

# Disciplina: Linguagem de Programação

## Função

Prof<sup>a</sup>. Dr<sup>a</sup>. Giovana Angélica Ros Miola  
giovana.miola@fatec.sp.gov.br

**Fatec**  
Presidente Prudente

**CPS**  
Centro  
Paula Souza

# Função/Método

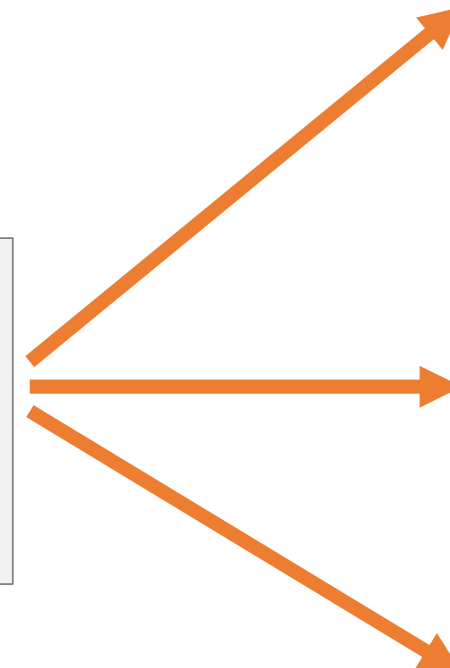
- Sempre é possível **dividir** um problema grande em problemas menores.
- Desta maneira, cada parte menor tem um algoritmo mais simples. Esta parte menor do programa, é chamada de **função**.
- Uma função é uma **parte** de um programa (ou um **módulo**), que pode efetuar diversas operações computacionais, tais como: entrada, processamento e saída).
- A função é criada para cumprir uma **tarefa** particular.
- Uma das principais razões para o uso de funções, é **evitar** que programadores escrevam o mesmo código **diversas** vezes.
- Você já utilizou algumas funções na linguagem Python, como:
  - `print()`
  - `input()`
  - `float()`
  - `int()`
- Agora chegou a hora de **você** aprender a criar as suas próprias funções.

# Função/Método

- Considere o seguinte trecho de código:

```
...  
funcionarios = 0  
somaSalarios = 0  
while funcionarios < 15:  
    salario= float(input("Informe o salário: "))  
    somaSalarios = somaSalarios + salario  
    funcionarios = funcionarios + 1  
...
```

- Agora imagine um programa em que você utilize esta lógica de somar salários em vários lugares.



```
... Alguma lógica  
...  
funcionarios = 0  
somaSalarios = 0  
while (funcionários < 15):  
    salario= float(input("Informe o  
salário: "))  
    somaSalarios = somaSalarios +  
salario  
    funcionarios = funcionarios + 1  
...  
...  
... Alguma lógica  
...  
funcionarios = 0  
somaSalarios = 0  
while (funcionários < 15):  
    salario= float(input("Informe o  
salário: "))  
    somaSalarios = somaSalarios +  
salario  
    funcionarios = funcionarios + 1  
...  
...  
... Alguma lógica  
...  
funcionarios = 0  
somaSalarios = 0  
while (funcionários < 15):  
    salario= float(input("Informe o  
salário: "))  
    somaSalarios = somaSalarios +  
salario  
    funcionarios = funcionarios + 1  
...  
...  
... Alguma lógica  
...  
etc
```

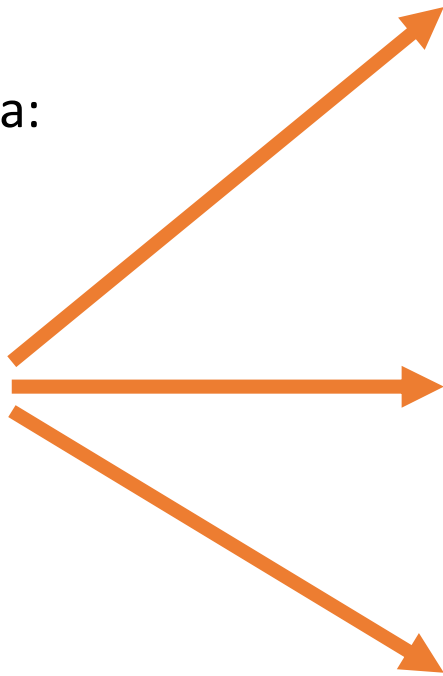
# Função/Método

- Exemplo de uma função e sua chamada no programa:

```
def somar_salarios () :  
    funcionarios = 0  
    somaSalarios = 0  
    while funcionarios < 20:  
        salario = float(input("Informe o salário: "))  
        somaSalarios = somaSalarios + salario  
        funcionarios = funcionarios + 1
```

