**04 - Definindo funções sem e com parâmetros**

Já conhecemos as *built-in functions*, mas e se quisermos resolver um problema específico para o qual não disponibilizamos de uma função pronta? Nesse caso, precisaremos criar nossas próprias funções - principalmente se for necessário utilizá-las em vários pontos do nosso código, afinal, funções são trechos de código reutilizáveis que permitem resolver problemas específicos.

Começaremos nossos estudos criando funções básicas, que não recebem parâmetros, e no futuro trabalharemos com funções mais complexas que recebem *inputs* e retornam valores. Vamos supor, por exemplo, que não exista uma função para calcular média, algo que faremos diversas vezes em nosso código.

Utilizaremos a palavra chave def seguida do nome da função e da abertura e fechamento de parênteses (que ficarão vazios, já que a função não receberá parâmetros) e, por fim, dois pontos. Na linha seguinte, precedidas por uma indentação, teremos as instruções que a função executará.

def <nome>():

<instruções>COPIAR CÓDIGO

Seguindo esse padrão, criaremos nossa função media().

def media():COPIAR CÓDIGO

Como queremos manter o código simples, iremos simplesmente calcular uma média. Nas instruções, criaremos uma variável valor para qual atribuiremos o cálculo (1 + 2 + 3) / 3 - ou seja, a soma de três valores e a sua divisão pelo número de valores. Por fim, imprimiremos o resultado guardado em valor.

Para testarmos, executaremos a nova função com media().

def media():

valor = (1 + 2 + 3) / 3

print(valor)

media()COPIAR CÓDIGO

Como retorno, obteremos o valor dessa operação.

2.0

Essa é a forma mais básica de uma função que calcula a média, e também a menos útil, afinal provavelmente precisaremos calcular as médias de valores específicos que irão variar em cada situação. Justamente para isso existem as funções com parâmetros. O esqueleto básico de uma função com parâmetros se mantém praticamente o mesmo, com a diferença sendo a inclusão desses parâmetros dentro dos parênteses.

def <nome>(<param\_1>, <param\_2>, ..., <param\_n>):

<instruções>COPIAR CÓDIGO

Definiremos então uma nova função media(). Entre os parênteses, definiremos a necessidade de três parâmetros, number\_1, number\_2 e number\_3. Calcularemos o valor, que será a soma dos três parâmetros recebidos e a subsequente divisão por três.

def media(number\_1, number\_2, number\_3):

valor = (number\_1 + number\_2 +number\_3) / 3

print(valor)COPIAR CÓDIGO

Ao executarmos, a função será salva na memória e poderá ser utilizada em qualquer ponto do nosso notebook. Agora poderemos chamar a função media() passando três valores quaisquer - nesse caso, 1, 2 e 3.

media(1,2,3)COPIAR CÓDIGO

2.0

A diferença é que agora podemos passar qualquer outro valor, como 23, 45 e 67.

media(23, 45, 67)COPIAR CÓDIGO

45

Outro ponto interessante é que, ao invés de passarmos esses valores sequencias, podemos utilizar uma lista. Criaremos outra função media(), que dessa vez recebe como parâmetro uma lista. Nas instruções, faremos o cálculo da média com o auxílio das *built-in functions* do Python: usaremos a função sum() para somarmos os itens da lista e dividiremos o resultado por len(lista), ou seja, a quantidade de valores que foram somados. Por fim, imprimiremos o valor.

def media(lista):

valor = sum(lista) / len(lista)

print(valor)COPIAR CÓDIGO

Com isso, não teremos mais a limitação de calcularmos sempre a média de três valores. Nesse exemplo, passaremos uma lista com 9 valores:

media ([1, 2, 3, 4, 5, 6, 7, 8, 9])COPIAR CÓDIGO

5.0

Uma coisa importante é que nossa função está imprimindo um valor, mas no momento não é possível atribuirmos tal valor a uma variável, afinal ele não é retornado. Por exemplo, se atribuirmos a chamada de media() a uma variável resultado, teremos a impressão do nosso valor normalmente.

resultado = media ([1, 2, 3, 4, 5, 6, 7, 8, 9])COPIAR CÓDIGO

5.0

Entretanto, se tentarmos exibir o conteúdo de resultado, nada acontecerá.

resultadoCOPIAR CÓDIGO

Ao verificarmos o tipo de resultado, receberemos um "NoneType".

type(resultado)COPIAR CÓDIGO

NoneType

Ou seja, a variável na verdade não tem valor nenhum. No próximo vídeo aprenderemos a criar funções que retornam valores que possam ser atribuídos a uma variável e reutilizados posteriormente no código.