

Lógicos

Tipo	Nombre	Descripción	Ejemplo
Booleano	`bool`	Representación de los valores lógicos Verdadero o Falso.	`True` `False`

Texto o Cadenas de caracteres

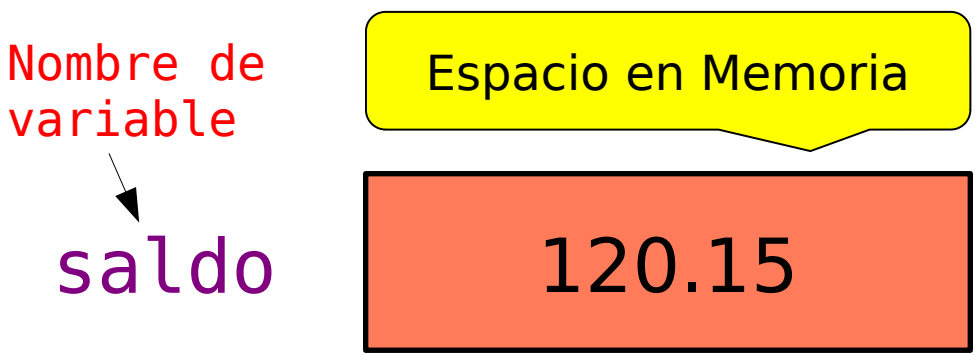
Tipo	Nombre	Descripción	Ejemplo
Cadenas	`str`	Expresiones (texto) formadas por caracteres. Se pueden representar indistintamente con comillas simples o dobles.	`'Hola'` `"Mundo"`

Definición y asignación de variables

Espacios de memoria para almacenar y recuperar datos

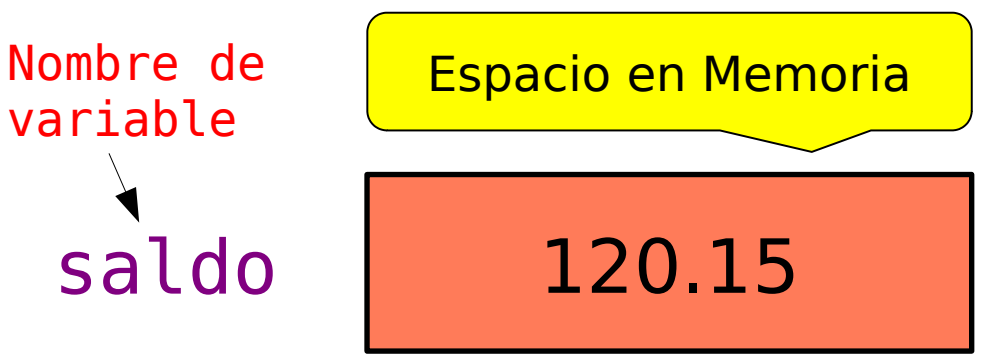
Variables

- Una variable es usada para almacenar, organizar y manipular la información en la memoria.
- Las variables son nombradas, es decir, hay que asignarles un nombre, que en Python debe seguir ciertas reglas



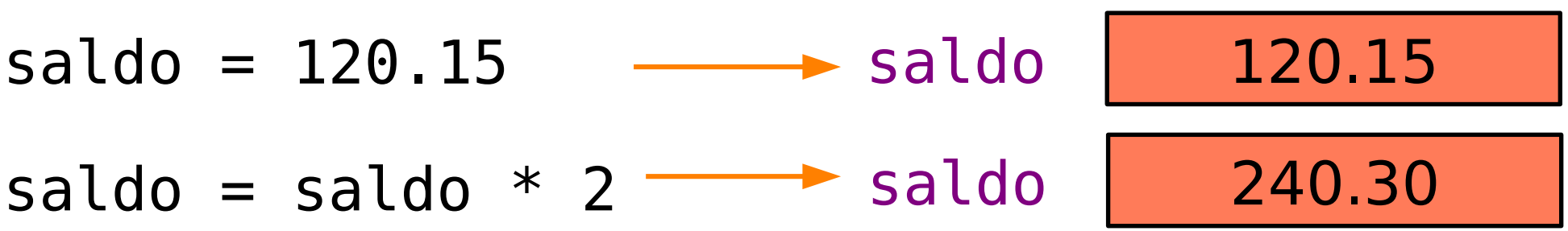
Nombres de Variables

- Sólo puede contener números, letras o el guión bajo, no utilizar espacios en blanco
- No puede iniciar con un número.
- No debe coincidir con una palabra reservada del lenguaje.



