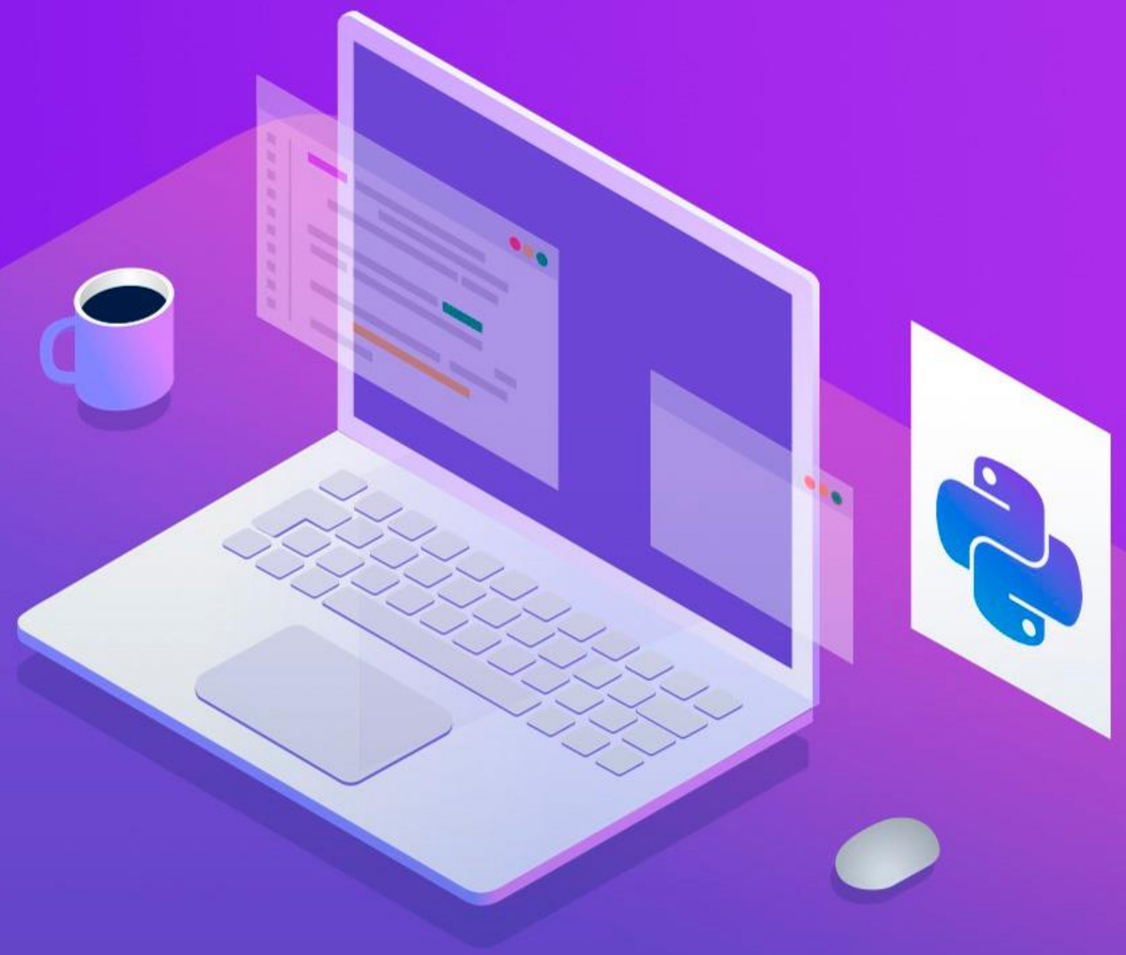
PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE



PROFUNDIZANDO FUNCIONES

Contenidos

- Importancia de las funciones
- La función como una máquina
- Scope
- Paso de parámetros
- Composición de funciones
- Retorno de más de un valor
- Funciones recursivas

Recordemos primero lo esencial

Un segmento de código

Con un nombre que realiza una tarea específica cuando se le llama.

El código puede estar parametrizado

Lo que hace que el resultado de su ejecución no sea la misma si los parámetros cambian.

