

Capítulo 11

Funciones

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Function Basics

- A function is a “wrapper” of statements performing some actions
- Every function has a name, receive some arguments used to make some transformations and return some values

```
def function_name (arg1, arg2, arg3) :
```

```
...
```

```
...
```

The logo for Cartagena99, featuring the text "Cartagena99" in a stylized font with a blue and orange gradient background.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Functions

- Python provide us with many functions to use directly. We have used some of these functions such as ***print()***, ***input()***, ***len()***, ***pow()***, etc.
- Additionally, the users can create their own functions.
- Python also permit us to collect a group of functions as a library. Later, we can use this library in our programs (importing these functions to our programs).
- The idea of using functions enable us to use the method of modular programming, where every module is responsible for a certain task.

The logo for Cartagena99 features the text "Cartagena99" in a stylized, dark green font. The "99" is significantly larger and more prominent than the word "Cartagena". The text is set against a light blue and white background with a subtle wave-like pattern. Below the text, there is a horizontal orange and yellow gradient bar.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Functions

- In order to follow modular programming we will divide a complex project into simpler tasks, every simple task can be codified as an independent block (function).
- So generally, in modular programming, the program is made of blocks of code called functions. Every block or function is responsible for a specific task.
- Any name but Python keywords
- Preserve indentation inside the function definition
- **Advantages of using functions:**
 - To implement new functionalities
 - To avoid repeating code
 - To have a better structure

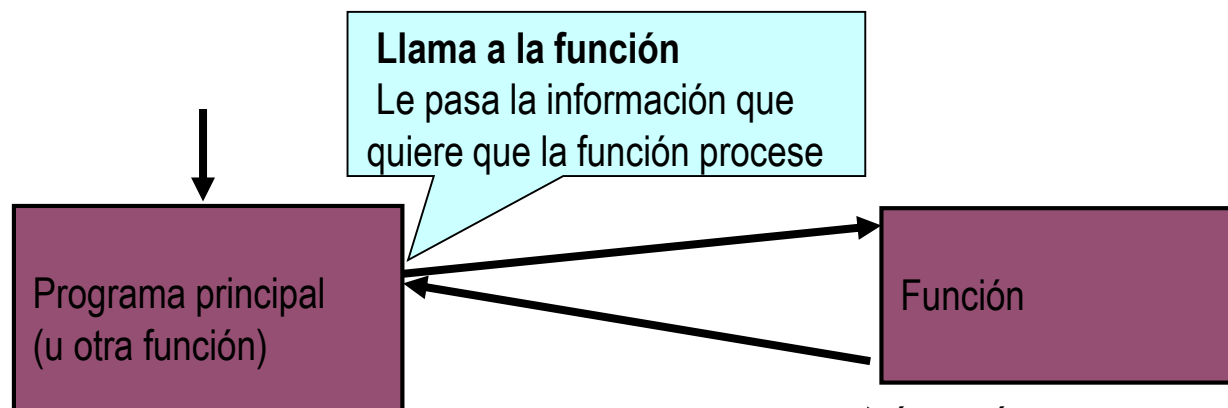
Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Functions

- **What is a function**: A function can be defined as a group of instructions (codes) that can do a specific task.
- Once the functions are written we can use them when necessary by invoking them. To activate the function we call the function.
- Every time a function is called the control of execution of the program will be transferred to that function. When the function is finished, the control of the execution goes back to the point where we called.
- We can call a function as many times as we want without limits. Even a function can call another function. By doing so, we can create very complex programs.



Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

llamada

Funciones

- Tipos
 - Predefinidas (built-in-functions.): abs, pow, round, etc.

```
In [9]:  
1 res=abs(-3)  
2 print(res)  
3  
In [10]:  
1 res=pow(2, 3)  
2 print(res)  
8  
In [7]:
```



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Funciones

- Tipos funciones

- Incorporadas en módulos. Deben importarse antes de usarse. Ejemplo: sin, cos

```
1 import math
2 seno_pi=math.sin(90*2*math.pi/360)
3 print(seno_pi)
```