Operación de asignación =

- El = es el operador de asignación
- ¿Para qué sirve?
  - Crear una nueva variable
  - Asignar un valor a una variable que ya existe
- La asignación, se efectúa de derecha a izquierda. Si hay operaciones a la derecha, éstas se calculan y luego se asigna el resultado a la variable



Otras asignaciones

- Asignación en la misma línea:

```
base = 5; altura = base + 2; area = base * altura
```

- Asignación múltiple (unpacking):

```
(base, altura) = (5.0, 7)
```

- Asignación del mismo valor:

```
base = altura = 2.5
```

- Acumulador sumativo y multiplicativo:

```
acum = acum + 2
mult = mult * 3
```

Operaciones y Expresiones
Aritméticas, Lógicas y relacionales

Operadores Aritméticos

Símbolo	Operación	Ejemplo	Resultado
+	Suma	2 + 4	6
-	Resta	8 - 5	3
*	Multiplicación	6 * 2	12
/	División	9 / 2	4.5
//	División (Entera)	9 // 2	4
%	Módulo	9 % 2	1
**	Potenciación	2 ** 3	8

Módulo

- Modulo realiza una división entera y retorna el residuo de ésta.
- Se utiliza el operador %
- Ejemplo:
  - 3 % 1 = 3 residuo 0, el valor retornado es 0.
  - 5 % 2 = 2 residuo 1, el valor retornado es 1.
  - 14 % 4 = 3 residuo 2, el valor retornado 2.

  Esto es realmente útil para determinar si un número es divisible por otro. - Si la operación retorna 0 es exactamente divisible.
  - Para determinar si un número es divisible para 2: N % 2 el residuo debe ser 0.

Operadores relacionales

Símbolo	Operación	Ejemplo	Resultado
==	Igual que	5 == 5	True
!=	Distinto que	8 != 5	True
>	Mayor que	6 > 9	False
<	Menor que	9 < 2	False
>=	Mayor o igual que	7 >= 3	True
<=	Menor o igual que	4 <= 2	False

Operadores Lógicos

Símbolo	Operación	Ejemplo	Resultado
`and`	Conjunción	`(2 >1) and (4 < 8)`	`True`
`or`	Disyunción	`(9 != 6) or (7 <= 3)`	`True`
`not`	Negación	`not (a < b)`	`True/False`

Operadores Lógicos

Existen dos posibles valores lógicos

- `True` (T)
- `False` (F)

Operadores lógicos

- `NOT`
- `AND`
- `OR`

a	NOT a
T	F
F	T

a	b	a AND b	a OR b
F	F	F	F
F	T	F	T
T	F	F	T
T	T	T	T

Con los cuales podemos construir predicados como:

