POLYTECHNIC
FROM PORTO
SCHOOL
HIGHER
MEDIA ARTS
AND DESIGN

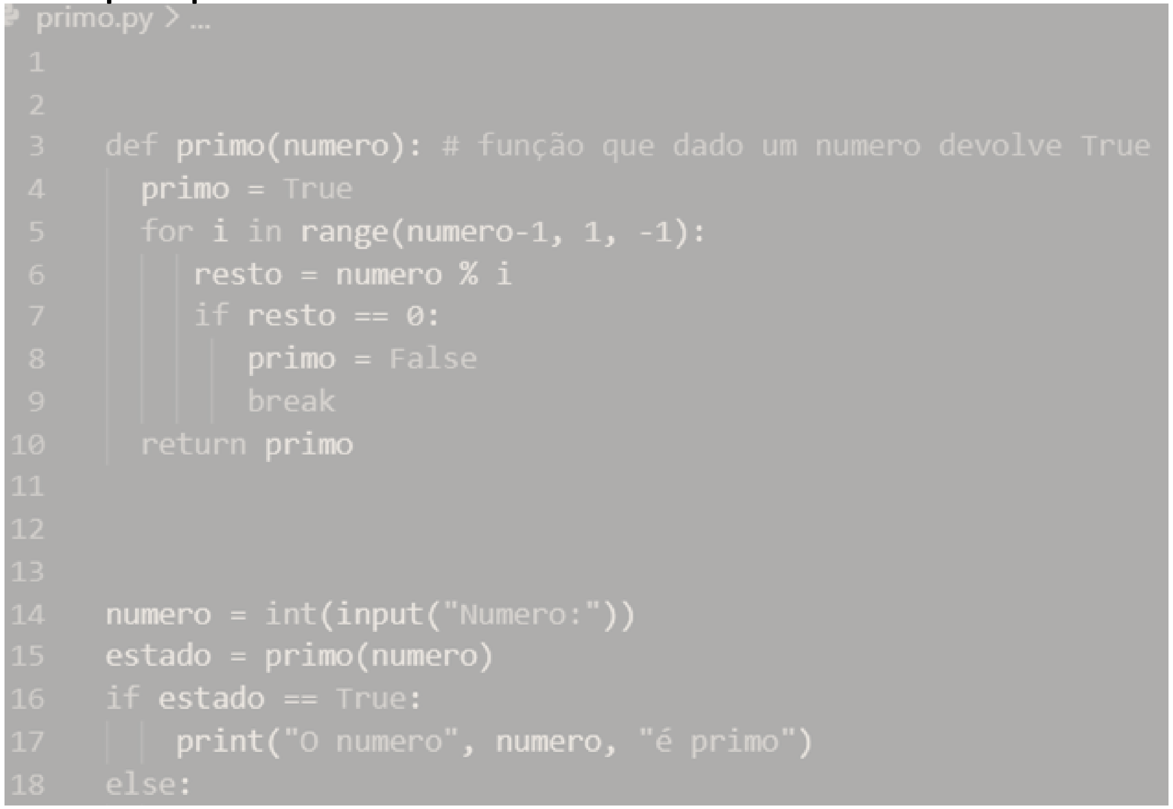
P.PORTO

ALGORITHMY AND DATA STRUCTURES

MODULE III
Modular Decomposition - Functions

Technologies and Information Systems for the Web

1. Modular Decomposition - Functions
   - ❑ Concept
   - ❑ Create a Function
   - ❑ Data return (*return*)
   - ❑ Parameters with default value
   - ❑ Undefined number of input parameters



primo.py > ...

```python


def primo(numero): # função que dado um numero devolve True
    primo = True
    for i in range(numero-1, 1, -1):
        resto = numero % i
        if resto == 0:
            primo = False
            break
    return primo



numero = int(input("Numero:"))
estado = primo(numero)
if estado == True:
    print("O numero", numero, "é primo")
else:
```

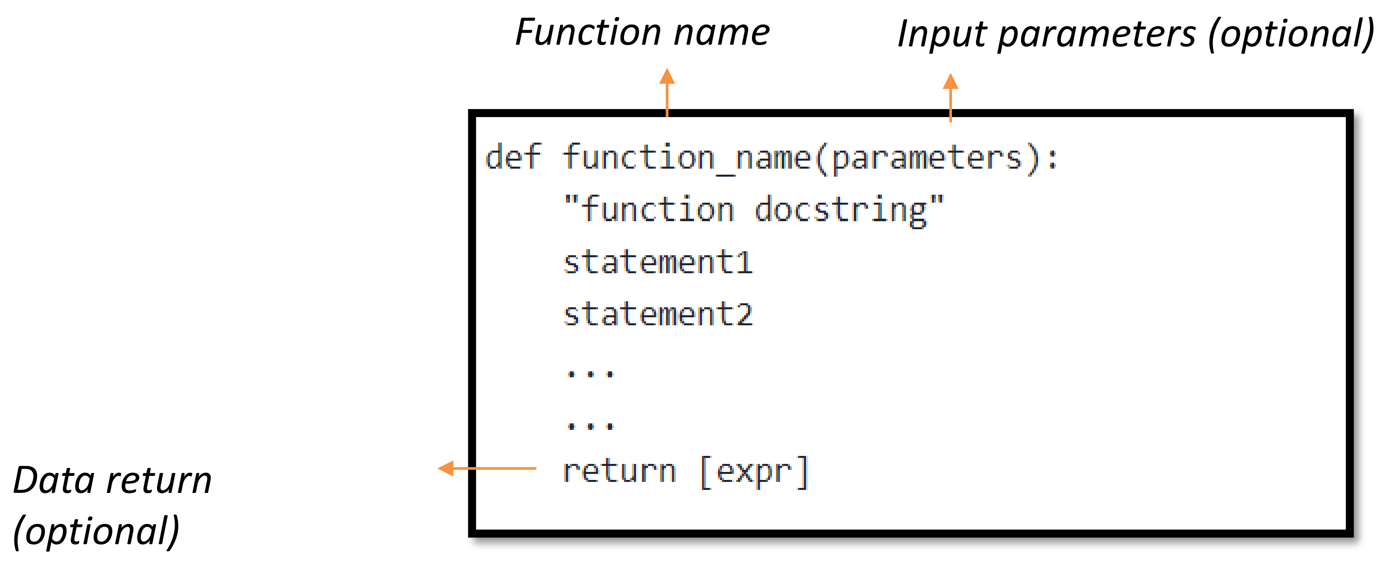
❖ Functions | Concept

- ❑ Set of code (or block of instructions), clearly delimited, and which performs a specific task
- ❑ A function behaves in the same way as a program, although on a different scale:
  - ❑ It is executed when invoked by the program that calls it;
  - ❑ When finished, it returns the execution of the program, from the place where it was called.
- ❑ The use of functions allows you to develop code in a structured way, facilitating:
  - ❑ Code reutilization
  - ❑ Code readability/readability
  - ❑ Code abstraction

❖ Functions | Concept

- A function is a reusable block of programming instructions designed to perform a certain task.

