INTRODUÇÃO A PROGRAMACÃO

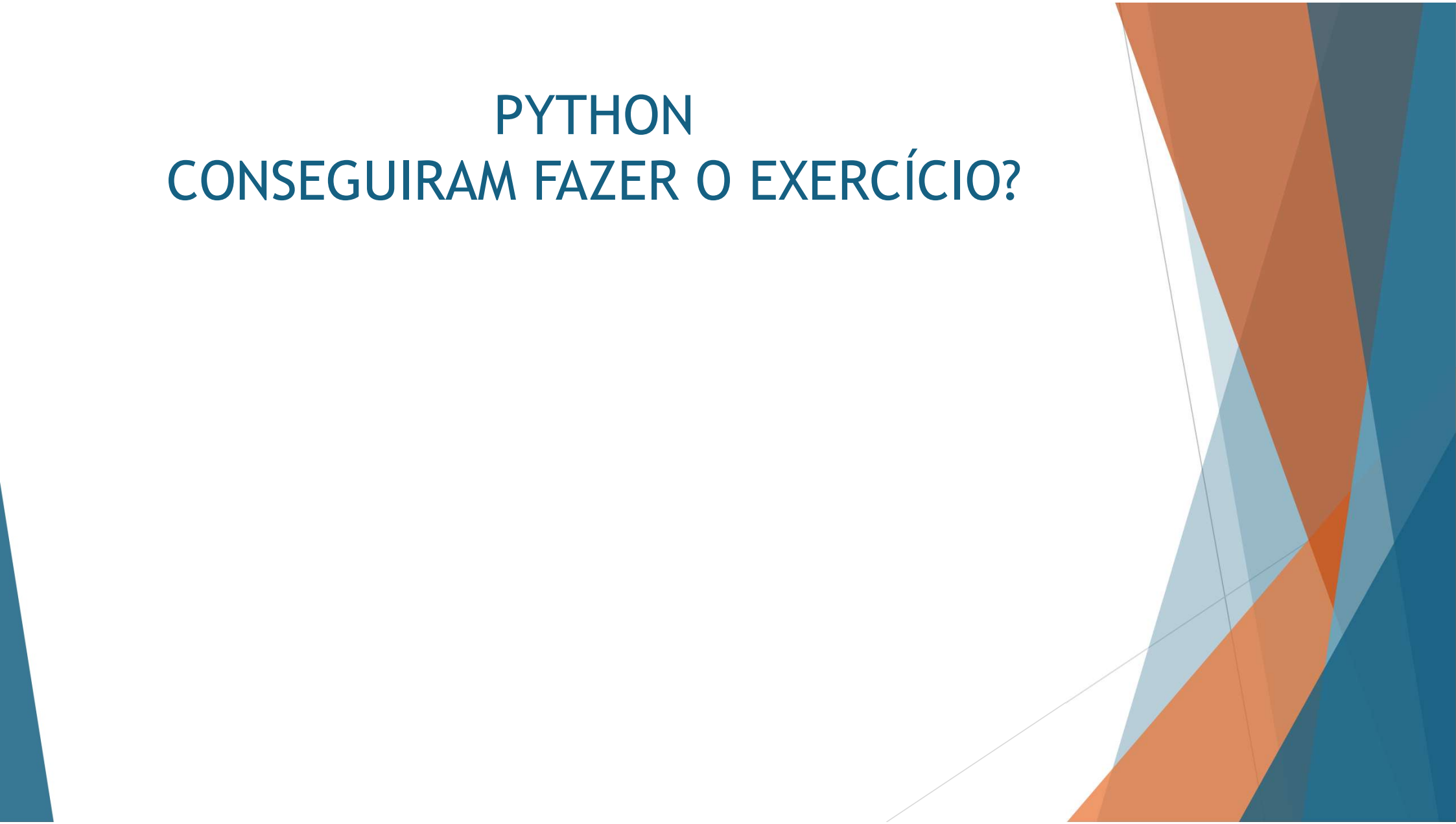
PROF. DEMÉTRIUS DE CASTRO

PROF2303@IESP.EDU.BR

83 9 87730383

WWW.DEMETRIUSDECASTRO.COM.BR

PYTHON
CONSEGUIRAM FAZER O EXERCÍCIO?



