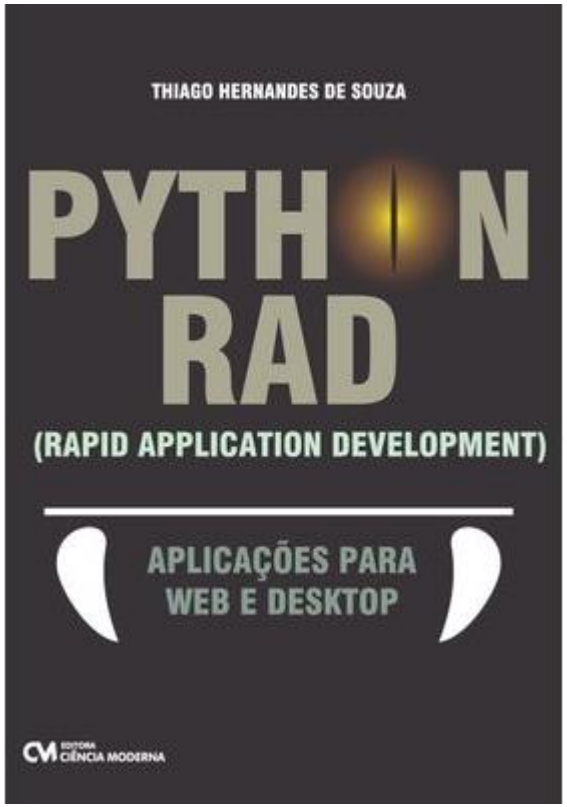
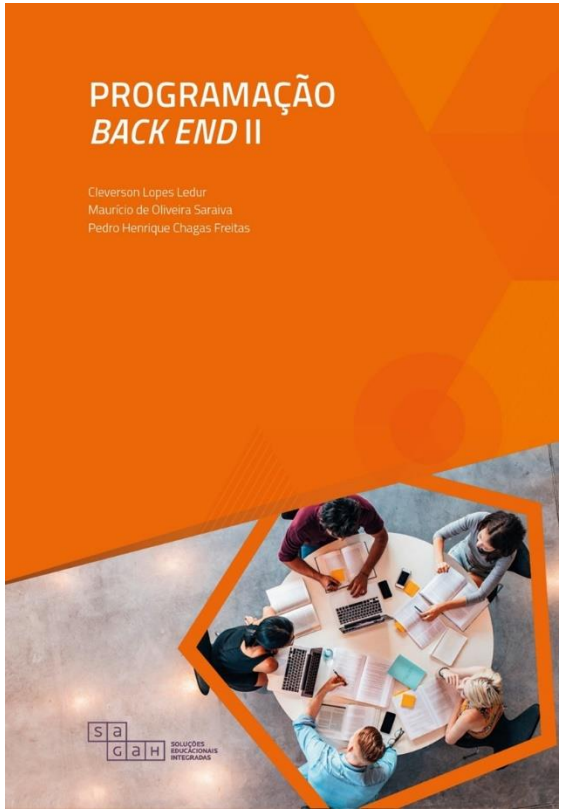
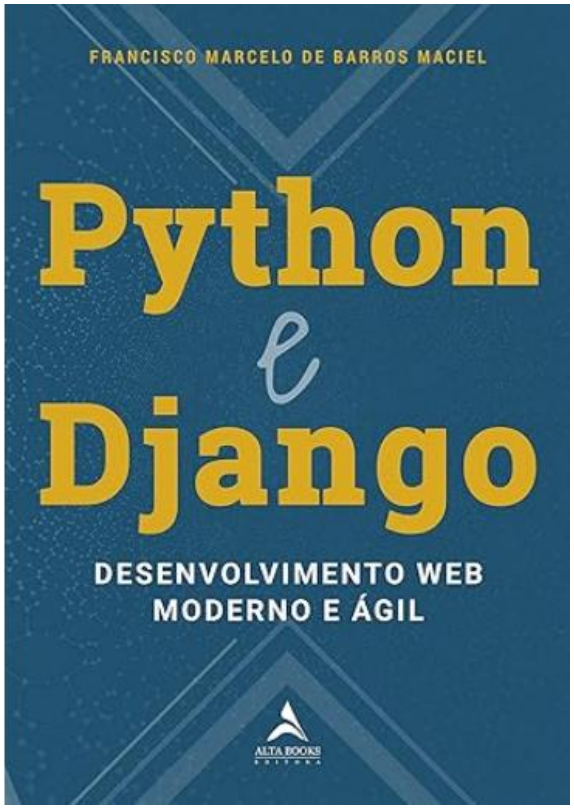
PROGRAMAÇÃO BACK-END







Python (Prática)

Funções

Funções Python

- Uma função é um bloco de código que só é executado quando é chamado.
- Você pode passar dados, conhecidos como parâmetros, para uma função.
- Uma função pode retornar dados como resultado.

teste01.py > ...

```
1   def funcao():
2       print("Executando uma função")
3
4   funcao()
```

Funções com Argumentos

- As informações podem ser passadas para funções como argumentos.
- Os argumentos são especificados após o nome da função, entre parênteses. Você pode adicionar quantos argumentos quiser, basta separá-los com vírgula.

