

PYTHON FUNCTIONS

```
def add_numbers(a, b):  
    return a + b
```

```
def greet(name='world'):  
    print('Hello', name)
```

ALGORITMIA E ESTRUTURAS DE DADOS

DECOMPOSIÇÃO MODULAR: MÉTODOS & FUNÇÕES

```
print('Hello world!')
```

LICENCIATURA EM

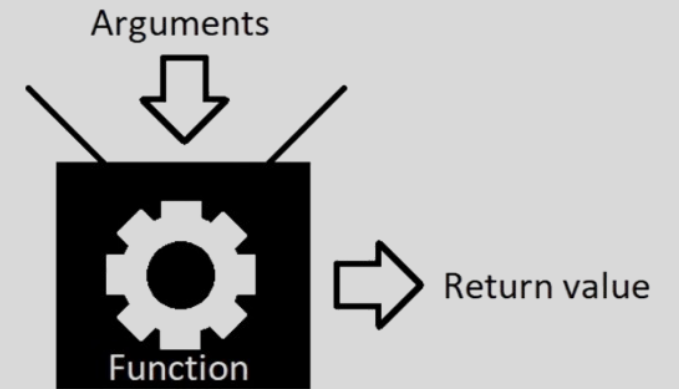
TECNOLOGIAS E SISTEMAS DE INFORMAÇÃO PARA A WEB

#ESMAD #P.PORTO

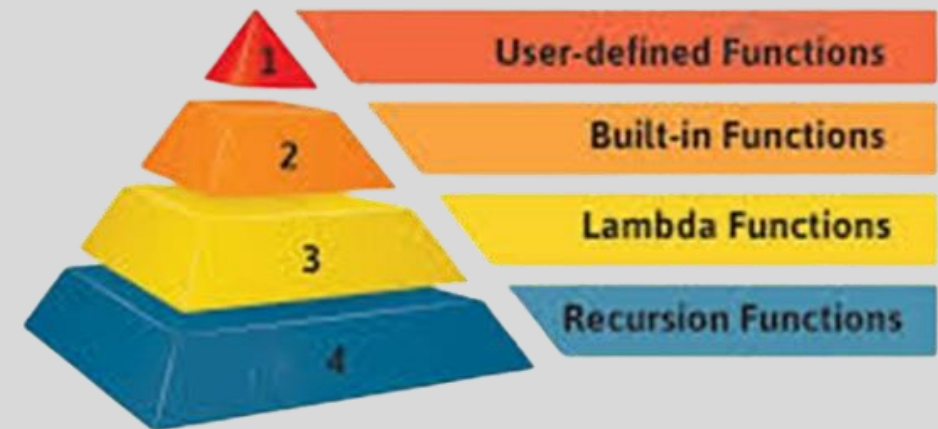
PYTHON LAND

Decomposição Modular – Funções

- ❑ Conceito
- ❑ Criar uma Função
- ❑ Devolução de dados (*return*)
- ❑ Parâmetros com valor por defeito
- ❑ Número de parâmetros de entrada indefinido