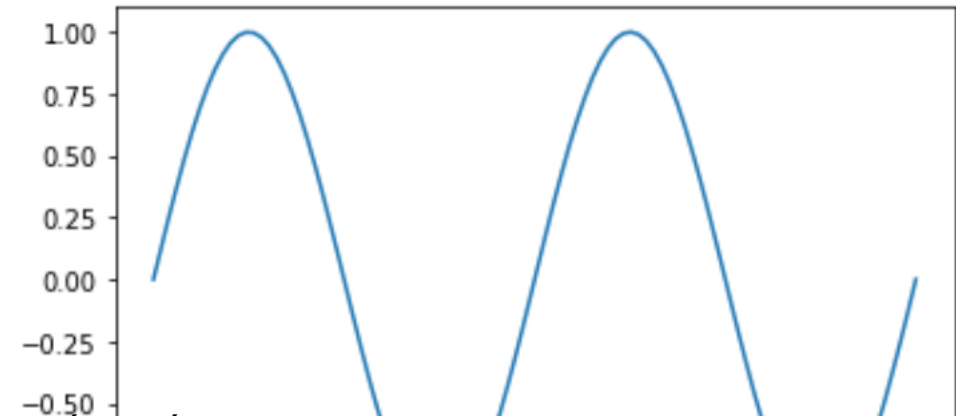
1.0

```
1 import numpy as np
2 notas=[4,5,6]
3 media=np.mean(notas)
4 print(media)
```

5.0

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 x=np.linspace(-2*np.pi,2*np.pi,100)
4 y=np.sin(x)
5 plt.plot(x, y)
```

[<matplotlib.lines.Line2D at 0x209f4b9dcf8>]



Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

1.0

Programmer-defined functions

- Tipos

- Creadas por el programador. Permite dividir los programas en un conjunto de pequeños componentes (autocontenidos), cada uno de los cuales con un propósito

```
Head { def nombre_funcion(param1, param2, ..., param_n):  
      { instruction_1  
        { instrucción_2  
          { ...  
            { instruction_n  
              { return val_1, val_2, ...val_n
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

separados por comas. Si la función devuelve uno o más valores, la llamada a la función se asigna a una o más variables. Si

la función no devuelve nada la llamada aparece sola.

Programmer-defined functions

```
def function_name(param1, param2, ...,param_n):  
    instruction_1  
    instrucción_2  
    ...  
    instruction_n  
    return val1, val2, ...valn
```

- `function_name`
 - Name of the function that is used to call the function anywhere in the program
 - Same limitations as any identifier name
- parameters (`parameter_1, parameter_2, ...parameter_n`)
 - They represent the names of data items that are transferred into the function from the calling portion of the program
 - They are also known as **formal parameters**
- An empty pair of parentheses **MUST** follow the function name if the function definition does not include any arguments
- `:` (Colon)
 - It marks the beginning of the body of the function (compound statement that defines the action to be taken by the function)

The logo for Cartagena99 features the text "Cartagena99" in a stylized, bold font. The "C" is large and green, while "artagena99" is in a smaller, dark green font. The text is set against a light blue and orange gradient background.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Functions

- Function calls:

- To call a function write the name of the function and a pair of parenthesis, inside the parenthesis you must indicate the parameters (variables) that the function needs separated by commas .
- If the function call does not require any arguments, an empty pair of parentheses must follow the name of the function
- The arguments appearing in the function call are referred to as **actual parameters** or arguments, in contrast to the **formal parameters** that appear in the first line of the function definition. The number of actual arguments must be the same as the number of formal arguments
- If the function returns one or more values, the function access is often written as an assignment statement; e.g.,

$y = \text{polynomial}(x)$

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Programmer-defined functions

- **return** var_1, var_2, ..., var_n
 - Last instruction in the body of the function
 - Causes an immediate exit from the function and returns the value of variables or expressions to the calling portion of the program
 - If the don't return any value, the **return** statement simply causes control to revert back to the calling portion of the program. Only in this case the **return** statement can be omitted, and the function will terminate on reaching the last sentence

```
def function_name(param1, param2, ...,param_n):  
    instruction_1  
    instrucción_2  
    ...  
    instruction_n  
    return val1, val2, ...valn
```

Function returning no values

```
In [29]: def func1(arg):  
         print("Just printing the argument:",arg)  
  
         out_func1=func1(100)  
         print(out_func1)
```

Function returning a value

```
In [31]: def func2(arg):  
         out= arg*2  
         return out  
  
         out_func2=func2(100)
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Formal and Actual Parameters

- Formal parameters:
 - Variables that receive the address of the arguments used in the function call
 - Variables that appear in the first line of the function definition
 - They are local variables:
 - Recognized only within the function
- Actual parameters:
 - Those variables used to refer to the function
 - Variables that appear in the function call