- A definição/criação de uma função é feita apenas UMA vez e pode ser chamada no programa tantas vezes quanto for necessária.

```
... Alguma lógica  
...  
...  
...  
...  
somar_salarios()  
...  
... Alguma lógica  
...  
...  
...  
...  
somar_salarios()  
...  
...  
... Alguma lógica  
...  
...  
somar_salarios()  
...  
...  
... Alguma lógica  
...  
etc
```



# Função/Método

- **Vantagens:**

- Modularização
- Eliminação de redundância de código fonte
- Agilidade de manutenção
- Reutilização de código fonte

# Função/Método

- Uma função em Python tem a seguinte forma geral:

```
def tipo nome_função (parâmetros) :  
    corpo da função.....  
    lógica de programação.....  
    corpo da função.....  
...
```

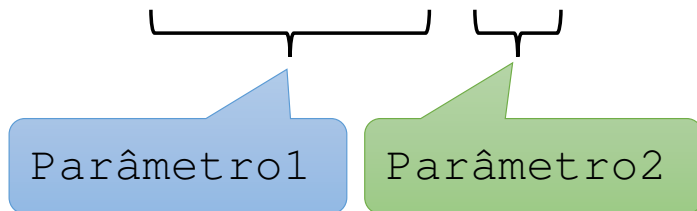
*tipo* é o tipo de valor que a função vai retornar (int, float, str, ...)

*nome\_função* é o nome dado a sua função, por exemplo: calcular\_salários(),  
somar\_dois\_números(), gerar\_relatório\_financeiro()

*parâmetros* são variáveis, quando tem mais que uma, separa-se por vírgula

# Função/Método

- Os parênteses são utilizados para separar os parâmetros de uma função.
- Esses parênteses são requeridos pelo interpretador Python.
- Um parâmetro é um valor passado para uma função, por exemplo na função **print**, a mensagem a imprimir.
- `print( 'O montante é', total)`



- Não precisa obrigatoriamente de uma função main() – embora seja uma boa prática ter uma.

# Formas de criação das funções

**nome\_funcao ( )** # sem retorno, sem parâmetro

**nome\_função ( parâmetros )** # sem retorno, com parâmetro

**nome\_função ( )** # com retorno, sem parâmetro

**nome\_funcao ( parâmetros )** # com retorno, com parâmetro



# Função sem retorno e sem parâmetro

- Função **sem parâmetro**, significa que **não é necessário ter nenhum valor externo (não venha nada da função main)** para que seja executada a lógica.
- Sem retorno indica que a lógica da função não necessita entregar algum valor para a função main(). **O que for determinado como lógica é realizado dentro da função.**

Na linguagem C#, por exemplo é assim:

```
void mensagem() {  
    ...  
}
```

```
# Função em Python que exibe  
# uma mensagem  
def mensagem():  
    print("Olá, tudo bem?")
```

- Em inglês, **void** quer dizer **vazio**.
- **void** em várias linguagens de programação é considerado um tipo, ele nos permite, definir funções que não retornam nada.

**Em Python não tem que escrever nenhum tipo, antes do nome da função, pois ocorre a tipagem dinamicamente**

# Função sem retorno e sem parâmetro

Sem valor  
Sem variável  
Sem parâmetro

```
def mensagem():  
    print("Olá, tudo bem!")  
  
def main():  
    mensagem()  
  
main()
```

Resultado apresentado:

>>> Olá, tudo bem?

