

# Funções, módulos e pacotes

Mapeamento

Função `map()`

# Funções, módulos e pacotes

## Mapeamento – Função map()

O mapeamento é feito através da função `map()`, que consiste em aplicar uma função a todos os itens de uma sequência, seja uma lista, tupla, etc., gerando uma outra sequência contendo os resultados e com o mesmo tamanho da sequência inicial. A função `map()` sempre retorna um gerador.

# Funções, módulos e pacotes

## Mapeamento – Função map()

A sintaxe da função map é:

```
map(função, objeto_iterável, ...)
```

Onde:

função: map() passa cada item do objeto iterável para a função informada.

objeto\_iterável: Objeto iterável que deve ser mapeado. Podemos passar mais de um objeto iterável.

# Funções, módulos e pacotes

## Mapeamento – Função map()

Para exemplificar, vamos criar uma função que retorna um número informado ao quadrado:

```
def calculaQuadrado(numero) :  
    return f"O quadrado de {numero} é: {numero * numero}"
```

# Funções, módulos e pacotes

## Mapeamento – Função map()

Em seguida vamos criar uma lista de números. Depois criaremos uma variável chamada “calculado” que receberá o resultado da função map(), onde vamos passar a função calculaQuadrado e a lista de números como parâmetros.

```
lista_numeros = [1, 2, 3, 4, 5, 6]  
calculado = map(calculaQuadrado, lista_numeros)
```

# Funções, módulos e pacotes

## Mapeamento – Função map()

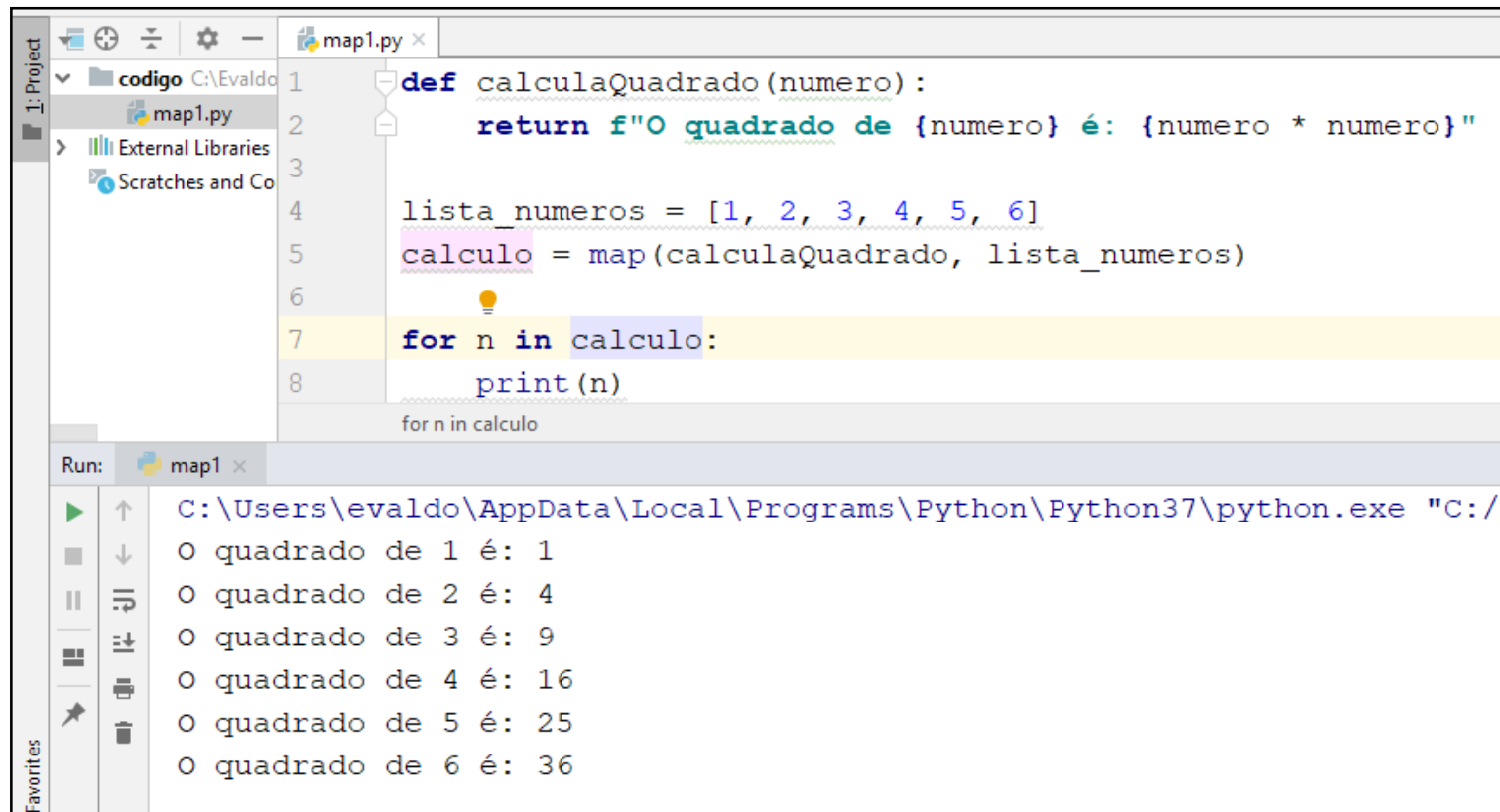
Por fim, vamos percorrer o objeto iterável “calculo”, que é uma lista, imprimindo cada item desta lista.

```
for n in calculo:  
    print(n)
```

# Funções, módulos e pacotes

## Mapeamento – Função map()

Agora veja o programa em execução.