Python Functions

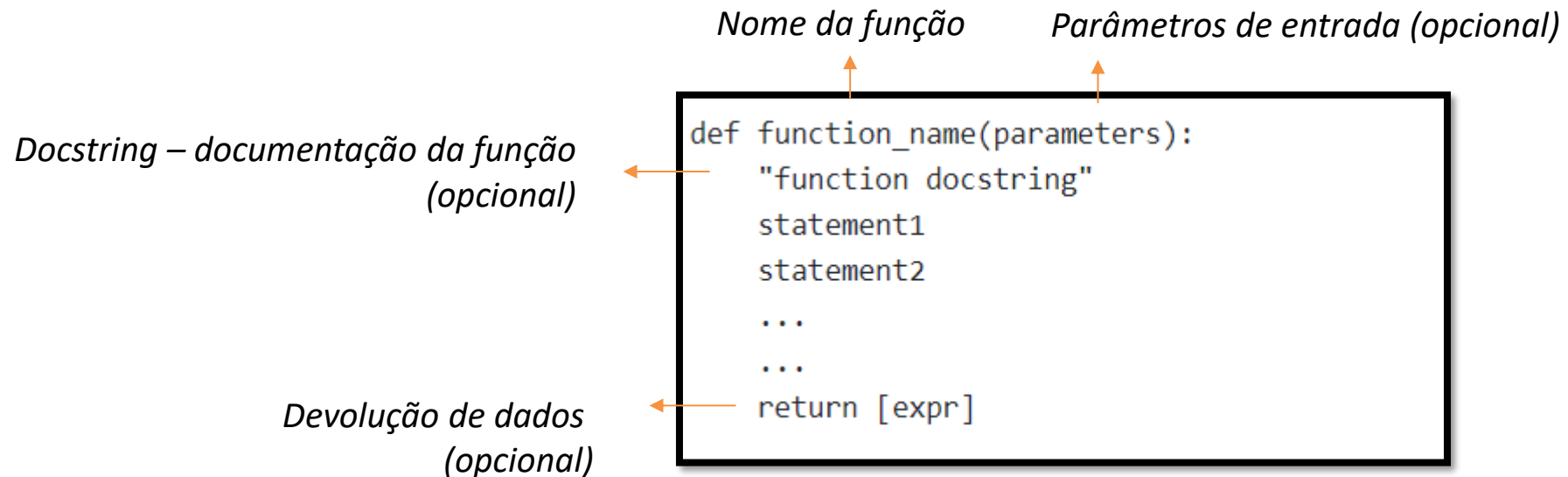


❖ Funções | Conceito

- ❑ Conjunto de código (ou bloco de instruções), delimitado de forma clara, e que executa uma tarefa específica
- ❑ Uma função comporta-se da mesma forma que um programa, embora numa escala diferente:
 - ❑ É executada quando invocada pelo programa que o chama;
 - ❑ Quando termina, devolve a execução do programa, a partir do local onde foi chamada.
- ❑ O uso de funções permite desenvolver código de forma estruturada, facilitando:
 - ❑ Reutilização do código
 - ❑ Leitura / legibilidade do código
 - ❑ Abstração do código

❖ Funções | Conceito

- ❑ Uma função é um bloco de código reutilizável, projetado para executar uma determinada tarefa.
- ❑ Para definir uma função, o Python fornece a keyword **def**. A seguir está a sintaxe para definir uma função



❖ Funções | Conceito

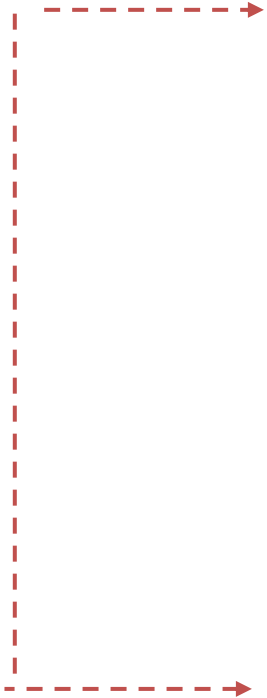
- ❑ As variáveis definidas no âmbito de uma função são **variáveis locais** - existem apenas dentro dessa função, em contraponto a **variáveis globais** - existem durante toda a execução do programa

Variáveis locais
(existem apenas dentro
da função)

Variável global

```
1 def fatorial(num):  
2     """  
3     recebe um número como argumento, calculando o sue fatorial  
4     Args: num: - número inteiro  
5     """  
6     if num >= 0:  
7         fatorial =1  
8         for i in range (num, 1, -1):  
9             fatorial*=i  
10        print("Fatorial de {:n} é {:n}" .format(num, fatorial))  
11    else:  
12        print("Argumento não pode ser um inteiro negativo")  
13  
14  
15 numero = int(input("Indique um número:"))  
16 fatorial(numero)  
17
```

❖ Criar uma Função



```
1 def fatorial(num):
2     """
3     recebe um número como argumento, calculando o sue fatorial
4     Args: num: - número inteiro
5     """
6     if num >= 0:
7         fatorial = 1
8         for i in range (num, 1, -1):
9             fatorial*=i
10        print("Fatorial de {:n} é {:n}" .format(num, fatorial))
11    else:
12        print("Argumento não pode ser um inteiro negativo")
13
14
15 numero = int(input("Indique um número:"))
16 fatorial(numero)
17
```

