

#### Estructura de una función

- Toda función comienza con def.
- El encabezado de la función debe terminar con "dos puntos".
- La sangría es obligatoria y define el alcance de la función.

@ Lic. Ricarde Thempsen

### Estructura de una función

- Las funciones deben escribirse <u>antes</u> de ser invocadas.
- El programa principal debe comenzar <u>después</u> de la última función.

#### Estructura de una función

Reglas para los nombres de funciones:

- Sólo se permiten letras, números y el guión bajo.
- No pueden comenzar con un número.
- No pueden coincidir con las palabras reservadas del lenguaje.

@ Lic. Ricardo Thempsen

### Estructura de una función

- El nombre de las funciones <u>debe</u> tener sentido.
- Deben evitarse nombres como "funcion" o "maxi".
- Es recomendable que el nombre de la función incluya un verbo en infinitivo que describa su tarea.

#### Estructura de una función

- La instrucción return termina la ejecución de la función y devuelve un valor a quien la haya llamado.
- Funciones que no retornen valores no necesitan llevar return.

@ Lic. Ricardo Thempsen

## **Importante**

- Cada función debe realizar <u>una sola actividad</u>.
- No deben leerse ni imprimirse valores dentro de una función que realice otra tarea.
- Jamás debe salirse de una función desde el interior de un ciclo.

#### Cadena de documentación

- La cadena de documentación (docstring) no es obligatoria.
- Va encerrada entre tres juegos de comillas (simples o dobles).
- Debe especificar <u>qué</u> hace la función, pero no <u>cómo</u> lo hace.
- Este texto aparece en la consola de Python al escribir help(<nombre>)

@ Lic. Ricarde Thempsen

# **Ejemplo 1**

Desarrollar una función para calcular el factorial de un número entero positivo.

```
def calcularfactorial(n):

""" Devuelve el factorial de un
número entero positivo """

fact = 1
for i in range(1, n+1):
    fact = fact * i
    return fact

a = int(input("Ingrese un número entero: "))
b = calcularfactorial(a)
print("El factorial es", b)
```

### **Variables locales**

- Toda variable creada dentro de una función se considera local, es decir que no existe fuera de ella.
- Las variables locales de una función no pueden ser utilizadas desde otra función, ni desde el programa principal.

#### **Parámetros**

- Los parámetros permiten que la función reciba datos para hacer su trabajo.
- Pueden pasarse 0 o más parámetros.
- Los paréntesis son obligatorios aunque no haya ningún parámetro, tanto en el encabezado como en la llamada de la función.

@ Lic. Ricarde Thempsen

#### **Parámetros**

- Los parámetros <u>formales</u> son los que aparecen en el encabezado de la función, que es la línea que comienza con <u>def</u>.
- Los parámetros <u>reales</u> son los que se escriben en la llamada o invocación.
- Los parámetros formales actúan en representación de los parámetros reales.