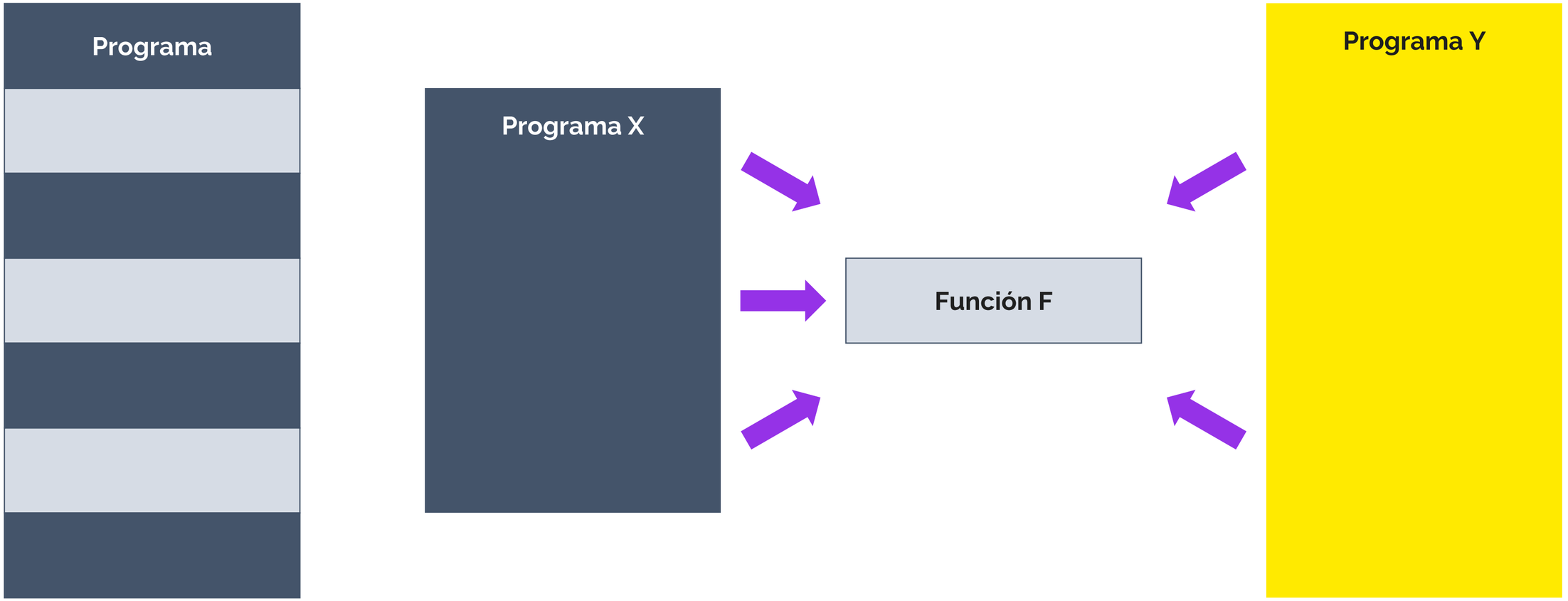
Como resultado de las acciones

El segmento puede retornar un valor.

Por qué son esenciales

1. Permiten escribir código mas fácil de entender.
2. El código es mas fácil de modificar.
3. Evita tener que repetir muchas veces el mismo segmento en el programa.
4. Pueden ser empaquetadas en forma de un módulo o librería que se puede poner a disposición de otros programas.

Gráficamente



Recuerda revisar la Ruta de ejercicios.
Ejercicio EM6-01 →

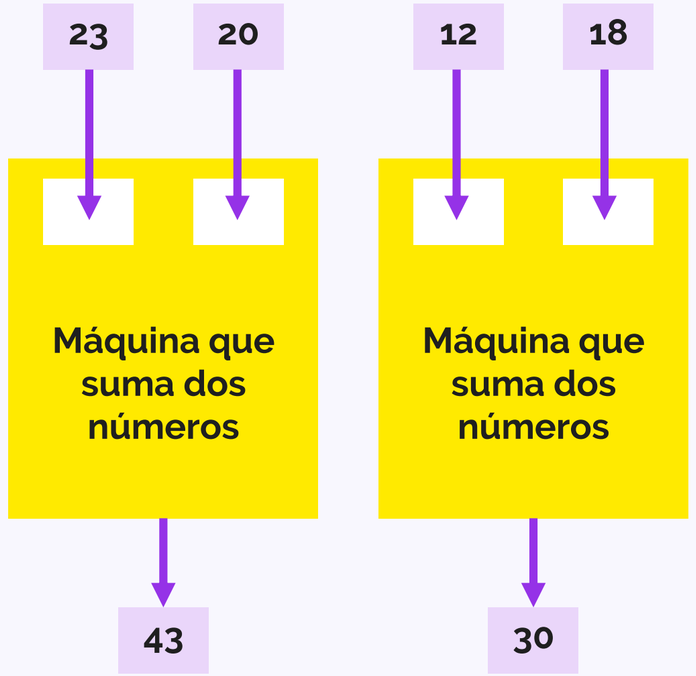
La función como una máquina

- Es llamada a ejecución pasándole una serie de valores en sus parámetros.
- La función utiliza esos valores para hacer su trabajo y devuelve un resultado.