- To define a function, Python provides the keyword **def**. Following is the syntax to define a function:



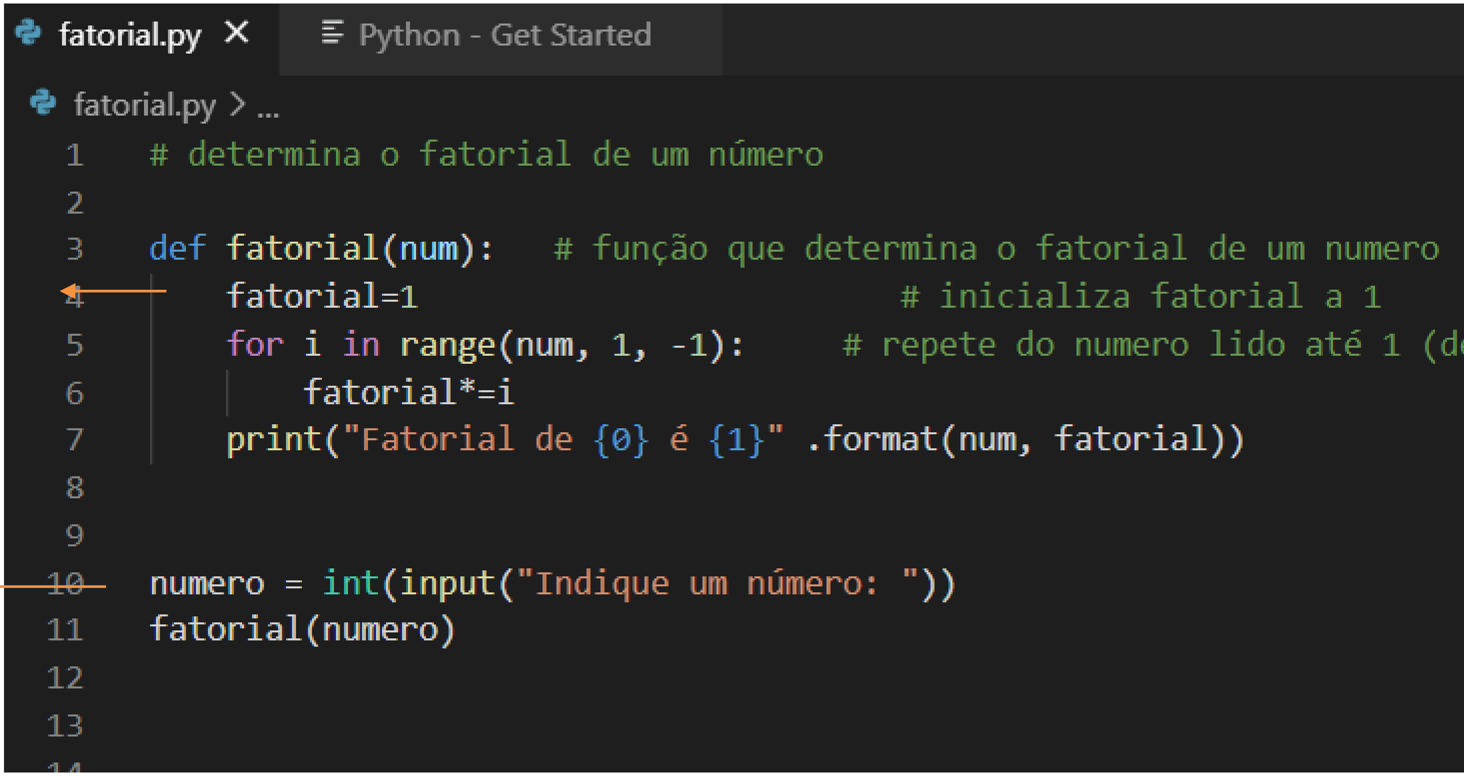
```
def function_name(parameters):
    "function docstring"
    statement1
    statement2
    ...
    ...
    return [expr]
```

❖ Functions | Concept

- ❑ Variables defined within the scope of a function are **local variables - they exist only** within this function, as opposed to **global variables –** they exist throughout the execution of the program

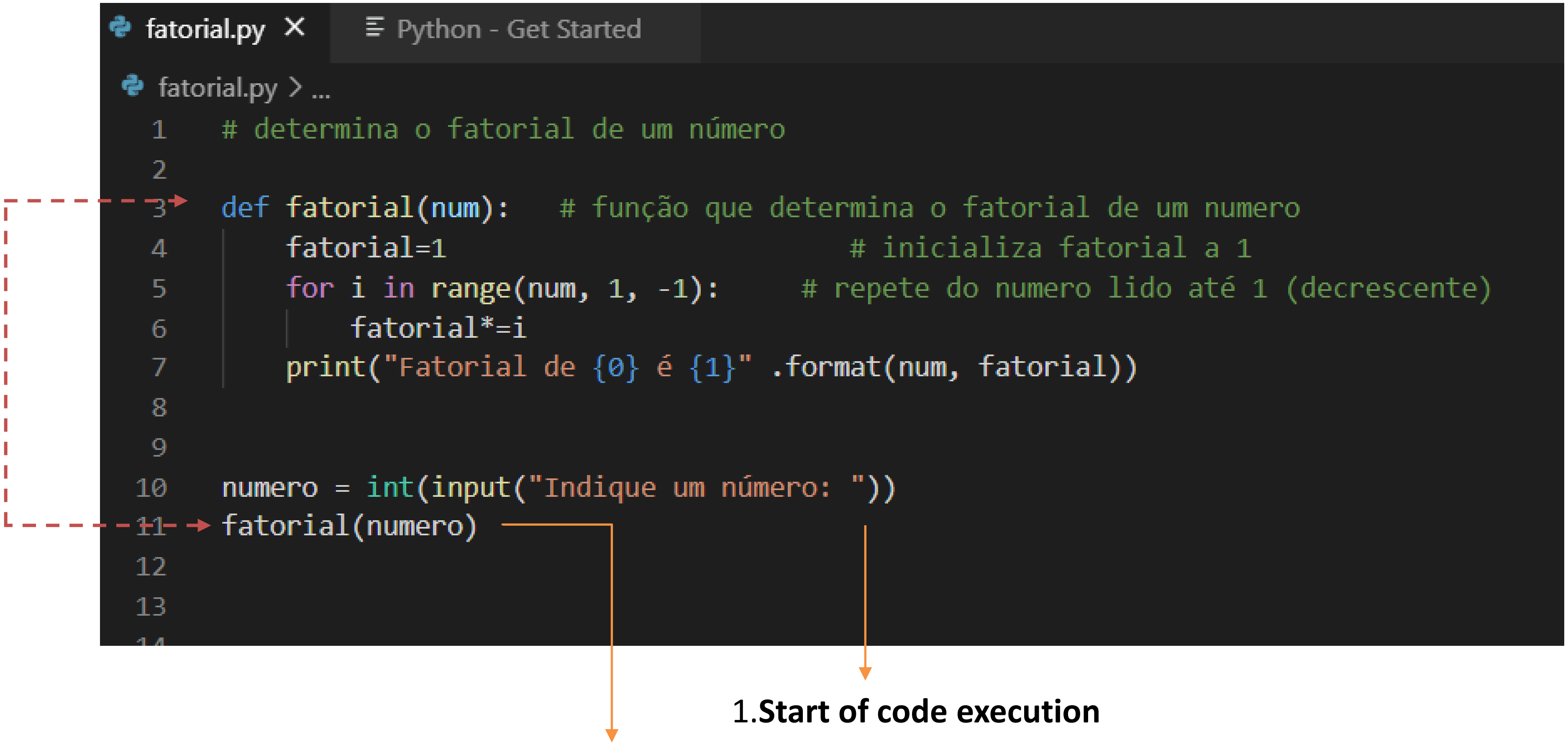
Local variables (there are only inside function)

