



- Conceitos de Métodos
- Criação de Métodos
- Exemplo de IMC
- Ordem dos Parâmetros

Python

Lógica de Programação

Funções em Python

01 Conceito de Métodos

- Qual foi o nosso primeiro código, lembram? Foi o tradicional *Hello World* (Olá Mundo). O print é um método que o Python tem para apresentar dados ao usuário final. E, neste resumo aprenderemos a usar métodos, também conhecidos por funções.

```
main.py x
1 print('Olá mundo!')
```

- Imagine-se desenvolvendo uma aplicação para ser utilizado em academias. E, em diversos locais do aplicativo, realizamos o cálculo do IMC.
- Se, hipoteticamente, o cálculo do IMC mude, o que vai acontecer? Você vai ter que modificar a forma de calcular o IMC em todos os locais.

- E se você esquecer de alterar em algum dos locais? Zebrou, concorda?

```
1 altura = float(input("Digite sua altura (em metros): "))
2 peso = float(input("Digite seu peso (em kg): "))
3 imc = peso / altura**2
```

- Para resolver isto, precisamos usar métodos, como veremos na seção a seguir.

02 Criação de Métodos

- Para criar as nossas primeiras funções, devemos usar a palavra-chave def, conforme o exemplo a seguir.

```
1 def minhaFuncao():
2     print('Imprime algo')
3     x = 1
4     print(x)
5
6     print('Não tem nada a ver com a função')
7     minhaFuncao()
```

- No exemplo acima, criamos uma função com nome chamado *minhaFuncao*, seguido de (). Esta função possui 3 linhas, que são as linhas 2,3 e 4, devido ao recuo. A linha 6 não tem nada a ver com a função. E, na linha 7, chamamos essa função.

- Uma vez criada a função, podemos chamar quantas vezes quiser a função, como fizemos na linha 7.

```
1 def minhaFuncao(parametro1):
2     print(parametro1)
3
4     print('Não tem nada a ver com a função')
5     minhaFuncao('valor1')
```

Não tem nada a ver com a função
valor1 **Saída**

- No exemplo acima, criamos uma função diferente, pois usamos **parâmetros**. Parâmetros são informações que, quem for chamar (linha 5) passam informações para o método.
 - Na linha 5, passamos para o método *minhaFuncao* a String 'valor1'
 - Na linha 1, entre os parênteses () temos o parâmetro chamado *parametro1*, que receberá o 'valor1'. Este valor recebido é apresentado na linha 2.
- Como faremos para criar um método para cálculo do IMC? Vamos a primeira versão, que não é tão boa, mas funciona. E baseado na primeira versão, mostrarei a segunda versão.

```
1 def calculoImc():
2     altura = float(input("Digite sua altura (em metros): "))
3     peso = float(input("Digite seu peso (em kg): "))
4     imc = peso / altura**2
5     print(imc)
6
7     calculoImc()
```

- Esse método funciona, mas tem alguns probleminhas.

- #1 É bom fazer leitura (input) das informações para cálculo do IMC fora do método e, dentro do método ter somente o cálculo do IMC
- #2 É bom passar as informações como parâmetro, como fizemos anteriormente.
- #3 É bom retornar o valor ao invés de imprimir com print, dentro do método
- #4 Quero você deixar doidão por Python ☺



03 Cálculo do IMC (Versão 1)

- Abaixo temos a primeira versão. Note que, com relação a:
 - #1, as variáveis são lidas fora do método (linhas 5 e 6).
 - #2 Passamos a altura e peso (lidos nas linhas 5 e 6) como parâmetros (linha 7). Note que, o método recebe os valores na linha 1.

```
1 ▼ def calculoImc(alt, pes):
2     imc = pes / alt**2
3     print(imc)
4
5 altura = float(input("Digite sua altura (em metros): "))
6 peso = float(input("Digite seu peso (em kg): "))
7 calculoImc(altura, peso)
```

- Note que, o item #3 ficamos devendo, mas vamos desvendar na seção a seguir.

04 Cálculo do IMC (Versão 2)

- Faltou retornar, mas o que é retornar? Vimos que no código acima fizemos um print dentro do método? Pois bem, isto não é uma boa prática. O ideal é, você devolver (retornar) a quem chamar o valor que ele precisa (neste caso, o valor do IMC) e quem chamou, se quiser, imprime.
- Um bom exemplo de um método que retorna algo é o input, que retorna uma String.
 - No exemplo a seguir, chamamos input, que devolve uma String. E, pegamos esse retorno (devolução) e atribuímos a uma variável (nome).

```
1 nome = input("Digite nome")
```

- E um exemplo de métodos que não tem retorno? Print 😊

```
main.py x
1 print('Olá mundo!')
```

- Para retornar, usamos a palavra-chave return, como veremos no exemplo a seguir. Note que:
 - Tiramos o print de dentro do método;
 - O método passou a retornar o valor 3
 - Ao chamar o método (linha 7), usamos a variável chamada retorno para “pegar” o retorno do método

```
1 ▼ def calculoImc(alt, pes):
2     imc = pes / alt**2
3     return imc
4
5 altura = float(input("Digite sua altura (em metros): "))
6 peso = float(input("Digite seu peso (em kg): "))
7 retorno = calculoImc(altura, peso)
8 print(retorno)
```

- Para deixar claro, no exemplo anterior, chamamos o método usando valores lidos do usuário. Mas, podemos chamar os métodos diretamente, passando os valores “na mão”, como podemos verificar no exemplo a seguir.

```
1 ▼ def calculoImc(alt, pes):
2     imc = pes / alt**2
3     return imc
4
5 retorno = calculoImc(1.8, 85)
6 print(retorno)
```

Possíveis erros que você pode encontrar:

- Chamar método sem parâmetros, quando o método tem parâmetros
- Chamar método passando 1 parâmetro, quando o método “pede” 2 parâmetros
- Em resumo, se o método tem N parâmetros, você tem que passar N parâmetros ao chamar.

Exemplos a seguir:



```
1 ▼ def calculoImc(alt, pes):
2     imc = pes / alt**2
3     return imc
4
5 retorno = calculoImc(85)
```

```
1 ▼ def calculoImc(alt, pes):
2     imc = pes / alt**2
3     return imc
4
5 retorno = calculoImc()
```

05 Ordem dos Parâmetros

- Note que, abaixo temos um método comum com 2 parâmetros. Entretanto, chamamos o método de 2 maneiras:
 - A primeira forma (linha 4) é como usualmente realizamos, onde os argumentos são posicionados. Neste cenário, passamos 178 e 88, que implica que, parametro1 recebe 178 e parametro2 recebe 88.
 - Neste cenário, a ordem dos argumentos é essencial
 - A segunda forma (linha 5) passamos o nome do parâmetro seguindo de um ‘=’ e o valor. Desta maneira, a ordem não importa, de forma que, no exemplo passamos parametro2 antes do parametro1, tal como acontece no método.

```
1 ▼ def metodoTeste(parametro1, parametro2):
2     print(parametro1, parametro2)
3
4 metodoTeste(178, 88)
5 metodoTeste(parametro2=88, parametro1=178)
```