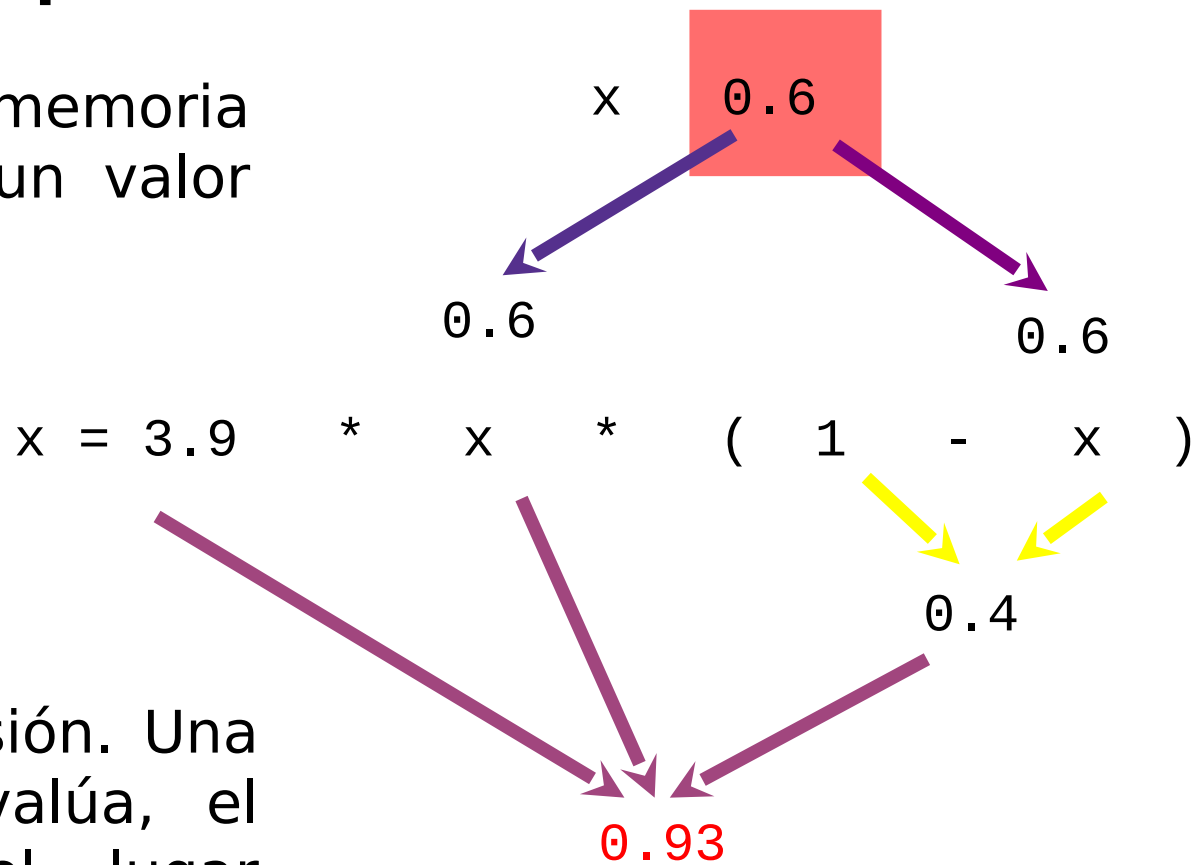
```
(a and b) or (c and (not d))
```

Operadores de incremento/decremento

Símbolo	Ejemplo	Equivalente a
+=	A += 5	A = A + 5
-=	A -= 5	A = A - 5
*=	A *= 5	A = A * 5
/=	A /= 5	A = A / 5
%=	A %= 5	A = A % 5

Expresiones

Una variable es un lugar de la memoria que se usa para almacenar un valor (0.6)



El lado derecho es una expresión. Una vez que la expresión se evalúa, el resultado se coloca en (el lugar asignado a) x.

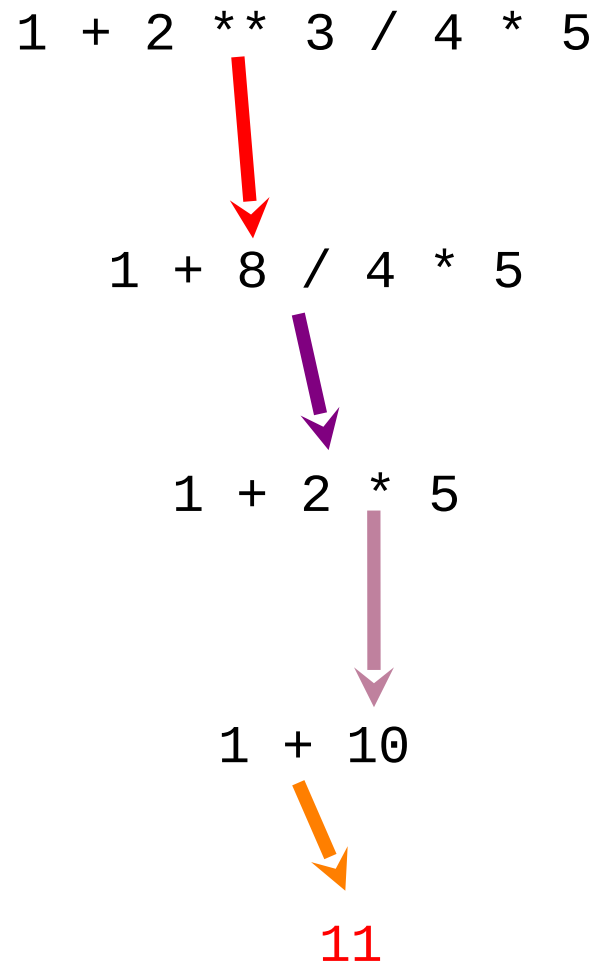
Reglas de Precedencia de Operador

- Reglas de precedencia en orden descendente:
  - Los paréntesis siempre son respetados
  - Potenciación
  - Multiplicación, División, y Módulo
  - Adición y Sustracción
  - De izquierda a derecha