Global variable



```
# determina o fatorial de um número

def fatorial(num):   # função que determina o fatorial de um numero
    fatorial=1                          # inicializa fatorial a 1
    for i in range(num, 1, -1):     # repete do numero lido até 1 (d
        fatorial*=i
    print("Fatorial de {0} é {1}" .format(num, fatorial))


numero = int(input("Indique um número: "))
fatorial(numero)
```

❖ Create a Function



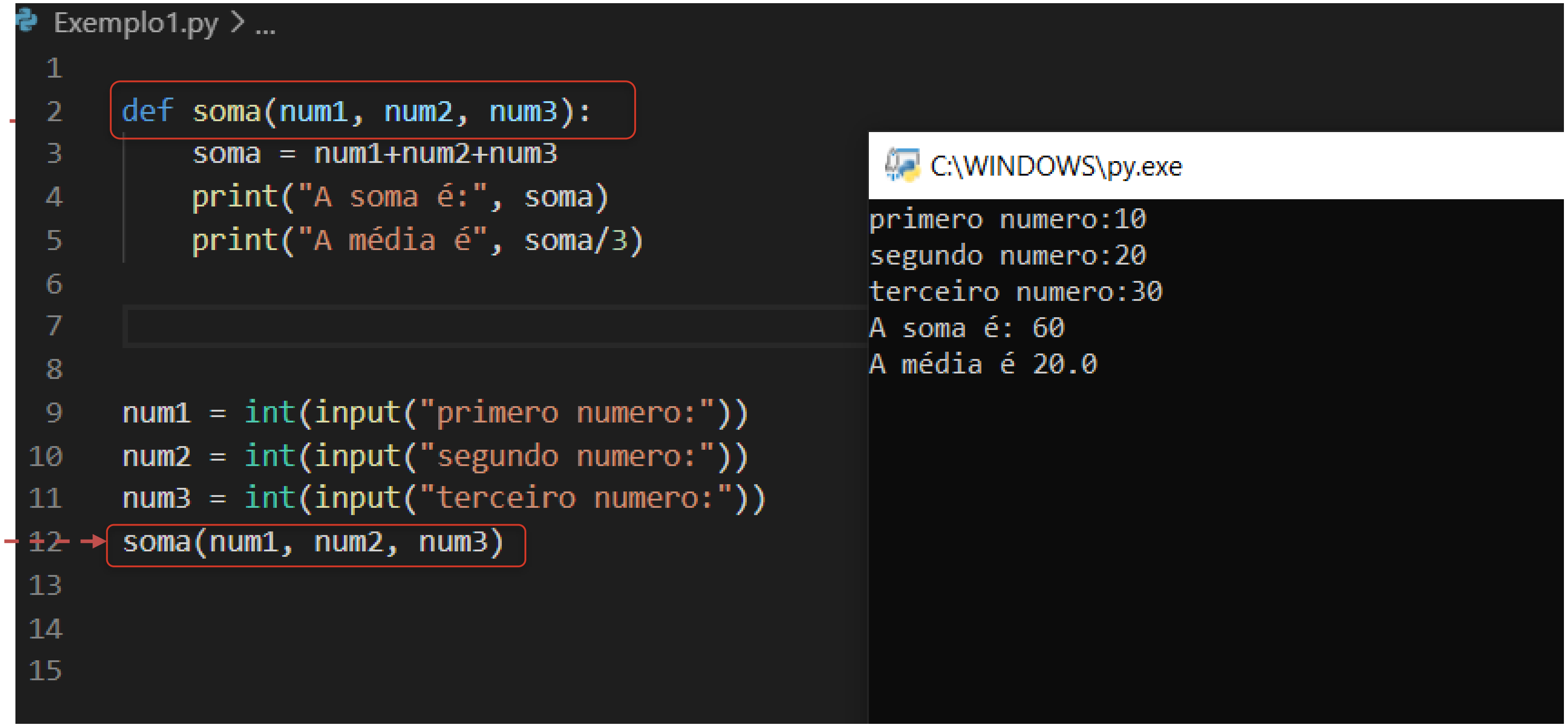
```python
# determina o fatorial de um número

def fatorial(num):   # função que determina o fatorial de um numero
    fatorial=1                            # inicializa fatorial a 1
    for i in range(num, 1, -1):      # repete do numero lido até 1 (decrescente)
        fatorial*=i
    print("Fatorial de {0} é {1}" .format(num, fatorial))


numero = int(input("Indique um número: "))
fatorial(numero)
```

1.Start of code execution

2.Invokes the fatorial function

❖ Create a Function

By default, a function must be called with the correct number of arguments.
Which means if your function **receives** 3 arguments, I should invoke the function with 3 arguments



