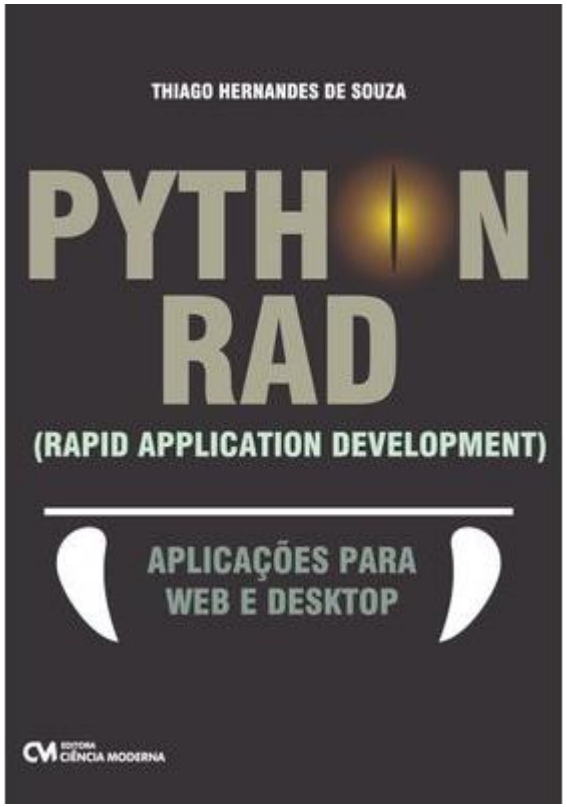
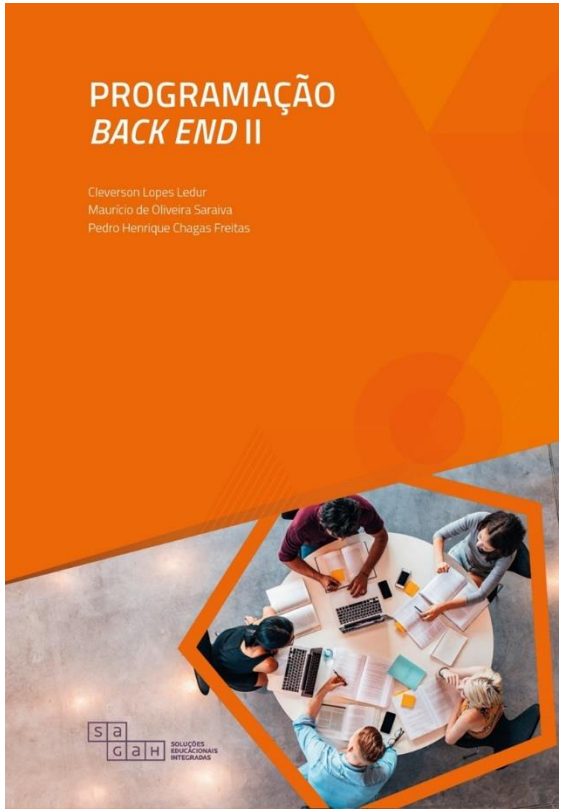
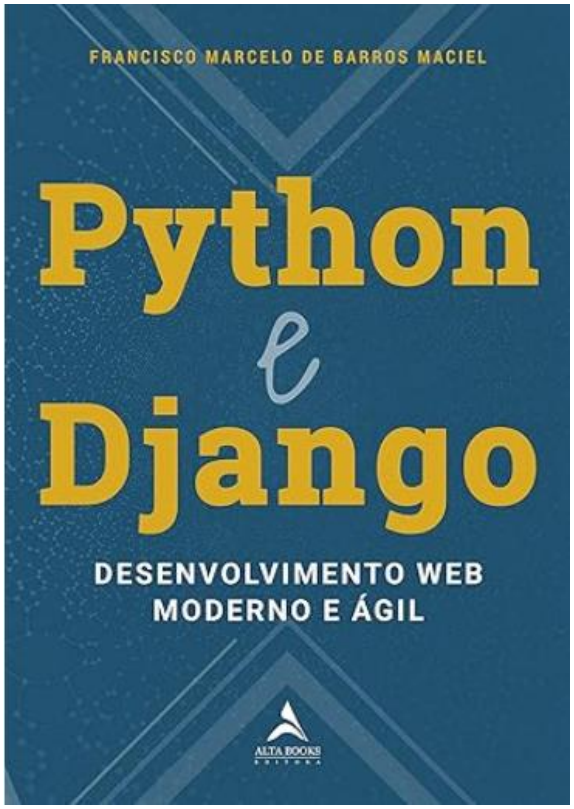
PROGRAMAÇÃO BACK-END