The screenshot shows a Python IDE with a project named 'codigo' containing a file 'map1.py'. The code in 'map1.py' defines a function 'calculaQuadrado' that takes a number and returns a formatted string of its square. It then creates a list 'lista\_numeros' with values [1, 2, 3, 4, 5, 6], uses 'map' to apply 'calculaQuadrado' to each element, and prints the results in a loop. The 'Run' output shows the execution of the script, displaying the squares of the numbers 1 through 6.

```
1 def calculaQuadrado(numero):  
2     return f"O quadrado de {numero} é: {numero * numero}"  
3  
4 lista_numeros = [1, 2, 3, 4, 5, 6]  
5 calculo = map(calculaQuadrado, lista_numeros)  
6  
7 for n in calculo:  
8     print(n)
```

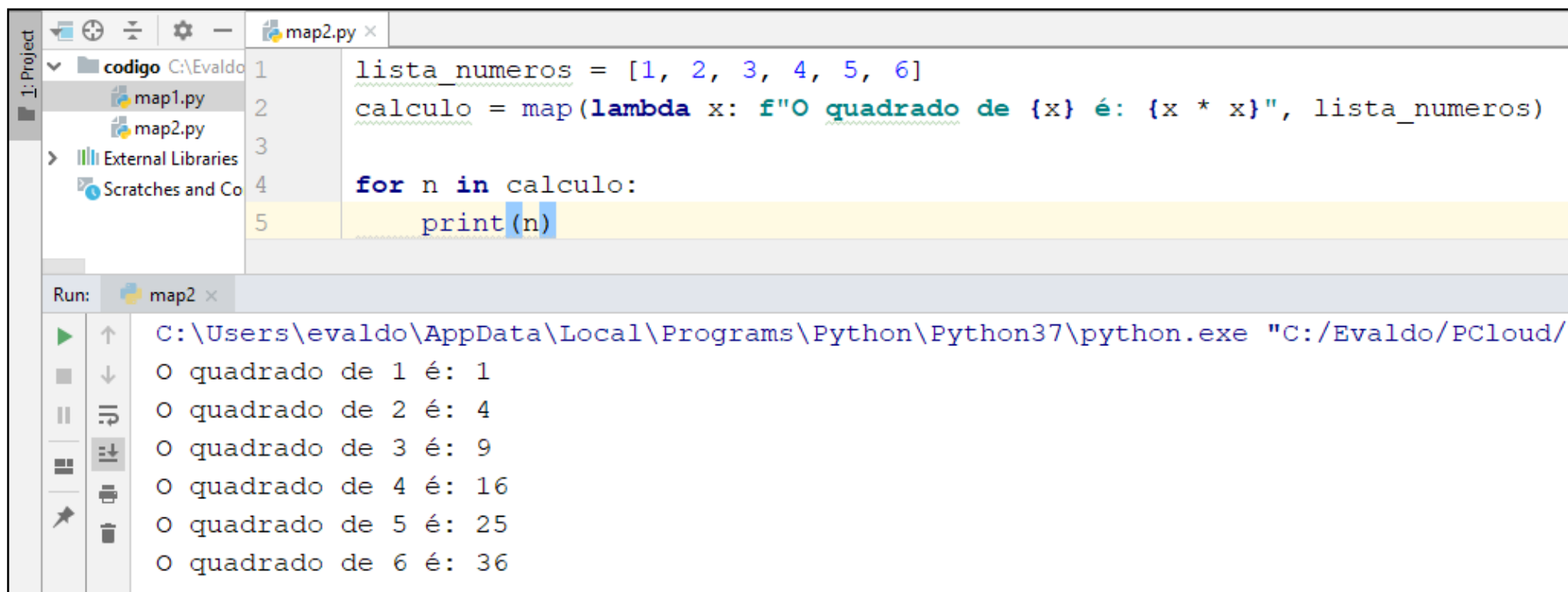
Run: map1 x

C:\Users\evaldo\AppData\Local\Programs\Python\Python37\python.exe "C:/.../  
O quadrado de 1 é: 1  
O quadrado de 2 é: 4  
O quadrado de 3 é: 9  
O quadrado de 4 é: 16  
O quadrado de 5 é: 25  
O quadrado de 6 é: 36

# Funções, módulos e pacotes

## Mapeamento – Função map()

Podemos usar também expressões *lambda* como função para a função map().



```
1 lista_numeros = [1, 2, 3, 4, 5, 6]
2 calculo = map(lambda x: f"O quadrado de {x} é: {x * x}", lista_numeros)
3
4 for n in calculo:
5     print(n)
```