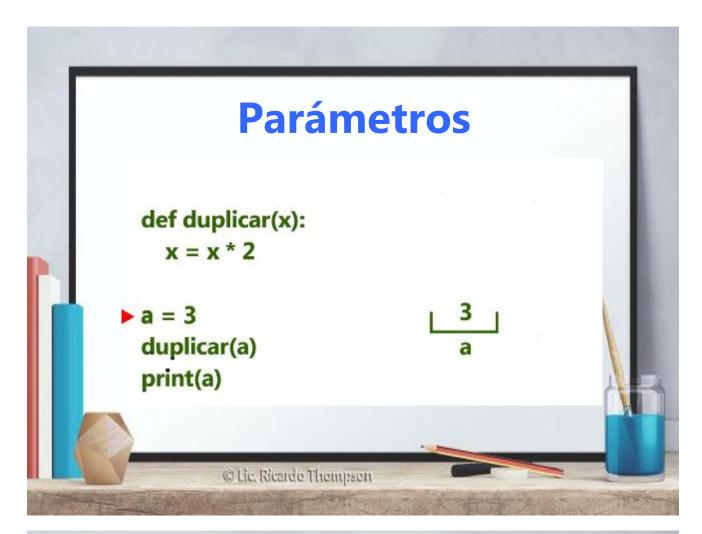
#### **Parámetros**

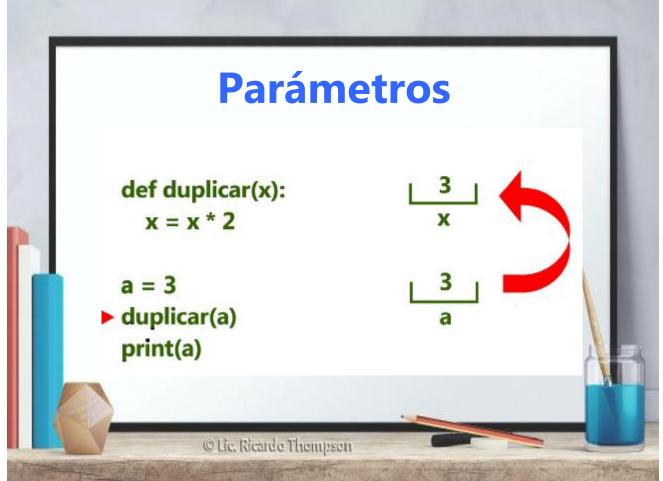
- Los parámetros se clasifican en mutables e inmutables.
- Las variables simples, las cadenas de caracteres y las tuplas son inmutables.
- Las listas, los conjuntos y los diccionarios son mutables.

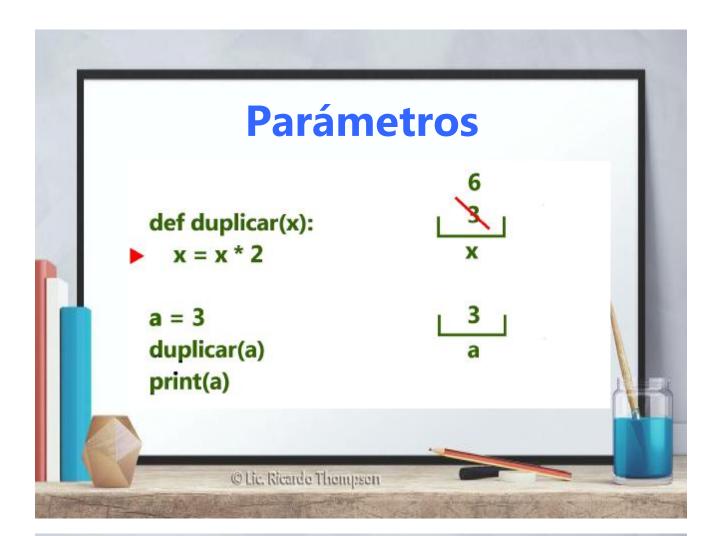
@ Lic. Ricardo Thempsen

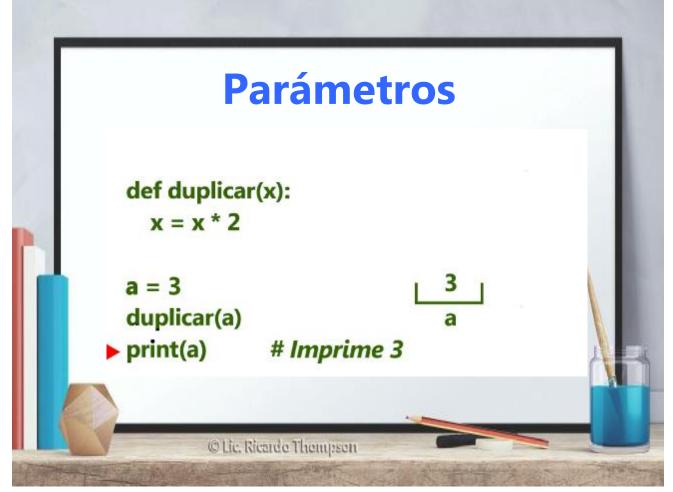
#### **Parámetros**

- Si un parámetro formal <u>mutable</u> es modificado dentro de la función, el cambio afecta al parámetro real correspondiente.
- Si la modificación se hace sobre un parámetro formal <u>inmutable</u>, el parámetro real no resulta afectado.











- Python permite que las funciones devuelvan <u>varios</u> valores.
- Esto evita tener que devolver resultados a través de los parámetros.

@ Lie. Ricarde Thempsen

## Ejemplo 2

Cómo devolver más de un valor de retorno

Escribir una función para ingresar una fecha por teclado.

```
def leerfecha():

""" Lee una fecha por teclado
y devuelve tres enteros """
dia = int(input("Dia?"))
mes = int(input("Mes?"))
año = int(input("Año?"))
return dia, mes, año

d, m, a = leerfecha()
print(d, "/", m, "/", a, sep="")
```

#### **Parámetros**

- Los parámetros pueden tener valores por omisión.
- Ante la ausencia de algún parámetro se utiliza el valor por omisión.
- En la práctica ésto permite invocar a una función con menos parámetros de los previstos.



