

# Tipos de datos, variables, y operadores básicos

Fundamentos de Programación

FIEC04341

MSc. Marco Calderon A.

## Tipos de datos primitivos.

# Tipo de Dato

- Un tipo de dato es un atributo de los datos que indica al ordenador sobre la clase de datos que se va a trabajar

# 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 de caracteres (tipo de dato estructurado).

# DATOS NUMÉRICOS

Tipo	Nombre	Descripción	Ejemplo
<i>Enteros</i>	<i>int</i>	<i>Números sin parte fraccionaria</i>	<i>52 0 -318</i>
<i>Reales o de punto flotante</i>	<i>float</i>	<i>Números con parte fraccionaria o expresados en notación de potencias de 10</i>	<i>6.37 -0.089 4.1e-3</i>
<i>Complejos</i>	<i>complex</i>	<i>Números con un componente real y uno imaginario</i>	<i>(9-3j) (2.5+6.4j)</i>

# DATOS LÓGICOS

Tipo	Nombre	Descripción	Ejemplo
<i>Booleano</i>	<i>bool</i>	<i>Representación de los valores lógicos Verdadero o Falso.</i>	<i>TRUE FALSE</i>

# CADENAS DE CARACTERES

Tipo	Nombre	Descripción	Ejemplo
<i>Cadenas</i>	<i>str</i>	<i>Expresiones (texto) formadas por caracteres. Se pueden representar</i>	<i>‘Hola’ “Mundo”</i>

# Definición y asignación de variables.



# VARIABLES

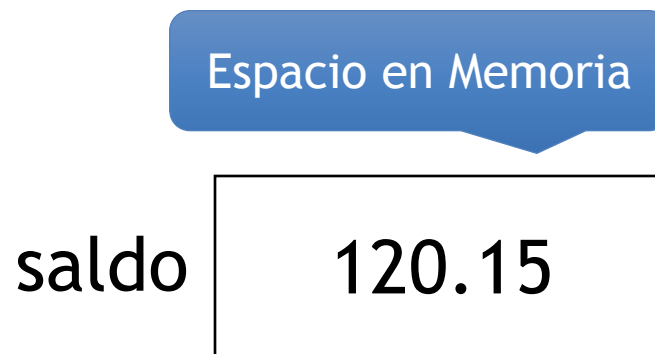
- Una **variable** es una referencia a una dirección en memoria RAM, cuyo valor puede cambiar durante un cálculo o en la resolución de un problema.
- A través de las variables se puede almacenar, organizar y manipular la información en la memoria (RAM).

Espacio en Memoria

120.15

# NOMBRES DE VARIABLES

- Para trabajar con variables hay que asignarles un nombre, que en Python debe seguir ciertas reglas:
  - Sólo puede contener números, letras o el carácter `_`
  - No puede iniciar con un número.
  - No debe coincidir con una palabra reservada del lenguaje.



## BUENAS PRÁCTICAS PARA NOMBRES DE VARIABLES

- Elegir un nombre significativo que tenga relación con el dato que representará.
- Se debe mantener consistencia en el estilo a utilizar en nombres que contengan más de una palabra, por ejemplo:

*fecha\_actual* o *fechaActual*

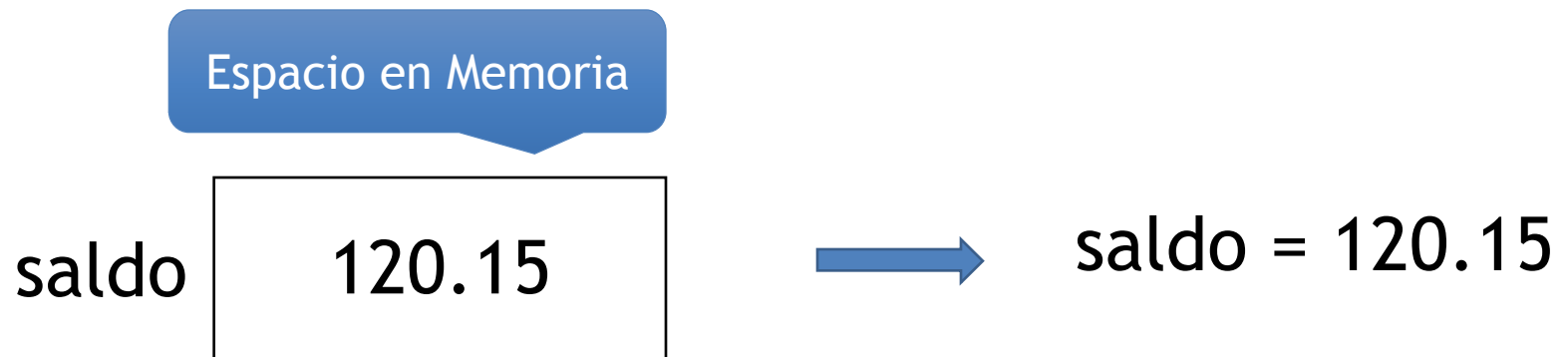
- Seguir las tradiciones de lenguaje, por ejemplo que el nombre de la variable inicie con una letra minúscula.
- No elegir nombres demasiado largos que podrían ocasionar problemas. Mantener un máximo de 15 caracteres.