Revisemos el siguiente ejemplo:

CÓDIGO

```
def sumar(a, b):
    return(a + b)
result = sumar(23, 20)
result = sumar(12, 18)
```



Scope

Recuerda revisar la Ruta de ejercicios. **Ejercicio EM6-02, EM6-03 y EM6-04 →**

Revisemos el siguiente ejemplo:

CÓDIGO	RESULTADO

`x = 10`
`def func1(y):`
`  print(y)`
`func1(x)`
`func1(2 * x)`
	`10`
	`20`

CÓDIGO	RESULTADO

`x = 10`
`def func2(y):`
`  print(y)`
`  x = 1`
`func2(x)`
`func2(x)`
	`10`
	`10`

CÓDIGO	RESULTADO

`x = 10`
`def func3(y):`
`  y = y + 1`
`  print(y)`
`func3(x)`
`print(y)`
	`11`
	`NameError: name 'y' is not defined`

Recuerda revisar la Ruta de ejercicios. **Ejercicio EM6-05 →**

Cuidado al pasar una lista

Si se pasa una **lista**, puede ser modificada al interior de la función.

CÓDIGO	RESULTADO

`x = [10, 20, 30]`
`def pick_first(x):`
`  print(x)`
`  x[0] = 0`
 	
`pick_first(x)`
`pick_first(x)`
	`[10, 20, 30]`
	`[ 0, 20, 30]`

Recuerda revisar la Ruta de ejercicios. **Ejercicio EM6-06 →**

Lo mismo con los diccionarios

Si se pasa un **diccionario**, puede ser modificada al interior de la función.

CÓDIGO	RESULTADO

`x = {'a':10, 'b':20, 'c':30}`
`def pick_b(x):`
`  print(x['b'])`
`  x['b'] = 100`
 	
`pick_b(x)`
`pick_b(x)`
	`20`
	`100`

Combinación de funciones

Las funciones que retornan valores pueden ser usadas en composición.

CÓDIGO	RESULTADO

`def  sum3(x, y, z):`
`  w = x + y + z`
`  return w`
`print (sum3(1,3,5))`
`print (sum3(1,2,sum3(3, 4, 5)))`
	`9`
	`15`

Puede retornarse mas de un valor en forma de tupla.

CÓDIGO	RESULTADO

`def first_and_third(x, y, z):`
`  return x, z`
 	
`x, y  =`
`first_and_third('esto','no','funciona')`
`print(x + " " + y)`
	`esto funciona`

Recuerda revisar la Ruta de ejercicios. **Ejercicio EM6-07 y EM6-08 →**

Recuerda revisar la Ruta de ejercicios. **Ejercicio EM6-09 →**

Uso de nombres en los parámetros

Los parámetros pasados pueden ser usando los nombres.

CÓDIGO	RESULTADO

`def func2(x, y, z):`
`  return x + 2 * y + 3 * z`
 	
`value = func2(1, 2, 3)`
`print(value)`
 	
`value = func2(z=3, x=1, y=2)`
`print(value)`
	`14`
	`14`

Recuerda revisar la Ruta de ejercicios. **Ejercicio EM6-10 →**

Parámetros por defecto

Los parámetros pueden tener valores por defecto (si no se especifican).

CÓDIGO

```
def func2(x=0, y=0, z=0):
  return x + 2 * y + 3 * z

value = func2(1, 2, 3)
print(value)

value = func2(1, 2)
print(value)

value = func2(1)
print(value)

value = func2()
print(value)
```

