
Ricardo Sérgio
Introdução à Programação e Algoritmia
API B – 12º Ano
MacBook Pro

Python

```
10 {

    #Funções

}
```


Ricardo Sérgio

Funções{

- As funções podem ser muito úteis para evitar repetição de código
- Forma otimizada de organizar o código
- Imaginem um jogo com 5 inimigos iguais. Todos com a mesma quantidade de vida, a mesma forma de morrer e o mesmo movimento no cenário de jogo.
- Para definir todas estas ações teríamos de repetir código para cada um deles.
- Com recurso a funções basta chamar no código a função respetiva e a mesma pode ser chamada quantas vezes quisermos sem repetir código. Recursividade.

}

Funções{

- As funções podem:

  - ✓ **Aceitar ou Não aceitar parâmetros (argumentos)**
  - ✓ **Retornar ou não valores**

- Devemos definir as funções na parte superior do nosso programa de forma a ficarem organizadas

}

Funções{

- Sintaxe:

- Sem argumentos:

```
def nome_função():
    Conteúdo da função aqui
    [return] valor
```

- Com argumentos:

```
def nome_função(argumento1, argumento2,…,argumentoN):
    Conteúdo da função aqui
    [return] valor
```

}

Funções{

- Chamada? Como chamar a função?

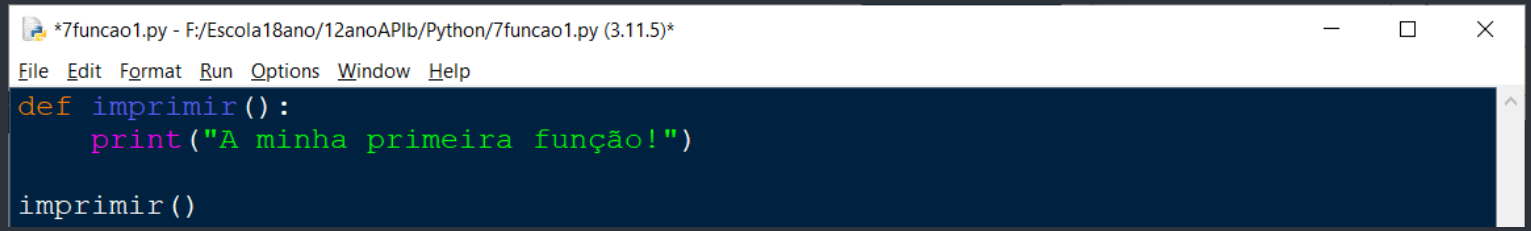
- Fácil, em qualquer parte do programa basta digitar o nome da função

- Exemplo:

```
def teste():
    Conteúdo da função aqui
(…)
teste() #chamada da função. Permite ser executada nesta linha de código
}
```

Funções{

- Exemplo sem argumentos:



- Resultado:

```
============= RESTART: F:/Escola18ano/12anoAPIb/Python/7funcao1.py =============
A minha primeira função!
```

}

Funções{

Exemplo com argumentos:

7funcaocomarg.py - F:/Escola18ano/12anoAPIb/Python/7funcaocomarg.py (3.11.5)

File Edit Format Run Options Window Help

```
def somar(a,b):
    c=a+b
    print("A soma de "+str(a)+" com "+str(b)+" é igual a "+str(c))

somar(2,3)
```

Resultado:

IDLE Shell 3.11.5

File Edit Shell Debug Options Window Help

```
>>> 
========== RESTART: F:/Escola18ano/12anoAPIb/Python/7funcaocomarg.py ==========
A soma de 2 com 3 é igual a 5
>>> 
```

Ln: 56 Col: 0

}

Funções{

Exemplo com retorno:

7funcaocomarg.py - F:/Escola18ano/12anoAPIb/Python/7funcaocomarg.py (3.11.5)

File Edit Format Run Options Window Help

```
def somar(a,b):
    c=a+b
    return c

x=2
y=3
print("A soma de "+str(x)+" com "+str(y)+" é igual a "+str(somar(x,y)))
```

Retorna valor aqui

Resultado:

IDLE Shell 3.11.5

File Edit Shell Debug Options Window Help

```
>>> ========== RESTART: F:/Escola18ano/12anoAPIb/Python/7funcaocomarg.py ==========
A soma de 2 com 3 é igual a 5
>>>
```

Ln: 65 Col: 0

}

Variáveis locais e globais{

- As variáveis definidas até agora foram sempre locais
- As variáveis locais só existem no local onde são definidas
- Se definidas dentro de uma função apenas existem aí
- Se definidas fora de uma função, em Python, existem em qualquer lugar
- Porém, ao serem manipuladas fora do programa principal, mantêm o valor definido no programa principal

}

Variáveis locais e globais{

Exemplo:

7funcaogloballocal.py - F:/Escola18ano/12anoAPIb/Python/7funcaogloballocal.py (3.11.5)

File Edit Format Run Options Window Help

```
def valor():
    print(x)

x="Eu sou global, mas só posso ser alterada no programa principal"
valor()
```

Resultado:

IDLE Shell 3.11.5

File Edit Shell Debug Options Window Help

```
>>> ======== RESTART: F:/Escola18ano/12anoAPIb/Python/7funcaogloballocal.py ========
Eu sou global, mas só posso ser alterada no programa principal
>>>
```

Ln: 106 Col: 0

}

Variáveis locais e globais{

Exemplo:

7funcaogloballocal.py - F:/Escola18ano/12anoAPIb/Python/7funcaogloballocal.py (3.11.5)

File Edit Format Run Options Window Help

```
def valor():
    x="Tento alterar o valor de x do programa principal"
    print(x)


x="Eu sou global, mas só posso ser alterada no programa principal"
valor()
print(x)
```

Resultado:

IDLE Shell 3.11.5

File Edit Shell Debug Options Window Help

```
Tento alterar o valor de x do programa principal
Eu sou global, mas só posso ser alterada no programa principal
>>>
```

Ln: 117 Col: 0

}

Variáveis locais e globais{

Exemplo:

7funcaogloballocal.py - F:/Escola18ano/12anoAPIb/Python/7funcaogloballocal.py (3.11.5)

File Edit Format Run Options Window Help

```
def valor():
    global x #forma de aceder à variável global e a poder alterar
    x="Tento alterar o valor de x do programa principal"
    print(x)

x="Eu sou global, mas só posso ser alterada no programa principal"
valor()
print(x)
```

Resultado:

IDLE Shell 3.11.5

File Edit Shell Debug Options Window Help

```
Tento alterar o valor de x do programa principal
Tento alterar o valor de x do programa principal
>>>
```

Ln: 121 Col: 0

}

Variáveis locais e globais{

Exemplo:

7funcaogloballocal.py - F:/Escola18ano/12anoAPIb/Python/7funcaogloballocal.py (3.11.5)

File Edit Format Run Options Window Help

```
def valor():
    global x #forma de aceder à variável global e a poder alterar
    x="Tento alterar o valor de x do programa principal"
    print(x)
    y="Eu sou local e só existo aqui"

x="Eu sou global, mas só posso ser alterada no programa principal"
valor()
print(x)
print(y)
```

Resultado:

```
Tento alterar o valor de x do programa principal
Tento alterar o valor de x do programa principal
Traceback (most recent call last):
  File "F:/Escola18ano/12anoAPIb/Python/7funcaogloballocal.py", line 10, in <mod
ule>
    print(y)
NameError: name 'y' is not defined
```

}

```
1
2
3
4   Para Hoje{
5
6
7
8
9
10
11
12
13  }
14
```