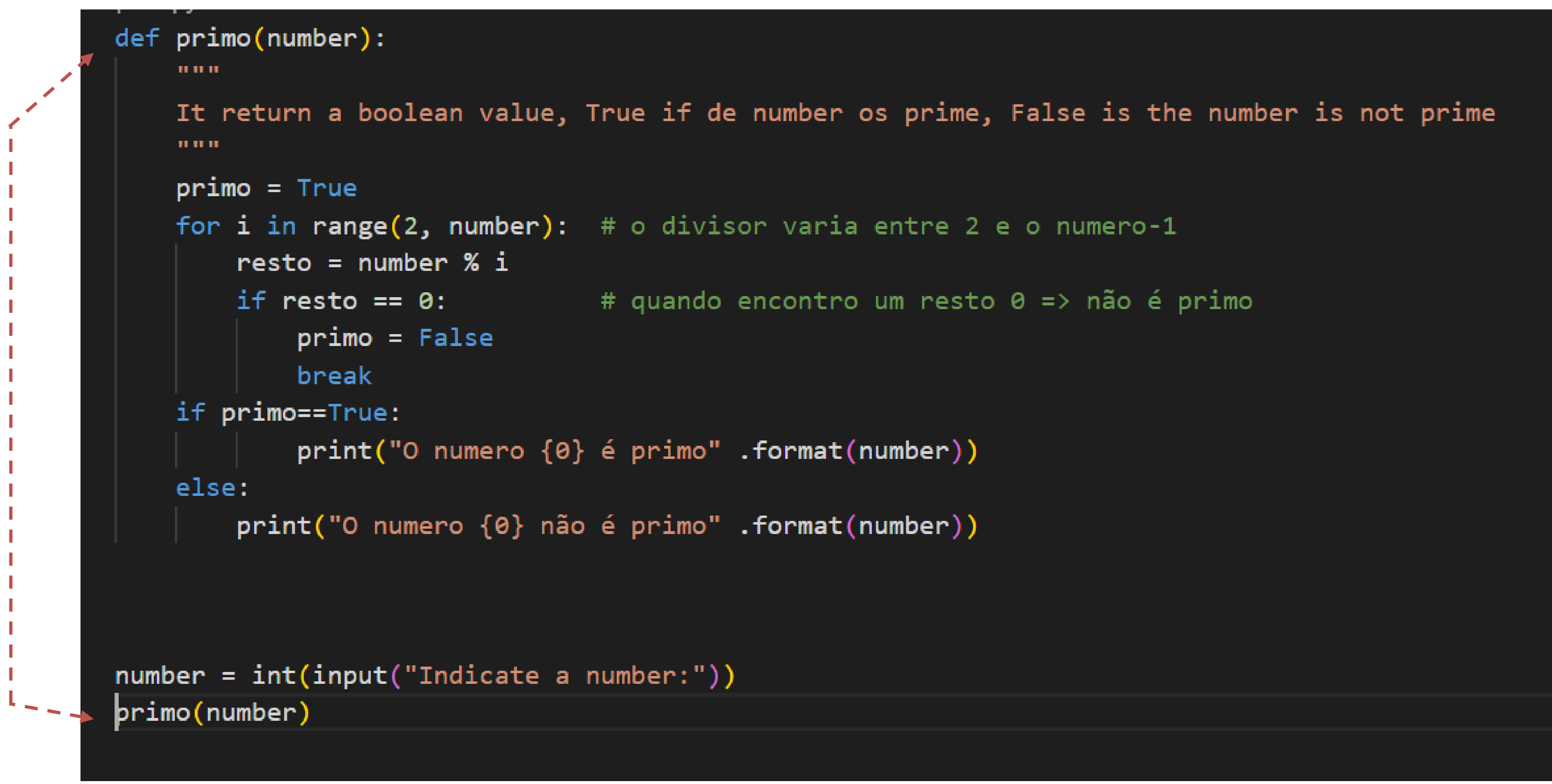
Exemplo1.py > ...

```python

def soma(num1, num2, num3):
    soma = num1+num2+num3
    print("A soma é:", soma)
    print("A média é", soma/3)



num1 = int(input("primero numero:"))
num2 = int(input("segundo numero:"))
num3 = int(input("terceiro numero:"))
soma(num1, num2, num3)
```

C:\WINDOWS\py.exe

```
primero numero:10
segundo numero:20
terceiro numero:30
A soma é: 60
A média é 20.0
```

❖ Create a Function

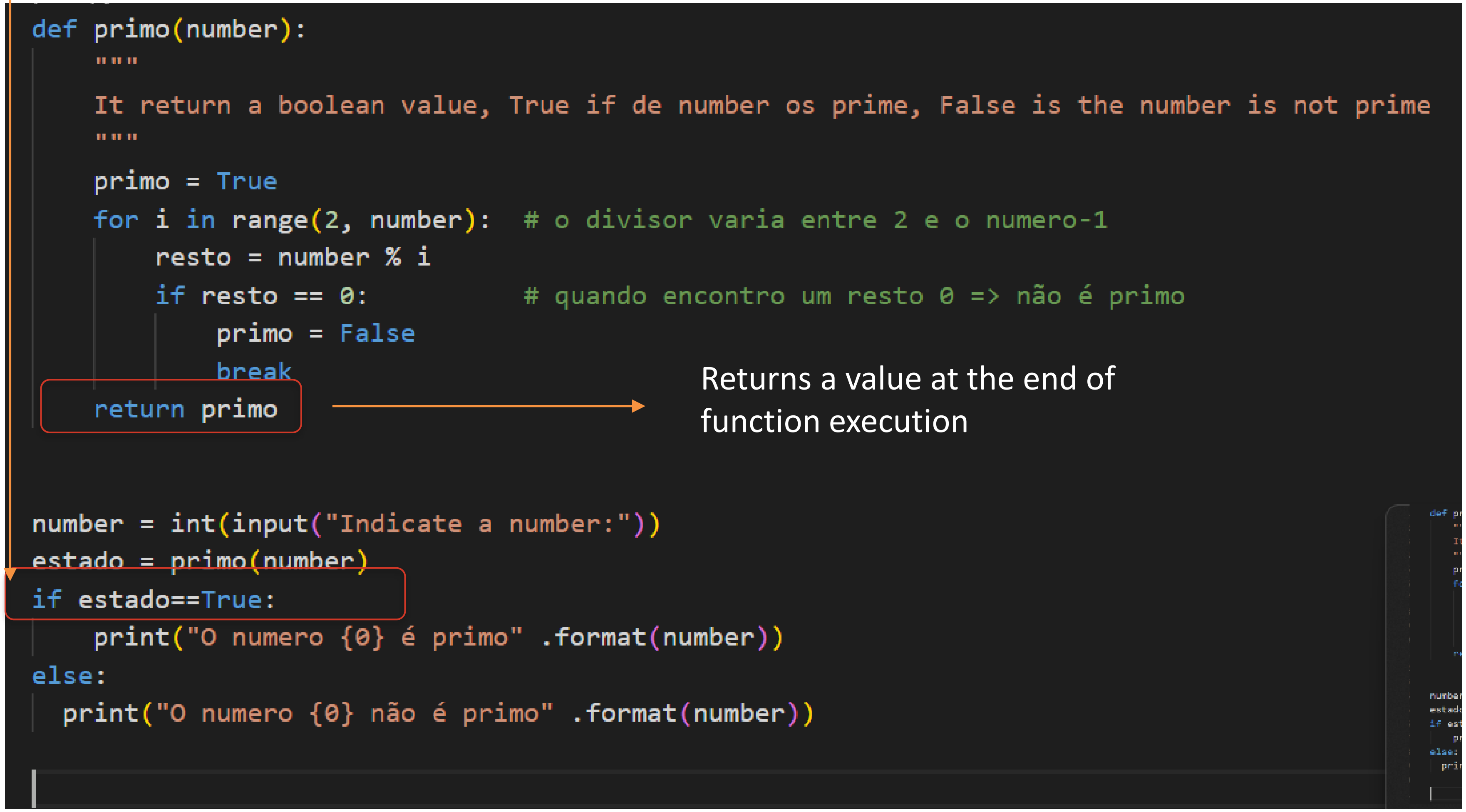


```
def primo(number):
    """
    It return a boolean value, True if de number os prime, False is the number is not prime
    """
    primo = True
    for i in range(2, number):  # o divisor varia entre 2 e o numero-1
        resto = number % i
        if resto == 0:          # quando encontro um resto 0 => não é primo
            primo = False
            break
    if primo==True:
            print("O numero {0} é primo" .format(number))
    else:
        print("O numero {0} não é primo" .format(number))



number = int(input("Indicate a number:"))
primo(number)
```