PYTHON
UM POUCO MAIS DE FUNÇÕES

Ao escrever um programa, muitas vezes precisamos repetir o mesmo código várias vezes para entradas diferentes. Para evitar duplicação, o Python permite que você chame uma função pré-determinada para aproveitar o código que ela contém. As funções ajudam a reduzir a quantidade de linhas em um código, facilitam o processo de depuração e permitem criar programas mais legíveis e fáceis de manter.

PYTHON
UM POUCO MAIS DE FUNÇÕES

O que são e para que servem?

Sendo estruturas essenciais do código, as funções permitem definir um bloco de código reutilizável que pode ser executado muitas vezes dentro de um programa. É uma estrutura que agrupa partes do código, criando soluções modulares para problemas complexos.

As funções em Python cumprem objetivos específicos definidos pelo usuário ou pela linguagem. Eles recebem como parâmetros dados de entrada chamados argumentos que são indicados pelo usuário ou automaticamente, sendo processados e retornando como dados de saída.

PYTHON
UM POUCO MAIS DE FUNÇÕES

O que são e para que servem?

As funções Python servem para:

- Dividir e classificar o código em partes mais simples para depurar e programar com maior facilidade.
- Reutilizar o código, evitando repetições desnecessárias em um programa.

PYTHON
UM POUCO MAIS DE FUNÇÕES

Tipos de funções

Podemos distinguir dois tipos de funções:

- Nativas (Built-in functions): opções disponíveis que já estão integradas no Python.
- Personalizadas: criadas pelo usuário. Para usá-las, você precisa definir as funções que atendam às necessidades do seu projeto.

PYTHON
UM POUCO MAIS DE FUNÇÕES

Funções nativas

Ex.:

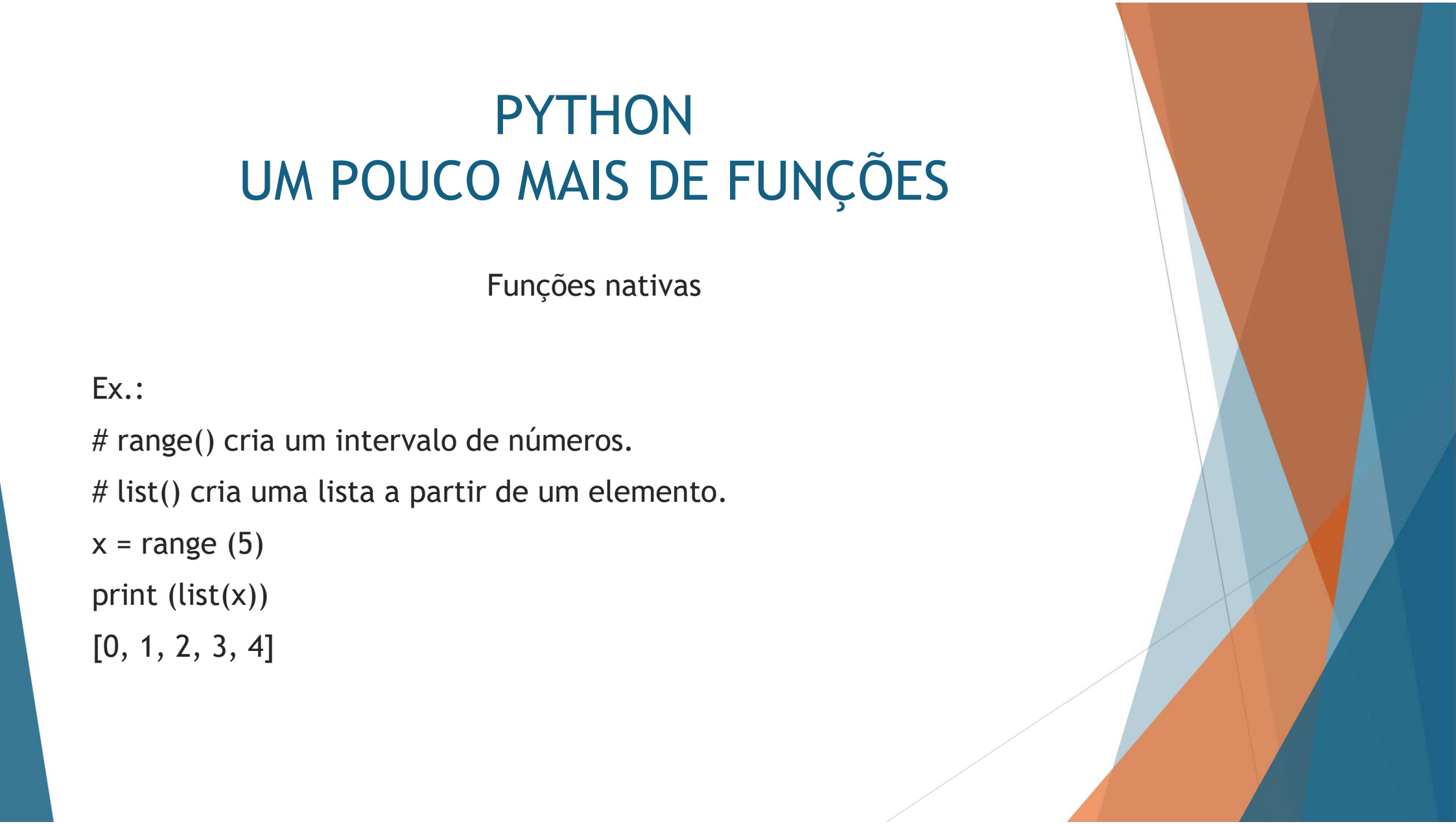
```
# print() mostra o argumento na tela;
print ("Oi")
Oi
# len() determina a quantidade de caracteres em uma cadeia deles.
len("Oi Python")
11
```

