

# PROGRAMACIÓN I



## FUNCTIONS

---

*Business Analytics*

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

---

- ▶ *Definition*
- ▶ *Arguments and parameters*
- ▶ *Annotations/type hinting, docstrings*
- ▶ *Execution frames and namespaces*
- ▶ *Arbitrary length 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

## FUNCTIONS

---

- ▶ *Definition*
- ▶ *Arguments and parameters*
- ▶ *Annotations/type hinting, docstrings*
- ▶ *Execution frames and namespaces*
- ▶ *Arbitrary length arguments*
- ▶ *Lambda (anonymous functions)*

Cartagena99

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

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

- Groups of related statements
- Parameters, returns a value
- DRY: do not repeat yourself
- When invoked, body of the function is called before control returns.

```
def function_name():  
    ...  
    ...  
  
    # Start of program  
    ...  
    ...  
function_name()
```

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

**Built-in functions** are those provided by the language and we have seen several of these already:

- For example, both `print()` and `input()`

**User-defined functions** are those written by developers



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

```
def function_name(parameter list):  
    """docstring"""  
    statement  
    statement(s)
```

- All (named) functions are defined using the keyword `def`
- A function name which uniquely identifies (naming convention lower case \_)
- List of parameters (optional): allow data to be passed into the function.
- Colon is used to mark the end of the function header
- Function body, with statements (indented)
- Docstring: documentation string (optional)



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

```
def print_msg():  
    print('Hello World!')
```

```
print_msg()
```

```
def print_my_msg(msg):  
    print(msg)
```

```
print_my_msg('Hello World')
```

```
def square(n: int) -> int:  
    return n * n
```

```
# Store result from square in a variable
```



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

---

- ▶ *Definition*
- ▶ *Arguments and parameters*
- ▶ *Annotations/type hinting, docstrings*
- ▶ *Execution frames and namespaces*
- ▶ *Arbitrary length 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



- A **parameter** is a variable defined **as part of the function header** and is used to make data available within the function itself.
- An **argument** is the actual value or data **passed into the function** when it is called. The data will be held within the parameters.

parameters

```
def make_list_of_range(start: int, end: int) -> list:
    """
    Makes a list of consecutive integers between start and end
    :param start:    first integer
    :param end:      last integer
    :return:         a list with consecutive integers
    """

    new_list = list(range(start, end))
```



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

- Named arguments (or keyword arguments):

We provide the name of the parameter we want an argument/value to be assigned to

```
def make_list_of_range(start: int, end: int) -> list:
    """
    Makes a list of consecutive integers between start and end
    :param start:    first integer
    :param end:      last integer
    :return:         a list with consecutive integers
    """

    new_list = list(range(start, end))
    return new_list
```

Keyword arguments

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

- Positional arguments:

Arguments assigned to parameters based on position

```
def make_list_of_range(start: int, end: int) -> list:
    """
    Makes a list of consecutive integers between start and end
    :param start:    first integer
    :param end:      last integer
    :return:         a list with consecutive integers
    """

    newList = list(range(start, end))
    return newList
```



Cartagena99

Positional arguments

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

- Blend of positional and keyword arguments

```
def make_list_of_range(start: int, end: int) -> list:
    """
    Makes a list of consecutive integers between start and end
    :param start:    first integer
    :param end:      last integer
    :return:         a list with consecutive integers
    """

    newList = list(range(start, end))
    return newList
```

Blend of positional and keyword arguments

```
make_list_of_range(2, end=4)
```



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

- With default values:

Function arguments can have default values in Python.