Chamada da função main/principal, nela **centraliza-se** a chamada de todas as outras funções do programa

# Atividade

- Elabore uma função chamada *divisores1* que leia um inteiro (do usuário) e imprima seus divisores.
- Elabore também a função *main()* com a respectiva chamada à função e que pergunte se o usuário deseja informar um novo número.

```
def divisores1():  
    num = int(input('Informe um número: '))  
    for i in range(1, num + 1):  
        if num % i == 0:  
            print(num, ' é divisível por ', i)
```

```
def main():  
    divisores1()
```

```
main()  
'''Executa a função APENAS uma vez'''
```

```
Informe um número: 10  
10 é divisível por 5  
10 é divisível por 2
```

```
def divisores2():  
    num = int(input('Informe um número: '))  
    for i in range(1, num + 1):  
        if num % i == 0:  
            print(num, ' é divisível por ', i)
```

```
def main():  
    resp = 's'  
    while resp == 'S' or resp == 's':  
        divisores2()  
        resp = input('Outro número (S/N)? ')
```

```
main()  
'''Executa a função VÁRIAS vezes'''
```

```
Informe um número: 4  
4 é divisível por 2  
Outro número (S/N?) s  
Informe um número: 6  
6 é divisível por 3  
6 é divisível por 2  
Outro número (S/N?)
```

# Função sem retorno e com parâmetro

- Numa função os parâmetros viabilizam entradas e saídas de valores.
- Além da entrada pelo usuário (digitação), uma função (sub-algoritmo/sub-rotina) pode receber um ou vários valores, por meio da **passagem de parâmetro, neste caso passagem**

- Exemplo:

- `print ( 'Olá, tudo bem?' )`

  
**Parâmetro**

`import math`

`math.pow( base, expoente )`

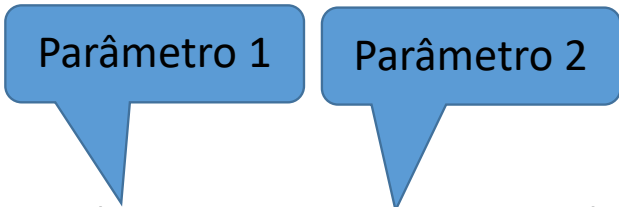
**1º. parâmetro**

**2º. parâmetro**

A seta representa a **entrada de valores** na nesta função. Estes valores são passados como **argumentos** na **chamada da função**, por exemplo, pela `main()`

# Parâmetros x Argumentos

- Na **chamada de uma função**, quando necessário, é fornecido valores concretos, chamados de **argumentos**, que será repassado aos parâmetros.
- Os argumentos devem ser **compatíveis** com o **tipo de parâmetro**, mas o nome do argumento usado no código de chamada, **não precisa ser o mesmo que o parâmetro nomeado definido no método**.
- Exemplo de **passagem de parâmetro por valor, na função `somar(numero1, numero2)`**



```
def somar(numero1, numero2):  
    r = numero1 + numero2  
    print("O resultado é", r)
```



```
def main():  
    a = 5  
    b = 7  
    somar( a, b )
```

main()

# Valores Padronizados na Passagem de Parâmetro

- Valores padronizados podem ser ou são usados na passagem de parâmetro numa função em Python
- Importante, o valor padrão do parâmetro é avaliado no momento da **declaração** da função, e não na **sua chamada**.

- Exemplo:

```
def salario_descontado_imposto(salario, imposto = 15):  
    res = salario - (salario * imposto / 100)  
    print('Salário líquido:', res)
```

- Esse **valor padrão, reflete diretamente na chamada**:

```
>>> salario_descontado_imposto(5000)  
>>> Salário líquido: 3650.0
```

```
def somar(a, b = 10):  
    res = a + b  
    print('A soma é ', res) # 12  
  
def main():  
    somar(2)  
  
main()
```

# Dois Tipos de Passagem de Parâmetro

- 1. Passagem de parâmetro POR VALOR:** Envia valores/variáveis para a função, mas estes valores não são vistos/obtidos fora dela, ou seja, os valores apenas entram numa função para atendimento de uma lógica.
  - 2. Passagem de parâmetro POR REFERÊNCIA:** Envia valores/variáveis para a função, utiliza-se de alguma lógica que altere-os e estes valores são vistos/obtidos fora dela, ou seja, esse valores entram e saem de uma função sem a utilização da instrução return. Vetor, matriz, estrutura(estudaremos posteriormente), entre outros, são de referência.
- A diferença entre os dois tipos de passagem de parâmetro, é que alguns valores são acessados de forma direta (por valor) e outros por meio de uma referência (endereço de memória), tais como, lista, tupla, dicionários, que em Python são objetos de dados.