PYTHON
UM POUCO MAIS DE FUNÇÕES

Funções nativas

Ex.:

```
# range() cria um intervalo de números.
# list() cria uma lista a partir de um elemento.
x = range (5)
print (list(x))
[0, 1, 2, 3, 4]
```



PYTHON
UM POUCO MAIS DE FUNÇÕES

Funções nativas

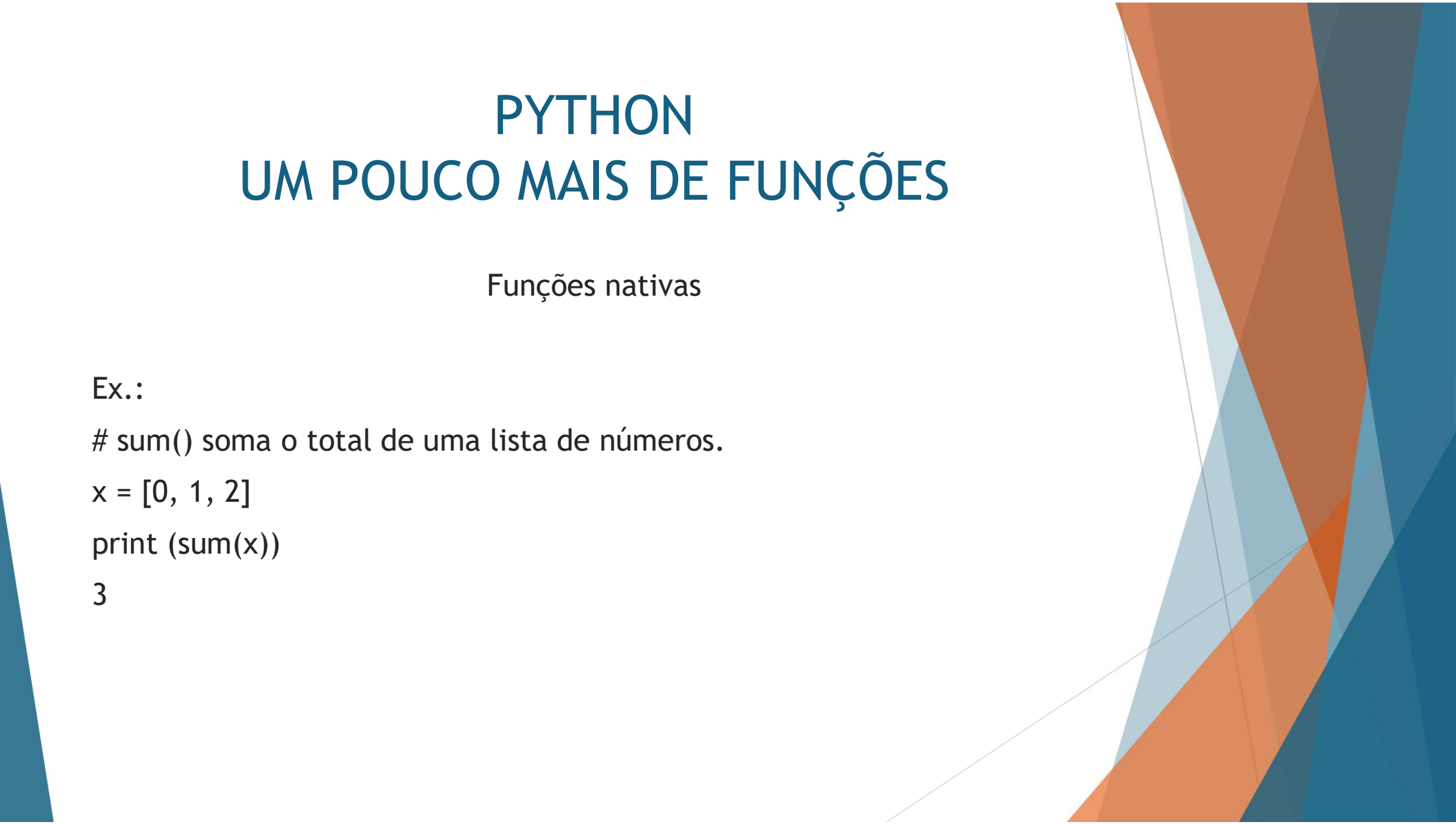
```
# max() determina o valor máximo entre um grupo de números.
x = [0, 1, 2]
print (max(x))
2
# min() determina o valor mínimo entre um grupo de números.
x = [0, 1, 2]
print (min(x))
0
```