We can provide a default value to an argument by using the assignment operator (=).

```
def make_list_of_range_default(start: int, end: int = 10) -> list:
    """
    Makes a list of consecutive integers between start and end
    :param start:    first integer
    :param end:      last integer
    :return:         a list with consecutive integers
    """

    newList = list(range(start, end))
    return newList
```



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

- Any number of arguments in a function can have a default value.
- Once we have a default argument, all the arguments to its right must also have default values.
- That is, non-default arguments cannot follow default 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

- `return` statement to return a value (or values) to the calling code
- function with terminate
- Return value can be used at the point that the function was invoked

```
def square(n: int) -> int:  
    return n * n
```

```
# Store result from square in a variable  
result = square(4)  
print(result)
```

```
# Send the result from square immediately to another function  
print(square(5))  
# Use the result of a function call
```



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

- It is possible to return multiple values from a function
- We can then assign the values returned to variables at the point when the function is called:

```
def swap(a: int, b: int) -> tuple[int, int]:  
    return b, a
```

```
a = 2  
b = 3  
x, y = swap(a, b)  
print(x, ',', y)
```



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

---

- ▶ *Definition*
- ▶ *Arguments and parameters*
- ▶ ***Annotations/type hinting, docstrings***
- ▶ *Execution frames and namespaces*
- ▶ *Arbitrary length 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

- Python uses dynamic typing
- To provide consistency, we use annotation to “hint” the type of variables
- Particularly useful in parameters and return values of functions
- *No type checking happens at runtime*
- Python assumes separate off-line type checker which users can run over their source code voluntarily
  - This is PyCharm in our case



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

# Docstring and annotation (type hinting)

```
def make_a_list_annotated(start: int, end: int) -> List:
    """
    Makes a list of consecutive integers between start and end
    :param start:      first integer
    :param end:        last integer
    :return:           a list with consecutive integers
    """

    new_list = list(range(start, end))
    return new_list
```

Docstrings



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

# Docstring and annotation (type hinting)

```
def make_a_list_of_range_default(start: int, end: int = 10) -> list:
    """
    Makes a list of consecutive integers between start and end
    :param start:      first integer
    :param end:        last integer
    :return:           a list with consecutive integers
    """

    new_list = list(range(start, end))
    return new_list
```

- Good practice
- IDEs will show documentation when used
- After 6 months, any code, even yours, will look like new.

```
8
9 def make_list_of_range_default(start: int, end: int = 10) -> list:
10     """
11     """
12     def make_list_of_range_default(start: int,
13                                     end: int = 10) -> list
14
15     Makes a list of consecutive integers between start and end :param
16     start: first integer :param end: last ingeter :return: a list with
17     consecutive integers
```

Cartagena99

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

## FUNCTIONS

---

- ▶ *Definition*
- ▶ *Arguments and parameters*
- ▶ *Annotations/type hinting, docstrings*
- ▶ *Execution frames and namespaces*
- ▶ *Arbitrary length 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

- Python creates an execution frame and passes parameters
- Calling code waits for return
- Each execution frame has its own namespace (more on this later)



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



Calculate a function that returns the nth Fibonacci number

$$F_0 = 0$$

$$F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}$$

The beginning of the sequence is thus:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...



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



# Recursive option

```
def fib_recursive(n: int) -> int:
    """
    Recursive function to calculate the n-th fibonacci number
    :param n:    position in the sequence
    :return:     Fn
    """
    if n == 0:
        # By definition
        return 0
    elif n == 1:
        # By definition
        return 1
    else:
        return fib_recursive(n - 1) + fib_recursive(n - 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

Time complexity:  $T(n) = T(n-1) + T(n-2)$  which is exponential.





```
def fib_dynamic_array(n: int) -> int:
    """
    Fibonacci Series using Dynamic Programming, using array
    :param n:    position in the sequence
    :return:     Fn
    """
    if n < 2:
        return n
    table = np.zeros(shape=(n + 1,), dtype=int)
    table[1] = 1
    for i in range(2, n + 1):
        table[i] = table[i - 1] + table[i - 2]
    return table[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



```
def fib_dynamic_list(n):  
    """  
    Fibonacci Series using Dynamic Programming, using list  
    :param n:    position in the sequence  
    :return:    Fn  
    """  
    if n < 2:  
        return n  
    results = [0, 1]  
    for i in range(2, n + 1):  
        results.append(results[i - 1] + results[i - 2])  
    return results[-1]
```