Uso de parámetros por omisión Escribir una función para calcular la raíz n-ésima de un número.

@ Lie. Ricarde Thempsen

def calcularraiz(radicando, indice=2):
return radicando \*\* (1/indice)

a = float(input("Ingrese el radicando: "))
r2 = calcularraiz(a)
r3 = calcularraiz(a,3)
print("Raiz cuadrada:", r2)
print("Raiz cúbica", r3)



- Los parámetros también pueden pasarse por nombre, en lugar de hacerlo por posición.
- Los parámetros con nombre deben escribirse después de los pasados por posición,

@ Lie. Ricarde Thempsen

# Ejemplo 4

Uso de parámetros con nombre

```
def calcularraiz(radicando, indice=2):
  return radicando ** (1/indice)

r5 = calcularraiz(indice=5, radicando=32)
  print("Raíz quinta:", r5)
  print(calcularraiz(27, indice=3))
```

#### **Funciones Lambda**

- Son funciones pequeñas, anónimas, desechables y de una sola línea.
- Se pueden usar en cualquier contexto donde se admita una función.
- Pueden reemplazarse por funciones normales.



cuadrado = lambda x: x\*\*2
print(cuadrado(3)) # imprime 9

...que es equivalente a:

def cuadrado(x):
 return x\*\*2
print(cuadrado(3))

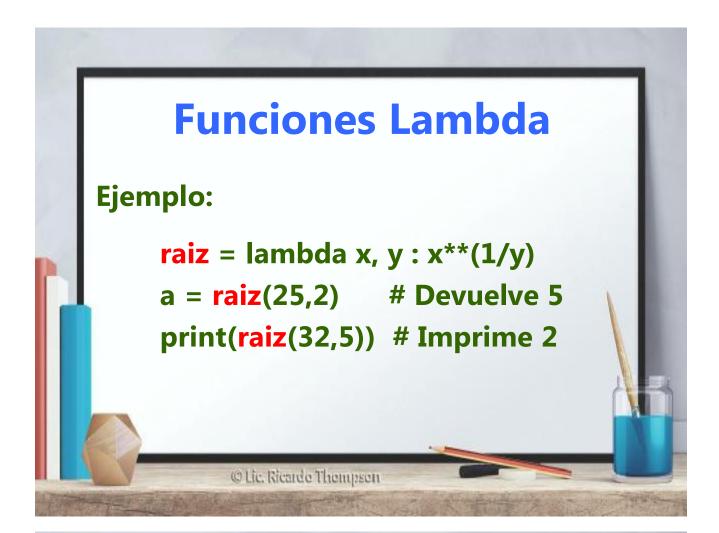
© Lie. Ricarde Thempsen

#### **Funciones Lambda**

#### **Sintaxis:**

<var> = lambda <params> : <valor de retorno>

La variable ubicada a la izquierda del signo igual pasa a comportarse como una función.



### **Funciones Lambda**

También pueden usarse parámetros con valores por omisión:

raiz = lambda x, y=2 : x\*\*(1/y) print(raiz(81)) # Imprime 9

#### **Funciones Lambda**

- Las funciones lambda se agregaron a Python por compatibilidad con el lenguaje Lisp.
- Su utilidad será más evidente al aplicarlas a listas, utilizando las funciones map() y filter().

@ Lie. Ricardo Thempsen

## Módulos y paquetes

- Un módulo es un conjunto de funciones.
- Un paquete es un conjunto de módulos.
- Equivalen a las bibliotecas de otros lenguajes.

## Módulos y paquetes

- La Biblioteca Standard de Python incluye una gran cantidad de módulos para tareas comunes.
- Es necesario importar al módulo deseado antes de poder utilizar sus funciones.

@ Lic. Ricarde Thempsen

## Módulos y paquetes

#### import math

a = math.pi # Constante pi

b = math.log(2) # Logaritmo

c = math.cos(a/2) # Coseno

 El nombre del módulo y el de la función van separados por un punto.

## Módulos y paquetes

- > Además de math, otros módulos de uso frecuente son:
- os: Funciones del sistema operativo.
- random: Números al azar.
- datetime: Manejo de fechas y horas.

@ Lic. Ricardo Thempson

### **Funciones incorporadas**

- Algunas funciones forman parte del intérprete Python, por lo que no es necesario importar ningún módulo.
- abs(): Valor absoluto
- len(): Longitud
- chr(): Carácter Unicode
  - ord(): Valor Unicode de un carácter