Python (Prática)

Listas

Listas

Listas são um dos 4 tipos de dados integrados em Python usados para armazenar coleções de dados.

As listas são usadas para armazenar vários itens em uma única variável.

As listas são criadas usando colchetes:

teste40.py > ...

```
1   lista = ["banana", "laranja", "maça"]
2   print(lista)
```

Imprime o número de itens da lista

teste41.py > ...

```
1    lista = ["banana", "laranja", "maça"]
2    print(len(lista))
```

Lista de dados do tipo string, int e booleanos

teste42.py > ...

```
1    lista1 = ["banana", "laranja", "maça"]
2    lista2 = [1, 5, 7, 9, 3]
3    lista3 = [True, False, False]
4    print(lista1)
5    print(lista2)
6    print(lista3)
```

Imprimir tipo da lista

teste43.py > ...

```
1   lista = ["banana", "laranja", "maça"]
2   print(type(lista))
```

Usando o construtor list() para fazer uma lista

teste43.py > ...

```
1   lista = list(("banana", "laranja", "maça"))
2   print(lista)
```

Imprimindo o segundo item da lista

teste44.py > ...

```
1   lista = list(("banana", "laranja", "maça"))
2   print(lista[1])
```

Imprime o terceiro, quarto e quinto itens da lista

teste45.py > ...

```
1   cores = ["vermelho", "azul", "verde", "amarelo", "roxo", "laranja", "preto"]
2   print(cores[2:5])
```

Imprime desde o primeiro, segundo e terceiro itens da lista

teste46.py > ...

```
1   cores = ["vermelho", "azul", "verde", "amarelo", "roxo", "laranja", "preto"]
2   print(cores[:3])
```

Imprime desde o quinto, sexto e sétimo itens da lista

teste46.py > ...

```
1   cores = ["vermelho", "azul", "verde", "amarelo", "roxo", "laranja", "preto"]
2   print(cores[4:])
```

Verifica se um item está contido na lista

teste47.py > ...

```python
cores = ["vermelho", "azul", "verde", "amarelo", "roxo", "laranja", "preto"]
if "azul" in  cores:
    print("Tem a cor azul na lista")
```

Substituir o terceiro item da lista

teste48.py > ...

```python
cores = ["vermelho", "azul", "verde", "amarelo"]
print(cores)
cores[2] = "laranja"
print(cores)
```

Insere um item na lista

teste49.py > ...

```python
cores = ["vermelho", "azul", "verde", "amarelo"]
cores.append("laranja")
print(cores)
```

Insere um item na lista na segunda posição

teste50.py > ...

```python
cores = ["vermelho", "azul", "verde", "amarelo"]
cores.insert(1,"laranja")
print(cores)
```

O método remove() remove o item especificado.

teste51.py > ...

```
1    cores = ["vermelho", "azul", "verde", "amarelo"]
2    print(cores)
3    cores.remove("vermelho")
4    print(cores)
```

Se houver mais de um item com o valor especificado, o método remove() remove a primeira ocorrência.

teste52.py > ...

```
1    cores = ["vermelho", "azul", "verde", "amarelo","vermelho"]
2    print(cores)
3    cores.remove("vermelho")
4    print(cores)
```