PYTHON
UM POUCO MAIS DE FUNÇÕES

Funções nativas

Ex.:

```
# sum() soma o total de uma lista de números.
x = [0, 1, 2]
print (sum(x))
3
```



PYTHON
UM POUCO MAIS DE FUNÇÕES

Como definir uma função?

Uma função é declarada usando a seguinte sintaxe:

```
def nome (parâmetro):
    bloco de código
    return
```

PYTHON
UM POUCO MAIS DE FUNÇÕES

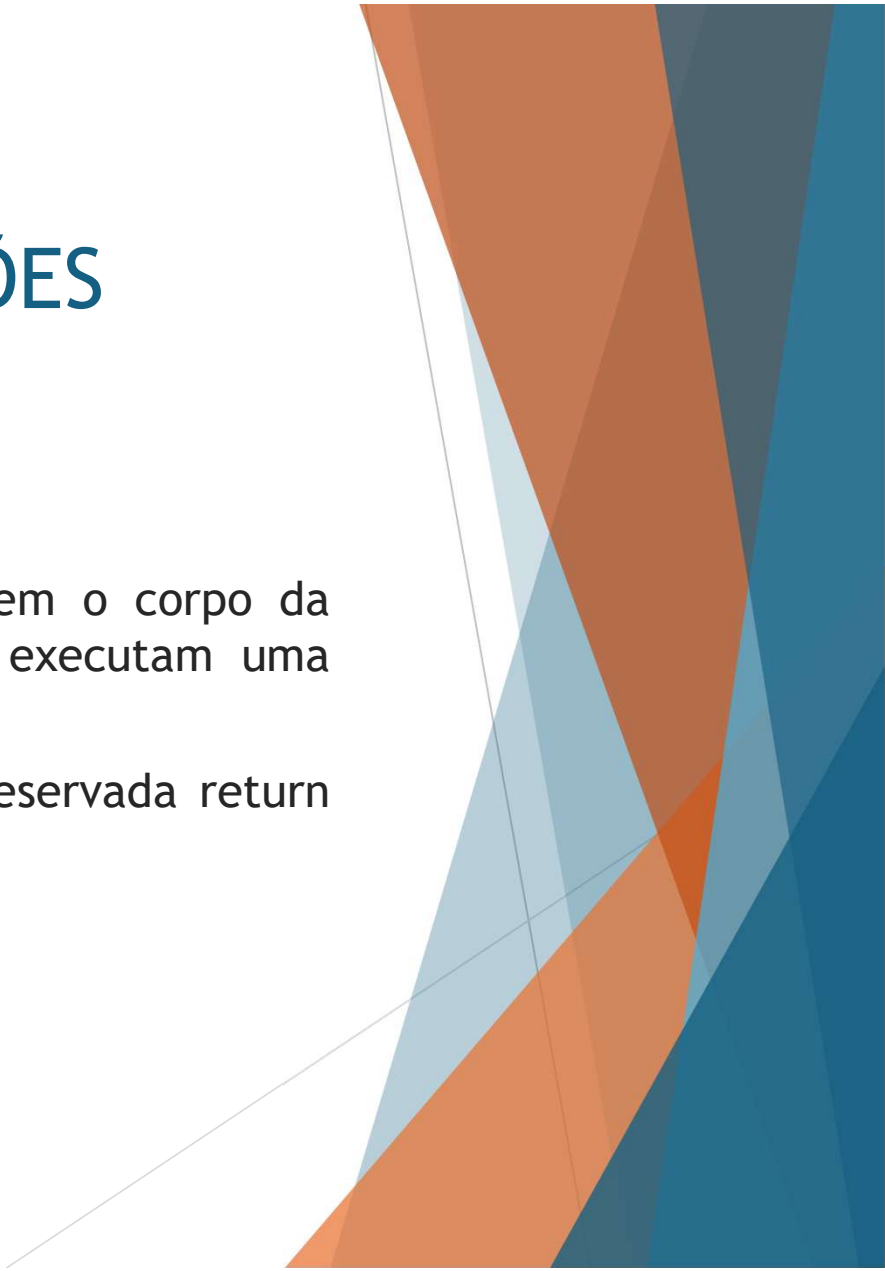
Como definir uma função?