## BUENAS PRÁCTICAS PARA NOMBRES DE VARIABLES

Incorrecto	Correcto
variable	edad
A B	deposito retiro
1numero 2numero	numero1 numero2
caso-1 caso-2	caso_1 caso_2
<b>input</b>	entrada

# OPERACIÓN DE ASIGNACIÓN

- Esta operación se utiliza para definir variables y asignar un valor a su contenido.
- Se efectúa de derecha a izquierda. Si hay operaciones, éstas se calculan, luego se asigna el resultado a la variable.
- Cualquier valor que haya tenido la variable antes de la asignación, se pierde y es sobrescrito con el nuevo valor.



# OTRAS ASIGNACIONES

- Asignación en la misma línea:

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

- Asignación múltiple:

base, altura = 5,7

→ base = 5 y altura = 7

- Asignación del mismo valor:

base = altura = 2.5  
2.5

→ base = 2.5 y altura =

- Asignación de intercambio:

base, altura = altura, base

→ base contendrá  
el valor de altura y viceversa

## Manejo de entrada y salida.

# Salida de Datos

- La instrucción para la salida de datos es *print*, la cual puede recibir cadenas de caracteres o variables, según lo que se desea mostrar por pantalla:

```
print("Hola Mundo")
```

```
suma = 20  
print(suma)
```



# Secuencias de Escape

- Se utilizan para presentar por la pantalla caracteres especiales.

Secuencia	Acción
\\	<i>Muestra el caracter backslash.</i>
\'	<i>Muestra el caracter de comilla simple.</i>
\"	<i>Muestra el caracter de comilla doble</i>
\a	<i>Sonido de alerta.</i>
\n	<i>Nueva línea. Coloca el cursor al inicio de la</i>
\t	<i>Tabulación. Mueve el cursor avanzando en la misma</i>

# Formatos de Salida

- Mostrar comillas dentro de un string.
- Escribir múltiples valores en un solo string.
- Escribir un string separado en múltiples líneas.
- Especificar el final de un string con *end*.
- Mostrar un string en múltiples líneas utilizando triple comillas.
- Concatenar strings con el operador +

```

# Game Over - Version 2
# Demonstrates the use of quotes in strings

print("Program 'Game Over' 2.0")

print("Same", "message", "as before")

print("Just",
      "a bit",
      "bigger")

print("Here", end=" ")
print("it is...")

print(
    """
    _____
   /  /  /  /  /  /
  /  /  /  /  /  /
 /  /  /  /  /  /
/  /  /  /  /  /

   \  \  \  \  \  \
  \  \  \  \  \  \
 \  \  \  \  \  \
  \  \  \  \  \  \
   \  \  \  \  \  \

    _____
   /  /  /  /  /
  /  /  /  /  /  /
 /  /  /  /  /  /
/  /  /  /  /  /

   \  \  \  \  \  \
  \  \  \  \  \  \
 \  \  \  \  \  \
  \  \  \  \  \  \
   \  \  \  \  \  \

    """
)

input("\n\nPress the enter key to exit.")