[redacted]

❖ Data return | return

When a function **returns a value (keyword return)**, we must assign the function to a variable

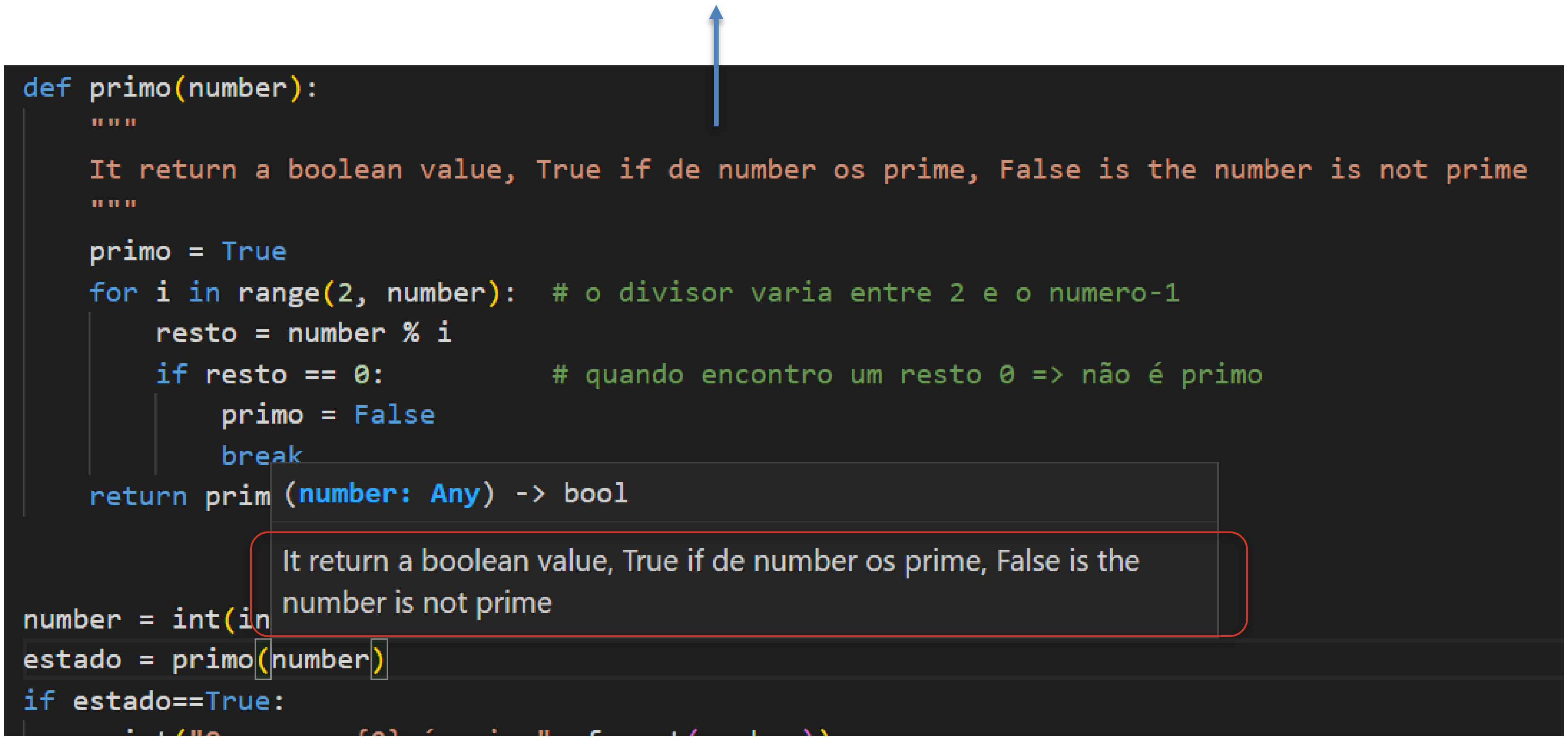


```
def primo(number):
    """
    It return a boolean value, True if de number os prime, False is the number is not prime
    """
    primo = True
    for i in range(2, number):  # o divisor varia entre 2 e o numero-1
        resto = number % i
        if resto == 0:          # quando encontro um resto 0 => não é primo
            primo = False
            break
    return primo


number = int(input("Indicate a number:"))
estado = primo(number)
if estado==True:
    print("O numero {0} é primo" .format(number))
else:
  print("O numero {0} não é primo" .format(number))
```