- Para criar uma nova função em Python, a palavra reservada def é utilizada.
- Em seguida, é atribuído um nome ao identificador que será usado para chamá-la.
- Depois do nome, uma lista opcional de parâmetros é incluída entre parênteses.
- O cabeçalho da função termina com dois pontos.

PYTHON
UM POUCO MAIS DE FUNÇÕES

Como definir uma função?

- Nas linhas seguintes, com uma indentação maior vem o corpo da função, ou seja, uma sequência de sentenças que executam uma operação.
- Por último, opcionalmente, é adicionada a palavra reservada return para devolver um resultado.



PYTHON
UM POUCO MAIS DE FUNÇÕES

Como definir uma função?

Uma vez que a função é definida, você pode reutilizá-la várias vezes no seu programa.

Agora, como exemplo, podemos criar uma função simples de soma com dois parâmetros:

```
def soma(a, b):
    resultado = a + b
    return resultado
```

PYTHON
UM POUCO MAIS DE FUNÇÕES

Como definir uma função?

Para obter o resultado, chamamos a função digitando seu nome. Indicamos argumentos para parâmetros entre parêntesis e solicitamos "imprimir" para observar o resultado na tela:

```
print(soma(3,7))
>>10
```

PYTHON
UM POUCO MAIS DE FUNÇÕES

Com uma função é chamada?

A função a seguir não requer argumentos ou retorno:

```
def bem-vindo():
    print("Bem-vindo ao Python!")
```

Para aplicar a função que acabamos de definir, basta chamar seu nome.

```
bem-vindo()
```

Bem-vindo ao Python!

PYTHON
UM POUCO MAIS DE FUNÇÕES

Com uma função é chamada?

Se adicionarmos um parâmetro à função, ao chamá-la teremos que inserir um argumento:

```
def bem-vindo(nome):
    print("Bem-vindo ao Python " + nome + "!")


bem-vindo ("Pedro")
>>Bem-vindo ao Python Pedro!
```

PYTHON
UM POUCO MAIS DE FUNÇÕES

Se houver vários argumentos, podemos adicioná-los de duas formas:

Posicional: Os argumentos são adicionados na mesma ordem em que aparecem os parâmetros correspondentes na definição da função.