# Passagem de Parâmetro por Referência

```
def inserir_vetor(v):
```

```
    v.append(56)
```

```
    v.append(78)
```

```
    v.append(99)
```

```
def main():
```

```
    vet = []
```

```
    print(vet)
```

```
    inserir_vetor(vet)
```

```
    print(vet)
```

```
main()
```

Resultado:

>>> []

>>> [56, 78, 99]

Nome da variável	Endereço de armazenamento <b>referência</b>	Conteúdo
p1	0001	9.5
	0002	
	0003	
	...	
ano	0005	2000
	0006	
	...	
	...	
vet[0]	0100	56
vet[1]	0104	78
vet[2]	0108	99



# Função com retorno sem parâmetro

- Uma função, em seu objetivo, pode precisar devolver/retornar um valor a quem, a chamou, por exemplo a main() ou uma outra função, e dar sequência a lógica. Para esta situação dentro da função deve-se usar a palavra reservada **return**.
- O comando **return** tem a seguinte sintaxe:  
**return** valor\_retorno
- Quando uma função é executada e encontra o comando **return**, ela é **encerrada imediatamente** e devolve o valor de retorno a main, ou a qualquer outra função

# Função com retorno e sem parâmetro

Quando não tem parâmetro, é provável ter que digitar valores, depende da lógica requerida

**def** somar():

```
n1 = int(input('Digite o primeiro número: '))
n2 = int(input('Digite o primeiro número: '))
resultado = n1 + n2
return resultado
```

Não tem argumento

**def** main():

```
print( 'A soma é', somar() )
```

main()

A função somar() é chamada **dentro** da função print, para apresentar o valor numérico resultante do cálculo

ou

**def** somar():

```
n1 = int(input('Digite o primeiro número: '))
n2 = int(input('Digite o primeiro número: '))
resultado = n1 + n2
return resultado
```

Com a chamada da função somar(), é atribuído o valor numérico resultante a variável x, que posteriormente é apresentado na frase

**def** main():

```
x = somar()
print( 'A soma é', x )
```

main()

Dois  
parâmetros

# Função com retorno e com parâmetro

```
def somar(n1, n2):  
    resultado = n1 + n2  
    return resultado
```

```
def main():  
    a = int(input('Digite o primeiro número: '))  
    b = int(input('Digite o primeiro número: '))  
    print( 'A soma é', somar(a, b) )
```

main()

ou



```
def somar(n1, n2):  
    n1 = int(input('Digite o primeiro número: '))  
    n2 = int(input('Digite o primeiro número: '))  
    resultado = n1 + n2  
    return resultado
```

```
def main():  
    a = int(input('Digite o primeiro número: '))  
    b = int(input('Digite o primeiro número: '))  
    x = somar()  
    print( 'A soma é', x )
```

main()

Dois  
argumentos

# Função – return sem valor

```
''' Lê 2 números e exibe a divisão'''  
  
def dividir():  
    a = float(input("Informe um número: "))  
    b = float(input("Informe um número: "))  
    if b == 0.0:  
        print("Divisão por zero!");  
        return  
    print("Resultado da divisão: ", a / b)  
  
def main():  
    dividir()  
  
main()
```

Informe um número: 8  
Informe um número: 0  
Divisão por zero!

Informe um número: 6  
Informe um número: 3  
Resultado da divisão:2.0

# Função – return com valor

```
def dividir() :  
    a = float(input("Informe um número: "))  
    b = float(input("Informe um número: "))  
    if b == 0.0:  
        print("Divisão por zero!");  
        return  
    else:  
        return a / b  
  
def main():  
    print('Resultado da divisão:',dividir())  
    #ou  
    r = dividir()  
    print('Resultado da divisão:', r)  
  
main()
```

Informe um número: 15  
Informe um número: 4  
Resultado da divisão: 3.75