```

# Salida por Pantalla

```
Program 'Game Over' 2.0  
Same message as before  
Just a bit bigger  
Here it is...
```

```
  _ _ _ _ _  
 /  /  /  /  /  
|  |  |  |  |  
|  |  |  |  |  
|  |  |  |  |  
 \  \  \  \  \  
  _ _ _ _ _  
 /  /  /  /  /  
|  |  |  |  |  
|  |  |  |  |  
|  |  |  |  |  
 \  \  \  \  \  
  _ _ _ _ _  
 /  /  /  /  /  
|  |  |  |  |  
|  |  |  |  |  
|  |  |  |  |  
 \  \  \  \  \  
  _ _ _ _ _
```

```
Press the enter key to exit.
```

# Entrada de Datos

- La instrucción para la entrada de datos es *input*. Si se desea mostrar un mensaje al usuario, se envía como una cadena de caracteres.
- Se debe definir una variable que almacenará en la memoria el dato ingresado por el usuario.

nombre = input("Ingrese su nombre")

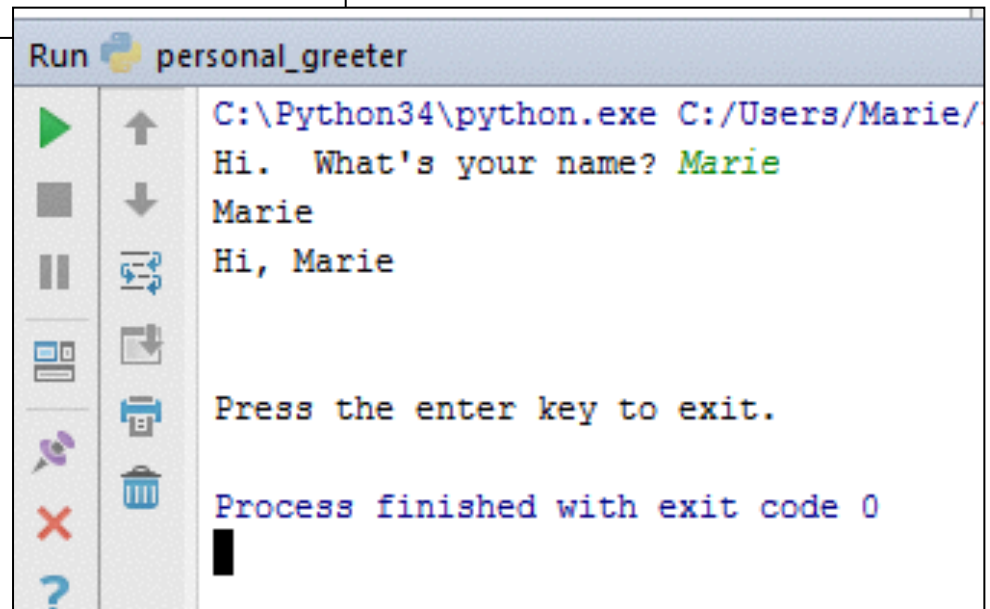
```
# Personal Greeter
# Demonstrates getting user input

name = input("Hi.  What's your name? ")

print(name)

print("Hi,", name)

input("\n\nPress the enter key to exit.")
```



```
Run personal_greeter
C:\Python34\python.exe C:/Users/Marie/
Hi.  What's your name? Marie
Marie
Hi, Marie

Press the enter key to exit.

Process finished with exit code 0
```

# Conversiones entre tipos de datos.

# CONVERSIONES

- Se puede realizar conversiones entre tipos de datos cuando se requiera, siempre que el contenido sea compatible.