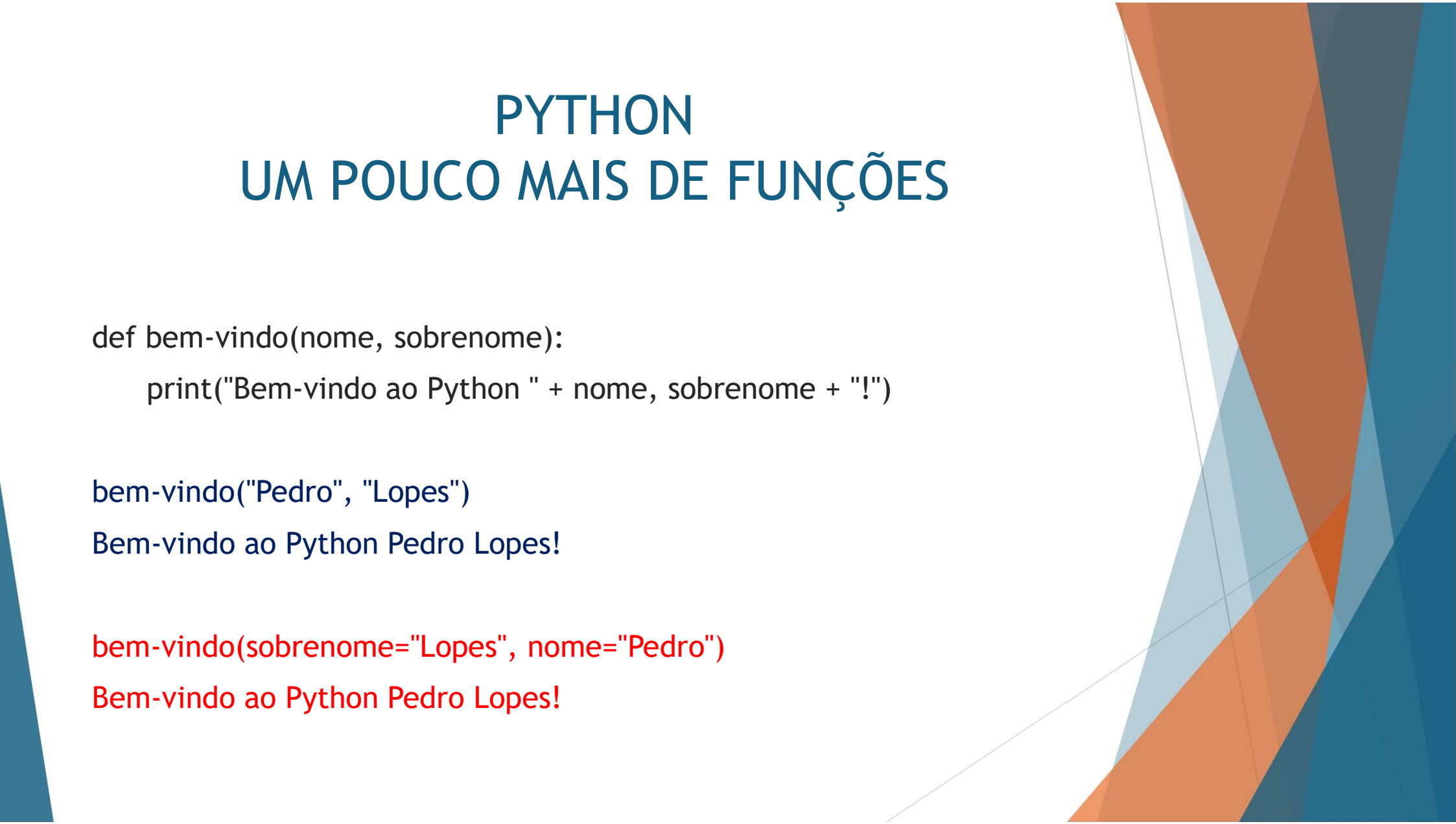
Nomeado: Independente da ordem, é especificado o nome do parâmetro ao qual se associa um argumento.

PYTHON
UM POUCO MAIS DE FUNÇÕES

```
def bem-vindo(nome, sobrenome):
    print("Bem-vindo ao Python " + nome, sobrenome + "!")

bem-vindo("Pedro", "Lopes")
Bem-vindo ao Python Pedro Lopes!

bem-vindo(sobrenome="Lopes", nome="Pedro")
Bem-vindo ao Python Pedro Lopes!
```



PYTHON
UM POUCO MAIS DE FUNÇÕES

A cada parâmetro pode ser atribuído um argumento padrão. Se a função for chamada sem atribuir um argumento a esse parâmetro, será utilizado um argumento padrão.

```
def bem-vindo(nome, idioma = "Python"):
    print("Bem-vindo ao " + idioma, nome + "!")
bem-vindo(“Thiago")
>>Bem-vindo ao Python Thiago!
bem-vindo(" Thiago ", "Java")
>>Bem-vindo ao Java Thiago!
```

PYTHON
UM POUCO MAIS DE FUNÇÕES

Às vezes, você precisa criar uma função com uma série de argumentos indeterminados para um parâmetro. Para fazer isso, colocamos um asterisco antes do nome do parâmetro e, ao chamarmos a função, adicionamos argumentos separados por vírgulas.