1. Início da execução do código

2. Invoca a função fatorial

❖ Criar uma Função

Docstring – documentação da função
(opcional) ←

- `"""`
- *texto a documentar objetivo da função*
- *Args:*
- *Returns:*
- `"""`

```
1 def fatorial(num):
2     """
3     recebe um número como argumento, calculando o sue fatorial
4     Args: num: - número inteiro
5     """
6     if num >= 0:
7         fatorial =1
8         for i in range (num, 1, -1):
9             fatorial*=i
10        print("Fatorial de {:n} é {:n}" .format(num, fatorial))
11    else:
12        print("Argumento não pode ser um inteiro negativo")
13
14
15 numero = int(input("Indique um número:"))
16 fatorial(numero)
17
```

```
12 def fatorial(num)
13     recebe um número como argumento, calculando o sue fatorial
14     Args: num: - número inteiro
15
16     Full name: Exemplo.fatorial
17 fatorial(numero)
18
```

❖ Criar uma Função



Por defeito, uma função deve ser chamada com o número correto de argumentos. O que significa que se sua função **recebe** 3 argumentos, devo invocar a função com 3 argumentos

```
def soma(num1, num2, num3)  
    Recebe três números e calcula a sua soma e a média  
    Args: num, num2, num3: - números inteiro
```

Full name: Exemplo.soma

[View Problem \(Alt+F8\)](#) [Quick Fix... \(Ctrl+.\)](#)

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

```
1 def soma(num1, num2, num3):  
2     """  
3     Recebe três números e calcula a sua soma e a média  
4     Args: num, num2, num3: - números inteiro  
5     """  
6     somaNumeros = num1 + num2 + num3  
7     print("A soma é {:n}" .format(somaNumeros))  
8     print("A média é {:n}" .format(somaNumeros/3))  
9  
10 soma(10,20,30)
```




❖ Devolução de dados | return

Quando uma função **devolve um valor** (keyword **return**), ao invocarmos a função devemos atribuir a função a uma variável

```
1 def primo(numero):
2     """
3     Recebe um número e devolve um valor boolean: True se o número é primo. False se não
4     Args: numero inteiro
5     Returns: booleano (True or False)
6     """
7     primo = True
8     for i in range(2, numero): # o divisor varia entre 2 e o numero-1
9         if numero % i == 0: # quando encontro um resto 0 => não é primo
10             primo = False
11             break
12     return primo
13
14
15 numero = int(input("Indique um número:"))
16 estado = primo(numero)
17 if estado == True:
18     print("O número {0} é primo".format(numero))
19 else:
20     print("O número {0} não é primo".format(numero))
21
22
```

Devolve um valor no final da execução da função

Função devolve valor

❖ Devolução de dados | *return*

```
1 def fatorial(num):  
2     """  
3     recebe um número como argumento, calculando o sue fatorial  
4     Args: num: - número inteiro  
5     """  
6     if num >= 0:  
7         fatorial =1  
8         for i in range (num, 1, -1):  
9             fatorial*=i  
10        print("Fatorial de {:n} é {:n}" .format(num, fatorial))  
11    else:  
12        print("Argumento não pode ser um inteiro negativo")  
13  
14  
15 numero = int(input("Indique um número:"))  
16 fatorial(numero)  
17
```

Exemplo de função fatorial
sem devolução de resultado

```
1  
2 def fatorial(num):  
3     """  
4     recebe um número como argumento, calculando o sue fatorial  
5     Args: num: - número inteiro  
6     """  
7     if num >= 0:  
8         fatorial = 1  
9         for i in range (num, 1, -1):  
10            fatorial*=i  
11            return fatorial  
12    else:  
13        return "Argumento inválido"  
14  
15  
16 numero = int(input("Indique um número:"))  
17 print("Fatorial de {:n} é {:n}" .format(numero, fatorial(numero)))  
18
```

Exemplo de função fatorial
com devolução de resultado

❖ Parâmetros com valor por defeito



Parâmetros definidos por defeito podem ser omitidos quando chamo a função

```
1 def fatorial(num=0):  → Valor por defeito, quando invoco a
2     """              função sem passar argumento
3     recebe um número como argumento, calculando o sue fatorial
4     Args: num: - número inteiro
5     """
6     if num >= 0:
7         fatorial = 1
8         for i in range (num, 1, -1):
9             fatorial*=i
10        return fatorial
11    else:
12        return "Argumento inválido"
13
14
15
16 print("Fatorial de {:n} é {:n}" .format(5, fatorial(5)))
17
18 print("Fatorial de {:n} é {:n}" .format(4, fatorial(4)))
19
20 print("Fatorial de {:n} é {:n}" .format(0, fatorial())) →
```

```
Fatorial de 5 é 120
Fatorial de 4 é 24
Fatorial de 0 é 1
Press any key to continue . . . |
```