O método pop() remove o índice especificado.

teste53.py > ...

```
1   cores = ["vermelho", "azul", "verde", "amarelo"]
2   print(cores)
3   cores.pop(2)
4   print(cores)
```

Se não for especificado o índice, o método pop() removerá o último item.

teste54.py > ...

```
1   cores = ["vermelho", "azul", "verde", "amarelo"]
2   print(cores)
3   cores.pop()
4   print(cores)
```

O método clear() esvazia a lista.

teste55.py > ...

```
1   cores = ["vermelho", "azul", "verde", "amarelo"]
2   print(cores)
3   cores.clear()
4   print(cores)
```

A palavra-chave del exclui completamente a lista.

teste56.py > ...

```
1   cores = ["vermelho", "azul", "verde", "amarelo"]
2   print(cores)
3   del cores
4   print(cores)
```

Pode-se percorrer os itens da lista usando um loop for.

teste57.py > ...

```
1   cores = ["vermelho", "azul", "verde", "amarelo"]
2   for x in cores:
3       print(x)
```

Pode-se percorrer os itens da lista consultando seu número de índice. Use as funções range() e len() para criar uma iteração.

teste58.py > ...

```
1   cores = ["vermelho", "azul", "verde", "amarelo"]
2   for i in range(len(cores)):
3       print(cores[i])
```

Pode-se imprimir todos os itens, usando um loop while para percorrer todos os números de índice.

teste59.py > ...

```
1   cores = ["vermelho", "azul", "verde", "amarelo"]
2   i = 0
3   while i < len(cores):
4       print(cores[i])
5       i = i + 1
```

Pode-se usar um pequeno loop manual for que imprimirá todos os itens de uma lista

teste60.py > ...

```
1   cores = ["vermelho", "azul", "verde", "amarelo"]
2   [print(x) for x in cores]
```

Os objetos de lista possuem um método sort() que ordenará a lista alfanumericamente, em ordem crescente, por padrão.

teste61.py > ...

```
1   cores = ["vermelho", "azul", "verde", "amarelo"]
2   cores.sort()
3   print(cores)
```

Para classificar em ordem decrescente, use a palavra-chave reverse = True

teste62.py > ...

```
1   cores = ["vermelho", "azul", "verde", "amarelo"]
2   cores.sort(reverse = True)
3   print(cores)
```

O método sort() também pode ser utilizado para ordenar uma lista numericamente, em ordem crescente, por padrão.

teste63.py > ...

```
1    numeros = [23, 73, 7, 39, 58]
2    numeros.sort()
3    print(numeros)
```

Para classificar em ordem decrescente, use a palavra-chave reverse = True

teste64.py > ...

```
1    numeros = [23, 73, 7, 39, 58]
2    numeros.sort(reverse = True)
3    print(numeros)
```

Pode-se fazer uma cópia de uma lista com o método copy().

teste65.py > ...

```
1   cores = ["vermelho", "azul", "verde", "amarelo"]
2   novacores = cores.copy()
3   print(novacores)
```

Outra maneira de fazer uma cópia é usar o método integrado list().

TESTE66.PY > ...

```
1   cores = ["vermelho", "azul", "verde", "amarelo"]
2   novacores = list(cores)
3   print(novacores)
```

Junte-se a duas listas.

teste67.py > ...

```
1    cores = ["vermelho", "azul", "verde", "amarelo"]
2    numeros = [23, 73, 7, 39, 58]
3    coresnum = cores + numeros
4    print(coresnum)
```

Anexando todos os itens de uma lista à outra lista, um por um.

teste68.py > ...

```
1    cores = ["vermelho", "azul", "verde", "amarelo"]
2    numeros = [23, 73, 7, 39, 58]
3    for x in numeros:
4      cores.append(x)
5    print(cores)
6
```

PROGRAMAÇÃO BACK-END

Os métodos integrados que pode-se utilizar em listas.

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list