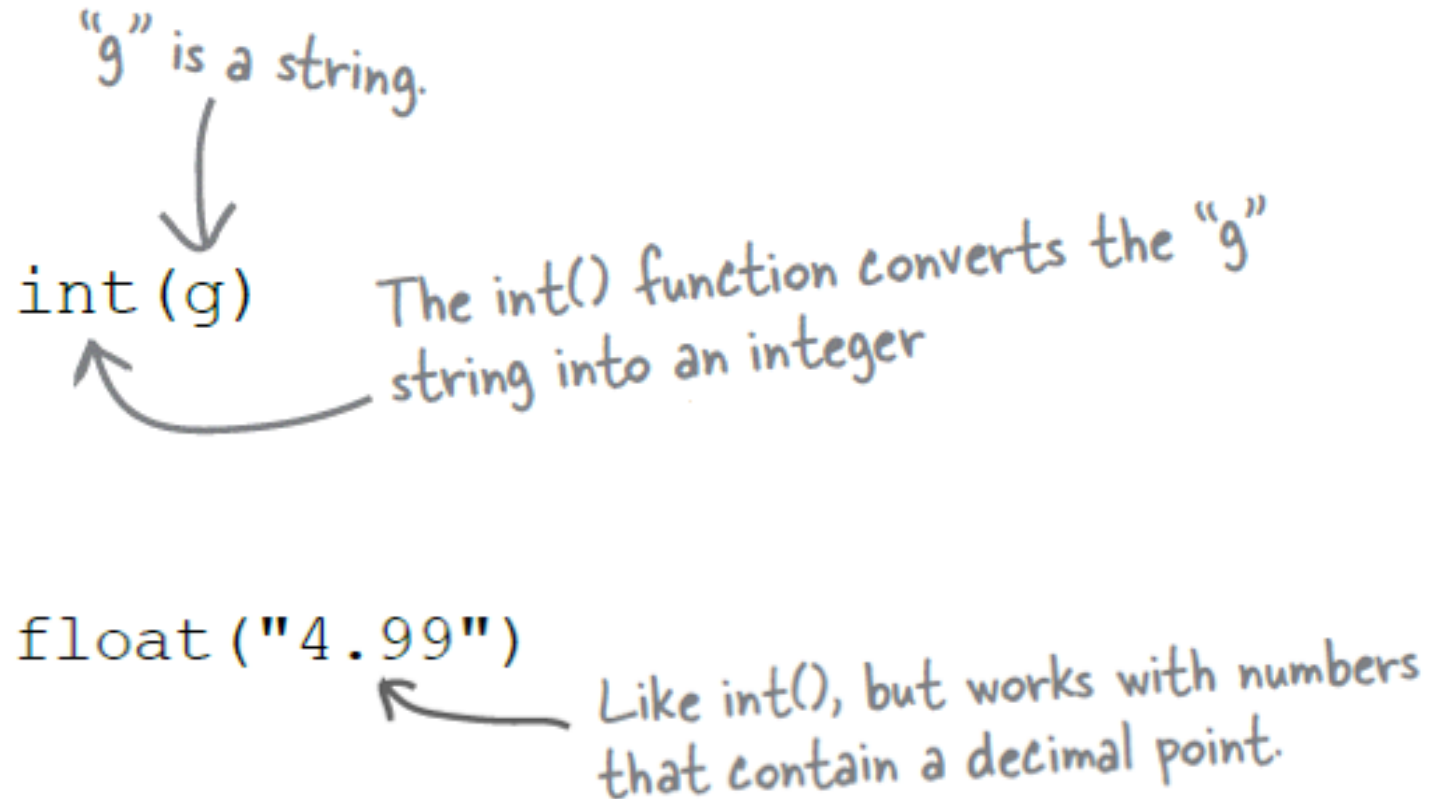


- Para la conversión, se debe preceder el dato con la especificación que corresponda al tipo de dato requerido:

(tipo de dato requerido) dato



## CONVERSIONES ENTRE TIPOS DE DATOS



# EJEMPLOS DE CONVERSIÓN DE TIPOS

<u>Dato</u>	<u>Conversión</u>	<u>Resultado</u>
saldo <input type="text" value="120"/>	(float)saldo	<input type="text" value="120.0"/>
saldo <input type="text" value="120"/>	(str)saldo	<input type="text" value="'120'"/>
saldo <input type="text" value="'120'"/>	(int)saldo	<input type="text" value="120"/>
saldo <input type="text" value="'120.0'"/>	(float)saldo	<input type="text" value="120.0"/>
saldo <input type="text" value="'120.0'"/>	(int)saldo	<input type="text" value="Error"/>
saldo <input type="text" value="'x120'"/>	(int)saldo	<input type="text" value="Error"/>

# Operadores lógicos y relacionales.

# OPERADORES ARITMÉTICOS

- Permiten realizar operaciones aritméticas utilizando directamente símbolos del teclado.

Símbolo	Operación	Ejemplo	Resultado
+	<i>Suma</i>	$2 + 4$	6
-	<i>Resta</i>	$8 - 5$	3
*	<i>Multiplicación</i>	$6 * 2$	12
/	<i>División</i>	$9 / 2$	4.5
//	<i>División</i>	$9 // 2$	4
%	<i>Módulo</i>	$9 \% 2$	1
**	<i>Potenciación</i>	$2 ** 3$	8

# OPERADORES RELACIONALES

- Se utilizan para evaluar condicionales; al operarlos se obtiene como resultado valores booleanos.