Returns a value at the end of function execution

❖ Data return | return

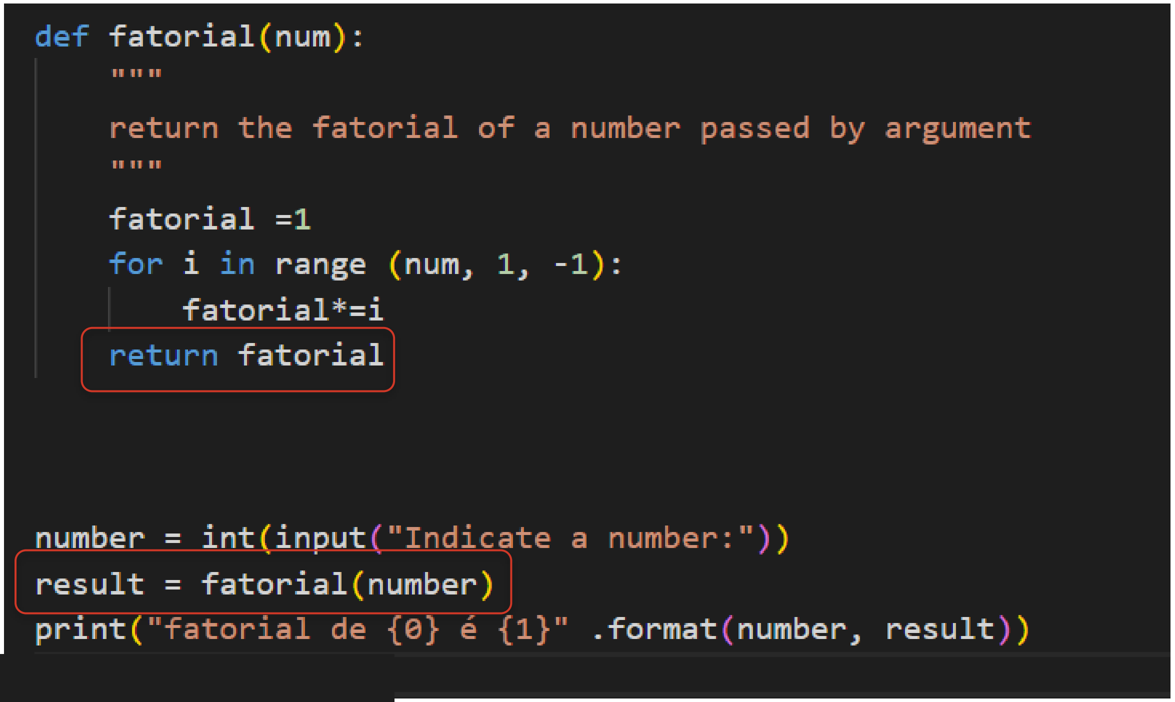
DocString



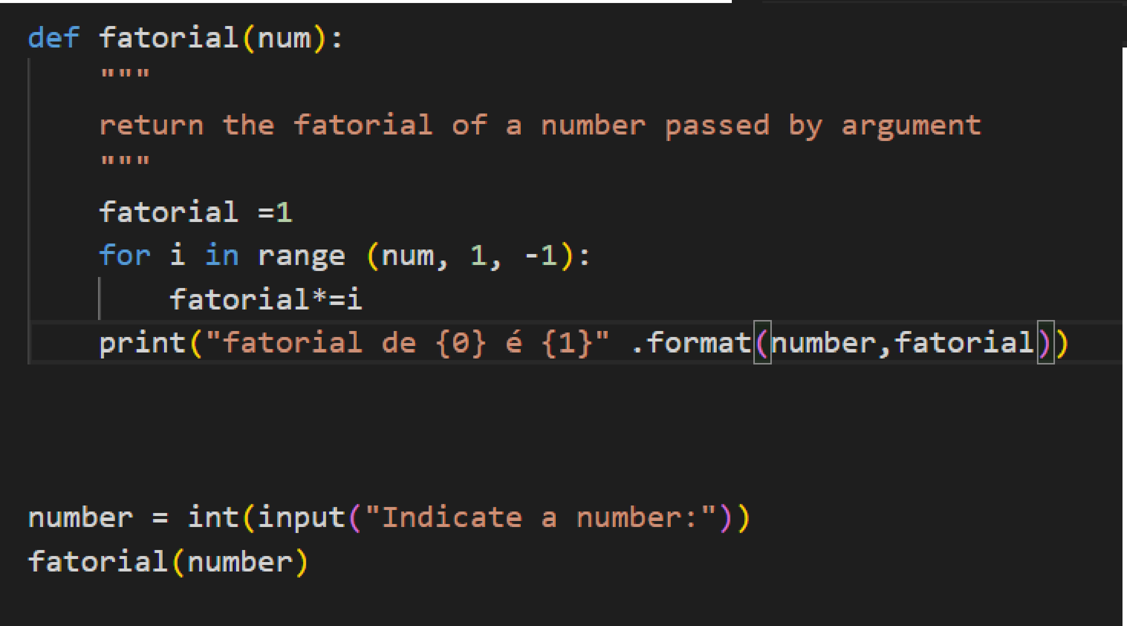
```
def primo(number):
    """
    It return a boolean value, True if de number os prime, False is the number is not prime
    """
    primo = True
    for i in range(2, number):  # o divisor varia entre 2 e o numero-1
        resto = number % i
        if resto == 0:          # quando encontro um resto 0 => não é primo
            primo = False
            break
    return prim (number: Any) -> bool

            It return a boolean value, True if de number os prime, False is the
            number is not prime
number = int(in
estado = primo(number)
if estado==True:
```

❖ Data return |return

Example of a function that returns a value



```
def fatorial(num):
    """
    return the fatorial of a number passed by argument
    """
    fatorial =1
    for i in range (num, 1, -1):
        fatorial*=i
    return fatorial



number = int(input("Indicate a number:"))
result = fatorial(number)
print("fatorial de {0} é {1}" .format(number, result))
```

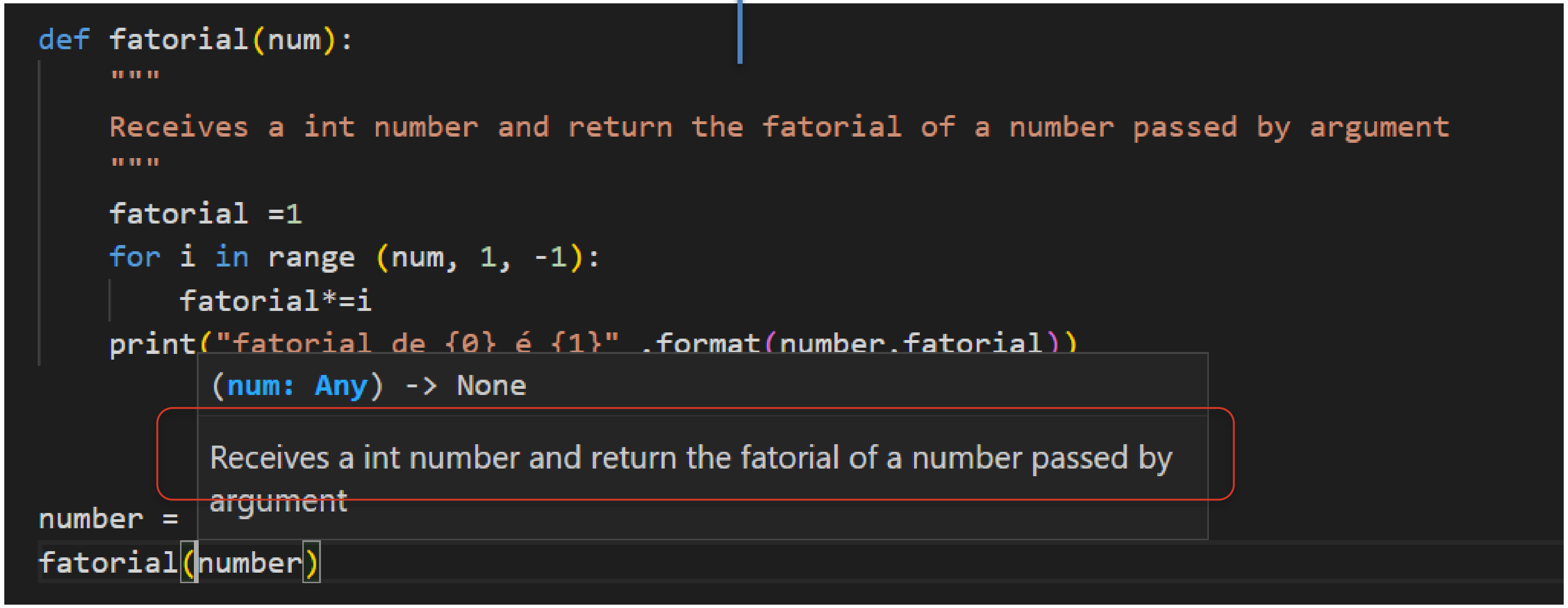


```
def fatorial(num):
    """
    return the fatorial of a number passed by argument
    """
    fatorial =1
    for i in range (num, 1, -1):
        fatorial*=i
    print("fatorial de {0} é {1}" .format(number,fatorial))




number = int(input("Indicate a number:"))
fatorial(number)
```