PYTHON
UM POUCO MAIS DE FUNÇÕES

```
def cumprimento (*nomes):
    for nome in nomes:
        print(“Oi " + nome + "!")

cumprimento ("Paulo", "Monica", “Thiago")
```

Oi Paulo!

Oi Monica!

Oi Thiago!

PYTHON
UM POUCO MAIS DE FUNÇÕES

Se o código não tiver uma instrução de retornar o valor, a função não vai devolver nada. Se executamos a função do exemplo sem instrução, aparecerá escrito None na tela.

```
def quadrado (x):
    quadrado = x ** 2

print (quadrado (5))
```

None

PYTHON
UM POUCO MAIS DE FUNÇÕES

O Python também permite que sejam devolvidos vários valores:

```
def quadrado_e_cubo (x):
    quadrado = x ** 2
    cubo = x ** 3
    return quadrado, cubo
print (quadrado_e_cubo(5))
```

25, 125

PYTHON
UM POUCO MAIS DE FUNÇÕES

As funções definidas não podem estar vazias. Mas, se por algum motivo, você tiver uma definição de função sem conteúdo e se quiser reservá-la para mais tarde, adicione a instrução pass para evitar um erro.

```
def funcao_reservada():
    pass
```

PYTHON
UM POUCO MAIS DE FUNÇÕES

Chamadas externas

Até agora aprendemos que podemos criar e chamar uma função no arquivo, mas como vimos anteriormente, criamos funções para que elas possam ser reutilizadas várias vezes e por vários arquivos diferentes.

Neste caso, podemos criar um arquivo chamado “funções.py” e adicionar todas as nossas funções neste arquivo.

Neste caso, todo arquivo onde será necessário chamar estas funções, precisaremos fazer o import do arquivo e mudar um pouco a sua chamado no arquivo.

PYTHON
UM POUCO MAIS DE FUNÇÕES

Chamadas externas

Ex.:

No arquivo que fará a chamada da função, precisaremos incluir o código:

Import “nome_do_arquivo_da_função”

E na chamada da função, precisamos colocar o nome do arquivo.nome_da_função

PYTHON
UM POUCO MAIS DE FUNÇÕES

As funções têm várias vantagens:

- Ao criar uma nova função, você define um bloco de instruções para fazer com que o seu programa seja mais fácil de ler, entender e depurar.
- As funções ajudam a reduzir linhas repetitivas. Elas permitem que você escreva o código apenas uma vez.
- Ao dividir um programa longo em funções, você pode revisá-lo e depurá-lo por partes antes de montar um código inteiro.
- As funções bem projetadas podem servir para muitos programas. Tendo escrito uma vez, você pode reutilizar a função quando quiser.

PYTHON
UM POUCO MAIS DE FUNÇÕES

Faça

Desenvolva um sistema que receba o nome e o ano de nascimento do usuário.

Crie uma função que retorne a quantidade de caracteres do nome do usuário.

Crie uma função que informe qual a idade do usuário na data de 31/12/2024.

Faça utilizando chamadas externas.

PYTHON
UM POUCO MAIS DE FUNÇÕES

Faça

Desenvolva um sistema que exiba uma lista de produtos e seus valores para o usuário (notebook – US$ 750.00 e Smartwatch – US$ 250.00). Quando o usuário escolher o produto, faça conversão do valor para REAL (1US$ = 5.09R$) e exiba para o usuário.

Após, peça para o cliente escolher o local para onde o produto será enviado, para cálculo do frete (Brasil – US$52.00 e Argentina US$75.00) e exiba para o usuário o valor convertido.

Faça uma função que realize essa conversão e por fim, exiba o valor total, em reais, para o usuário.