Paréntesis
Potencia
Multiplicación
Adición
Izquierda a derecha

```
>>> x = 1 + 2 ** 3 / 4 * 5
>>> print(x)
11
>>>
```

Paréntesis
Potencia
Multiplicación
Adición
Izquierda a derecha



Más de Evaluar expresiones

- Expresiones con operadores de diferente tipo, se evalúan primero las operaciones aritméticas, luego las relacionales, y finalmente las lógicas.

```
( (3+4*x) > 10*(y-5) )  and  ( (a+b)/c  !=  9*(4 + c ) )
```

- Dos niveles de agrupamiento por paréntesis, primero paréntesis más internos.

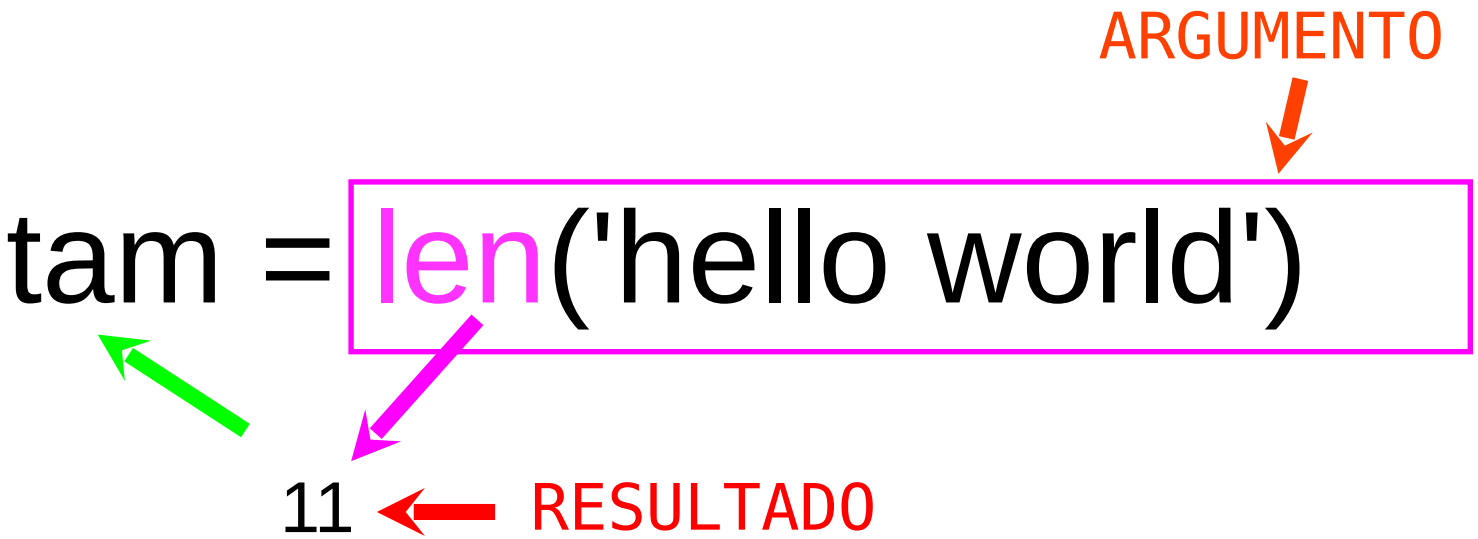
Aritméticos
Relacionales
Lógicas

Funciones predeterminadas

Funciones built-in de Python

Funciones en Python

- Hay dos tipos de funciones en Python.
  - Funciones internas (built-in) que vienen incluidas en Python - input(), type(), float(), int(), len() ...
  - Funciones que definimos nosotros y luego usamos
- Tratamos los nombres de funciones internas como nuevas palabras reservadas (i.e. evitamos usarlas como nombres de variables)




```
>>> tam = len('hello world')
>>> print(tam)
>>> 11
>>> tiny = min('helloworld')
>>> print(tiny)
>>> d
```