example of a function that does not return a value

❖ Data return |return

DocString



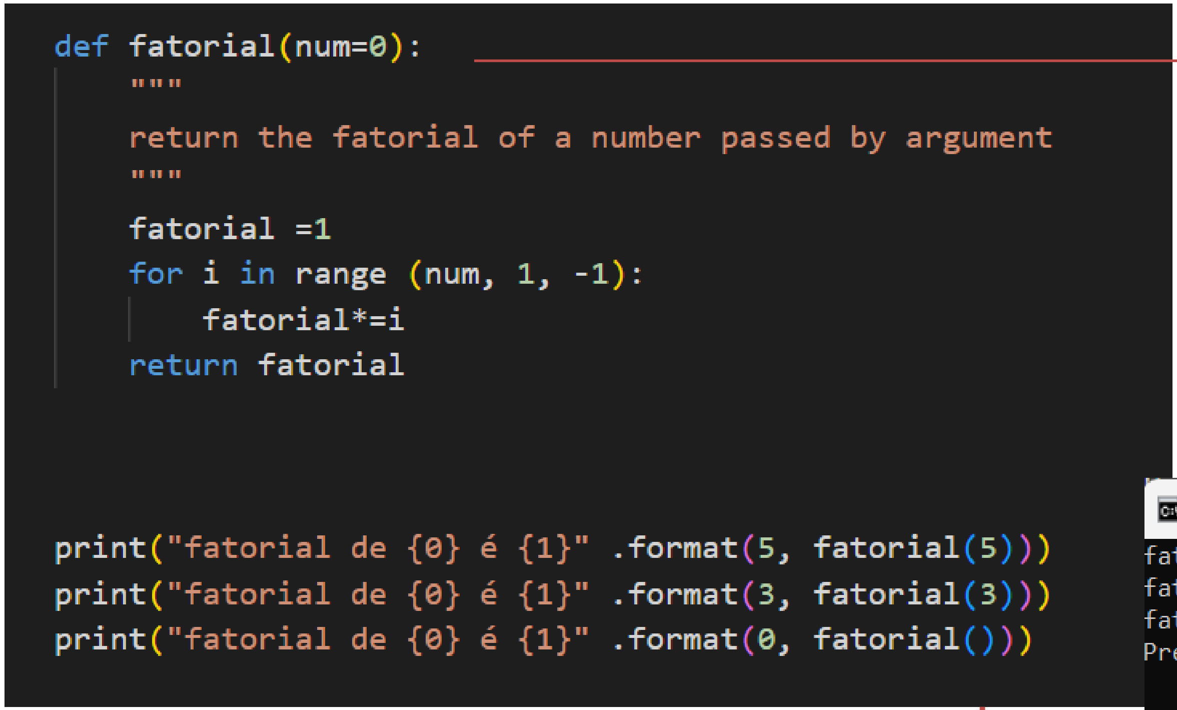
```
def fatorial(num):
    """
    Receives a int number and return the fatorial of a number passed by argument
    """
    fatorial =1
    for i in range (num, 1, -1):
        fatorial*=i
    print("fatorial de {0} é {1}" .format(number,fatorial))
```

```
(num: Any) -> None

Receives a int number and return the fatorial of a number passed by
argument
```

```
number =
fatorial(number)
```

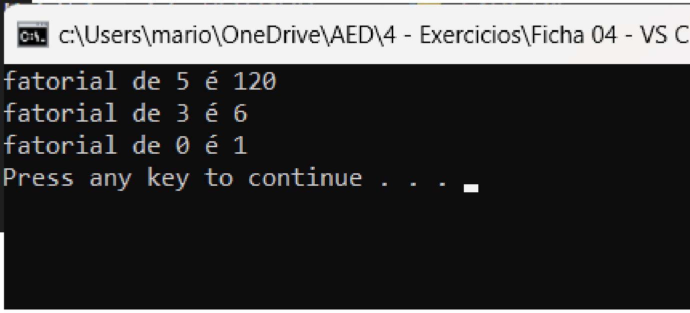
❖ Parameters with default value

**Parameters defined by default
can be omitted when I call the function**



```python
def fatorial(num=0):
    """
    return the fatorial of a number passed by argument
    """
    fatorial =1
    for i in range (num, 1, -1):
        fatorial*=i
    return fatorial


print("fatorial de {0} é {1}" .format(5, fatorial(5)))
print("fatorial de {0} é {1}" .format(3, fatorial(3)))
print("fatorial de {0} é {1}" .format(0, fatorial()))
```

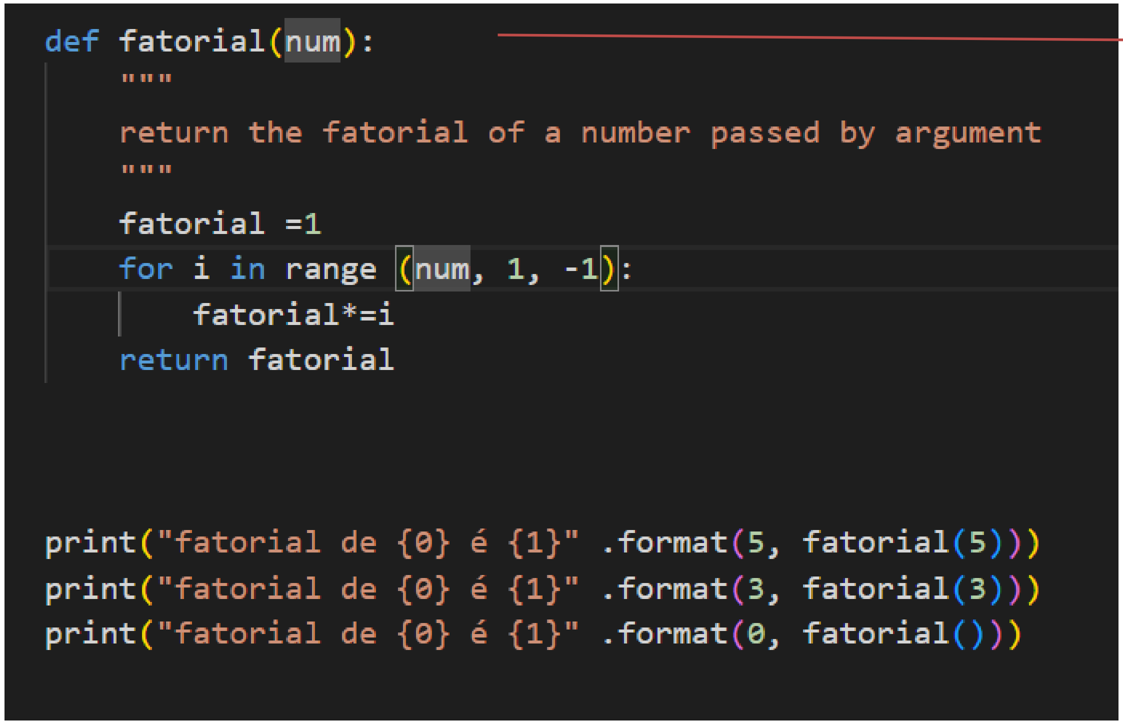
Default value, when I call the function if I pass an argument



```
c:\Users\mario\OneDrive\AED\4 - Exercicios\Ficha 04 - VS C
fatorial de 5 é 120
fatorial de 3 é 6
fatorial de 0 é 1
Press any key to continue . . .
```