```
def maximun(a, b):  
    if a > b:  
        result = a  
    else:
```

Formal parameters

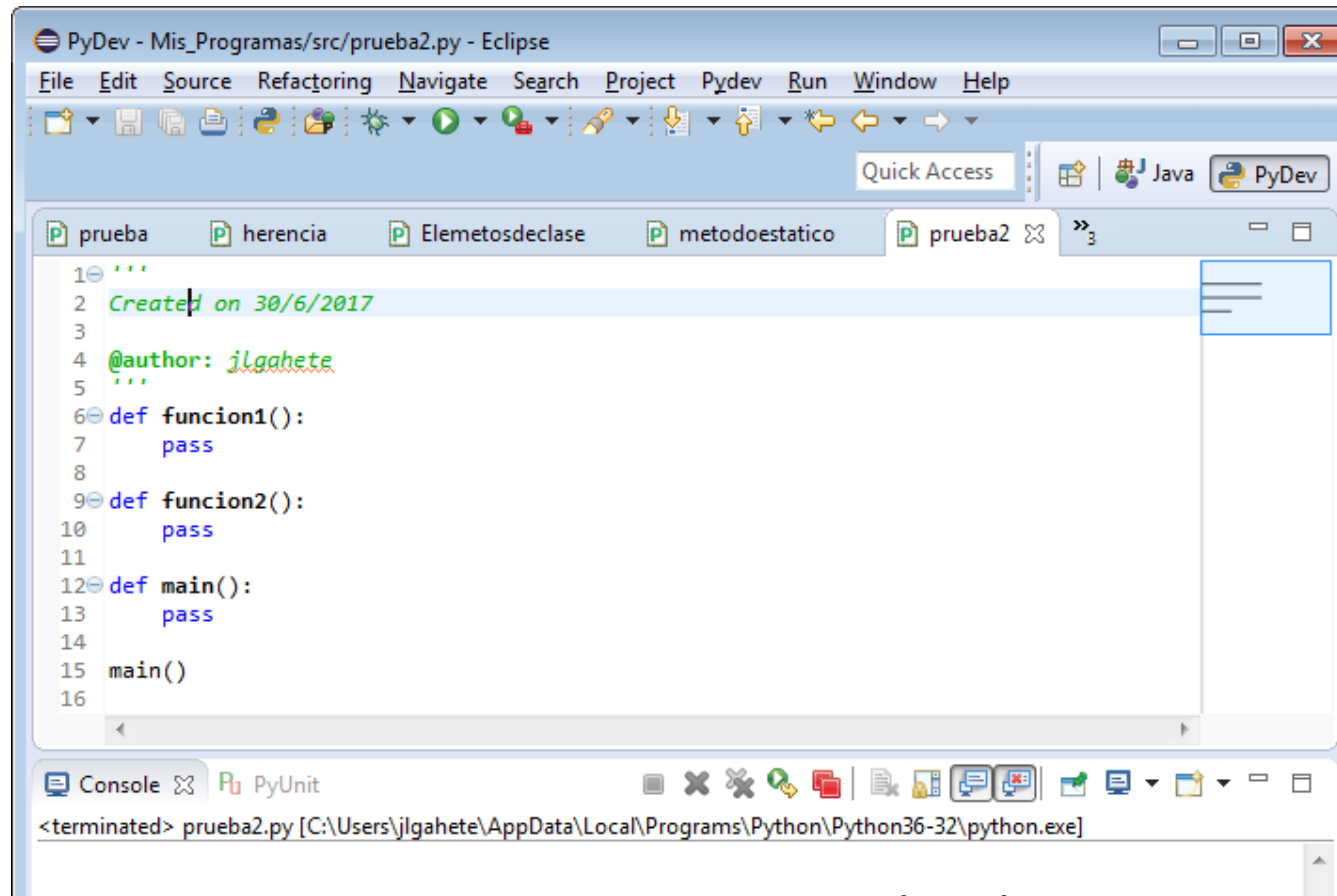
Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

```
col = maximun(1, num2)
```

Organización de las funciones en el programa



```
PyDev - Mis_Programas/src/prueba2.py - Eclipse
File Edit Source Refactoring Navigate Search Project Pydev Run Window Help
Quick Access Java PyDev
prueba herencia Elementosdeclase metodoestatico prueba2
1 '''
2 Created on 30/6/2017
3
4 @author: jlgahete
5 '''
6 def funcion1():
7     pass
8
9 def funcion2():
10    pass
11
12 def main():
13    pass
14
15    main()
16
Console PyUnit
<terminated> prueba2.py [C:\Users\jlgahete\AppData\Local\Programs\Python\Python36-32\python.exe]
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Writable Insert 2:7