teste02.py > ...

```
1   def funcao(fargumento):
2       print("Linux "+fargumento)
3
4   funcao("10.0")
5   funcao("9.5")
6   funcao("16.4")
```

Esta função espera 2 argumentos e obtém 2 argumentos.

teste03.py > ...

```
1   def funcao(farg1, farg2):
2       print(farg1+" "+farg2)
3
4   funcao("Linux","16.0")
5   funcao("Windows","11.0")
6   funcao("Android","14.0")
```

Caso a função não passe os argumentos, ocorre um erro.

teste04.py > ...

```
1   def funcao(farg1, farg2):
2       print(farg1+" "+farg2)
3
4   funcao("Linux","16.0")
5   funcao("Windows","11.0")
6   funcao("Android")
```

Se o número de argumentos for desconhecido, adicione um *antes do nome do parâmetro.

teste05.py > ...

```
1   def funcao(*farg1):
2       print(farg1)
3
4   funcao("Linux","Windows","Android")
```

teste06.py > ...

```
1   def funcao(*farg1):
2       print(farg1[1])
3
4   funcao("Linux","Windows","Android")
```

Argumentos de palavras-chave

Pode-se enviar argumentos com a sintaxe chave = valor. Assim, a ordem dos argumentos não importa.

teste07.py > ...

```
1   def funcao(farg3, farg2, farg1):
2       print("Item selecinado "+farg2)
3
4   funcao(farg1="Windows",farg2="Linux",farg3="Android")
```

Argumentos de palavras-chave arbitrárias, **kwargs

Se é conhecido quantos argumentos de palavra-chave serão passados para a função, adicione dois asteriscos: **antes do nome do parâmetro na definição da função. Assim a função receberá um dicionário de argumentos, e poderá acessar os itens de acordo.

teste08.py > ...

```
1   def funcao(**kwargs):
2       print("O Sobrenome é "+kwargs["Sobrenome"])
3
4   funcao(Nome = "Carlos", Sobrenome="Henrique")
```

Valor do parâmetro padrão

O exemplo a seguir mostra como usar um valor de parâmetro padrão. Se for chamado a função sem argumento, ela usará o valor padrão

teste09.py > ...

```
1   def funcao(pais = "EUA"):
2       print("O país é " + pais)
3
4   funcao("Brasil")
5   funcao("Russia")
6   funcao()
7   funcao("Israel")
```

Passando uma lista como argumento

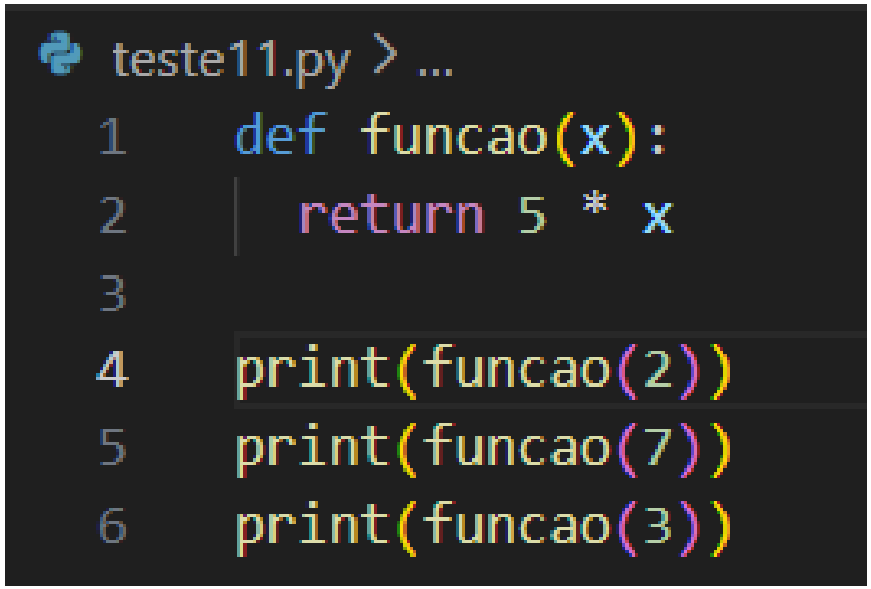
Pode-se enviar qualquer tipo de argumento de dados para uma função (string, número, lista, dicionário etc.), e ele será tratado como o mesmo tipo de dados dentro da função.

teste10.py > ...

```
1   def funcao(fcores):
2     for x in cores:
3       print(x)
4
5   cores = ["vermelho", "azul", "verde", "amarelo", "verde"]
6   funcao(cores)
```

Valores de retorno

Para permitir que uma função retorne um valor, use a return instrução.



Recursão

Python também aceita recursão de função, o que significa que uma função definida pode chamar a si mesma. A recursão é um conceito matemático e de programação comum. Isso significa que uma função chama a si mesma. Isso tem a vantagem de significar que você pode percorrer os dados para chegar a um resultado.

teste12.py > ...

```python
def fatorial(n):
    if n == 0:
        return 1
    else:
        return n * fatorial(n-1)

print(fatorial(3))
print(fatorial(5))
print(fatorial(6))
```