RESULTADO

```
14
5
1
0
```

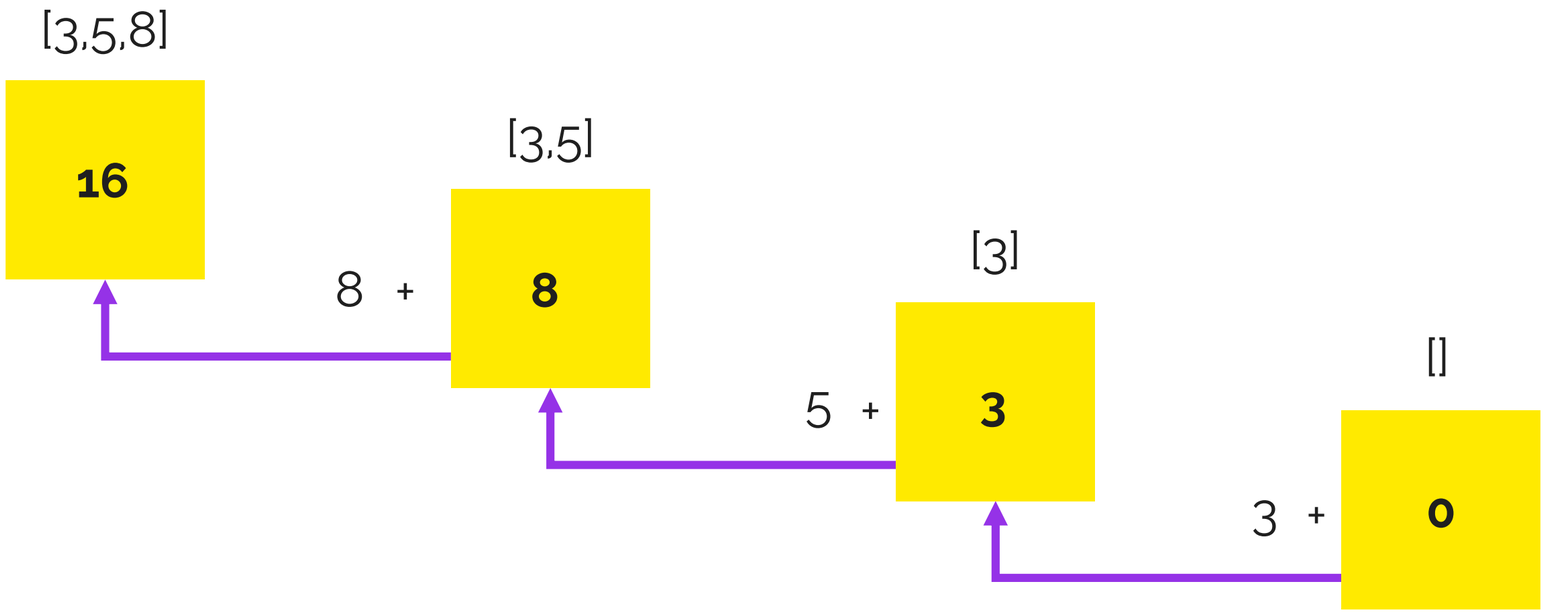
Funciones recursivas

1. Uno de los aspectos que más asombra a los que comienzan a programar.
2. Una función puede llamarse a si misma!
3. Entonces, ¿no terminaría nunca?
4. El truco es que al llamarse siempre lo hace con parámetros que representan una reducción del problema.
5. Cuando los parámetros son suficientemente pequeños, se detiene el proceso.

Ejemplo: suma de los valores de una lista

Si la lista es vacía, el resultado es cero.

Si la lista tiene n valores, la suma es igual al último en la lista + la suma de la lista sin él.



En código

Recuerda revisar la Ruta de ejercicios. **Ejercicio EM6-11 →**

CÓDIGO

```
def suma(x):
  l = len(x)
  if l == 0:
    return 0
  else:
    last = x.pop()
    print(last, x)
    return suma(x) + last

print(suma([7, 4, 23]))
print ('=' * 15)
print(suma([1,2,3,4,5]))
```

RESULTADO

```
23 [7, 4]
4 [7]
7 []
34
===============
5 [1, 2, 3, 4]
4 [1, 2, 3]
3 [1, 2]
2 [1]
1 []
15
```

Resumen

- Las funciones son una parte esencial en el desarrollo de cualquier pieza de software.
- Representan un trozo de funcionalidad independiente que puede ser parametrizada.
- Los parámetros de entrada se usan en el cálculo y se retorna un valor.
- Si las funciones retornan valores, pueden ser aplicadas unas sobre otras.
- Si lo que se pasa es una lista o diccionario, puede ser modificado al interior de una función.
- Una función puede llamarse a sí misma y se denomina función recursiva.

Referencias bibliográficas

- Gaddis, T. (2017). Starting Out with Python, 4th Ed. & Tearson.
- Matthes, E. (2015). Python Crash Course: A Hands-On, Project-Based Introduction to Programming. No Starch Press.

Cierre

Has finalizado la revisión de los contenidos de esta clase.

A continuación, te invitamos a realizar las actividades y a revisar los recursos del módulo que encontrarás en plataforma.