Returning values

- The most common structure is returning some variable created inside

```
In [47]: ▶ import math
def area(radius):
    temp = math.pi * radius**2
    return temp
```

```
print("The area of circle with radius 2 is: ", area(2))
print("The area of circle with radius 3 is: ", area(3))
print("The area of circle with radius 4 is: ", area(4))
print("The area of circle with radius 5 is: ", area(5))
```

```
The area of circle with radius 2 is: 12.566370614359172
The area of circle with radius 3 is: 28.274333882308138
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Using that functions
make the code more
readable and reduces
repeated statements

Cartagena99

Returning values

- Temporary variables sometimes can be skipped
- But using variables makes debugging easier

```
In [48]: ▶ import math
def area(radius):
    return math.pi * radius**2

print("The area of circle with radius 2 is: ",area(2))
print("The area of circle with radius 3 is: ",area(3))
print("The area of circle with radius 4 is: ",area(4))
print("The area of circle with radius 5 is: ",area(5))
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Returning values

- More than one return statement is allowed

```
In [49]: ▶ import math
def area(radius):
    if radius < 0:
        return -1
    elif radius == 0:
        return 0
    else :
        return math.pi * radius**2

print("The area of circle with radius -3 is: ",area(-3))
print("The area of circle with radius 0 is: ",area(0))
print("The area of circle with radius 1 is: ",area(1))
print("The area of circle with radius 2 is: ",area(2))
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Dead code

- Note that all the code after a return statements will never be executed

```
In [50]: import math
def area(radius):
    temp = math.pi * radius**2
    return temp
    perimter = 2 * math.pi * radius
    print("The area of a circle with radius {} is: {}".format(radius,temp))
    print("The perimter of a circle with radius {} is: {}".format(radius,perimter))

print("The area of circle with radius 2 is: ",area(2))
print("The area of circle with radius 3 is: ",area(3))
print("The area of circle with radius 4 is: ",area(4))
print("The area of circle with radius 5 is: ",area(5))
```

This is dead code

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Composition

- Functions can be called from within another
- A function can call itself

```
1 from datetime import datetime
2 def addTimeStamp():
3     now = datetime.now()
4     return "timestamp: " +str(now)
5
6 def checkIn(arg):
7     log = arg + " is checking on "+addTimeStamp()
8     return log
9
10 def checkout(arg):
11     log = arg + " is checking on "+addTimeStamp()
12     return log
13
14 def cardReader(name,action):
15     if action=="IN":
16         log=checkIn(name)|
17     else:
18         log=checkOut(name)
19     return log
```

```
1 log=cardReader("José Luis","IN")
2 print(log)
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

TIPOS DE ARGUMENTOS EN LLAMADAS A FUNCIONES: POSICIONALES Y NOMBRADOS

- Hasta el momento, deben coincidir en número los parámetros actuales (reales o ficticios) con el número de parámetros formales
- Orden:
 1. Posicional: Se respeta el orden de los parámetros formales y se asignan los valores de los parámetros actuales en su correspondiente formal
 2. Nombrados: No se respeta el orden de los parámetros formales. Se debe usar en los argumentos el formato nombre_param_formal = valor
- Se pueden mezclar, pero si se utiliza un argumento nombrado ya no puede aparecer después uno posicional

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Passing arguments

- Arguments are not mandatory

```
In [5]: 1 from datetime import datetime
        2 def addTimeStamp():
        3     now = datetime.now()
        4     return "timestamp: |" +str(now)
        5 print(addTimeStamp())
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Passing arguments

- Normally they have one or several

```
1 import math
2 def areaCircle(radius):
3     area = math.pi * radius**2
4     return area
5
6 print(areaCircle(5))
```

78.53981633974483

```
1 def concat(string1,string2):
2     newstring= string1+" "+string2
3     return newstring
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Passing arguments

- We can assign default values

```
1 from datetime import datetime
2 import pytz
3
4 def currentTimeStamp(timezone='Europe/Madrid'):
5     tz = pytz.timezone(timezone)
6     now = datetime.now(tz)
7     return "Timestamp in {}: {}".format(timezone,now)
```

```
1 print(currentTimeStamp())
```

Timestamp in Europe/Madrid: 2020-04-01 11:37:45.232613+02:00

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Passing arguments

- The arguments order is relevant

Mandatory

Optional

Optional

```
In [48]: ▶ def registerUser(name, city="Madrid", job="Student"):
out="User {} from city: {} with job: {} created".format(name,city,job)
return out
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Passing arguments

- How can we sort this issue out?

Mandatory

Optional

Optional