Símbolo	Operación	Ejemplo	Resultado
<b>==</b>	<i>Igual que</i>	<b>5 == 5</b>	<b>True</b>
<b>!=</b>	<i>Distinto que</i>	<b>8 != 5</b>	<b>True</b>
<b>&gt;</b>	<i>Mayor que</i>	<b>6 &gt; 9</b>	<b>False</b>
<b>&lt;</b>	<i>Menor que</i>	<b>9 &lt; 2</b>	<b>False</b>
<b>&gt;=</b>	<i>Mayor o igual</i>	<b>7 &gt;= 3</b>	<b>True</b>
<b>&lt;=</b>	<i>Menor o igual</i>	<b>4 &lt;= 2</b>	<b>False</b>

# OPERADORES LÓGICOS

- Permiten construir expresiones lógicas, obteniendo como resultado valores booleanos.

Símbolo	Operación	Ejemplo	Resultado
<i>and</i>	<i>Conjunción</i>	<i>2 &gt; 1 and 4 &lt;</i>	<i>True</i>
<i>or</i>	<i>Disyunción</i>	<i>9 != 6 or 7 &lt;=</i>	<i>True</i>
<i>not</i>	<i>Negación</i>	<i>not True</i>	<i>False</i>

# OPERADORES DE INCREMENTO/DECREMENTO

- Proveen instrucciones de operación aritmética resumida, asignando el resultado a la misma variable.

Símbolo	Ejemplo	Equivalente a
<b><code>+=</code></b>	<b><code>A+=5</code></b>	<b><code>A=A+5</code></b>
<b><code>-=</code></b>	<b><code>A-=5</code></b>	<b><code>A=A-5</code></b>
<b><code>*=</code></b>	<b><code>A*=5</code></b>	<b><code>A=A*5</code></b>
<b><code>/=</code></b>	<b><code>A/=5</code></b>	<b><code>A=A/5</code></b>
<b><code>%=</code></b>	<b><code>A%=5</code></b>	<b><code>A=A%5</code></b>

# EXPRESIONES

- Una expresión es una secuencia de valores unidos por operadores, que al ser evaluada se simplifica en otro valor.
- En las expresiones se utiliza la misma precedencia de operadores que en aritmética.
- Si una expresión contiene operadores de diferente tipo, se evalúan primero las operaciones aritméticas, luego las relacionales, y finalmente las lógicas.
- Se puede utilizar paréntesis para indicar la precedencia de los operadores.

$( (3+4*x) > 10*(y-5) ) \text{ and } ( (a+b)/c \neq 9*(4/a + (9+b)/c ) )$



# EVALUANDO EXPRESIONES

- Número  $x$  entre 0 y 10

$(x \geq 0) \text{ and } (x \leq 10)$

- Número  $x$  fuera del intervalo  $[0, 10]$

$\text{not } ((x \geq 0) \text{ and } (x \leq 10))$

o también

$(x < 0) \text{ or } (x > 10)$

# Aleatoriedad.

# ALEATORIEDAD

- Para introducir en los programas el factor “azar” o “suerte”, podemos utilizar la generación de números aleatorios.
- Python genera números aleatorios basándose en una fórmula (por lo tanto no son realmente aleatorios, pero son suficientes para la mayoría de aplicaciones).
- El módulo **random** es una librería de Python que contiene funciones para generar aleatorios. Para acceder a él se debe cargar al programa con la instrucción **import**.

**import random**

# FUNCIONES ALEATORIAS BÁSICAS

- `random()` genera un número aleatorio entre 0 y 1.
- `randint(a,b)` genera un aleatorio en el rango especificado, incluyendo a y b.
- `randrange(x)` genera un aleatorio entre 0 y x-1

*Ejemplo:* Simular el lanzamiento de un dado.

`dado=randint(1,6)` → valores entre 1 y 6

`dado=randrange(6) +1` → valores entre 0 y 5  
(se “desplaza” en 1)

# BIBLIOGRAFÍA

- Mike Dawson, Python Programming, Third Edition, 2010, CENGAGE Learning.
- Paul Barry, Head First Python, Second edition, 2010, O'Reilly
- Luis Rodríguez, Python Programación

# Próxima sesión

- **ESTRUCTURAS DE CONTROL  
CONDICIONALES E ITERATIVAS**