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



# Dynamic programming, using iterative approach

```
def fib_dynamic_iter(n):  
    """  
    Fibonacci Series using Dynamic Programming, iterative approach  
    :param n:    position in the sequence  
    :return:     Fn  
    """  
    previous, current = (0, 1)  
    for i in range(2, n + 1):  
        previous, current = (current, previous + current)  
    return current
```

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



```
def fib_formula_2(n: int) -> int:
    """
    Fibonacci Series using Binets Formula
    :param n:    position in the sequence
    :return:     Fn
    """
    phi = (1 + 5 ** 0.5) / 2
    return round(phi ** n / 5 ** 0.5)
```

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

## FUNCTIONS

---

- ▶ *Definition*
- ▶ *Arguments and parameters*
- ▶ *Annotations/type hinting, docstrings*
- ▶ *Execution frames and namespaces*
- ▶ *Arbitrary length 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

Sometimes, we do not know in advance the number of arguments that will be passed into a function.

- Arbitrary arguments:
  - We can handle this situation through function calls with an arbitrary number of arguments.
  - Syntax: an asterisk (\*) before the parameter name to denote this kind of argument.

```
def greeter(*names: str) -> None:  
    for name in names:  
        print('Welcome', name)
```

```
Welcome John  
Welcome Denise  
Welcome Phoebe  
Welcome Adam  
Welcome Gryff
```



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

## Use:

- Available as a tuple inside the function, with the name used after the \*
- We can use indexes to access individual items (names[0])
- It is iterable: use in a for or as as an argument to another function that takes in an iterable

```
def sorted_greeter(*names: str) -> None:  
    print(f'Sorted names: {sorted(names)}')
```

```
sorted_greeter('John', 'Denise', 'Phoebe', 'Adam', 'Gryff', 'Jasmine')
```



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

It is also possible to pass arbitrary length keyword arguments.

- Syntax `**kwargs` (doble `*` followed by an identifier)
- `kwargs` becomes a dictionary that maps each keyword to the value that we pass alongside it
- We can iterate using dictionary methods like `.item()`, `.values()`, `.keys()`

```
def values_to_lowercase(**kwargs: str):  
    for key, value in kwargs.items():  
        print(f"wisper {key} value, {value.lower()}")
```

```
values_to_lowercase(first='I', mid='SAY', last='SHOUT')
```

```
wisper first value, i
```



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

---

- ▶ *Definition*
- ▶ *Arguments and parameters*
- ▶ *Annotations/type hinting, docstrings*
- ▶ *Execution frames and namespaces*
- ▶ *Arbitrary length arguments*

Cartagena99

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

When we define a function, we use a name and the object is stored in the namespace

Sometimes we need to build a short, simple function for “immediate” consumption:

- we don't want to keep it in the name space (to keep in clean and simple, and to avoid names clashes).
- We want to improve readability of the code

Defined using the keyword lambda (lambda functions)

```
def sorting_criteria(student: tuple[str, str, int]) -> int:  
    return student[2]
```

becomes, at the point of consumption:



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



Consider a nested data structure (container):

```
student_tuples = [  
    ('john', 'A', 15),  
    ('jane', 'B', 12),  
    ('dave', 'B', 10),  
]
```

We want to sort this structure. After sorting, we will return tuples of str, str, int, the elements of the list.

The question is:



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



`sorted(iterable, key=None, reverse=False)`

Sorted() has a key parameter to specify a function (or other callable) to be called on each list element prior to making comparisons.

The value of the key parameter should be a function (or other callable) that takes a single argument and returns a key to use for sorting purposes.

```
student_tuples = [  
    ('john', 'A', 15),  
    ('jane', 'B', 12),  
    ('dave', 'B', 10)]
```



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



We are going to sort the tuples based on the integer of the tuple.

Work in groups:

1. Define a function called `sorting_criteria` that takes in an element of the list and returns the integer
2. Create the variable `student_tuples`

```
student_tuples = [  
    ('john', 'A', 15),  
    ('jane', 'B', 12),  
    ('dave', 'B', 10),
```

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

argument to the parameter key