```
In [48]: ▶ def registerUser(name, city="Madrid", job="Student"):  
          out="User {} from city: {} with job: {} created".format(name,city,job)  
          return out
```

```
1 def registerUser(name, city="Madrid", job="Student"):  
2   out="User {} from city: {} with job: {} created".format(name,city,job)  
3   return out
```

```
1 print(registerUser("José Luis"))
```

User José Luis from city: Madrid with job: Student created

```
1 print(registerUser("José Luis", "Córdoba"))
```

User José Luis from city: Córdoba with job: Student created

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

```
1 print(registerUser("José Luis", "Córdoba", "PND"))
```


Passing arguments

- Where there exists ambiguity, use explicit naming for the arguments

```
1 print(registerUser("José Luis", job="Phd"))
```

User José Luis from city: Madrid with **job: Phd** created

- Note that Python is not TYPESAFE, so the type does not help with disambiguation

```
In [65]: 1 def registerUser(name, city="Madrid", job="Student"):
2         out="User {} from city: {} with job: {} created".format(name,city,job)
3         return out
```

```
In [27]: 1 print(registerUser("José Luis"))
```

User José Luis from city: Madrid with job: Student created

```
In [28]: 1 print(registerUser("José José Luis", 100))
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

Passing arbitrary arguments

- Sometimes the number of arguments is variable

Arbitrary arguments

As tuples

```
1 def newUsers(*args):  
2     print("Arguments type:", type(args))  
3     print("Arguments: ", args)
```

```
1 newUsers()
```

Arguments type: <class 'tuple'>

Arguments: ()

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Passing arbitrary arguments

- More flexibility can be achieved using dictionaries

As dictionaries

```
1 def newParkingPlaces(**args):  
2     print("Arguments type:", type(args))  
3     print("Arguments: ", args)
```

```
1 newParkingPlaces()
```

```
Arguments type: <class 'dict'>  
Arguments: {}
```

```
1 newParkingPlaces(engineering=5, it=10, sales=20, nrbh=5)
```

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Passing arbitrary arguments

- Or combine both solutions

```
def newBulding(city, *users, **parking):  
    print("City variable type:", type(city))  
    print("City: ", city)  
    print("Users variable type:", type(users))  
    print("Users: ", users)  
    print("Parking variable type:", type(parking))  
    print("Parking: ", parking)
```

```
newBulding("Madrid", "Mia", "Brian", "Roman", "Elena", engineering=5, it=10, sales=20, rrhh=5)
```

```
City variable type: <class 'str'>
```

```
City: Madrid
```

```
Users variable type: <class 'tuple'>
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

Ejercicios

- 1.- Programa que pide 3 números, num1, num2 y num3 llama a la función calculos() que devuelve el mayor, el menor y la media de los tres números y los muestra por pantalla en el programa principal.
- 2.- Programa que pide un nombre y llama a una función que muestra un mensaje de bienvenida particularizado para esa persona
- 3.- Programa que pide un número num1 y llama a la función factorial() que devuelve el factorial del número y lo muestra por pantalla en el programa principal.
- 4.- Programa que calcula el combinatorio de m sobre n.
- 5.- Programa que pide un número y llama a la función esPrimo() que devuelve verdadero si el número es primo y falso en caso contrario
- 6 - Programa que pide dos números y muestra por pantalla todos los números

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Ámbito (scope) de la variables

- El ámbito de una variable es la zona de programa donde puede ser utilizada.
 - Variable local: variable definida dentro de la función (incluido el main) y que tiene como ámbito de actuación (utilización) solo la propia función.
 - Variable global: variable definida fuera de todas las funciones, al principio del programa. Su ámbito de utilización es todo el programa, por lo que puede ser usada en cualquier función

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Ámbito (scope) de la variables

- Diferentes niveles: el más alto es el programa principal, el siguiente nivel son las funciones incluidas en el programa principal y cada vez que hay una función incluida dentro de otra estaríamos bajando un nivel.
- Principios:
 - Cada variable pertenece a un ámbito determinado: programa principal o función
 - Las variables son completamente inaccesibles en los ámbitos superiores al ámbito al que pertenecen
 - las variables pueden ser accesibles o no en ámbitos inferiores al ámbito al que pertenecen

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Functions Scope

- All the values defined inside the function's body are local. You cannot use them outside

This is the scope of the function

```
def concat(string1,string2):  
    newstring= string1+" "+string2  
    return newstring
```