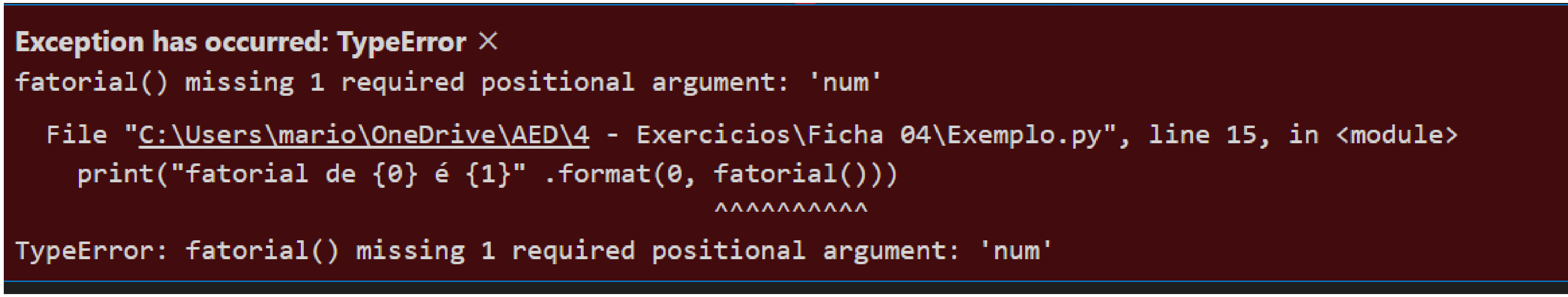
I invoke the factorial function without passing any value

❖ Parameters with default value



```python
def fatorial(num):
    """
    return the fatorial of a number passed by argument
    """
    fatorial =1
    for i in range (num, 1, -1):
        fatorial*=i
    return fatorial



print("fatorial de {0} é {1}" .format(5, fatorial(5)))
print("fatorial de {0} é {1}" .format(3, fatorial(3)))
print("fatorial de {0} é {1}" .format(0, fatorial()))
```

In this case it gives an error, as the parameter does not contain any default value

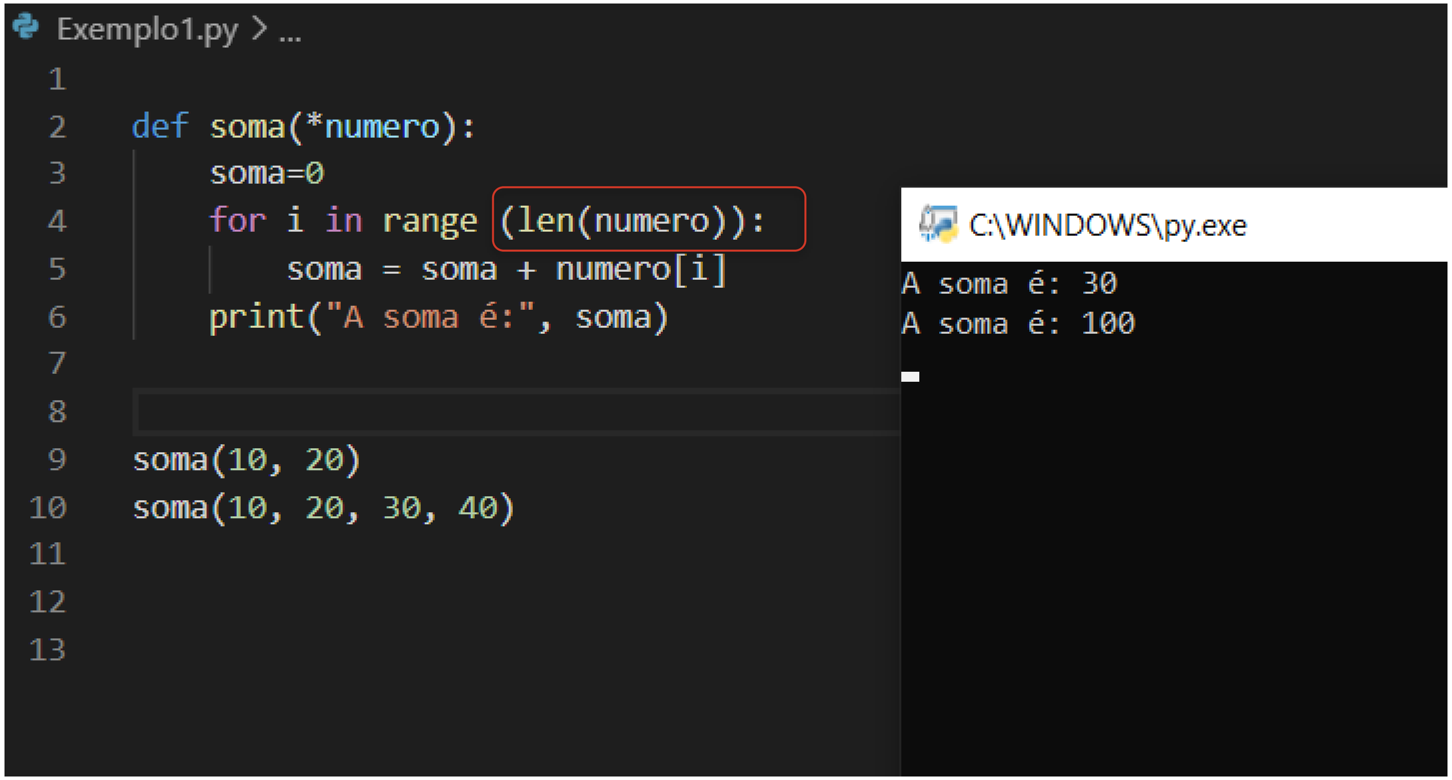


```
Exception has occurred: TypeError ×
fatorial() missing 1 required positional argument: 'num'

  File "C:\Users\mario\OneDrive\AED\4 - Exercicios\Ficha 04\Exemplo.py", line 15, in <module>
    print("fatorial de {0} é {1}" .format(0, fatorial()))
                                             ^^^^^^^^^^^
TypeError: fatorial() missing 1 required positional argument: 'num'
```

❖ Number of parameters undefined

function len()

counts the number of arguments that were passed to the function



Exemplo1.py > ...

```python

def soma(*numero):
    soma=0
    for i in range (len(numero)):
        soma = soma + numero[i]
    print("A soma é:", soma)


soma(10, 20)
soma(10, 20, 30, 40)
```

C:\WINDOWS\py.exe

```
A soma é: 30
A soma é: 100
```