Run: map2 x

C:\Users\evaldo\AppData\Local\Programs\Python\Python37\python.exe "C:/Evaldo/PCloud/t

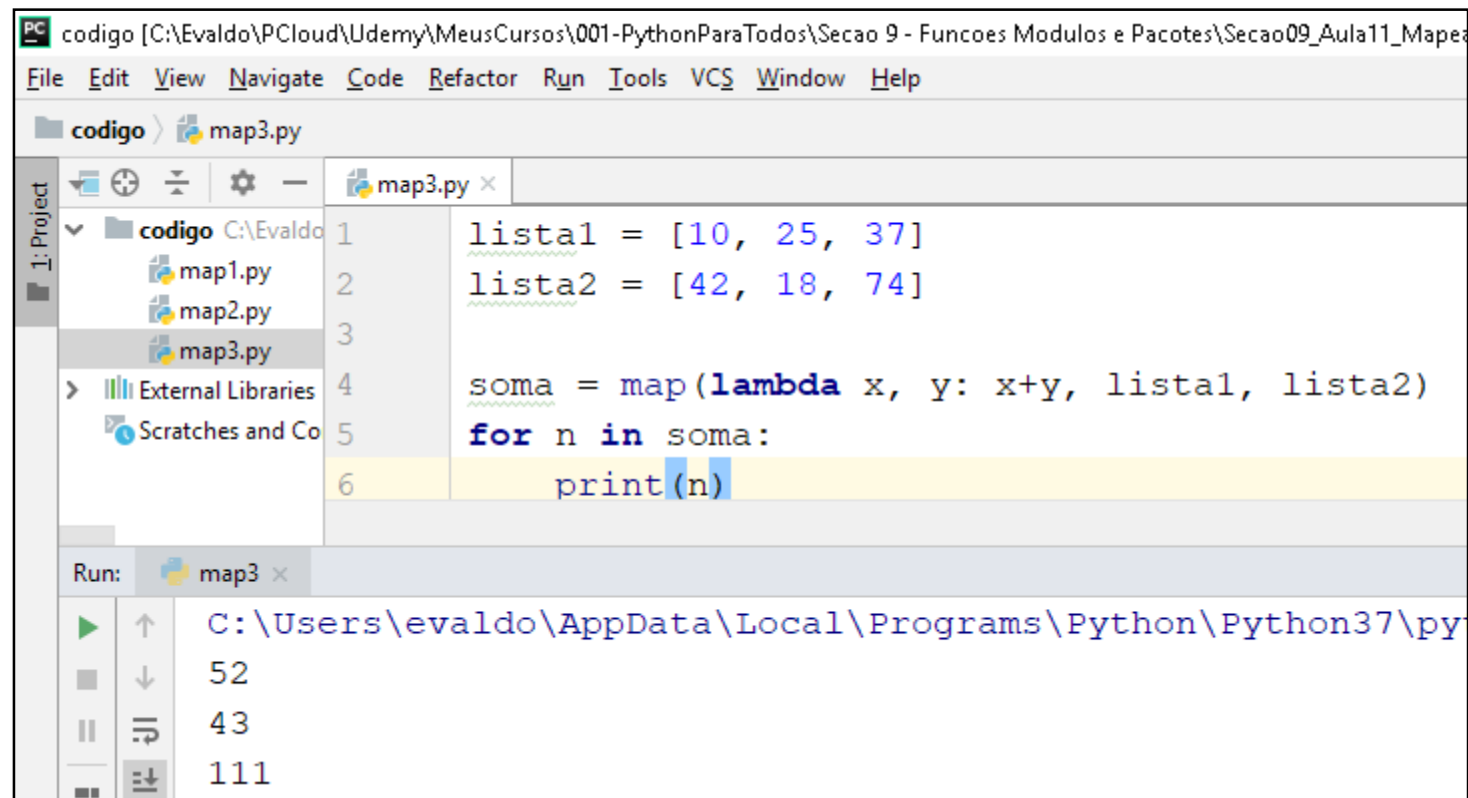
O quadrado de 1 é: 1  
O quadrado de 2 é: 4  
O quadrado de 3 é: 9  
O quadrado de 4 é: 16  
O quadrado de 5 é: 25  
O quadrado de 6 é: 36



# Funções, módulos e pacotes

## Mapeamento – Função map()

Podemos passar mais de um iterador para o map().



The screenshot shows a Python IDE with a project named 'codigo' containing three files: 'map1.py', 'map2.py', and 'map3.py'. The 'map3.py' file is open and contains the following code:

```
1 lista1 = [10, 25, 37]
2 lista2 = [42, 18, 74]
3
4 soma = map(lambda x, y: x+y, lista1, lista2)
5 for n in soma:
6     print(n)
```

The code is executed, and the output is displayed in the Run console:

```
Run: map3 x
C:\Users\evaldo\AppData\Local\Programs\Python\Python37\py
52
43
111
```

# FIM