```
begining="This is the begining part"  
ending="and this is the ending part"  
sentence = concat(begining, ending)  
print(sentence)
```

```
This is the begining part and this is the ending part
```

```
print(newstring)
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

NameError: name 'newstring' is not defined

Functions Scope

```
from datetime import datetime
def addTimeStamp():
    now = datetime.now()
    return "timestamp: " +str(now)

def checkIn(arg):
    log = arg + " is checking on "+addTimeStamp()
    return log

def checkout(arg):
    log = arg + " is checking on "+addTimeStamp()
    return log

def cardReader(name,action):
    if action=="IN":
        log=checkIn(name)
    else:
        log=checkOut(name)
    return log
```

Cannot concatenate a datetime type with an string type without conversion

We can use the same variable name in different functions because scopes are different

An outer function can invoke an inner function that uses the same variable name

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Functions Scope

```
1 def Validation(x,y):#x and y are local variables
2     if y>0:
3         res=x//y #res is local variable
4         print ("result= ", res)
5     else:
6         print ("Error...")
7     return
8 a=7
9 b=3
10 Validation(a,b)
11 a=14
12 b=0
13 Validation(a,b)
14 print(res)
15 print(x)
```

result= 2

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

`>python input_4_4c20d015b2f5c_in_2modules`

Functions Scope

```
1 def ejemplo():  
2     a = 3  
3     print(a)  
4     return  
5  
6 a = 99  
7 ejemplo()  
8 print(a)
```

3
99

```
1 def ejemplo():  
2     a = 3  
3     print(a)  
4     return  
5  
6 ejemplo()  
7 print(a)
```

3

NameError

<ipython-input-1-09b8c

5

```
1 def ejemplo():  
2     print(a)  
3     a = 3  
4     return  
5  
6 ejemplo()  
7 print(a)
```

UnboundLocalError

<ipython-input-2-88f9

4 return

5

----> 6 ejemplo()

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Argumentos y devolución de valores

- En la mayoría de los lenguajes de programación, en los que las variables son contenedores donde se guardan valores, cuando se envía una variable como argumento en una llamada a una función suelen existir dos posibilidades:
 1. Paso por valor o copia: se envía simplemente el valor de la variable, en cuyo caso la función no puede modificar la variable origen, pues la función sólo conoce una copia de su valor, pero no la variable que lo almacenaba.
 2. Paso por referencia: se envía la dirección de memoria de la variable, en cuyo caso la función sí que puede modificar la variable, normalmente a través de punteros.
- En Python no se hace ni una cosa ni otra. En Python cuando se envía una variable como argumento en una llamada a una función lo que se envía es la referencia al

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Functions Scope

x is immutable
LISTA mutable

```
1 def example_passing(x,y):
2     x=x**3
3     y.append(10)
4     print("\nValor de x en la función: ", x)
5     print("Valor de y en la función: ", y)
6     return
7 x=2
8 y=[2,4,6,8]
9 print("Valor de x en el programa: ", x)
10 print("Valor de y en el programa: ", y)
11 example_passing(x,y)
12 print("\nValor de x en el programa despues de llamar función: ", x)
13 print("Valor de y en el programa despues de llamar función: ", y)
```

Valor de x en el programa: 2

Valor de y en el programa: [2, 4, 6, 8]

Valor de x en la función: 8

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

valor de y en el programa despues de llamar función: [2, 4, 6, 8, 10]

Functions Scope

DICCIONARIO mutable

```
def passing_dic(student):  
    student['telephone']=111111  
    print("\nMuestro el diccionario MODIFICADO en la función ")  
    print(student)  
    return
```

```
student_data={}  
str=input("Introduce student name:")  
id=int(input("Introduce student id:"))  
student_data['name']=str  
student_data['ID']=id  
print("\nMuestro el diccionario en el programa principal")  
print(student_data)  
passing_dic(student_data)  
print("\nMuestro el diccionario en el programa principal")  
print(student_data)
```

```
Introduce student name:Pepe Luis  
Introduce student id:1234
```

```
Muestro el diccionario en el programa principal  
{'name': 'Pepe Luis', 'ID': 1234}
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

```
Muestro el diccionario en el programa principal
```

```
{'name': 'Pepe Luis', 'ID': 1234, 'telephone': 111111}
```

Anonymous functions

- Are small functions
- They have no name
- Can only have a single expression
- Also known as LAMBDA functions
- Very common on MAP operations

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Anonymous functions

- Note the differences