Invoco a função fatorial sem passar qualquer valor

❖ Parâmetros com valor por defeito

```
2 def fatorial(num):  
6     """  
7     if num >= 0:  
8         fatorial = 1  
9         for i in range (num, 1, -1):  
10            fatorial*=i  
11            return fatorial  
12     else:  
13         return "Argumento inválido"  
14  
15  
16  
17     print("Fatorial de {:n} é {:n}".format(5, fatorial(5)))  
18  
19     print("Fatorial de {:n} é {:n}".format(4, fatorial(4)))  
20  
21     print("Fatorial de {:n} é {:n}".format(0, fatorial()))
```

Neste caso dá erro, pois o parâmetro de entrada não contém nenhum valor por defeito

Exception has occurred: TypeError ×
fatorial() missing 1 required positional argument: 'num'
File "C:\Users\mario\OneDrive\AED\2023-24\4 - Exercicios\Ficha 04\Exemplo.py", line 21, in <module>
 print("Fatorial de {:n} é {:n}" .format(0, fatorial()))
 ^^^^^^^^^^^
TypeError: fatorial() missing 1 required positional argument: 'num'

❖ Parâmetro: tipo de dados



Definir tipo de dados aceite como argumento da função

```
1
2 def fatorial(num:int):
3     """
4     recebe um número como argumento, calculando o sue fatorial
5     Args: num: - número inteiro
6     """
7     if num >= 0:
8         fatorial = 1
9         for i in range (num, 1, -1):
10             fatorial*=i
11         return fatorial
12     else:
13         return "Argumento inválido"
14
15
16
```

```
def fatorial(num: int)
recebe um número como argumento, calculando o sue fatorial
Args: num: - número inteiro
Full name: Exemplo.fatorial
```

```
l de {:n} é {:n}" .format(5, fatorial(5)))
l de {:n} é {:n}" .format(4, fatorial(4)))
l de {:n} é {:n}" .format(0, fatorial()))
```

❖ Parâmetro: tipo de dados



Definir tipo de dados aceite como argumento da função

```
1 def primo(numero:int):  
2     """  
3     Recebe um número e devolve um valor boolean: True se o número é primo. False se não for  
4     Args: numero inteiro  
5     Returns: booleano (True or False)  
6     """  
7     primo = True  
8     for i in range(2, numero): # o divisor varia entre 2 e o numero-1  
9         if numero % i == 0: # quando encontro um resto 0 => não é primo  
10             primo = False  
11             break  
12     return primo  
13
```

❖ Numero de parâmetros de entrada indefinido

função len() conta o número de argumentos que foram passados para a função

```
1
2 def soma(*numeros):
3     """
4     recebe uma lista de números, calculando o sua soma
5     Args: numeros: - número inteiro
6     Return: inteiro com soma dos número
7     """
8     soma = 0
9     for i in range (len(numeros)):
10         soma+= numeros[i]
11     return soma
12
13 print(soma(10,20))
14 print(soma(10,20,30, 40))
```

```
30
100
Press any key to continue . . .
```

❖ Funções

Avalia o teu conhecimento

❑ Função 1

Implemente a função **somatório** que receba 2 números inteiros, como argumento de entrada. A função deve **imprimir o somatório** de todos os números inteiros incluídos nesse intervalo

```
somatorio(1,3)      # imprime somatorio de numeros inteiros entre [1-3]  
somatorio(3,6)      # imprime somatorio de numeros inteiros entre [3-6]
```

❑ Função 2

Implemente uma função designada **maior** que receba **n** números inteiros positivos (n é variável, dependendo dos argumentos fornecidos à função)

A função deve devolver (return) o maior desses números

Exemplos da chamada da função:

```
maior (10,20)  
maior (10, 30, 20)  
maior (5, 15, 20, 14)
```


❖ Funções

Avalia o teu conhecimento

❑ Função 3

Implemente a função **heartRate(fc)** que receba a frequência cardíaca de um indivíduo durante o treino, e que **devolva (return)** o nível de esforço efetuado, de acordo com as seguintes condições:

- fc entre [50-80] – treino aeróbico
- Fc entre]80-100] – treino cardiovascular
- Fc entre]100-120] – treino aeróbico ideal
- Fc entre]120-100] – treino anaeróbico

Exemplo da chamada da função:

```
print(heartRate(118))  
print(heartRate(91))
```