```
def sum2values(value1,value2):  
    return value1+value2
```

```
sum2values(10,20)
```

30

```
f=sum2values  
f(10,20)
```

30

```
f=lambda value1,value2: value1+value2
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Anonymous functions

- Makes MAP operations straightforward. Mapping means applying the same transformation to all the elements of a collection

```
file=("Ann;18;Single","Leo;24;Single","Ron;55;Divorced")

def splitColumns(line):
    return line.split(";")

matrix=[]
for line in file:
    matrix.append(splitColumns(line))
```

Cartagena99

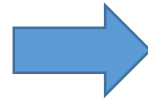
CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

["Ron", "55", "Divorced"]

Anonymous functions

```
file=("Ann;18;Single","Leo;24;Single","Ron;55;Divorced")  
  
def splitColumns(line):  
    return line.split(";")  
  
matrix=[]  
for line in file:  
    matrix.append(splitColumns(line))
```



```
file=("Ann;18;Single","Leo;24;Single","Ron;55;Divorced")  
  
def splitColumns(line):  
    return line.split(";")  
  
matrix = map( splitColumns , file )
```



```
file = ("Ann;18;Single","Leo;24;Single","Ron;55;Divorced")  
matrix = map( lambda line: line.split(";") , file )
```

This is the most common way of

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Handling exceptions

- Whenever a runtime error occurs, it creates an exception
- The default behaviour is halting the program and display a message
- No critical on amateur codes
- Mandatory on professional services

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Handling exceptions

- Why this code launch an exception?

```
In [6]: ▶ distance=100
        elapsed_time=0
        speed=distance/elapsed_time
        print("Speed: ",speed)
```

```
-----
ZeroDivisionError                                Traceback (most recent call last)
<ipython-input-6-3f5d4cfccce44> in <module>
      1 distance=100
      2 elapsed_time=0
----> 3 speed=distance/elapsed_time
      4 print("Speed: ",speed)
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Handling exceptions

- How do we deal with it?
- Use try/except statements
 - `try`: code to “protect”
 - `except`: what to do if an exception is thrown

```
In [7]: distance=100
        elapsed_time=0
        try:
            speed=distance/elapsed_time
        except ZeroDivisionError:
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Handling exceptions

- Remember this code?

```
from datetime import datetime
import pytz

def currentTimeStamp(timezone='Europe/Madrid'):
    tz = pytz.timezone(timezone)
    now = datetime.now(tz)
    return "Timestamp in {}: {}".format(timezone,now)
```

- What would happen if we run this statement?

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Handling exceptions

- This time zone does not exist
- The function from library `pytz` raises an exception
- The function “`currentTimeStamp`” does not know what to do with

```
currentTimeStamp("Alcobendas")
```

```
-----  
UnknownTimeZoneError                                Traceback (most recent call last)  
<ipython-input-111-33cc67bb76ee> in <module>  
----> 1 currentTimeStamp("Alcobendas")  
  
<ipython-input-110-631cd05f76fc> in currentTimeStamp(timezone)  
      3  
      4 def currentTimeStamp(timezone='Europe/Madrid'):  
----> 5     tz = pytz.timezone(timezone)  
      6     now = datetime.now(tz)  
      7     return "Timestamp in {}: {}".format(timezone,now)  
  
~\Anaconda3\lib\site-packages\pytz\_init_.py in timezone(zone)  
    179         fp.close()  
    180     else:  
--> 181         raise UnknownTimeZoneError(zone)
```

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

Handling exceptions

- We can capture specific exceptions or capture any of them

Note that we don't specify the Exception

```
from datetime import datetime
import pytz
def currentTimeStamp(timezone='Europe/Madrid'):
    try:
        tz = pytz.timezone(timezone)
        now = datetime.now(tz)
        return "Timestamp in {}: {}".format(timezone,now)
    except:
        print("Time zone unkown")
        print("Check available list:")
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Handling exceptions

- When the input time zone does not exist, now the code knows what to do
- It will never fail

```
currentTimeStamp("Australia/Perth")
```

```
'Timestamp in Australia/Perth: 2019-08-28 15:44:14.322528+08:00'
```

```
currentTimeStamp("Alcobendas")
```

```
Time zone unknown
```

```
Check available list:
```

```
https://en.wikipedia.org/wiki/List\_of\_tz\_database\_time\_zones
```

```
'Timestamp in UTC: 2019-08-28 09:44:15.400472'
```

```
currentTimeStamp(123)
```

```
Time zone unknown
```

```
Check available list:
```

```
https://en.wikipedia.org/wiki/List\_of\_tz\_database\_time\_zones
```

```
'Timestamp in UTC: 2019-08-28 09:44:16.373994'
```

```
currentTimeStamp("Zulu")
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

```
'Timestamp in UTC: 2019-08-28 09:44:16.920903'
```

Cartagena99

Custom Exceptions

- There exists a limited number of exceptions
- We can implement our own

Create new Exception

```
class RetrialsExceeded(Exception):  
    pass
```

```
counter=0  
while True:  
    password=input("Chose password: ")  
    if not(password.isalpha() or password.isnumeric()):  
        print("Password chosen")  
        break
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Functions and their error messages

```
from datetime import datetime
def addTimeStamp():
    now = datetime.now()
    return "timestamp: " +now

def checkIn(arg):
    log = arg + " is checking on "+addTimeStamp()
    return log

def checkout(arg):
    log = arg + " is checking on "+addTimeStamp()
    return log

def cardReader(name,action):
    if action=="IN":
        log=checkIn(name)
    else:
        log=checkOut(name)
    return log
```

This concatenation will fail

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Functions and their error messages

- The error trace shows all the functions that were “active” during the failure (called TRACEBACK)
- The inner trace (the bottom one) shows the real failure

```
cardReader("Luis","IN")
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-45-6bef1cd84a78> in <module>
----> 1 cardReader("Luis","IN")

<ipython-input-44-025162e26bbb> in cardReader(name, action)
    14 def cardReader(name,action):
    15     if action=="IN":
----> 16         log=checkIn(name)
    17     else:
    18         log=checkOut(name)

<ipython-input-44-025162e26bbb> in checkIn(arg)
     5
     6 def checkIn(arg):
----> 7     log = arg + " is checking on "+addTimeStamp()
     8     return log
     9

<ipython-input-44-025162e26bbb> in addTimeStamp()
     2 def addTimeStamp():
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Recursion

- Functions can be called from within another
- But a function can also call itself
- Beware of RECURSION and infinity interactions
- That can make complex functions very compact but can be extremely difficult to debug

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Recursion

- Compute N! (read N factorial)

$$N! = N * (N - 1) * (N - 2) * \dots * 1$$

```
def factorial(n):  
    print("- Computing {}".format(n))  
    if n == 1:  
        print("- Deepest level reached")  
        return 1  
    else:
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Recursion

- Compute N! (read N factorial)

```
In [56]: ► factorial(5)
- Computing 5!
- Computing 4!
- Computing 3!
- Computing 2!
- Computing 1!
- Deepest level reached
- Intermediate result for 1! is 2
- Intermediate result for 2! is 6
- Intermediate result for 3! is 24
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Recursion

What would happen if we run **factorial(-5)**?

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

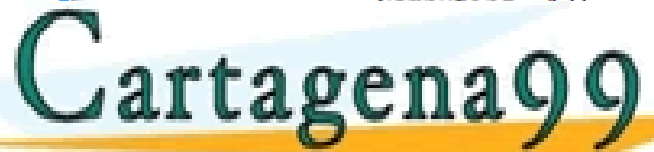
Recursion

```
In [58]: ▶ def factorial(n):
          print("- Computing {}".format(n))
          if n == 1:
              print("- Deepest level reached")
              return 1
          else:
              intermediate = n * factorial(n-1)
              print("- Intermediate result for {}! is {}".format(n-1,intermediate))
              return intermediate

factorial(-5)

- Computing -5!
- Computing -6!
- Computing -7!
- Computing -8!
- Computing -9!
- Computing -10!
- Computing -11!
- Computing -12!
```

To stop this we need to reboot the kernel



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70
- - -
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Boolean functions

- Functions can return Boolean values
- This is commonly used for hiding complicated tests
- We can solve factorial(-5) if we implement some verifications before running the function

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Boolean functions

```
def RunFactorial(n):  
    if not isinstance(n, int):  
        print("This is not an integer")  
        return False  
    elif n <= 0:  
        print("This is not a positive number")  
        return False  
    else:  
        return True
```

This code only returns TRUE when the number is an integer > 0

```
def factorial(n):  
    if RunFactorial(n)==True:  
        print("- Computing {}".format(n))  
        if n == 1:  
            print("- Deepest level reached")  
            return 1  
        else:  
            intermediate = n * factorial(n-1)  
            print("- Intermediate result for {}! is {}".format(n-1,intermediate))  
            return intermediate  
    else:  
        print("- Error")
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70